Applying Heuristic and Machine Learning Strategies to Product Resolution

Oliver Strauß[1][a], Ahmad Almheidat[2] and Holger Kett[1][b]

[1]*Fraunhofer-Institute for Industrial Engineering IAO, Nobelstraße 12, 73760 Stuttgart, Germany*

[2]*Institute of Human Factors and Technology Management, University of Stuttgart, Nobelstraße 12, 73760 Stuttgart, Germany*

Keywords: Product Resolution, Duplicate Detection, Machine Learning, Classification Web and Social Media Analytics.

Abstract: In order to analyze product data obtained from different web shops a process is needed to determine which product descriptions refer to the same product (product resolution). Based on string similarity metrics and existing product resolution approaches a new approach is presented with the following components: a) extraction of information from the unstructured product title extracted from the e-shops, b) inclusion of additional information in the matching process, c) a method to compute a product similarity metric from the available data, d) optimization and adaption of model parameters to the characteristics of the underlying data via a genetic algorithm and e) a framework to automatically evaluate the matching method on the basis of realistic test data. The approach achieved a precision of 0.946 and a recall of 0.673.

1 INTRODUCTION

Product resolution (also known as product matching, or de-duplication) is the task of disambiguating different appearances of a product in various diverse sources like e-shops. The use case that motivated this work is the need to analyze product reviews obtained from different e-shops via web-scraping. This is only possible, if the same product can be identified in the different sources in order to associate the collected reviews with it. Product resolution is also relevant for other e-Commerce applications like online search engines, product data aggregation for web mining applications, and product tracking across different e-shops.

Relevant data for product resolution is not always present as structured data on e-shop pages that typically contain a descriptive title, free text descriptions and tables. Extracting such structured data (e.g. product name, brand, model id, product number, units, etc.) and normalizing the extracted values can significantly improve matching precision. This paper focuses on product resolution based on the product title and product attributes previously extracted from tables and descriptions.

In this paper we describe three heuristic methods to extract structured information from unstructured product titles like `MIELE Trockner TMG 840 WP, A+++, 8 kg`: brand names (`MIELE`, Section 3.1.1), unitized values (`8 kg`, Section 3.1.2) and potential product ids (`TMG 840 WP`, Section 3.1.3). Furthermore we investigate a number of pre-filters based on this and other available information in order to reduce the number of needed comparison of product pairs in the matching process (Section 3.2.2 and 5.1.1). The actual classification is performed using two approaches that employ supervised learning: a heuristic method based on a combination of string similarity metrics with adjustable thresholds and weights which are subsequently adapted to the characteristics of the test data (Section 3.2.3) and a Random Forest classifier that uses feature vectors constructed from combinations of string similarities on different parts of the product description. A Java-based framework to perform the matching experiments has been developed and is described in Section 4.3. Results of various experiments on two real world data sets are presented in Section 5 followed by a short discussion and conclusions in Sections 6 and 7.

The contribution of this paper is not an entirely new method for product resolution but a new combination of exiting approaches and aims to help inform other experiments on the effects and performance of the prefilters and classification methods used.

[a] https://orcid.org/0000-0003-1421-2744
[b] https://orcid.org/0000-0002-2361-9733

Strauß, O., Almheidat, A. and Kett, H.
Applying Heuristic and Machine Learning Strategies to Product Resolution.
DOI: 10.5220/0008069402420249
In *Proceedings of the 15th International Conference on Web Information Systems and Technologies (WEBIST 2019)*, pages 242-249
ISBN: 978-989-758-386-5

2 RELATED WORK

The problem to identify descriptions of the same product in different web shops (product resolution) is a special case of the well researched problem of identifying duplicates in a set of records, see e.g. (Elmagarmid et al., 2007) for a summary. Most of these methods use text similarity measures to compute document similarities by measuring the degree of similarity between words, sentences, paragraphs or even whole documents. Since in the first phase of the project mainly test data were available, which contained only the product title, methods were first analyzed, which try to determine product matches exclusively on the basis of this single piece of information.

