Product data is mostly embedded in webpages with Microdata markup [2]. For the experiment presented in this paper, we use a subset of the extracted Microdata consisting of 9,454,403 product offers. There are two main vocabularies with which products are marked up in Microdata: schema[5], data-voc[6]. The most common class for marking up products in the two vocabularies is **Product** with more than 95% usage. The most frequent Microdata attributes that are used to describe products are *title* and *description*, with 86% and 61% presence respectively. The *title*s usually contain the name of the product plus several other product features which are considered important for marketing the product. The *description* attribute contains a free-text describing the product and potentially also comparing it with competing products. All other properties that are defined in the *schema* and *data-voc* vocabularies for describing products are used within less than 50% of the descriptions [2].

2. **Product Classification** As we want to restrict our investigation to electronic products, we need to determine the subset of all offers that fall into this product category. Products on the web are primarily described with text, therefore, classification methods, such as Naive Bayes, K-Nearest Neighbors or SVM, can be used. Various classification hierarchies have been already developed, for instance, Amazon product classification (Table 1). Adopting an existing classification hierarchy helps validate the applied method.

3. **Product Feature Extraction** Microdata product offers are not structured enough for identifying which e-shops offer the same product using identity resolution techniques. Thus as a pre-processing step, we need to extract more structured product features such as brand, model, memory size, screen size, etc. from the titles and descriptions of the products. Product features can be extracted from a textual description either by manually defining regular expressions [3] or by learning extraction models using existing well-structured data as training data [4]. We use a combination of both approaches and learn feature extractors from Amazon data.

4. **Identity Resolution** Once distinguishing product features have been extracted from the Microdata, identity resolution techniques [5] can be applied in order to find out which offers from different e-shops refer to the same product. For this task, we employ the Silk Link Discovery Framework[7], an open-source tool for discovering relationships between data items that are represented as RDF. Silk relies on genetic programming to learn which attributes to compare using which similarity metric and how to combine the similarity scores for different attributes [6].

5. **Data Fusion** Within Microdata, the descriptions of related entities such as reviews or ratings refer to the described product entity. Thus after we have identified all e-shops that offer a specific product, we can combine the ratings and reviews for the product from all shops.

While related work [2–4] covers Microdata extraction, product feature extraction, and identity resolution as separate tasks, our work covers the whole pipeline from product data extraction to integration.

[5] `http://schema.org/`
[6] `http://data-vocabulary.org`
[7] `https://www.assembla.com/spaces/silk/wiki`

Table 1: Precision and Recall table for product classification

Category	Precision %	Recall %
Books	86.58	87.95
Movies, Music, & Games	89.81	70.63
Electronics & Computers	**92.98**	**88.00**
Home, Garden & Tools	73.81	60.78
Grocery, Health & Beauty	70.20	72.86
Toys, Kids, Baby & Pets	75.00	64.85
Clothing, Shoes& Jewelry	88.56	89.93
Sports & Outdoors	72.83	67.90
Automotive & Industrial	73.06	65.50
Average	***80.31***	***74.26***

2. PRODUCT CLASSIFICATION

This Section describes the classification techniques that we used to assign each Microdata offer to one of the 9 main product categories of the Amazon product catalog and to determine the subset of the offers that fall into the category electronics.

Since we only have *title* and *description* property values for most offers, we need to treat the task as a text classification problem and learn a classifier for these two properties. As a training set for our model we have chosen 18,000 labeled products with titles and descriptions (Editorial review), from Amazon.com, classified into 9 product categories. We filtered the Microdata product descriptions that we extracted from the Common Crawl to only contain English descriptions that are at least 20 words long (descriptions shorter than 20 words do not contain enough information to be successfully classified). This reduced the number of offers from 9,454,403 to 1,986,359.

The 4-step process of feature generation for the classification is based on generating a word vector from the documents (title + description) from our training set. The first step is tokenizing (by non-alpha characters) the text and removing stop words. The second step employs multiple pruning techniques in order to reduce the number of features. First we prune by the relative frequency of the terms in the documents, with a lower bound of 0.2% and a upper bound of 98% (the bounds were set by empirical testing). In addition to the relative frequency pruning, weighted association analysis was done in order to determine the terms association to a certain class. The computed weights served as a pruning bound if a certain term is only weakly associated with any class (the maximum weight given to a term is smaller than 0.2) or if a certain term is highly associated with multiple classes (the second highest weight is within 50% of the max weight assigned to a term). The third step is creating 1 to 4-grams out of the remaining terms that resulted in ~3600 features. As a last step, we computed TF-IDF for the ~3600 features in order to get normalized feature vectors.

