Product Discovery From E-Commerce Listings via Deep Text Parsing

Uma Sawant
IIT Bombay,
Mumbai
uma.sawant@gmail.com

Vijay Gabale
Huew,
Bangalore
vijay.gabale@gmail.com

ABSTRACT

Understanding unstructured text in e-commerce catalogs is important for product search and recommendations. In this paper, we tackle the product discovery problem for fashion e-commerce catalogs where each input listing text consists of descriptions of one or more products; each with its own set of attributes. For instance, `[this RED printed short top paired with blue jeans makes you go green]` contains two products: item `top` with attributes {pattern=printed, length=short, brand=RED} and item `jeans` with attributes {color=blue}. The task of product discovery is rendered quite challenging due to the complexity of fashion dictionary (e.g. RED is a brand or green is a metaphor) added to the difficulty of associating attributes to appropriate items (e.g. associating RED brand with item top). Beyond classical attribute extraction task, product discovery entails parsing multi-sentence listings to tag new items and attributes unknown to the underlying schema; at the same time, associating attributes to relevant items to form meaningful products. Towards solving this problem, we propose a novel composition of sequence labeling and multi-task learning as an end-to-end trainable deep neural architecture. We systematically evaluate our approach on one of the largest tagged datasets in e-commerce consisting of 25K listings labeled at word-level. Given 23 labels, we discover label-values with F1 score of 92.2%. To our knowledge, this is the first work to tackle product discovery and show effectiveness of neural architectures on a complex dataset that goes beyond popular datasets for POS tagging and NER.

CCS CONCEPTS

• **Information systems → Information extraction**;

KEYWORDS

Product discovery, Sequence labeling, Multi-task learning

ACM Reference Format:
Uma Sawant and Vijay Gabale. 2018. Product Discovery From E-Commerce Listings via Deep Text Parsing. In *CoDS-COMAD '18: The ACM India Joint International Conference on Data Science & Management of Data, January 11–13, 2018, Goa, India.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3152494.3152503

1 PRODUCT DISCOVERY

In an e-commerce product marketplace such as Amazon, Myntra or Flipkart, geographically distributed sellers upload their product listings to a central product listing platform. While some brands and sellers provide structured descriptions consisting of key-value pairs, the vast majority of sellers only provide unstructured natural language descriptions. Typically, such unstructured listing text consists of a title and a description; where the title is a collection of important keywords, and the description details attributes or specifications of the product possibly spread over multiple sentences with ill-formed text and non-standard terminologies. For fashion e-commerce, which is the focus of this work, the products are typically apparels (shirt, top, jeans) or accessories (footwear, purses) and the attributes are typically brand name, color, fabric, pattern etc.

We parsed more than 2 million fashion listings across prominent fashion e-commerce portals in India and discovered that as high as *35% of listings contain multiple products in the text*, each with its own set of attributes. Take for instance an example shown in Table 1. Although the title of this listing signifies the underlying product as a top, it also contains an occurrence of a secondary item, legging, with its own set of attributes. We selected about 25K of these fashion listings, spread across different items, to manually inspect and then quantify the number of listings with multiple items and attributes. Table 2 shows the distribution of attributes for a few selected fashion items, along with the distribution of co-occurring secondary fashion items and their respective attributes on 25K tagged listings.

Text-based search engines on e-commerce platforms work on the principle of keyword indexing. However, owing to the presence of multiple fashion items in the text, many search queries, today, fetch irrelevant fashion items resulting in poor user experience. Apart from search engines, obtaining structured information of multiple products and identifying the dominant product is immensely useful for recommendation engines and advertisement businesses that serve contextual and relevant advertisements based on an e-commerce page, or any Web page content.

1.1 Challenges in product discovery

Transforming such unstructured text into appropriate structured form involves several challenges.

- **Ungrammatical text:** Bulk of product listings uploaded on e-commerce and marketplace platforms do not exhibit grammatical structure of natural language; many times text contains contains typographical errors and abbreviations.

