Modeling the Evolution of Product Entities

Priya Radhakrishnan
IIIT, Hyderabad, India
priya.r@research.iiit.ac.in

Manish Gupta[*]
Microsoft, Hyderabad, India
gmanish@microsoft.com

Vasudeva Varma
IIIT, Hyderabad, India
vv@iiit.ac.in

ABSTRACT

A large number of web queries are related to product entities. Studying evolution of product entities can help analysts understand the change in particular attribute values for these products. However, studying the evolution of a product requires us to be able to link various versions of a product together in a temporal order. While it is easy to temporally link recent versions of products in a few domains manually, solving the problem in general is challenging. The ability to temporally order and link various versions of a single product can also improve product search engines. In this paper, we tackle the problem of finding the previous version (predecessor) of a product entity. Given a repository of product entities, we first parse the product names using a CRF model. After identifying entities corresponding to a single product, we solve the problem of finding the previous version of any given particular version of the product. For the second task, we leverage innovative features with a Naïve Bayes classifier. Our methods achieve a precision of $\sim$88% in identifying the product version from product entity names, and a precision of $\sim$53% in identifying the predecessor.

Categories and Subject Descriptors

H.2.8 [**Information Systems**]: Database Applications—*Data Mining*; H.3.4 [**Information Storage and Retrieval**]; H.4.0 [**Information Systems Applications**]: General

Keywords

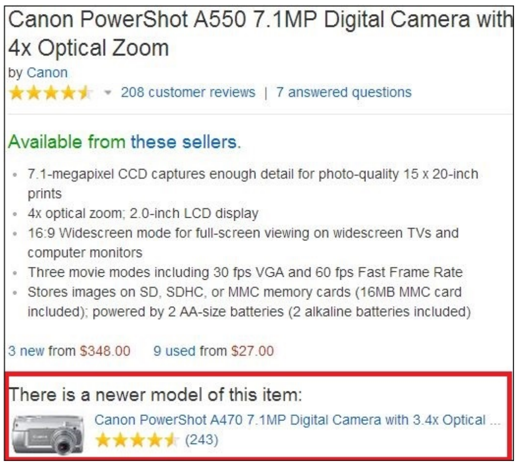
Entity Mining; Product Predecessor Prediction; Product Name Parsing; Entity Evolution

1. INTRODUCTION

A large number of users search on the web for product entities with a variety of intents: knowing product specifications, comparing products, buying products, selling products, reading reviews, etc. Online data on consumer products is also increasing day by day thanks to a large number of e-commerce and review websites. While such e-commerce websites already host information about millions of products, new products and new versions keep appearing frequently. Many of such products have a long history. Linking various versions of the same product temporally can help us understand evolution trends for particular products. Also, the ability to link all versions of the same product together, and the ability to rank them temporally can improve product search engines.

Predecessor Prediction Problem: Given a set of versions of a product entity, one can construct a product version tree by identifying parent-child relationships. Such a product version tree can be constructed by linking a child node to a parent such that the parent product version was released just before the child, and is the closest to the child node in terms of its specifications. Thus, the central problem in creating such a product version tree is to find the immediate predecessor version of a particular product version. We focus on the predecessor prediction problem in this paper.

Example Application: Product predecessors have already been used to improve user experience on product portals like Amazon as shown in Figure 1. But it is unknown as to how Amazon achieves this. We use such product pairs obtained from Amazon as the golden set for our experiments.



Canon PowerShot A550 7.1MP Digital Camera with 4x Optical Zoom
by Canon
208 customer reviews | 7 answered questions

Available from these sellers.