We have chosen the Naive Bayes classification model for the classification. This allows products to be labeled as non-matching against any of the 9 classes, which in turn allows better precision in classifying. The model is trained with the generated features, as explained above. To improve the precision of the model two threshold functions are applied. Firstly, a flat threshold of 0.15 is applied: if a products' highest probability given by the Naive Bayes model is lower than 0.15 the product is assigned to the 'Other Products' category. To determine the threshold value we simplified the problem to binomial classification (match or a non-match

Title	AppleMacBook Air MC968/A 11.6-Inch Laptop
Description	Faster Flash Storage with 64 GB Solid State Drive and USB 3.0. 720p FaceTime HD Camera. The new 1.6 GHz Intel Core i5 Processor with Intel HD Graphics 3000 enabling beautiful rendering and 4GB DDR3 RAM. 11.6" LED display with the best resolution…

Title	Apple MacBook Air 11-in, Intel Core i5 1.60GHz, 4 GB, 64 GB, Mac OS X Lion 10.7
Description	The MacBook Air MC 968/A powered by Intel Core i5(1.6GHz, 3MB L3). 64 GB SSD and 4096 MB of DDR3 RAM. 29.464cm (11.6") TFT 1366x768, Intel HD Graphics, IEEE 802.11a/b/g, Bluetooth 4.0, FaceTme camera, OS X LIon

(a)

Brand	Apple
Model	MacBook Air MC968/A
Processor	1.6 GHz Core i5-560UM
Storage	64GB
Display	11.6-Inch
…	…

(b)

Figure 2: (a) WDC product offers; (b) Amazon product specification

against a certain class) and compared the ROC curves[8]. Secondly, an analysis on the probabilities given for all the classes for a specific product was conducted. The analysis determined if the maximum probability for a product is greater than the other probabilities with a margin big enough to allow greatest certainty of the prediction. This in turn is used as the second threshold for improving our models' precision. Table 1 shows the per-class and macro average precision and recall out of a 10 fold cross validation. For experiment purposes we used a Rapidminer[9] implementation of the Naive Bayes classification model.

Further steps of the integration process were applied to the electronics product category, however, the methods can be easily extended to other domains. The dataset contains 219,118 electronics products. Figure 2(a) shows an example of two electronics product offers from WDC dataset describing the same product. Matching these product offers is a non-trivial task since (1) user generated product descriptions often follow different patterns and/or different levels of detail (the top product description contains less technical description than the bottom one); (2) numeric values are often imprecise, e.g. due to rounding (the top product description contains an 11.6 inch laptop versus the 11 inch laptop in the bottom one); (3) abbreviations are used differently (the bottom product description uses SSD as abbreviation, while the top refers to the full name Solid State Drive).

3. PRODUCT FEATURE EXTRACTION

Isele et al. [6] shows that identity resolution with genetic programming works well with structured clean data. However, product offers that are described with free text often lack well defined structure and contain a lot of noise. Thus as a pre-processing step, we need to extract more structured product features such as brand, model, memory size, screen size, etc. from the titles and descriptions of the products. The required product feature extractors can either be set up manually or learned automatically. In the case of automatically learning the extractors, the process consists of two steps: (1) building a model for each of the items' properties from structured data (training data), and (2) applying the model on the data containing the free-text, i.e extraction of properties from the free-text.

Köpcke et al. [7] perform information extraction on product offers. While the domain is the same as ours, only free-text properties were used for entity resolution in the study. Product features were extracted from the title by manually defining regular expressions. Similarly, record linkage between free-text product offers and structured product specifications has been introduced in [4]. Structured product specifications were used to learn a model, which in turn was used for extracting product features from unseen product offers. Our approach combines the extraction principles from these two previous works.