(Vandic et al., 2012) use regular expressions to extract so-called ”model words” from the product title that consist of both numeric and non-numeric characters and are potential candidates for product numbers. The similarity is then calculated as a weighted average of the cosine similarity of the title and the average Levenshtein similarity of the model words involved. (van Bezu et al., 2015) extends this approach to include product attributes in the matching. Another approach to obtaining structured information from unstructured product titles using regular expressions is provided by (Horch et al., 2015). With the help of a unit ontology, unit-related variables in the product title are identified, extracted and made available for product comparison.

Approaches to enrich the information contained in product titles by web searches and subsequent addition of characteristics and features contained in the search results are presented by (Gopalakrishnan et al., 2012) and (Londhe et al., 2014). (Londhe et al., 2014) use a graph based Community Detection algorithm for matching, while (Gopalakrishnan et al., 2012) calculate importance scores for title tokens from web searches and use the Cosine similarity of token pairs weighted by their importance.

Other more recent approaches use machine learning and neutral networks to approach product matching. (Ristoski et al., 2016) use a Conditional random Field model (CRF) in combination with text embeddings to extract product features from the product title and description and compare the approach with a dictionary based approach and a random forest classifier (Breiman, 2001). (Ristoski and Mika, 2016) also used named entity recognition to features as extract key-value pairs for subsequent matching.

(Shah et al., 2018) use a classification based approach using shallow neural network based on fastText and a similarity based approach based an Siamese networks to approach the product resolution problem.

Microdata embedded in shop web sites is used by (Petrovski et al., 2014) to match products using a bag-of-words approach. (Petrovski et al.,) extends this work with a genetic algorithm for learning regular expressions for feature extraction.

3 APPROACH

3.1 Pre-processing

The purpose of pre-processing is to prepare the data in a way that makes the matching process as efficient and precise as possible. In order to achieve this goal pre-processing tries identify and extract semantically meaningful information from the input data. In this paper we employ *brand extraction* (sec. 3.1.1), *unit extraction* (sec. 3.1.2) and *model word extraction* (sec. 3.1.3) to this end. These pre-processing steps extract data and remove it from the title. Therefore the sequence of pre-processing operations matters. To achieve the best results, the operations *brand extraction* and *unit extraction* should to be performed first, since *model word extraction* partly matches the same tokens but provides a weaker semantic context. Therefore the pre-processing workflow applied to the raw data looks as follows:

1. Trim leading an trainling whitespace
2. Perform *brand extraction*
3. Perform *unit extraction*
4. Perform *model word extraction*
5. Convert strings to lowercase

3.1.1 Brand Extraction

Brands are extracted from product titles using a heuristic method based on the assumption that the brand will be the first token in almost every title. The method works as follows:

1. Extract the first token from every product title.
2. Normalize the tokens.
3. Build a histogram of the tokens.
4. A token t_i is treated as brand, when its relative frequency $f_{rel}(t_i) = N_i/N$ is above a specified threshold f_{min} value where N_i is the number of occurences of t_i and N is the total number of tokens.

Experiments on the given test data yield good results for $f_{min} \approx 0.004$. This value was determined by manual inspection of the resulting brand list and subsequent lowering of the threshold, until the first false positives were encountered. The concrete value depends on the data used and has to be determined for every data set. The described approach works best with bigger data sets, since for small sets deviations have much more influence on the result and lead to a higher number of false positives.

Once a list of brands has been identified, a second pass through all product titles is performed to extract and/or remove brands from the title (even if they are not the first token in the title).

3.1.2 Unit Extraction and Normalization

The purpose of unit extraction is to assign semantic meaning to tokens in a product description that describe units of measurement as well a the measured quantity. Simple examples are `8 kg` or `1500 U/min`. In the following we call a quantity with an associated unit an *amount*. To be able to compare amounts with different unit scales (e.g. `mm` and `m`) within one *unit category* (e.g. *weight* or *frequency*), the amounts have to be converted to a the common *base unit* in this unit category. If this is the case, amounts can be compared by just looking at the quantities. Each unit category has a base unit that can be chosen according to the commonality of the different scales in the problem domain.

