Using Text Reviews for Product Entity Completion

Mrinmaya Sachan Tanveer Faruquie L. V. Subramaniam Mukesh K. Mohania
IBM Research India
New Delhi, India
{mrsachan, ftanveer, lvsubram, mkmukesh}@in.ibm.com

Abstract

In this paper we address the problem of obtaining structured information about products in the form of attribute-value pairs by leveraging a combination of enterprise internal product descriptions and external data. Product descriptions are short text strings used internally within enterprises to describe a product. These strings usually comprise of the Brand name, name of the product, and its attributes like size, color, etc. Existing product data quality solutions provide us the capability to standardize and segment these descriptions into their composing attributes using domain specific rulesets. We provide techniques that can leverage the supervision provided by these existing rulesets for extracting missing values from other external text data sources accurately. We use a large real life data collection to demonstrate the effectiveness of our approach.

1 Introduction

Enterprises usually store information of its products in the form of unstructured text strings. Such product descriptions contain the name of the product and its specific attributes. These product descriptions are usually written by multiple people and could contain overlapping information or even the same information written differently. For example, a superstore may source the same product from many different vendors and each vendor may give varying descriptions of the same product. Information could be scattered through various departments and held by certain employees or systems instead of being available centrally. This results into varying standards and vocabulary.

Consider the following product descriptions obtained from an enterprise selling cameras. They are provided by different vendors supplying the cameras to the enterprise:
Nikon D90 4288×2848 703 g Digicam F/1.8
Nikon Digital 90 Cam 12.3MP 1.8 F-Len(1.55lb)
Nikon D-90 Camera with Nikkor 50mm 1.8D
Expert knowledge specific to the domain (that D90, D-90 and Digital 90, 4288 × 2848 and 12.3 MP, F-number 1.8 and 1.8 D focal length, and 703 g and 1.55lbs are same) is required to conclude that the entries above are the same product. Coupled with data entry errors, the problem of identifying a standard representation of the product becomes even harder.

The problem of obtaining a structured representation of such product descriptions is similar to the 'Attribute-Value pair' mining problem. Attribute represents an aspect of the product. It could be anything from a manufacturing detail like model number to information like color, size and weight.

Due to its practical applications, the problem has drawn interest from the research community as well as the industry. There are many products which provide solutions for standardizing, matching, merging and validating such descriptions. Popular ones include Oracle middleware, Silvercreek, Ethoscontent product-copy, Trilliumsoftware and IBM Data Stage-Quality Stage. Since rules are easy to understand, manage and give good accuracies in practice, they are widely used by these solutions. Overall, the product data cleansing solution is achieved by a collection of rule sets, each tackling a given product vertical. A ruleset scales to descriptions within its vertical.

Often, the product descriptions are very brief and do not convey the entire information about the product. Critical information about the product could be missing. Because of this, an enterprise does not have a complete view of its products and services. Suppose an enterprise wants to have manufacturer wise information about its products for making important demand-supply decisions. If

Proceedings of the 5th International Joint Conference on Natural Language Processing, pages 983–991,
Chiang Mai, Thailand, November 8 – 13, 2011.

the manufacturer information is erroneously captured or missing, it wouldn't be possible. This lack of information makes it difficult for a potential customer to compare products and make an informed decision about it. At the same time, low data quality impedes business. An enterprise may loose its competitive edge due to poor customer service and advertising resulting from incorrect reporting and incomplete view of its products and services. Having a standard and complete structured representation of a product can help in consolidation and is useful in various business intelligence applications. Given the purchasing history of its customers, a better view of enterprise products would result into a host of benefits like better targeted advertising, better recommender systems and reduced maintenance effort.

At the same time, as more and more people are beginning to write reviews, blogs and opinions about their experiences in using products, it is possible to obtain a lot of information about the products on the web. Popular merchant sites like Ebay and Amazon contain a large number of product reviews from its customers. These reviews not only contain reviewer sentiments but also contain key information which can be used to create an enriched view of the products. Our goal is to obtain a complete view of the products by extracting attribute values from such sources.