Fashion listing	`title`: Perfectly designed printed tops `description`: This mandarin top from Indigo fashion will make you go gaga this season, dress up for that perfect cocktail party. This georgette top is soft against your skin, team it with a black jegging to accentuate blue color, waist length, 3/4 sleeves.

Expected output	**product 1**: {apparel = top, pattern = printed, neck = mandarin, fabric = georgette, color = blue, sleeve = three-fourth, length = waist-height, occasion = cocktail-party, brand: Indigo fashion}
	product 2: {apparel = jegging, color = black}

Table 1: Example listing in fashion commerce containing two products. Note that dress is not a valid keyword for this example whereas indigo, which is also a color, is part of the brand name.

primary item	primary attributes	secondary items	secondary attributes
top	color 84%, pattern 65%, fabric 59%, brand 29%	blouse 17%, jeans 16%	color 33%, pattern 6%
kurti	color 89%, length 71%, pattern 57%, fabric 54%	legging 14%, jegging 11%	color 23%, pattern 2%
dress	color 79%, length 67%, sleeve 46%, shape 31%	pump 13%, wedge 8%	color 17%, pattern 2%

Table 2: Distribution of attributes and secondary products for sample fashion items. As an example, 84% of listings for fashion item top have at least one color value, 16% have mentions of jeans as one of the secondary fashion items, and 33% have mentions of secondary color attribute that is attached to secondary fashion items

- **Partially known schema:** Though one can define product schema in terms of item types such as apparel or footwear and attribute types such as color, fabric or brand, such schema is often volatile containing tens (e.g., colors) or even thousands (e.g., brands) of values that get added or deleted due to fast moving fashion trends. For instance, 8 new colors and 241 new brands were added in a fashion season of two quarters on prominent Indian fashion marketplaces. Hence, it becomes imperative to *discover* previously unknown fashion item values and attribute values from the listing text.
- **Keyword ambiguity:** 'Indigo Fashion', 'Red', 'Blue', 'House of Cocktail', 'Tip Tops', 'Bodycon' are a few examples of fashion commerce brand names that also share popular color values, shape values or occasion values. 'High' is a neck value as well as length value; 'long' is a sleeve value as well as length value; 'shirt' in 'shirt dresses' is a shape value and 'dress' is commonly used to refer to upper or full-body wear. During curation and tagging process, we found that *more than 30%* of label-values either have at least two labels or used as a colloquial term. Moreover, unlike popular POS or NER datasets, e-commerce listings contain tens of sentences in description and the context around a fashion item is typically spread over multiple sentences. Thus, it is quite challenging to identify important fashion items and their attributes from such text data.
- **Product-attribute attachment ambiguity :** Table 1 shows a representative example where complementary apparel or footwear items often appear side-by-side in the same text description. One needs to correctly associate attributes such as color or brand to appropriate items to form a product. Moreover, to index a listing with relevant items and attributes, we must identify the dominant product in such cases.

1.2 Problem definition

Keeping in mind all the above challenges, we now formally define the *product discovery* problem.

Given a partial schema consisting of items and attributes and an unstructured text snippet possibly consisting of multiple sentences, **product discovery** entails the discovery of one of more products from the text through the identification of values of items and attributes; and the affiliation of each attribute with an appropriate item.

Item in our case is a fashion item of type apparel or footwear, e.g., 'dress' is one of the possible value for type 'apparel'. Similarly, attributes in our case are fashion attributes such as sleeve, brand. 'Sleeveless' for instance, is one of the possible values that we want to tag with attribute type 'sleeve'. We do not assume that we know in advance all possible values of an attribute type in our schema. But we assume that we know the taxonomy of fashion items and attributes (examples in table 3), which includes the names of the products and the attributes.

keyword-types	labels (known schema)	values (not known)
fashion item	apparel footwear	tops, jeans wedges, pumps
attributes	sleeve brand color	angel, puff indigo fashion cornsilk, dust

Table 3: Examples of candidate words and phrases to be tagged with appropriate labels