The logic that determines unit amounts is able to handle the following cases:

- The amount is given as a simple number (e.g. `10 kg` or `2.50 m`)
- The amount is given as a fraction of two integer numbers (e.g. `3/8` in or `50/60 Hz`). Two cases are distinguished: a) If the numbers do not have a common divisor the amount is treated as a fraction and is converted to a floating point number (e.g. `3/8` $\Rightarrow$ 0.375). b) If the numbers share a common divisor the amount is treated as a range (e.g. `50/60` $\Rightarrow$ [50..60]).
- The amount is given as a range of two integer numbers (e.g. `50-60 Hz`): The amount is treated as a range [50..60] in this case.

In addition to the units that are quantified by a number, energy efficiency labels of the form `A+++` are supported as a special case.

3.1.3 Model Word Extraction

Model words are tokens of a product title that contain both numeric and alphabetic/punctuation characters (Vandic et al., 2012; de Bakker et al., 2013). These tokens often represent product numbers of productIDs and are therefore relevant to determine if two product titles are matching. We slightly modified the above mentioned approach in that we also check for adjacent tokens that are either model words or tokens that contain either only upper case letters or only numeric characters. Such adjacent tokens are subsequently merged to one model word. Model words are extracted using a regular expressions and post processed to merge adjacent model words into one word. As an example the tokens `MIELE`, `TMG`, `840` and `WP` would have been extracted from the title `MIELE Trockner TMG 840 WP, A+++, 8 kg`. Since the last three extracted tokens are adjacent, the resulting model words are `MIELE` and `TMG840WP`.

3.2 Matching Strategies

3.2.1 General Approach

To identify matching products in a set of N product descriptions the following strategy is applied:

1. Formation of all possible pairings of product descriptions. This results in N^2 product pairs. Under the assumption, that the comparison function is symmetric (i.e. that comparison of product A with product B yields the same result as comparing B to A) reduces the number of pairs to $N(N-1)/2$.
2. For every pair of product descriptions the binary decision has to be made whether the two descriptions describe the same product or not. This decision can be made in several ways:
   a. By means of hard criteria, e.g. "Products are the same when their titles exactly match " (see Section 3.2.2).
   b. By means of similarity functions that are mostly based on string similarity metrics (e.g., Levenshtein, Cosine, or Jaccard, see (Elmagarmid et al., 2007)) that are combined in different ways and are subsequently applied on different parts of the product description (see Section 3.2.3). A similarity function returns a value between 0 (different) and 1 (equal) to which a threshold is subsequently applied. If the similarity is above the threshold, products are classified as equal, otherwise as different.
   c. Using a learnt classifier (see Section 3.2.4).
3. Based on the classification result a clustering strategy can optionally be applied to derive groups of similar products from the set of pair comparisons.

3.2.2 Pre-filtering

Since product resolution is based on the pairwise comparison of products, it has a complexity of $O(N^2)$

in number N of the products to be compared. To accelerate the comparison and to allow comparisons of larger product quantities filters can be applied to reduce the number of needed pair comparisons.

In principle, filters can be used in two ways: a) as *rejecting filters RF* whose goal to reduce the number of pair comparisons and b) as *accepting filters AF* that can already be considered as part of the actual matching process. They should use clear computationally inexpensive decision criteria.

Performance measures for rejecting pre-filters are pair completeness PC, reduction rate RR and the their harmonic mean F_{Filter} whereas accepting filter performance can be measured with the usual measures precision P, recall R and their harmonic mean F_1 (see Section 4.2). Pre-filters are a special case of blocking strategies (see e.g. (Köpcke, 2014)).