Here we note that existing rulesets used by data cleansing solutions can give us critically important supervision to drive the attribute-value extraction. This would not only help us in achieving greater accuracy but also allow us to extract true product values. Infact, a key differentiator that puts this work apart from its peers is that the supervision provided by the rules allows us to extract attribute 'values' in the real sense and not merely sentiment words as in most previous works.

Overall, the task of creating a complete view of the products comprises of standardization of appropriate features present in the product descriptions along with enrichment using web data. We carry out product description segmentation by writing handcrafted rules on top of a domain specific dictionary. Then, we show that the same rules can be reused on web data to fill-in missing values for many of the product attributes. Our paper is organized as follows. Section 2 gives us the necessary background and describes related work. Then, Section 3 describes our approach in detail. In Section 4, we report experimental results on real datasets. Finally, Section 5 concludes.

2 Related Work and Background

There has been significant work in information extraction from text data for products. In particular the extraction of product attributes and user sentiments has received wide attention. One of the methods (Hu and Liu, 2004) is to use frequent item sets of nouns along with the opinion words to mine infrequent product attributes. This method is further improved (Zhuang et al., 2006) by using domain knowledge along with noun phrases for attribute extraction. Another refinement (Qiu et al., 2009) uses extraction rules based on different relations existing between opinion words and attribute words. These relations are syntactic and are propagated in an iterative manner.

Some approaches detect product attributes along with opinion extraction. (Liu et al., 2005) first detects attributes by using a rule miner to find noun phrases. Further, it finds polarity descriptors for these noun phrases. (Popescu and Etzioni, 2005) computes the point wise mutual information between noun phrases and product class specific discriminators to determine whether a noun phrase is a product attribute. It finds part-whole patterns by querying the web and uses a part-whole pattern for attribute mining. It further finds the sentiments of these attributes. In contrast, our work finds actual values for the attributes and not merely sentiments expressed by users.

An approach to finding attributes and sentiments jointly is to mine patterns of aspects-evaluation (Kobayashi et al., 2007) using statistical and contextual cues. Here aspects are attributes for a particular product and evaluations are the opinions expressed. The significance of discovered patterns are computed based on their statistical strength. (Wang and Wang, 2008) uses iterative boot strapping to find opinion words from attributes and then finding attributes from opinion words in an alternating fashion. It uses mutual information to measure association between them and linguistic rules to identify infrequent attributes and opinion words. In one of the most interesting works of its kind, (Zhai et al., 2010) groups domain synonyms to form feature groups using a naive Bayesian EM formulation iteratively on the labeled and unlabeled data. It leads to each unlabeled example being assigned a posterior proba-

Table 1: Sample Segmented Product Descriptions

Description	BrandName	ProductName	Product	Lens	Resolution	Price
Fujifilm Quick Snap Single Use Camera	Fujifilm	Quick Snap	Camera	NULL	NULL	NULL
$25 cannon EF f/2.8 USM Lens	Canon	EF	Lens	f/2.8 USM	NULL	$ 25
Sony DSCP 8 3.2 MP Camra	Sony	DSCP 8	Camera	NULL	3.2 MP	NULL

bility of belonging to a group giving it the ability to find multigram attribute terms. However, these methods concentrate on detecting product attributes and do not find associated values.

Among machine learning approaches CRF (Peng and McCallum, 2006) (Stoyanov and Cardie, 2008) has been effective if we have training data and want to extract template attributes and values. However, we do not always have a predefined list of explicit attributes and values which have to be extracted and populated. The method in (Ghani et al., 2006) finds attributes and values together using a naive bayes algorithm combined with EM. This method requires labeled training data which is often difficult and expensive to produce. Also, since both the attribute and value extraction is automatic, the accuracy is low and hence, not useful for most business applications. Our method differs from it as we propose an automatic approach to detect potential attributes and a combination of rules and semi-supervised learning to extract values. Our method also has the flexibility to detect values of pre-defined attributes.