While prior works [12, 14] attempt to extract attributes from e-commerce product listings, these techniques consider only the title text, consisting of only a few words, ignoring descriptions. Hence, these techniques do not capture the long-term dependencies and relationships in an unstructured product listing. Most of the recent literature [5, 9, 11, 15, 16] on classical sequence labeling or NER problems has been designed and evaluated on popular POS tagging and NER datasets that have well formed grammatical English. Online commerce listings have characteristics of high variability in the formation and structure of sentences with product context often spread across multiple sentences. Performance testing on such datasets has not received much attention in prior work. Furthermore, none of the prior work attempts product discovery considering multiple items in a listing text. Note that we not only need to identify key items and attributes from the text, but also associate attributes to appropriate items to form meaningful products. The following example, *"title : Arcilia frill tops and blouses, description : comes in blue and off white pattern, combine this top with cropped blue jeans and black wedges"*, we wish to discover the following products *(1) label:apparel-item value:top, label:color values blue, off-white, (2) label:apparel-item value:jean, label:color value blue, label:style value cropped, (3) label:footwear-item wedges,label:color value black*; along with the fact that the first product is the dominant product (to be indexed by search engines appropriately).

1.3 Our contributions

Inspired by the recent advances in deep learning [1, 7, 9–11, 15, 17], we design a novel deep neural architecture that composes sequence labeling with multi-task learning to jointly solve attribute and fashion item labeling and attribute-to-fashion-item affiliation. Our architecture consists of an end-to-end trainable neural network with layers of character-embedding, Convolutional Neural Network (CNN), Bidirectional Long Short Term Memory Network (BLSTM) and *cascaded* Conditional Random Fields (CRF). The contributions of our work are as follows.

- To our knowledge, this is the first work to formulate the product discovery problem on unstructured text snippets and propose an end-to-end design based on deep neural architectures to solve the problem.
- Going beyond popular POS tagging and NER datasets, we experiment and evaluate our approach on one of the largest tagged datasets in e-commerce domain consisting of 23 labels and 25K labeled listings (or SKUs).
- We achieve state-of-the-art results for product discovery with 92.1% accuracy and 92.2% F1 score. Compared to prior work [5] on parsing e-commerce listings, we achieve more than 13% gain in accuracy.

This paper is organized as follows. In Section 2, we discuss the challenges and present our solutions. In Section 3 we discuss our experiments. In Section 4 we provide an overview of the related work and finally outline the future work and our conclusions in Sections 5 and 6 respectively.

2 DESIGN: COMPOSITION OF SEQUENCE LABELING AND MULTI-TASK LEARNING

Products to be discovered in the input text listing can be completely specified with the help of two types of labels for each word:

keyword labels (e.g. red[color] dress[apparel] with[-] black[color] shoes[footwear]) [1]) and attribute-product **affiliation labels** (red[P1] dress[P1] with[-] black[P2] shoes[P2]) where P1 is dress and P2 is shoes.

This labeling scheme works fine for almost all of the product listings, except those which have more than one product of the same type, e.g. two tops or two leggings. However, we found that such occurrences are rare; less than 0.05% a sample of 25K tagged listings had such more than 1 product of the same type. Furthermore, after parsing 2 million listings, we find that a listing at the most contains 5 different products and hence we keep 5 as the fanout for affiliate label prediction.

Discovery of above two types of labels converts our problem to a sequence labeling or Named Entity Recognition (NER) problem with two distinct classes of labels. Formally, this problem can be described as follows. Let $\mathcal{L}_a$ and $\mathcal{L}_c$ be the sets of all possible keyword labels and attribute-product affiliation labels respectively. Given an input text sequence $\mathcal{X} = \{x_1, \ldots, x_T\}$, the corresponding keyword labels can be specified as

$$\mathcal{Y}_a = \{y_1^a, \ldots, y_T^a \mid y_i^a \in \mathcal{L}_a\}$$

Similarly, attribute-product affiliation can be specified via another label sequence

$$\mathcal{Y}_c = \{y_1^c, \ldots, y_T^c \mid y_i^c \in \mathcal{L}_c\}$$

Given $\mathcal{X}$, the task of product discovery is to predict $\mathcal{Y}^* = \text{argmax}_{\mathcal{Y}_a, \mathcal{Y}_c}\, p(\mathcal{Y}_a, \mathcal{Y}_c | \mathcal{X})$

2.1 Design requirements

In this section we design a framework to discover products from a listing. We first enlist the key characteristics of the product discovery task. These characteristics motivate our choice of components in the end-to-end trainable neural network.