- 7.1-megapixel CCD captures enough detail for photo-quality 15 x 20-inch prints
- 4x optical zoom; 2.0-inch LCD display
- 16:9 Widescreen mode for full-screen viewing on widescreen TVs and computer monitors
- Three movie modes including 30 fps VGA and 60 fps Fast Frame Rate
- Stores images on SD, SDHC, or MMC memory cards (16MB MMC card included); powered by 2 AA-size batteries (2 alkaline batteries included)

3 new from $348.00   9 used from $27.00

There is a newer model of this item:
Canon PowerShot A470 7.1MP Digital Camera with 3.4x Optical ...
(243)

Figure 1: "Newer Model" Feature on Amazon

Challenges: Predicting the predecessor version poses the following challenges. (1) There is no common convention followed in naming versions or products. Even the same manu-

[*]The author is also affiliated with IIIT-Hyderabad

facturer does not follow any standard convention. Information extraction from short listing titles present a unique challenge, with the lack of informative context and grammatical structure. (2) Product descriptions are mostly provided in unstructured natural language form [5]. Product mentions in the description do not follow any canonical name. For example, the product *JVC S-VHS Camcorder* may appear as *Super VHS Camcorder* or *S VHS Camcorder* in the product description. Linking such variations is challenging.

Brief Overview of the Proposed Approach: We solve the problem of finding the predecessor version of a given query product version in two stages. In the first stage, given a set of product entity names, we parse the product names to identify the "brand", "product", and "version" indicating words from the product name. The first stage thus gives us all the product entities belonging to the same product as the given query product version. The second stage ranks these candidates and chooses the most probable predecessor version from the candidate set. Both the stages follow a supervised approach with interesting features.

Contributions: The contributions of this paper are as follows.

- We propose the novel problem of predicting the predecessor version of a given query product version.
- We present a two-stage approach to solve the problem: (1) parsing the product title, and (2) predicting the predecessor version given all candidates.
- Experiments on a dataset crawled from Amazon show that our methods achieve a precision of ∼88% for the first stage, and a precision of ∼53% for the second stage.

2. RELATED WORK

Our work is related to research on (1) extracting attribute values for product entities, and (2) identifying related entities.

Mauge et al. [5] structure items into descriptive properties using a two-step method (unsupervised property discovery and extraction, and supervised property synonym discovery). They mine this data from product descriptions. Raju et al. [8] perform the task of automatically discovering attributes of products from text descriptions using an unsupervised approach. Ghani et al. [3] present a semi-supervised co-EM algorithm for attribute-value entity extraction from product descriptions. Unlike these approaches, we aim at extracting information from product titles only. Putthividhya et al. [7] present a named entity recognition (NER) system for extracting product attributes and values from listing titles. However, they focus on clothing and shoes categories only and design methods specific to those domains. None of these works focus on extracting the version information from the product listing titles, which is the focus of the first stage of the proposed work.

Relationship extraction between entities has been studied recently [1]. However all of these works [4] focus on semantic relation between entities, while the second stage of the proposed work deals with identifying temporally related product entities. To understand the relationship between statistically related entities, Fang et al. [2] propose the problem of entity relationship explanation. The temporal relationship discovered by the proposed work is self-explanatory.

3. PREDICTING PREDECESSOR VERSION: THE PROPOSED APPROACH

We solve the problem of finding the predecessor version of a given product version in two stages. In the first stage, given a set of product entity names, we parse the product names to identify the "brand", "product", and "version" indicating words from the product name. The first stage thus gives us all the product entities belonging to the same product as the given product version. The second stage ranks these candidates and chooses the most probable predecessor version from the candidate set. We discuss the supervised approach for both the stages in this section.

3.1 Parsing the Product Title

We crawled product titles and descriptions for our experiments from Amazon. Thus, for every product, we have the title, and other details like the product ID, product description, reviews, etc. Typically a product title consists of product brand, product name, product attributes, attribute values, version information, and accessory words. We aim at labeling such product titles with the following tags: `brand name, product name, version` and `others`. For example, a product in this dataset has product title *Leica D-Lux 6 digital camera*. In this example, the *Leica* will be labeled as "brand name", *D-Lux* will be labeled as "product name", *6* will be labeled as "version" and *digital camera* will be labeled as "Others".