3.2.3 Rule-based Approach

The pursued rule-based approach is based on the Title-Model-Word method described in (de Bakker et al., 2013) ans (Vandic et al., 2012). This approach combines multiple simple string similarity functions to more complex functions. In the following, T_1 and T_2 are the two product titles, and $tok(T_1)$ and $tok(T_2)$ are the sets of tokens that each respective title is composed of. The approach is comprised of the following steps:

1. Comparison of brands and rejection of pairs (T_1, T_2) with different brands.
2. Comparison of all product titles using a simple cosine similarity metric $tCosSim(T_1, T_2)$ and accept a product pair if the similarity is above a threshold α.
3. Pairwise comparison of the model words (i.e. the product number candidates) of the two products. Comparison is performed by splitting each model word into a letter part (containing all letters in the model word) and a number part (containing all numbers). Only model words with matching letter and number parts are considered as match.
4. Breaking down the product title into individual tokens $tok(T_{1/2})$ and calculating the mean Levenshtein similarity of all combinations of token pairs $avgLevSim(tok(T_1), tok(T_2))$, where longer words receive a higher weight. The resulting title similarity *tSim* is the weighted sum of *tCosSim* and *avgLevSim* with the adjustable weight β.

$$\begin{aligned} tSim(T_1, T_2, \alpha) = \beta\, tCosSim(T_1, T_2) \\ +(1-\beta)\, avgLevSim(tok(T_1), tok(T_2)) \end{aligned} \tag{1}$$

5. Calculation of the mean Levenshtein similarity $avgLevSim(MW_{T1}, MW_{T2})$ of all model words MW_{T1} und MW_{T2}. The resulting final similarity *tSim* is the weighted sum of *avgLevSim* and *tSim* with the adjustable weight β.

$$\begin{aligned} sim(T_1, T_2, \delta) = \delta\, avgLevSim(MW_{T1}, MW_{T2}) \\ +(1-\delta)\, tSim(T_1, T_2) \end{aligned} \tag{2}$$

This approach only considers information that is available in the product title and can therefore be applied to both the basic and extended datasets. In addition, steps 1. to 3. do not depend directly on the language of the product descriptions. This generality of the approach comes with a lower performance in precision and recall.

3.2.4 Machine-learning Approach using a Random Forest Classifier

In the learning phase a Random Forest classifier (Breiman, 2001) creates a predetermined number of decision trees, which evaluate randomly selected components of a feature vector. In the classification phase, the feature vectors of the product pairs to match are classified by each of the generated decision trees. The classification result is the majority vote of all individual trees results.

The creation of meaningful feature vectors is essential for good classificaton results. A feature vector describes the properties of a pair of product descriptions. The vectors is composed of a number of components which are represented as floating point numbers. Each component encodes some information about the product pair. Examples are:

- The similarity of the product title as measured by some string similarity metric
- The similarity of the brand candidates
- The average similarity of all model word combinations
- The maximum similarity of all model word combinations
- The degree of correspondence between unit and their associated quantities
- The degree of agreement between the extracted features of the two products

The Random Forest classifier is a supervised learning method and, like the rule-based approach, requires the presence of labels in the training data that encode the information about the actual matching products.

4 EVALUATION

4.1 Test Data

The evaluation of our approach is based on two real word data sets provided by an industrial partner:

- A *basic data set* that consists of 44,758 product descriptions extracted from ten different web shops. Covered languages are German, English, French and Dutch and the data mainly consist of the product titles (e.g. `CANDY CH 647 B Glaskeramik-Kochfeld (590 mm breit, 4 Kochfelder)`). The products are from different domains ranging from coffee machines to washing machines.
- An *extended data set* that represents a subset of the basic data set and consists of 3,629 product descriptions from four German web shops. The descriptions were enhanced with structured data parsed from embedded micro formats and with product properties in the form of key-value pairs obtained by custom extractors (e.g. SKU, price, currency, etc.).

All data was obtained by web scraping and was subsequently labeled manually. Due to the data set's size the quality of labels could be fully checked. Samples have shown that the data set contains a limited number of false labels but the error rate is comparatively low, especially in the extended data set.