- **The language of fashion:** Capturing the fashion domain grammar and language such as 'accessarize', 'complements', "bold and bright colors", 'luxe' is important for the precise attribute and product label identification. Some phrases such as "winter clothing" or "formal look", though not specifying any attribute or clothing item in particular, can heavily influence the probability distribution of possible attributes found in the surrounding text; in this case fashion items such as jackets and office full shirt respectively. Furthermore, such influence may extend to words quite far away in the text. This necessitates *a robust mechanism to extract features out of fashion listings and model long-term word dependencies.*
- **Dependence between keyword and affiliation labels:** Although we specify our problem in terms of two labeling

[1]OTHER is a catch-all negative label indicating that the word is not a clothing-item or attribute

tasks, it is important to note the the two sets of labels exhibit dependencies. The information that `black` is marked as a color in `[BLUE releases a range of black dresses]` increases the chance of BLUE being tagged as a brand instead of color (even if BLUE is not specified in capital case). Knowing there exist multiple neck-type keyword labels in a given text increases the chance that the attributes are affiliated to different products. For text snippets such as `[half-sleeve top in red color]`, the fact that the previous word mentions a sleeve-type makes it more likely that the next label could be an apparel type than waist-size label. Dresses and sandals are more likely to co-occur in a listing than dresses and jeans.
Capturing such dependencies between keyword labels and attribute-product affiliation labels is imperative for correct labeling.

- **Importance of word and character level features** : The importance of *word and character features* such as capitalization, phrasing, function words in keyword or natural language text processing is undisputed. Product description data is no different, and it is important to consider both character and word level representations or features of the input text.

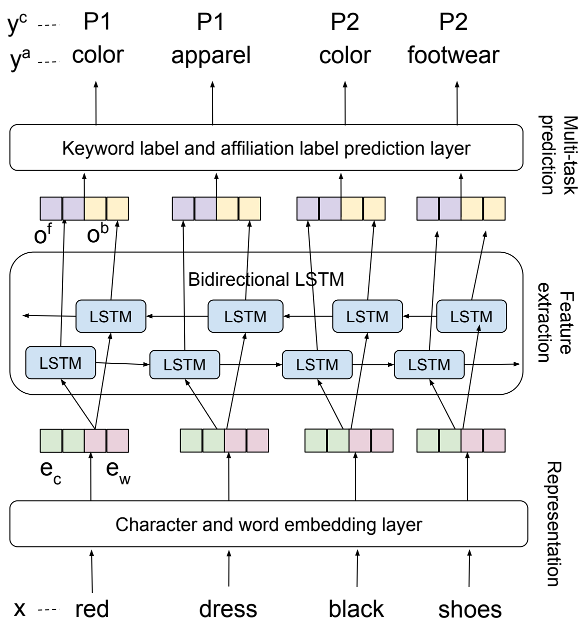

y^c --- P1 P1 P2 P2
y^a --- color apparel color footwear
Keyword label and affiliation label prediction layer
Multi-task prediction
o^f o^b
Bidirectional LSTM
LSTM
Feature extraction
e_c e_w
Character and word embedding layer
Representation
x --- red dress black shoes

Figure 1: System Architecture (e_w, e_c, o^f, o^b shown only for the word `red` to reduce clutter.)

Each of the above aspects naturally leads to certain design choices in our deep learning framework. For the input representation, we use both **character and word representations**. We make use of both char2vec and word2vec representations to represent a word in our vocabulory. For feature extraction, we use **Long Short Term Memory** (LSTM) networks, which are a class of recurrent neural networks designed to capture long-distance dependencies. As the data contains both forward and backward dependencies, we use **bi-directional LSTMs**. We model the entire listing text, consisting of multiple sentences and hence few tens of words, using a bi-directional LSTM. Our task is then to predict two labels for each of the words in a listing. For label predictions, we use **conditional random fields (CRFs)**, on the top of LSTM, which are a natural choice for sequence learning problems. Finally, given the dependency of our two set of labels, we propose joint training of our two tasks through a **multi-task framework**. Our multi-taks framework jointly learns to predicts the two sets of labels.