The task of parsing the product title translates to categorizing words in the product title as brand name, product name, version and others. Typically a product title consists of "brand" before "product" with a high probability. Thus, the order of words in a product title follows a sequential pattern. Hence, we choose a Conditional Random Fields (CRF) based approach to solve this problem. CRFs are a class of statistical modelling method and are widely used for labeling or parsing of sequential data. The CRF is trained using manually labeled data for a few product titles. The CRF tagger gets trained on the (word, label) pairs using features obtained from the product description, context patterns surrounding the labels and linguistic patterns frequently associated with the labels in the training set. It is then used to label words in product titles from the test set. We used the MALLET [6] toolkit.

Next, we will discuss the set of features we used for the CRF model. We used three main types of features. Consider a product title *Apple iPad Mini (White)*. We will use this example to describe a few of the features.

Linguistic features

We analysed the Part-Of-Speech (POS) tags of the words in the product title to identify POS tag patterns. We check if the words of the product title have POS tag in the set {NNP, MD, VB, JJ, NN, CD, NNS, IN, RB, DT, VBP, VBD, CC, VBN, JJS, VBZ, LS, VBG, FW, PRP$, PRP, SYM}. We have a binary feature for presence or absence of each of these parts of speech. For example, the word *White* in the above example will have the linguistic feature JJ set as TRUE and all other linguistic features set as FALSE.

Context and Word Characteristics Features

The contextual text of the brand name, product name and version in the product title conveys valuable information. For each word token, we define the following features: (1) Position of the word from the beginning of the product title, in terms of number of words; (2) Is the word the last word in the title; (3) Is the word alphabetic; (4) Does the word represent a color; (5) Is the word numeric; (6) Is the word surrounded by parenthesis; (7) Is the previous word "for".

In the above example, the word *White* has the following context features set to TRUE: the word is alphabetic, the word is surrounded by parenthesis, the word represents a color, and the word appears at position 4 in the title. If one has an exhaustive dictionary of all brand names, features like "Is the word a brand?" could also be considered.

Product Description Features: Product description specifies the attributes and the values for these attributes of the entity. We have a binary feature corresponding to every attribute, which checks for presence or absence of that attribute's value in the product description. We identify nine such features as follows: description itself, weight, review date, review title, model, category, bought next, bought along with, URL.

We label the product titles using a CRF model based on the above set of features. Next, we group together product entities that have the same brand name and product name but differ only in the version part. Given any query product version, we identify its (brand, product) cluster after labeling the query product entity title using the CRF. All members of this cluster are candidates for being its predecessor version.

3.2 Predicting Predecessor Version

As discussed in the previous subsection, the first stage provides a set of product entities with the same brand name and product name as compared to the query product entity. These are candidates for the second stage. The second stage as discussed in this subsection uses a classification based approach to identify the most probable product predecessor version from the candidate set for the query product entity.

The classification approach relies on the following intuitive set of binary features.

Lexical Ordering

This feature indicates whether the candidate lexically preceeds the given query product version. The version names often manifest alphanumeric patterns. For example, "Nokia Lumia 1020" preceeds "Nokia Lumia 1320" both wrt time of release as well as lexicographically. Hence, product versions can be ordered lexicographically.

Review-Date Based Ordering

This feature indicates whether the candidate is older than the given query product version. Products can be ordered temporally based on their earliest review date. We obtain posting date of the earliest review for all products in our dataset. We then determine the age of a product version using the review date of its earliest review. Intuitively, the version which got reviewed earlier has a higher probability of being older.

Mentions Based Ordering

This feature indicates that the candidate product version was *mentioned* in the given query product version's description or reviews. When describing a new version of any product, the product manufacturers often highlight improved features in this version compared to the previous version. Similarly, reviewers often compare the current version with the previous version when writing reviews. We use these *mentions* of product names in product reviews and description towards linking two versions of the same product. We use simple substring matches (with the `product name`) to link the mentions with the product entities.