4.2 Evaluation Metrics

Product resolution is a binary classification problem with two possible outputs (match / no match). In order to measure the quality of a product resolution approaches, the standard binary classification metrics precision P, recall R and the F_1-measure can therefore be used (see e.g. (Elmagarmid et al., 2007)).

For pre-filters used to narrow down the number of computationally expensive comparisons, we follow (Köpcke, 2014) to use the pairs completeness $PC = M_f/M$ with M_f being the truly matching pairs after filtering and M the total matching pairs. PC indicates, how many true matches have been retained by the filter.

The reduction ratio $RR = 1 - c/pn$ indicates the deduction of the search space. Here c is the number of pairs after filtering and pn is the number of total pairs that here is given by $pn = N(N-1)/2$ with N being the number of products to compare, since products are only compared to products with higher id.

The corresponding F-measure $F_{Filter} = 2 \cdot (PC \cdot RR)/(PC + RR)$ is given as the harmonic mean of PC und RR.

4.3 Evaluation Framework

To perform the matching exeriments a modular and extensible framework for the execution and evaluation of product resolution algorithms has been implemented in Java 8 that automates the following steps of the product resolution process:

1. *Data Pre-processing* as described in Section 3.1
2. *Generation of product pairs* for comparison: For N product descriptions $N(N-1)/2$ pairs are generated, since each pair is only considered once.
3. *Outer Filter Cascade*: A sequence of accepting and rejecting filters is applied to the product pairs. Pairs for which the filters can come to a decision are added to the final result. Undecided pairs are deferred to the next steps. The outer filter cascade reduces the number of pairs that need to be processed in the classification step. However the filters remove information from the training data, which could potentially decrease learning performance.
4. *Cross Validation*: The classification performed in the following steps 5.-7. is evaluated using an n-fold cross validation strategy. The pairs are divided into n random parts of about the same size. Training and classification is now performed n times, with one part being used as test data and all other parts as training data. In this way all pairs were used once for classification and $(n-1)$ times for training.
5. *Training*: In case of the rule-based approach (see Section 3.2.3) the training step consists of the optimization of the adjustable parameters (threshold and weights) in the similarity function. The framework supports optimization by Mesh-based Optimization Genetic Algorithms (Wilhelmstötter, 2018), Bound Optimization by Quadratic Approximation (BOBYQA) (Powell, 2009) and Covariance Matrix Adaptation Evolution Strategy (CMA-ES) (Auger and Hansen, 2005). In case of the machine learning approach training is performed according to the Random Forest algorithm (see Section 3.2.4).
6. *Inner Filter Cascade*: A sequence of accepting filters can be applied after rule-based or learnt classification if precise and efficient criteria to find matching pairs are available.
7. *Classification*: In this step, the trained rule-based or machine learning classifier is applied to the remaining product pairs and the final classification decision is arrived at.
8. *Summarizing and documentation* of the results.

5 RESULTS

5.1 Extended Data Set

The results in this section are based on the 3,629 product description of the extended data set, for which product features and structured data from embedded micro formats were available.

5.1.1 Pre-filter Results for the Extended Data Set

The following rejecting pre-filters (RF) have been considered:

- RF_{Brand}: Pairs with differing brand are rejected.
- RF_{GTIN13}: Pairs with defined but different GTIN13-value are rejected.
- RF_{SKU}: Pairs with defined but different SKU-value are rejected.
- $RF_{MWPrefix3}$: Pairs that have no model words pair with matching prefixes of length 3 are rejected.
- $RF_{FuzzyMW5}$: Pairs where all model word pairs have a Levenshtein distance greater than 5 are rejected.

The performance of these rejecting pre-filters is summarized in Table 1.

The following accepting pre-filters (AF) have been considered:

- $AF_{GTIN13/SKU}$: Pairs with defined and matching GTIN13 or SKU values are considered a match.
- AF_{MW8}: Pairs with perfectly matching model words of length > 8 are considered a match.
- AF_{Title}: Pairs with perfectly matching normalized titles are considered a match.
- Cascade of $AF_{GTIN13/SKU}$, AF_{MW8} und AF_{Title}.