As training data we used products from Amazon.com. The dataset contains 20 structured electronics products that were used to train the model, and were gathered using the Amazon Product API. The selection was based on the Amazon.com current offer that intersects with the WDC product dataset (August 2012). Similarly to the WDC product offers, Amazon product offers contain free text title and description for each product, however, additionally, they also contain structured description in terms of property-value pairs shown in Figure 2(b).

For the purposes of extracting features from textual descriptions, we implemented the Free Text Preprocessing tool[10], an open source tool that bundles free text transformation and extraction methods. The main goal of the tool is to produce a structured representation of data that contains free text. The tool takes as an input an RDF file with properties with free text values and an additional RDF file that contains structured data. The resulting output is an RDF file contaning the newly created structured objects.

3.1 Extractors

The first extractor uses **bag-of-words** as a model. The creation of the model consists of tokenizing the values and building lists of those tokens for every property contained in the training dataset (the structured data), shown in Figure 3(a). The extraction process consists of tokenizing the text of the free-text property and matching the tokens against the model explained above.

The bag-of-words extractor is suitable for extracting nominal and numerical (with units of measurements) attributes. However, the extractor is not suitable when it comes to extracting multi-token value attributes. Additionally as a trade off to the simplicity of the model, this extractor is unable to extract values that might refer to more than one property. For instance, from Figure 3(a) it can be seen that the value of 4GB can refer to Storage or Memory.

The second extractor uses **feature-value pair** as a model; it keeps a list of feature-value pairs per item belonging to the training data set. The lists are shown in Figure 3(b), where the first 3 pairs describe a laptop and the last 2 describe a TV. Similarly to the bag-of-words-extractor lists of values per property are built. Subsequently, tagging is performed

[8]Receiver Operating Characteristic (ROC), is a graphical plot which illustrates the performance of a binary classifier system as its discrimination threshold is varied, i.e. plotting precision vs. recall

[9]Rapidminer - `http://rapidminer.com/`

[10]`https://www.assembla.com/spaces/silk/wiki/Silk_Free-text_Pre-processing`

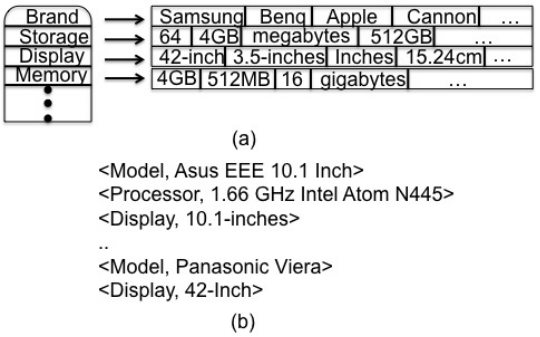

Figure 3: Extractors: (a) bag-of-words model; (b) feature-value pairs

on the target dataset. The tagging consists of taking all n-grams up to 4 and matching them against the previously created lists in order to get a possible property tag for a particular value. Given the tagging a parsing step is performed. Parsing is defined as taking the best combination of feature-value pairs, defined in the tagging step, that best describes an object from the training dataset.

As opposed to the bag-of-words extractor, the feature-value pair extractor is able to efficiently extract multi-token values. Still, both extractors are not able to efficiently extract values that can refer to more than one property. The 3rd extractor uses manually defined **regular expressions** to extract new values, similarly to [3]. The 4th extractor uses **dictionary search** by searching into a collection of words. Each item in the collection must be delimited by one of the conventional delimiters like tab, new line, semicolon, etc.

In our case we manually configured the extractors for 5 properties. The manually configured regular expressions for the 5 properties are listed below:

- Model: `^.*(\w+_[a_zA-Z0-9]+)_\d.*(gb|hd|p[x]|inche?s?|m).*$` - match a token only consisted of letters followed by an alphanumeric token and a token which resembles a number unit measurement pair.
- Storage: `\d+GB` - match an integer number followed by the string GB.
- Processor: `\d*\.?\d+GHz` - match an integer or real number followed by the string GHz.
- Display: `\d*\.?\d+` -inch - match an integer or real number followed by the string inch.
- Brand: dictionary search from list of Brands taken from Amazon.

By using the extractors that require manual configuration the extraction process can be fine-tuned according to the data leading to improved quality of the discovered links. However, configuring the extractors needs domain knowledge and understanding of the input data in order to efficiently pick extraction methods manually.