3 Our Approach

We pose the problem of Product Entity Completion as two sub-problems: Product description segmentation and Enrichment. The segmentation problem simulates the product data standardization done in the industry. It constructs a standard attribute value representation using product descriptions found in the enterprise database. A set of attributes $\{A_1 \ldots A_n\}$ is decided by applying some initial domain knowledge. We are given a set of product descriptions comprising of values for one or more attributes. Our task is to segment these descriptions and put each segment into one of the n attribute bins. Since the descriptions are incomplete, large number of attribute values are null after the segmentation step. Enrichment task leverages the supervision provided by the rules to extract unknown or missing values from external web data.

3.1 Product Description Segmentation

Enterprise descriptions of products are short strings containing information about one or more attributes like “Brand Name”, “Product Name”, “Model Number”, “Manufacturer” and few other product specific attributes. To attain a standard view of products and to conform to organization-wide specifications, product data cleansing solutions are used to segment the descriptions into these product attributes. Non-standard representations are converted to standard forms and misspellings, etc are corrected. Some typical examples of product descriptions and their corresponding standardized forms are shown in Table 1.

To begin with, the enterprise or a domain expert ascertains the attributes comprising the descriptions. To perform the task of moving free-form descriptions into these pre-determined fixed attribute columns, a dictionary classification for generic tokens like involved brand names and product names is maintained from various intellectual property organizations like UNSPSC[1] and WIPO[2], and common metric units and currency symbols from common knowledge. This dictionary of standards is often stored in the form of rich taxonomies and is fixed. Each token is assigned a classification symbol depending on its type. To account for all misspellings (“Camera”, “Camcorder” and “Camrecorder”), difference in vocabularies(“Oz” and “Ounces”), classifications, synonyms and abbreviations, and other non-standard representations, the native forms(like Camrecorder) are mapped to a standard form (Camcorder) using signature clustering techniques as described in (Prasad et al., 2011). This is conveniently achieved by popular string similarity measures and by looking at context of these native forms in the input descriptions. Such data driven context mining approaches to find various ways in which similar words (which often have the same classification entry) like “Camera”, “Camcorder” and “Camrecorder”, and “LCD”, “CFD” and “TFT” help reduce the manual effort in rule writing and dictio-

[1] http://www.unspsc.org/

[2] http://www.wipo.int/portal/index.html.en

nary building. Despite this, suitable additions to these dictionaries are sometimes needed from domain experts. Example classification for our running example is given in Table 2.

Table 2: Classification Entries

NativeForm	StandardForm	Classification	Symbol
Canon	CANON	Brand	B
Cannon	CANON	Brand	B
Sony	SONY	Brand	B
Fujifilm	FUJIFILM	Brand	B
Quick Snap	QUICK SNAP	Product Name	N
Camera	CAMERA	Product	P
Camra	CAMERA	Product	P
USM	USM	Lens Property	L
$	$	Currency	C
MP	MEGAPIXEL	Metric	M
F	F	Alphabet	A

Also, we generate symbols for each number or unknown word to help us make use of the context to write rules. The classifications lead to a pattern of symbols for every product description entry.

For the following product description,
"Canon Powershot SD1200IS 10 MPIXEL Digital Cam $f/3.5 - 6.3\ 8''$ LCD Screen $123"
the corresponding pattern would be:
"$B + @\#M + PA/\# - \#\#M + +C\#$"
Here, B is a brand name and P is a product name recognized from one of the catalogues lying with the enterprise or some intellectual property database. M is a metric unit and C is a currency symbol lying in the dictionary of metric standards and currencies, respectively. Other symbols include $+$ for an unknown word, $\#$ for a number and $@$ for an alphanumeric. Finally, a domain expert writes rules to move entries into appropriate database columns and complete the standardization. Rules are written to process important subpatterns from the left or the right and capture attribute values in one pass. Table 3 lists some hand crafted segmentation rules to capture Resolution, Focal Length and Price. People invest a great amount of time, effort and money in building and maintaining these rules for extracting information from product descriptions. The output of these rules are used to populate Data Warehouses and Product Information Management (PIM) systems. However, these rules are applied only to short product descriptions which do not lead to a complete view of the product. In the subsequent step, we provides techniques that employ these rules to discover missing values of existing attributes. Doing so reuses the time and effort spent in building the rules on external content which otherwise is very expensive and time consuming to build.