Label	P	R	F
`brand name`	0.98	0.65	0.77
`product name`	**0.89**	**0.58**	**0.69**
`version`	**0.69**	**0.48**	**0.55**
`others`	0.84	0.98	0.91

Table 1: CRF Accuracy for the Product Title Parsing Task (when `product name` and `version` are treated as Separate Labels)

Label	P	R	F
`brand name`	0.98	0.65	0.77
`product name/version`	**0.88**	**0.55**	**0.67**
`others`	0.85	0.98	0.91

Table 2: CRF Accuracy for the Product Title Parsing Task (when `product name` and `version` are treated as the Same Label)

Given a query product version, a classifier is used to rank all the candidate product versions and the most probable one is chosen.

4. EXPERIMENTS

In this section, we will describe the dataset and analysis of the results of our experiments.

4.1 Dataset

We crawled ~462K product description pages from Amazon. These pages were parsed to obtain product title, product description, reviews, etc. The dataset is publicly available[1].

4.2 Results

Parsing the Product Title

From the dataset, we selected 500 product titles from the *camera & photo* category, and hand labeled the words as `brand name, product name, version` or `others`. A CRF sequential learner was trained on this dataset to predict the labels. The Precision (P), Recall (R) and F-measure (F) from a five fold cross validation of this task is presented in Table 1. Note that the label `brand name` was identified with the highest precision. Even the product name was identified with a good precision. This implies that given a query product version, we obtain its candidate product predecessor versions with a high accuracy in the first stage. However, the precision for the `version` label is quite low.

We observed that the classifier incorrectly marked some `product name` words as `version` and vice versa. This was because for such product entities, the product name and the version indicator does not appear as two different words; instead it appears as a single word. For example, *ge dv1-co* and *foscam fi8918w*. In such cases, the words *dv1-co* and *fi8918w* serve both as the version indicators as well as product names. Hence, we also trained a CRF model for a three-class labeling task with labels as `brand name, product name/version` and `others`. The precision, recall and F1 score for this three-class labeling task are presented in Table 2.

If the 4-label CRF identifies product name and version as separate words, we use the output from the 4-label CRF,

[1]http://tinyurl.com/lclapy8

else we use the output from the 3-label CRF. For the output from 4-label CRF, clustering is performed on (brand name, product name) to cluster and obtain candidates for the query product version. For the output from the 3-label CRF, clustering is performed on the brand name alone.

Predicting Predecessor Version

As shown in Figure 1, Amazon lists the immediate preceeding version of a product, for some products. For example the product description page of *Canon Powershot A550 7.1MP Digital Camera with 4x Optical Zoom* lists *Canon Powershot A470 7.1MP Digital Camera with 3.4x Optical Zoom* as the immediate preceeding model. We collected all such pages from our dataset and extracted (version1, version2) ordered product pairs. We use this as our golden truth for the product immediate predecessor prediction task. 40 out of our 500 product pages have such golden predecessor version mentions[2].

Using the output from the CRFs, we cluster the product entities based on the brand name. For each cluster (i.e., brand), product versions in the golden truth are considered. For each product version, we list every possible candidate of the predecessor product version. Such (product query, candidate) pairs are used to define instances for the Naïve Bayes classifier which is then used to compute the probability that the candidate could be a predecessor of the queried version. Note that for this task, the dataset is highly imbalanced. The positive class (the correct (query, predecessor) pairs) is in minority.