The performance of these accepting pre-filters is summarized in Table 2.

Accepting and rejecting filters can be combined. The order in which the filters are applied is significant, since filters are processed in sequence where each subsequent filter is applied only to the pairs that have not been decided by the previous filters. Table 3 shows the performance of a sequence of two accepting and one rejecting filters ($AF_{GTIN13/SKU} \rightarrow AF_{MW8} \rightarrow RF_{Brand}$). Only three product pairings were wrongly marked as matching by the filters and nine actual pairings were not found. The number of pair comparisons left for classification is reduced from 6,147,612 to only 434,789.

Table 1: Rejecting pre-filter performance.

Filter	PC	RR	F_{Filter}
RF_{Brand}	98.61	93.39	95.93
RF_{GTIN13}	98.72	7.41	13.78
RF_{SKU}	83.87	33.51	47.89
$RF_{MWPrefix3}$	95.62	74.03	83.45
$RF_{FuzzyMW5}$	75.53	97.71	85.20

Table 2: Performance of various accepting pre-filters. The values refer to the decisions of the pre-filter and not to the total result of the matching process hence FN and TN are 0.

Filter	P	TP	FP	FN	TN
$AF_{GTIN13/SKU}$	100.00	359	0	0	0
AF_{MW8}	99.32	436	3	0	0
AF_{Title}	95.28	101	5	0	0
$AF_{GTIN13/SKU} \rightarrow AF_{MW8} \rightarrow AF_{Title}$	98.70	608	8	0	0

Table 3: Performence of the filter combination $AF_{GTIN13/SKU} \rightarrow AF_{MW8} \rightarrow RF_{Brand}$.

Filter	P	R	F_1
$AF_{GTIN13/SKU} \rightarrow AF_{MW8} \rightarrow RF_{Brand}$	99.50	98.50	99.00

5.1.2 Classification Results for the Extended Data Set

Four experiment based on the extended data set were performed:

1. Exp_1: The rule-based approach described in Section 3.2.3 was combined with the filters $AF_{GTIN13/SKU} \rightarrow AF_{MW8} \rightarrow RF_{Brand}$ in the outer filter cascade (see also Table 3).
2. Exp_2: The Random Forest classifier was combined with the filters $AF_{GTIN13/SKU} \rightarrow AF_{MW8} \rightarrow RF_{Brand}$ in the outer filter cascade.
3. Exp_3: Random Forest classifier was combined with the filters $RF_{GTIN13/SKU} \rightarrow RF_{GTIN13} \rightarrow RF_{SKU} \rightarrow RF_{Brand} \rightarrow AF_{Title}$ in the outer filter cascade.
4. Exp_2: The Random Forest classifier was combined with the filters $AF_{GTIN13/SKU} \rightarrow RF_{Brand} \rightarrow AF_{MW8}$ the outer filter cascade.

The results are shown in Table 4. It can be observed that the Random Forest method (Exp_2) achieves a significantly higher precision of $P = 98.51\%$ compared to the rule-based method (Exp_1) with $P = 54.41\%$. However, the rule-based method (Exp_1) achieved a higher recall of $R = 68.59\%$ compared to the $R = 63.46\%$ of the random forest classifier (Exp_2). The selection of filters and their order have a significant impact on the overall performance. An experiment with stronger rejecting filters (Exp_3) was able to increase the precision again to 99.7% at the expense of the recall. In this run, only a single product pair was incorrectly classified as matching. By interchanging the last two filters of experiment Exp_2, experiment Exp_4 achieved the most balanced result with the best value for the F_1 measure.

Table 4: Performance of the four experiments Exp_{1-4} performed on the extended data set.