Table 3: Rules for Segmentation Process

Subpattern	Rule[a]
# | M="MP"	COPY [1][2] "Resolution";
A="F" | / | # | - | #	COPY [1][2][3][4][5] "FocalLength";
C | #	COPY [1] "Currency"; COPY [2] "Price";

[a] [N] represents the N^{th} token in the subpattern
"Resolution", "FocalLength" and "Price" are Attribute columns
| represents a token separator

3.2 Missing Value Filling

We observe that the product descriptions after standardization have missing values for many predefined attributes (Nambiar et al., 2011). Next, to obtaining a complete view of the product we extract these values from the product reviews available on the web. Our approach is general and can be extended to other data sources like Sales and Marketing data, Product Catalogs, Website Product Listings and so on. Unlike many previous attempts, we extract meaningful 'values' of interesting attributes such as the shape, size and manufacturer's name and not merely use sentiments. We make use of the supervision provided by already existing rulesets frequently used for standardization to get a handle of such values. However as the reviews are verbose and noisy, these values too contain a lot of noise. Hence some text processing heuristics are used to choose most appropriate values.

However, we should note that the rules are meant for short product descriptions and not the user reviews. Due to high degree of verbosity and possibility of competing product talk in the reviews, direct rule application on the reviews yields many candidate values for each attribute. Hence, we devise a strategy to prune inappropriate candidates. Our approach to prune out unimportant candidates assesses the affinity of all the candidate values with their corresponding product descriptions and calculates a confidence level for each of them. Confidence level intuitively measures the likelihood that the candidate is a true value of the attribute and product in question. For each product and each of its attributes, the enterprise can then choose the value with the highest confidence (if its confidence is above a certain threshold).

Table 4: Sample Standardized Product Descriptions

Brand Name	Product Name	Product	Focal Length	Lens Diameter	Price	Weight
Canon	EF	Lens	2.8D	70-200mm	-	-

Table 5: Sample Product Reviews

... I bought a *Canon 2.8D lens*...certainly worth each of the **1369 bucks**.....**2.9 pounds** is a bit heavier...
... new *Nikon AF f/3.5-5.6G* ... fair price deal of **$ 685** ...

We compute the affinity for a candidate by looking at the appearances of known attribute values(values obtained by segmentation of the product description strings) of the product in its context. The 'known attribute values' are assigned certain weights and the confidence score for the candidate is computed using the weights carried by the 'known attribute values' in its context. Assuming all weights to be equal, the following example explains the idea in detail:
Consider the product description "Canon EF 70-200mm f/2.8D Lens" and its corresponding segmented output in table 4. Here - represents the missing values to be filled using the reviews. Consider the two reviews given in table 5. Out of the candidates "1369 bucks " and "$ 685" for the price attribute, "1369 bucks" is chosen as it contains more known attribute values (Canon, 2.8D, lens) in its context in review 1. On the other hand as "$ 685" has many competing attribute values (which do not match the known values) in its context (Nikon, AF, f/3.5-5.6G), it is rejected. Similarly, "2.9 pounds" being the only candidate for the weight attribute is accepted due to the presence of many matching known values in its context.

In the above example, we simply counted the number of known attribute values in each candidate's context to compute its confidence score. All the known values carried equal weights.