3.2 Evaluation

Table 2 shows the extraction accuracy of the properties used in the Combination configuration (explained in detail in Section 4.2) for 7 products, one from each category. The evaluation involved running the extraction with the Combination configuration on the reference links set and comparing the result against the Amazon dataset. The Brand property has the highest accuracy, while the Processor property has the least accurate extraction. This is mostly because product descriptions often fail to provide information on some of the properties. For instance, Figure 4(b) shows the extracted structure for iPod Nano 8GB; and while all other properties are correctly extracted, the Processor and Dimension properties could not be found in the product offer (shown in Figure 4(a)).

Table 2: Product/Property extraction accuracy matrix

	Brand	Model	Storage	Disp.	Proc.	Dim.
iPod Nano	.92	.98	.86	.49	.12	.78
Galaxy SII	.72	.87	.89	.81	.40	.91
GalaxyTab7.7	.80	.92	.89	.85	.72	.93
Ixus 120IS	1	.96	N/A	.89	N/A	.56
Vaio VPC	.99	.65	.81	.77	.73	.32
Viera 42	.95	.72	.N/A	.82	N/A	.64
Sansdisk	1	1	.85	N/A	N/A	.31

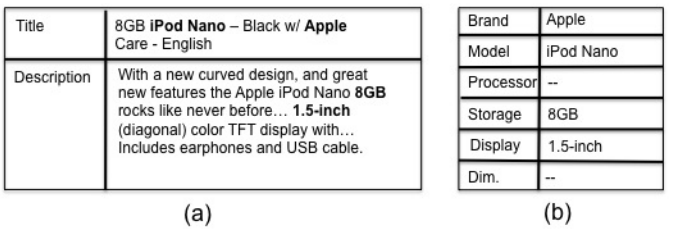

Figure 4: Example of an extracted offer: (a) WDC product offer; (b) Extracted properties

4. IDENTITY RESOLUTION

The problem of finding out which records in one or more data sets refer to the same real word entity has been studied under various labels including record linkage [4,8], duplicate detection [5,9], and identity or entity resolution [3,6,7,10]. All these studies give a palette of approaches for determining which offers refer to the same product. For instance Köpcke et al. [3] approach involves the use of an adaptive learning strategy with 3 string measures and a support vector machine. Another approach was introduced by Kannan et al. [4] where product specifications were classified as matching or non-matching by using logistic regression. In this paper, we use a genetic programing approach introduced in [6] and implemented in Silk Link Discovery Framework.

4.1 Learning Linkage Rules

The genetic programming approach [6] is used to learn a linkage rule that specifies how two data items are to be compared. The approach involves the following steps: A set of rules is used to generate an initial population of random linkage rule. First of all, a linkage rule is built consisting of up to two random comparisons. The comparison is done over a single property or a specific combination of properties. For each comparison a random pair from a pregenerated list of compatible properties is selected. Scores from the comparisons are aggregated in order to get a single score on the possible link between two data items. In

Table 3: Precision, Recall, and F-Measure per Configuration

Configuration	Precision %	Recall %	F-measure %
Baseline	69	90	78.1
Bag-of-words	75	82	77.9
Feature-value Pairs	80	77	78.4
Manually conf. methods	82	80	80.9
Combinations	85	80	82.4

order to efficiently combine comparators for different properties the genetic programming approach relies on different aggregation functions.

The population of linkage rules is breaded and the quality of the linkage rules is assessed by a fitness function relying on user-provided training data. The training data consists of a set of positive reference links (connecting entities identifying the same real world object) and a set of negative reference links (connecting entities identifying different objects). The prediction of the linkage rule is compared with the positive reference links, counting true positives and false negatives and the negative reference links, counting false positives and true negatives.

4.2 Experiment

The proposed methods of extracting product features from free-text and performing identity resolution based on these features are evaluated in 5 different configurations:

1. Baseline - link discovery on free text properties without extracting new properties; the details are given below.
2. Bag-of-words - link discovery with the bag-of-words model for property extraction.
3. Feature-value pair - link discovery with feature-value pairs model for property extraction.
4. Manually configured methods - link discovery with custom regular expression and dictionary search for particular property.
5. Combination - link discovery with manually defined combination of extractors.