Filter	P	R	F_1
Exp_1	54.41	68.59	60.68
Exp_2	98.51	63.46	77.19
Exp_3	99.73	38.78	55.84
Exp_4	94.59	67.27	78.63

5.2 Basic Data Set

The results in this section are based on the 44,758 product descriptions of the basic data set that provide only product titles for matching. To reduce the number of pair comparisons the product descriptions were split based on the brand extracted from the title. This resulted in 34 parts of different sizes, all of which, however, remained manageable. The effects of this split were a pair completion of $PC = 99.26$, a reduction ratio of $RR = 93.37$ and $F_{Filter} = 96.22$. Only 0.74% of actual matches were lost, but the number of comparisons was reduced from over one billion to less than 66.5 million.

The experiment conducted used pre-filtering with $AF_{Title} \rightarrow AF_{MW8} \rightarrow RF_{MWPrefix3} \rightarrow RF_{FuzzyMW5}$ and classification with the Random Forest classifier. It resulted in a precision of $P = 87.28$, a recall of $R = 61.89$ and $F_1 = 72.43$.

Compared with the results of the extended data set, which provides much more information about the products that can be exploited, the result is clearly worse. The additional information leads to a lower number of false positives and thus increases the precision of the matching to $P = 98.51\%$ for the extended data. The recall is similar for both data sets ($R = 63.46\%$ for the extended data set vs. $R = 61.89\%$ for the basic data set).

6 DISCUSSION

The experiments have confirmed that additional structured data can improve the precision of the matching. Comparison of the basic data set (only unstructured product title) and the extended data set (including product properties and metadata) resulted in an increase of precision from $P = 87.3\%$ to $P = 98.5\%$ and recall from $R = 61.9\%$ to $R = 63.5\%$. An even higher precision is to be expected when the names of product properties are not in the form of shop-dependent strings but unified with shop-independent identifiers. The *Random Forest classifier* achieved a higher precision of ($P = 98.5\%$) compared to the rule-based approach ($P = 54.4\%$). The recall is slightly lower though ($R = 63.5\%$ compared to $R = 68.6\%$). The best compromise was with a precision of $P = 94.6\%$, a recall of $R = 67.3\%$ and an F_1-value of $F_1 = 78.6\%$.

The experiments confirmed the big influence of *pre-processing* (in particular the extraction of structured information from unstructured text) on the results of the matching process. *Prefiltering* can reduce the number of comparison pairs significantly thus making the compute-intensive matching of a large number of products possible but there is a conflict of objectives between the reduction rate and the number of actual matches lost in the filtering. The right balance depends on the use case and the number of products to compare. When applied to big data volumes the presented methods reach their limit relatively quickly ($n \approx 50,000$) due to the $O(n^2)$ nature of the matching problem. Prefiltering can only partly solve this problem, since it does not change the nature of the problem, but only reduces the n.

7 CONCLUSIONS AND FUTURE WORK

With the developed method a fully automated product comparison process can be implemented, provided the expected errors are tolerable. If high precision is required, the Random Forest-based method yields very good results, as only less than two percent false positives are to be expected. All tested approaches show a recall of approx. $60-68\%$. If the use case tolerates that not all matching products are found, the approach is suited for automatic use.

The integration of our product comparison approaches into a *semi-automated process* requires the presence of confidence information for each classification decision in order for the system to decide which match decisions should be reviewed by a human expert. In the described evaluation framework this is currently not directly possible. In the heuristic approach confidence information can be derived from the calculated similarities and for the Random Forest method it can be obtained from a posteriori values provided by the classifier for each classification decision.

Next steps in the context of the presented use case is the introduction of confidence values and the comparison of the obtained result with similar approaches by performing the described experiments and the datasets of the WDC Gold Standards for product matching (Petrovski et al., 2017). Future work will be the application of word embeddings (Pennington et al., 2014) and character embeddings similar to (Ristoski et al., 2016) the problem of product resolution and to combine these approaches with the preprocessing and filtering methods described in this paper.

ACKNOWLEDGEMENTS

This work has been made possible by the Eurostars project E!10138 ReProsis ”Big Data Product Analysis in Real Time – Product Management System for International Markets”, sub-project ”Intelligent Product Data Extraction and Product Resolution” funded by the German Bundesministerium für Bildung und Forschung (BMBF) under the grant number 01QE1632B.