However, certain values occur much more rarely than others in free text. Hence, matching a rarer value in a candidate's context should give us more confidence that the review author is talking about the same product (as in the product description string) than one which occurs more frequently. For example, matching a value Focal length '18-55 mm' (which occurs rarely) instills more confidence than if the value color 'black' (a value which should occurs very frequently) is matched. We quantify this idea by assigning an 'importance' measure to all known values. The importance $I(v)$ of a known value v can be decided by the proportion of distinct product entities in the database(output of the standardization phase) that contain the value v. Since we wish to weigh rarer values more as they signify more importance, we use the inverse document frequency(IDF) formulation popularly used in previous text mining works to define:

$$I(v) = \begin{cases} log(\frac{N}{n(v)}) & \text{if n(v) } >0 \\ 0 & \text{otherwise} \end{cases}$$

where, N is the total number of distinct entities for the attribute in the database, and $n(v)$ is the number of distinct product entities that contain the value v.

We also note that matching the value for a certain attribute can signify greater confidence than others. For instance, if the "Product:Camera" or "Brand:Nikon" matches we still cannot be very sure because the author can be comparing two different cameras or two Nikon products. However if a value mentioning weight or lens of the camera matches, we can be more certain as it is unlikely to have two cameras with exactly the same weight or the same lens.

With this intuition in mind, we also give different weights to each 'attribute' in the standardization schema for matching. These weights intuitively signify our confidence on a value of the given attribute towards matching. In our running example, intuitively we should give more weight to 'Weight' or 'Lens' attributes than 'Product' and'Brand'. Weight of the attributes can be expressed by the domain expert during the schema decision process. Please note that this schema decision and weight assignment needs to be done only once per product vertical. For the matching to be effective, all real numbers were morphed to a common representation '#' and misspellings were corrected for terms known in the dictionary.

Finally, these weights and importance measures are tied together to estimate the confidence for each candidate value. We introduce a distance metric to quantify the distance between two words in a review. The effect of a value on the confi-

dence of a candidate dies off with its distance from the candidate. A formal description of the idea is given below.

Given: Attribute schema A=$\{A_1 \dots A_N\}$, set of products P and set of reviews R_p for each $p \in P$,

$$R = \bigcup_{p \in P} R_p$$

Let $A_i(p)$ be the value of attribute A_i for the product $p \in P$ from the segmentation step. Let $I_p(A_i, r)$ be the set of index positions at which $A_i(p)$ occurs in the review $r \in R_p$. Define:

$$B_p(A_i) = \begin{cases} 1 & \text{if } A_i(p) \text{ is known} \\ 0 & \text{otherwise} \end{cases}$$

Let attribute value A_j be missing for some product p after the segmentation step i.e. $B_p(A_j) = 0$ for some $j \in \{1 \dots n\}$. Given a set of candidates $C_p(A_j, r)$ (obtained by applying rules on a product review $r \in R_p$) for attribute A_j and product p, we define:

$$Q_p(c, r) = \sum_{x=1}^{n} \left(B_p\left(A_x\right) W(A_x) \sum_{i \in I_p(A_x, r)} I(i) d_r(i, c) \right)$$

$$\forall c \in C_p(A_j, r)$$

where d_r is a distance metric defined over every pair of words in a given review r.

In similar lines to the idea presented so far, occurrence of a value that contests a value already known from the standardization stage(segmentation output of the product description) can be used as an indicator that the author is talking of some other product or attribute. Hence, we have a case to reject candidates in its vicinity. Also, often people compare two products or brands while writing a review. Use of comparative adjective forms or coordinating conjunctions like "but, whereas, while, although, etc." that express a contrast mark such cases. To incorporate both of the above ideas, we can easily extend the objective function Q_p to account for these contingencies by adding terms to the summation that reduce the score of a candidate if conflicting attribute values and active comparison is found in its vicinity. Consider the following review:

I love the Point and Shoot mode in my new camera X. I was bored of using the same auto mode in my old cam Y. Though it lacks the auto-program feature, still it's worth its price in gold. While auto mode in Camera Y was a sham, night mode was a cool addition. A f/1.8 lens , preferably brand Z would just be the icing on the cake.