The baseline involves pairwise matching using just the title and description from both WDC and Amazon datasets. In order to get a more precise comparison we extract patterns from the title and description. From the title we extract the product brand and model by matching a regular expression: `^.*(\w+_[a_zA-Z0-9]+)_\d.*(gb|hd|p[x]|inche?s?|m).*$`. From the description we find a number/unit of measurement pairs, which usually correspond to numeric attributes like 5 m, 3.5 inches, 256 MB etc.

The second configuration involved using the bag-of-words extractor on every property in the training set. The goal is to test which properties are efficiently extracted by the bag-of-words extractor and if that helps the link discovery process. The third configuration involves using the feature-value pair extractor on every property in the training set. Analogously to the second configuration, the main goal is to test which properties are efficiently extracted and if that helps the link discovery process.

After testing the above approaches separately, combining extractors is a natural step forward. We assigned the bag-of-words extractor to be used for Brand, Storage and Display properties, since these properties are usually composed of one token. The feature-value pair extractor was used for extracting the Model and Dimension properties, because these properties are usually multi-token. Finally, we assigned a custom regular expression for the extraction of the Processor property, as we wanted to limit the extraction to the frequency value of the processor. The Brand and Model properties were searched in the title, while all the others were searched in the description of the product offer.

The experiment involves running the 5 different configurations for a 5,000 manually annotated reference links (2,500 positive/2,500 negative) a subset of reference links from the datasets described above. The results from the experiment are shown in Table 3.

Table 3 shows that the baseline configuration fails to produce high quality links (precision is 69%). An analysis of the links produced by the baseline, showed that failures can be explained by the fact that there is too much residual noise. Moreover, a large number of negative reference links were reported as positive, which also contributed to the low precision. On the other hand, the high recall can be explained by the fact that the model defined in the baseline correctly identifies the positive reference links. Namely, only 250 positive reference links were reported as negative.

Two conclusions can be drawn from the results presented in Table 3. Firstly, the quality of the discovered links improves when data is structured. Furthermore, the quality is the highest when using Combination configuration, i.e. when extractors are chosen in accordance to the properties. Secondly, we get lower recall with all our approaches with respect to the baseline. This can mostly be assigned to two factors. On one hand, information loss in terms of property values occurs during extraction, leading to a loss in a small portion of positive reference links. On the other hand as we strive to get better precision, the fitted model labels some positive reference links as outliers.

Figure 5 shows the number of the discovered offers per product when the Combination configuration is applied on the whole WDC electronics product dataset (219,118 product offers).

The results accommodate to a power law, i.e. there is a small number of products that dominate the offers found on the web, and a long tail. The product with most offers is iPod Nano 8GB with 829 offers. The product with least offers is the SeaGate FreeAgent 1TB with 19 offers. The total number of discovered offers for the 20 products is 3,610. The rest of the product offers in the WDC dataset describe other products.

5. DATA FUSION

As described in the previous Section, we were able to discover links between Amazon and WDC product data sets with high precision. The discovered links give us clusters of product offers, in which the same product is described by the data coming from different sources. Some of the sources also provide product ratings and reviews. Therefore, for each product we can fuse the rating and combine the reviews from the different sources. The results of the data fusion are presented in this section.

Reviews in the WDC dataset are usually represented by two main classes *schema:Review* and *data-voc:Review*; they are linked to a product offer through the schema:Product/re-

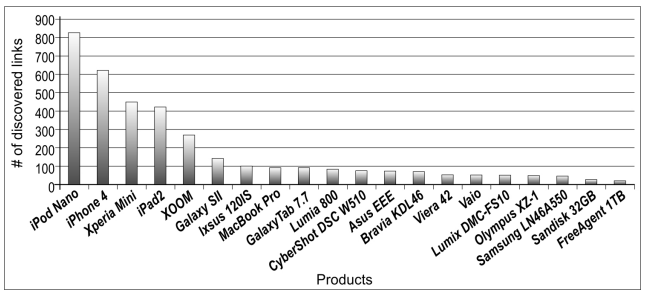

Figure 5: Discovered offers per product in the WDC electronics product dataset

Table 4: Number of reviews and ratings per product

Product	Offers	Reviews	Ratings
Apple iPod Nano 8GB	829	84	0
Apple iPhone 4 16GB	624	35	52
Sony Ericsson Xperia Mini	450	31	12
Apple iPad2 16GB	423	40	48
Motorola XOOM 32GB	270	12	0
Samsung Galaxy SII	142	8	0