2.2 Architecture

Figure 1 shows our end-to-end architecture composed of three main components.

Representation. The first component is responsible for converting the input text into a vector representation. For each word, we look up a vector e_w from a pre-trained embedding table. We consider publicly available word2vec embeddings and further fine-tune the word embeddings on the listings in our training dataset. To capture character-level features for each word, we use a convolutional neural network [11] over individual character embeddings, followed by a max-pooling layer to create a fixed length representation e_c for each word w. We consider different embeddings for different characters taking into account capitalization and other variations. We concatenate the above two vectors, a vector obtained via word2vec fine-tuning and a vector obtained via the application of CNN on character embeddings, to create a final vector representation [e_c, e_w] for each word and feed it to the next layer.

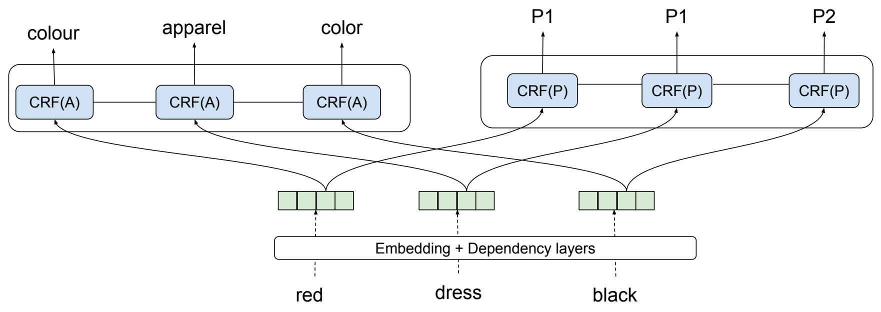
Feature extraction. The next component is responsible for extracting features and capturing long range dependencies from the input. We use LSTMs for the same. An LSTM can formally be described as :

$$\begin{aligned} i_t &= \sigma(W_i[h_{t-1}, x_t] + b_i) \\ \bar{C}_t &= \tanh(W_c[h_{t-1}, x_t] + b_C) \\ C_t &= f_t * C_{t-1} + i_t * \bar{C}_t \\ o_t &= \sigma(W_o[h_{t-1}, x_t] + b_o) \\ h_t &= o_t * tanh(c_t) \end{aligned}$$

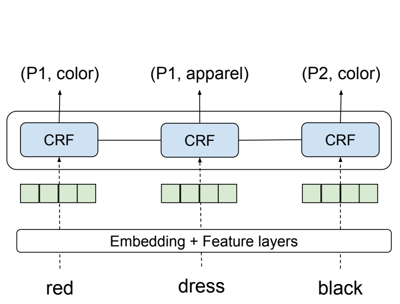
where x_t is the input ([e_c, e_w] in our case), W_i is the weight, b_i is the bias, h_t is the hidden node and o_t is the output at step t.
A bi-directional LSTM contains two LSTMs, one for each direction of a sequence. The output of both LSTMs (say o^f and o^b for forward and backward LSTM) is concatenated ([o^f, o^b]) and presented to the next layer as features.

Prediction. The final section is responsible for prediction of the two types of labels - keyword labels and product-attribute affiliation labels. We now discuss three possible designs for this layer.

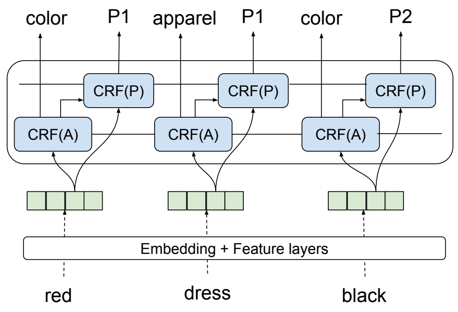
- *Multi-task disjoint prediction* : In this design, as shown in Figure 2a, keyword label prediction and product-attribute assignment label prediction is done via two separate prediction models which are trained together via a multi-task framework. In this design, the feature production layer remains shared; however, each label is predicted via a separate



(a) Label prediction via disjoint CRFs



(b) Prediction via joint state CRF



(c) Label prediction via cascaded CRFs

Figure 2: Multiple designs for Keyword label and affiliation label prediction layer (also see Figure 1)

CRF layer. Thus, keyword label prediction does not influence product-attribute assignment label prediction. Note that, both the models have the identical input ($[o^f, o^b]$) coming from the feature extraction layer. This design is similar to a prior work [15].