Here the author compares his old camera with the one he just purchased and finally talks about buying a different product (a lens). A naive extraction scheme will extract features for each of cameras X and Y, and lens Z. The problem could be further compounded if the author swings back and forth comparing two products leading us to the deep waters of Pronoun Resolution and Attribute Coreference Resolution. However, we easily find a way around them with the assumption that the switch will not happen too frequently in most reviews.

Finally, as the descriptions are often sparse, we do not have all representative values for an attribute. A better 'importance' measure for a value ($I(v)$) can be obtained from the reviews. Also, in the above description, Q_p values can be arbitrarily large or small. Hence, we translate this idea in the probabilistic sense to a scale between 0 and 1. In an informal manner, the affinity of each candidate with the product can be treated as a confidence measure (represented by $P(c)$ instead of discrete Q_p values) in a scale of 0 to 1.

We also give a linear time algorithm based on the above idea. The algorithm iterates over all the candidates, modifying their affinity and dispersing the change to all other candidates in the review.

Given:- A set of products P and a set of reviews R_p for every product $p \in P$. Let $C_{p,r}$ be the set of candidates for product p in review r. Let A be the set of attributes. $W_{sim}(A_k)$ and $W_{diff}(A_k)$ are the weights of the attribute A_k for a matching value and a competing value, respectively. Weight W_{comp} penalizes a comparative sentence in the affinity calculation. $d_r(c, c')$ is some distance metric defined on all candidate pairs in a review r. We set the distance between two candidates as the number of words between their occurrence in the review in our simulations.

for each $p \in P$:
  for each $r \in R_p$:
    for $c \in C_{p,r}$:
      Compute: $I(c)$
      Initialize: $P(c) = \frac{1}{2}$
      Init_value=P(c)
      for $k \in 1 \dots n$:
        if $c = A_k(p)$:
          $P(c) := W_{sim}(A_k) * P(c) + (1 - W_{sim}(A_k))$

end if
else if $c = A_k(p'), p' \in P, p'! = p$:
$P(c) := W_{diff}(A(k)) * P(c)$
end elseif
end for
if c lies in a comparative sentence:
$P(c) := W_{comp} * P(c)$
end if
Change(c)=$P(c)$-Init_value
for $c' \in C_{p,r}, c'! = c$:
$P(c) += Change(c') \times I(c') \times e^{-d_r(c',c)}$
end for
Normalize P
end for
end for
end for

Selection:- Finally for each product p and attribute A_j that misses a value for p, we choose the candidate with maximum affinity($P(c)$) i.e. choose c^*_{p,A_j} such that,

$$c^*_{p,A_j} = \underset{c' \in C_p(A_j,r), r \in R_p}{\text{argmax}} P(c', r).$$

Finally, c^*_{p,A_j} is filled in as a value for A_j if $P(c^*_{p,A_j}, r))$(r is the review containing c_{p,A_j}) is above a certain threshold. The algorithm runs with a complexity of $O(N + C^2)$ where N is the size of the reviews(number of words) and C is the number of candidates generated by applying rules on them. Since C is generally small, this is effectively linear with respect to size.

4 Evaluation

4.1 Dataset

The algorithms were tested on a real life dataset crawled from 'Amazon'. It contains reviews and short descriptions of 996 products in the Camera and Accessories space (Cameras, lenses, filters, etc). The total number of reviews was 23,337 leading to reviews per product ratio of about 24. Average number of words per review was around 40. Each product has a minimum of 20 reviews.

4.2 Experimental Setup

4.2.1 Product Description Segmentation

The experiments were carried out by first identifying the appropriate schema and 12 attributes were identified for description segmentation. They are given in Table 6. Rules (as described in Section 4.1) were used to standardize the descriptions into the composing attributes. The dictionary augmentation and rule writing together took nine man-hours. This is much lower than usual since we used Intellectual Property datasets on Brands and Products and clustering used to detect misspellings and varying vocabulary.