view and data-voc:Product/review properties. The total number of reviews found in the WDC electronics product dataset is 41,539. The ratings in the WDC dataset are usually represented by two classes: schema:AggregatedRating and data-voc:AggregatedRating. The ratings are linked to the product offers trough the schema:Product/aggregatedRatings and data-voc:Product/ratingValue properties. The total number of ratings found in the WDC electronics product dataset is 60,126.

We experimented with the reviews and ratings properties for the 6 products that have more than 100 offers in the WDC dataset. The number of reviews and ratings per product is shown in Table 4. On average ~6% of the product offers are linked to at least one review. Surprisingly, the number of ratings was lower than the reviews even though the ratings in general outnumber reviews. This can be explained by the fact that offers for the given products mainly came from e-shop aggregators that take ratings from the product offer source, rather than providing them themselves. On average only ~1% of the product offers contain ratings.

Despite having low number of reviews and ratings there is a big benefit in fusing these properties. Namely, because most of the products have no reviews, fusing the review property allows all products to share user reviews. For instance, from 829 iPod Nano 8GB offers 745 offers do not have a single review, but all will have 84 reviews in the fused dataset. This is also true for fusing the ratings since the aggregate Ratings property has a very low occurrence.

6. CONCLUSIONS

This paper presents a methodology covering the process of integration of products marked up with Microdata. A five step process of Microdata extraction, product classification, product feature extraction, identity resolution, and data fusion is proposed as an efficient method for identifying data items. The study indicates that link quality strictly depends on the preprocessing step of product feature extraction extraction. Namely, new item properties are best extracted when extractors are chosen in accordance to the properties. A keep all strategy was used when fusing data clusters gained from the identity resolution task.

Even though the domain of this paper is product offers, a generalization to other data items (persons, places, etc.) is possible. To improve the approach further research in automatically learning the extraction combinations by employing supervised learning methods can be implemented.

7. ACKNOWLEDGMENTS

The work presented in the paper is supported by the EU FP7 project *LOD2 – Creating Knowledge out of Interlinked Data*[11] (Ref. No. 257943).

8. REFERENCES

[1] I. Hickson. Html Microdata. `http://www.w3.org/TR/microdata/`, 2011.

[2] C. Bizer, K. Eckert, R. Meusel, H. Mühleisen, M. Schuhmacher, and J. Völker. Deployment of RDFa Microdata and Microformats on the Web A Quantitative Analysis. In *12th International Semantic Web Conference In-Use track*, pages 17–32, 2013.

[3] H. Köpcke, A. Thor, and E. Rahm. Evaluation of entity resolution approaches on real-world match problems. In *Proc. 36th Intl. Conference on VLDB / Proceedings of the VLDB Endowment 3(1)*, pages 484–493, 2010.

[4] A. Kannan, I. Givoni, R. Agrawal, and A. Fuxman. Matching unstructured offers to structured product descriptions. In *International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 404–412, 2011.

[5] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios. Duplicate record detection: A survey. *IEEE Trans. on Knowl. and Data Eng.*, pages 1–16, 2007.

[6] R. Isele and C. Bizer. Learning Linkage Rules using Genetic Programming. In *6th International Workshop on Ontology Matching*, pages 1638–1649, 2011.

[7] H. Köpcke, A. Thor, and E. Rahm. Tailoring entity resolution for matching product offers. In *Proc. 15th Intl. Conference on Extending Database Technology (EDBT)*, pages 545–550, 2012.

[8] W. E. Winkler. Overview of record linkage and current research directions. Technical report, Bureau of the Census, 2006.

[9] S. Sarawagi and A. Bhamidipaty. Interactive deduplication using active learning. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 269–278, 2002.

[10] O. Benjelloun, H. Garcia-Molina, D. Menestrina, Q. Su, S. Whang Euijong, and J. Widom. Swoosh: A generic approach to entity resolution. *The VLDB Journal*, 18(1):255–276, 2009.

[11] `http://lod2.eu/`