REFERENCES

Auger, A. and Hansen, N. (2005). A Restart CMA Evolution Strategy With Increasing Population Size. In *2005 IEEE Congress on Evolutionary Computation*, volume 2, pages 1769–1776, Edinburgh, Scotland, UK. IEEE.

Breiman, L. (2001). Random Forests. *Machine Learning*, 45(1):5–32.

de Bakker, M., Frasincar, F., and Vandic, D. (2013). A hybrid model words-driven approach for web product duplicate detection. In *Advanced Information Systems Engineering*, volume 7908 of *Lecture Notes in Computer Science*, pages 149–161. Springer, Berlin, Heidelberg.

Elmagarmid, A. K., Ipeirotis, P. G., and Verykios, V. S. (2007). Duplicate Record Detection: A Survey. *IEEE Transactions on Knowledge and Data Engineering*, 19(1):1–16.

Gopalakrishnan, V., Iyengar, S. P., Madaan, A., Rastogi, R., and Sengamedu, S. (2012). Matching product titles using web-based enrichment. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, pages 605–614, Maui, Hawaii, USA. ACM.

Horch, A., Kett, H., and Weisbecker, A. (2015). Extracting Product Unit Attributes from Product Offers by Using an Ontology. In *Proceedings of The Second International Conference on Computer Science, Computer Engineering, & Social Media*, Lodz, Poland. IEEE.

Köpcke, H. (2014). *Object Matching on Real-World Problems*. Dissertation, Universität Leipzig, Leipzig.

Londhe, N., Gopalakrishnan, V., Zhang, A., Ngo, H. Q., and Srihari, R. (2014). Matching titles with cross title web-search enrichment and community detection. *Proceedings of the VLDB Endowment*, 7(12):1167–1178.

Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Petrovski, P., Bryl, V., and Bizer, C. Learning Regular Expressions for the Extraction of Product Attributes from E-commerce Microdata. page 10.

Petrovski, P., Bryl, V., and Bizer, C. (2014). Integrating product data from websites offering microdata markup. In *Proceedings of the 23rd International Conference on World Wide Web - WWW '14 Companion*, pages 1299–1304, Seoul, Korea. ACM Press.

Petrovski, P., Primpeli, A., Meusel, R., and Bizer, C. (2017). The WDC Gold Standards for Product Feature Extraction and Product Matching. In Bridge, D. and Stuckenschmidt, H., editors, *E-Commerce and Web Technologies*, volume 278, pages 73–86. Springer International Publishing, Cham.

Powell, M. J. D. (2009). The BOBYQA algorithm for bound constrained optimization without derivatives. Technical Report, University of Cambridge, Cambridge, UK.

Ristoski, P. and Mika, P. (2016). Enriching Product Ads with Metadata from HTML Annotations. In *Proceedings of the 13th International Conference on The Semantic Web. Latest Advances and New Domains - Volume 9678*, pages 151–167, Berlin, Heidelberg. Springer-Verlag.

Ristoski, P., Petrovski, P., Mika, P., and Paulheim, H. (2016). A machine learning approach for product matching and categorization: Use case: Enriching product ads with semantic structured data. *Semantic Web*, 9(5):707–728.

Shah, K., Kopru, S., and Ruvini, J. D. (2018). Neural Network based Extreme Classification and Similarity Models for Product Matching. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 3 (Industry Papers)*, pages 8–15, New Orleans - Louisiana. Association for Computational Linguistics.

van Bezu, R., Borst, S., Rijkse, R., Verhagen, J., Vandic, D., and Frasincar, F. (2015). Multi-component similarity method for web product duplicate detection. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, pages 761–768, Salamanca, Spain. ACM Press.

Vandic, D., Van Dam, J.-W., and Frasincar, F. (2012). Faceted product search powered by the Semantic Web. *Decision Support Systems*, 53(3):425–437.

Wilhelmstötter, F. (2018). Jenetics - Library User's Manual 4.3. Technical Report, Vienna, Austria.