Table 6: Attributes guessed by the Rule Writer

Attribute	**Value**		**Attribute**	**Value**
Brand	Nikon		Product Type	Digital
Zoom	True		Color	Black
Lens	F/2.8 D		Retail Price	$ 250
Model	D 90		Product	Camera
Filter	UV		Size	Compact
Resolution	12MP		Features	Point&Shoot

4.2.2 Missing Value Filling

It was observed that around 60% of the attributes after the segmentation stage were null. So we used the reviews to fill in these missing values. Reusing the rules on the reviews led to a whopping 8434 candidates for the 12 attribute places in 969 products, which on using the confidence score based pruning scheme reduced to 5321. We set 0.5 as the weights for Brand Name, Product Name, Product Type and Color; and 1 for the remaining attributes. Recall that this is in-line with our arguments that there can be many products with the same Brand name, Product name, type and color but it is less likely for two products to have the same value for other attributes (say Retail Price or Weight). This choice of weights is done once and can be done at the time of the initial schema selection. Finally to select or reject candidates, we use a threshold. We used 0.4 times the mean(of the candidate confidence values) as the threshold for selection in our experiments. The threshold can be chosen based on the general confidence that the enterprise has on the correctness of the reviews. As the threshold is decreased, the number of values filled in increases but the precision-recall (and thereby F-Score) decreases.

4.3 Results

To evaluate our work, we also compute the entire attribute values view manually. Using this as ground truth, we calculate the Precision, Recall and F-Score of the overlap it has with our proposed techniques. Sometimes the values extracted are only partially correct for example, if the measurement unit is missing for the weight attribute.

Table 7: Evaluation Metrics for each Stage

Evaluation Stages	**100 Cameras**			**100 Lenses**		
Partially Correct	**Precision**	**Recall**	**F-Score**	**Precision**	**Recall**	**F-Score**
1.Standardization	1.000	0.909	0.952	1.000	0.925	0.961
2.Missing Value Extraction	0.888	0.705	0.786	0.837	0.700	0.762
Baseline 1	0.728	0.682	0.704	0.706	0.676	0.691
Baseline 2	0.771	0.709	0.739	0.741	0.703	0.712
Completely Correct	**Precision**	**Recall**	**F-Score**	**Precision**	**Recall**	**F-Score**
1.Standardization	0.998	0.901	0.947	1.000	0.917	0.957
2.Missing Value Filling	0.773	0.613	0.683	0.721	0.632	0.674
Baseline 1	0.644	0.607	0.625	0.618	0.596	0.607
Baseline 2	0.613	0.534	0.571	0.578	0.527	0.551

Hence, we evaluate results for both cases (when the values match perfectly or only partially).

4.3.1 Product Description Segmentation

As the Standardization stage is rule based, with time, close to perfect precision and recall can be achieved. In nine man-hours of effort, we achieved very high precision and recall as shown in Table 7.

4.3.2 Missing Value Filling

In Table 7, we give the precision and recall for the Enrichment phase. Here, we also draw two baseline comparisons for our work.

Baseline 1 is drawn by using the values extracted by the standardization stage from a training set (remaining 886 cameras) as seeds to train a CRF. The value extraction task is treated as a multi-class classification problem where each word is to be classified as a value to one of the attributes ($A_1 \dots A_n$) or as a "not a value" class. To generate a training set, every word is represented as a feature vector of $n + 20$ features. First n features are boolean entries and represent if the word matches an already known attribute value of the product. If the word matches with a value for A_j, then j^{th} feature is set to true and the rest to false. 10 words in the context of that word and their POS tags are remaining 20 features. The seeds are used to assign class labels to the training set. State of the art CRF is used for value extraction in a 10-fold setting. Please note that drawing the seeds from the standardization stage (which was rule based) gives this baseline the supervision provided by the rules. This is done to make a comparison with our approach fair. Recall that our technique for missing value Assignment leverages the supervision of existing rules to come up with meaningful values for the attributes.