Table 3 presents the results of the product predecessor prediction for the positive class in terms of the following metrics: True Positive Rate (TP), False Positive Rate (FP), Precision (P), Recall (R) and F1 score (F). The results were found to be statistically significant (95% confidence). Each row indicates the feature sets used. Note that none of the orderings individually could provide reasonable accuracy. Each of the orderings have drawbacks when used individually. For example, lexical ordering does not work in cases where no such ordering is followed. For example, *Google Nexus 7* was the earlier version of *Google Nexus 5*. Review-Date based ordering does not work for those products which was launched on Amazon later than their actual release dates. Similarly, Mentions based ordering does not work if the review text or product description does not contain mentions of previous versions, or the mentions could not be detected accurately. This stresses the need of combining various approaches for this task. The combination of the Review-Date Based Ordering and the Lexical Ordering performed the best in predicting the correct product predecessor version entity (∼53% precision). Note that this precision value is good, because we have reported numbers on the minority class only.

5. DISCUSSIONS

Besides the review-date based ordering, we tried two other interesting ways of ordering product versions as follows.

- The release date of product versions could be extracted from their manuals. But manuals are not easily available for most products, while reviews are in plenty.
- We queried a search engine with the brand name and the product name. Based on the order of the search results, assuming that the later versions appear at the top, one can rank product entities. But both the coverage as well as the precision of such an approach was found to be quite low.

[2]These titles are listed in the file "goldenData" in the dataset at http://tinyurl.com/lclapy8

Feature	TP	FP	P	R	F
Lexical + Review-Date	**0.632**	0.049	**0.533**	**0.632**	**0.578**
All features	0.579	0.049	0.512	0.579	0.543
Review-Date	0.579	0.061	0.458	0.579	0.512
Review-Date + Mentions	0.553	**0.047**	0.512	0.553	0.532
Lexical + Mentions	0.5	0.049	0.475	0.5	0.487
Lexical	0.5	0.056	0.442	0.5	0.469
Mentions	0.45	0.049	0.462	0.45	0.456

Table 3: Classifier Accuracy for the Positive Class for Product Predecessor Version Prediction

6. CONCLUSIONS

We proposed the novel problem of predicting the predecessor version of a given query product version. We presented a two-stage approach to solve the problem: (1) parsing the product title, and (2) predicting the predecessor given all candidates. Experiments on a dataset crawled from Amazon show that our methods achieve a precision of ∼88% for the first stage, and a precision of ∼53% for the second stage. The solution can be helpful for a variety of applications like building product version trees, studying entity evolution, product search engine ranking, comparing product versions, etc.

Though we tested the method for the *camera & photo* category only, the method is generic. We plan to test it on products from other domains in the future. Attribute values usually show a trend across product evolution. For example, pixel resolution of "Canon Powershot cameras" has been increasing, weight of "Samsung hard disks" has been increasing over time. We would like to extract such patterns in attribute evolution of product entities and use them to correct the temporal ordering of product versions.

7. REFERENCES

[1] N. Bach and S. Badaskar. A Survey on Relation Extraction. *Language Technologies Institute, Carnegie Mellon University*, 2007.

[2] L. Fang, A. D. Sarma, C. Yu, and P. Bohannon. REX: Explaining Relationships Between Entity Pairs. *PVLDB*, 5(3):241–252, Nov 2011.

[3] R. Ghani, K. Probst, Y. Liu, M. Krema, and A. Fano. Text Mining for Product Attribute Extraction. *SIGKDD Explorations*, 1:41–48, 2006.

[4] T. Hasegawa, S. Sekine, and R. Grishman. Discovering Relations Among Named Entities from Large Corpora. In *ACL*, pages 415–422, 2004.

[5] K. Mauge, K. Rohanimanesh, and J.-D. Ruvini. Structuring E-Commerce Inventory. In *ACL*, pages 805–814, 2012.

[6] A. K. McCallum. MALLET: A Machine Learning for Language Toolkit. http://mallet.cs.umass.edu, 2002.

[7] D. Putthividhya and J. Hu. Bootstrapped Named Entity Recognition for Product Attribute Extraction. In *EMNLP*, pages 1557–1567, 2011.

[8] S. Raju, P. Pingali, and V. Varma. An Unsupervised Approach to Product Attribute Extraction. In *ECIR*, pages 796–800, 2009.