(Ghani et al., 2006) laid down an ingenious technique to automatically extract candidate attribute-value seeds. They considered all pair of consecutive words (w_i, w_{i+1}) where w_i is a candidate value for the attribute w_{i+1}. Next, they computed the mutual information between all such candidate attribute-value pairs to prune down to fewer but cleaner seeds. Baseline 2 generates seeds by their technique and filters them for the attributes $\{A_1 \dots A_n\}$. Again, CRF is trained with the same feature space.

It can further be observed that due to rule based cues, our techniques do well even when completely correct values are expected. However, Baseline 2 (projection of (Ghani et al., 2006) on the attribute set) falls apart as most extracted values are incomplete and partial.

The entire experiment is also carried out on a similar dataset of 100 lenses to prove the generalization ability of our technique within the product domain(Photography). Results on the lens dataset (shown in Table 7) are very close to the camera dataset, and occasionally better.

5 Conclusion

In this work, we tackle the problem of creating the complete view of an enterprise's products and services. We utilize the rulesets developed by existing product data cleansing solutions for value extraction from unstructured text media. Hereby, we escaped the laborious process of writing annotators. Supervision provided by the rules helped us uncover values which can give us a better view of the product and not merely sentiments.

References

Rayid Ghani, Katharina Probst, Yan Liu, Marko Krema, and Andrew Fano. 2006. Text mining for product attribute extraction. *SIGKDD Explorations Newsletter*, 8(1):41–48.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the International Conference on Knowledge discovery and data mining*, pages 168–177.

Nozomi Kobayashi, Kentaro Inui, and Yuji Matsumoto. 2007. Extracting aspect-evaluation and aspect-of relations in opinion mining. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

Bing Liu, Minqing Hu, and Junsheng Cheng. 2005. Opinion observer: analyzing and comparing opinions on the web. In *Proceedings of the International Conference on World Wide Web*, pages 342–351.

U. Nambiar, A. Faruquie, K. H. Prasad, L. V. Subramaniam, and M. K. Mohania. 2011. Data augmentation as a service for single view creation. In *Proceedings of IEEE International Conference on Services Computing (SCC)*.

Fuchun Peng and Andrew McCallum. 2006. Information extraction from research papers using conditional random fields. *Information Processing Management*, 42(4):963–979.

Ana-Maria Popescu and Oren Etzioni. 2005. Extracting product features and opinions from reviews. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 339–346.

K. H. Prasad, T.A. Faruquie, S. Joshi, S. Chaturvedi, L. V. Subramaniam, and M. K. Mohania. 2011. Data cleansing techniques for large enterprise datasets. In *Proceedings of SRII Global Conference (SRII)*.

Guang Qiu, Bing Liu, Jiajun Bu, and Chun Chen. 2009. Expanding domain sentiment lexicon through double propagation. In *Proceedings of the International joint Conference on Artifical intelligence*, pages 1199–1204.

Veselin Stoyanov and Claire Cardie. 2008. Topic identification for fine-grained opinion analysis. In *Proceedings of International Conference on Computational Linguistics*, pages 817–824, Morristown, NJ, USA.

B. Wang and H. Wang. 2008. Bootstrapping Both Product Features and Opinion Words from Chinese Customer Reviews with Cross-Inducing. *Proceedings of International Joint Conference on Natural Language Processing*.

Zhongwu Zhai, Bing Liu, Hua Xu, and Peifa Jia. 2010. Grouping product features using semi-supervised learning with soft-constraints.

Li Zhuang, Feng Jing, and Xiao-Yan Zhu. 2006. Movie review mining and summarization. In *Proceedings of the International Conference on Information and knowledge management*, pages 43–50.