- *Joint prediction* : We treat the two labels as a single joint label $\{(y^a, y^c) \mid y^a \in \mathcal{L}_a, y^c \in \mathcal{L}_c\}$ and use a single prediction model in the joint label space. This is shown in Figure 2b. However, as a generic design choice this approach can result in a large label set and hence may not be scalable.
- *Multi-task cascaded prediction* : Here we propose to keep the label spaces separate, but train them together via a multi-task framework. Given the conditional dependency on affiliation labels from keyword labels, we tie up the two tasks in our multi-task prediction. As shown in Figure 2c, we supply the output distribution of attribute prediction model as additional features to the input of product-attribute affiliation prediction model. The reverse configuration, supplying affiliation prediction to keyword prediction, is also possible. However, it is intuitive to see that, for our specific problem of product discovery from e-commerce listings, the keyword labels strongly influence the affiliation labels (e.g., the occurrence of two distinct fashion items signifies at least two affiliation labels).

Given that our data is sequential, we chose CRFs as basic building blocks in both designs. A CRF is an undirected graphical model that captures the conditional probability distribution on labels given the input as follows. Let C be the set of cliques in the graphical model over input data x and labels y. Then,

$$p(y|x) = \frac{1}{Z(x)} \prod_{c \in C} \phi(x_c, y_c) \quad (1)$$

$$\phi(x_c, y_c) = \exp(\textstyle\sum_i w_i f_i(x_c, y_c)) \quad (2)$$

Here, ϕ is called potential function while Z is the partition function.

3 EVALUATION

In this section, we evaluate the techniques described in the previous section on e-fashion product listings tagged at word level. First, we describe our dataset and our data preparation procedure. We then evaluate the deep neural architectures described in the previous section against the state-of-the-art neural architectures for sequence labeling proposed in prior work. We further compare our approach with a prior approach, that utilizes a combination of word2vec [18] and CRF, that was proposed to solve named-entity recognition on the text that appears in title section of e-commerce listings. Finally, we extend our approach to solve the problem of identifying the dominant product of a listing in the presence of multiple products.

Our experiments are governed by the following questions:

(1) Are neural architectures, consisting of state-of-the-art components, effective on highly unstructured data (e-commerce product listings) for the task of product discovery?
(2) What is the degree of improvement one can expect on tasks that involve parsing large paragraphs by employing a neural architecture for multi-task learning against popular word2vec based approaches?

(3) How much unlabeled and labeled data do we need to achieve high performance via end-to-end trainable neural networks?

3.1 Dataset

samples	#sentences 5/50/95 pc(*)	#tokens	vocab size
train: 12760	1/11/17	1276K	8.3K
validate: 4950	1/12/19	293K	1.7K
test: 7280	1/11/25	563K	3.7K

Table 4: Summary of dataset used for training, validation and testing (*) pc indicates percentile. See text for explanation.

3.1.1 Size of dataset. We consider 24890 fashion listings that are tagged at word level for attribute and product discovery (data preparation described next). These listings are collected from 12 leading e-commerce and marketplace databases in India. The dataset split across train, validation and test samples is shown in table 4. The sentences column in this table shows 5th percentile, 50th percentile and 95th percentile number of sentences for each listing, when the listings are ordered by the number of sentences contained. For instance, 95% of listings in training dataset have at most 17 sentences, 50% of listings in the training set have at most 11 sentences and so on. The table also shows the number of tokens (at word level) and vocabulary size of the dataset. Note that the vocabulary size is quite large, partly due to the wide range of brand and merchant names, although we consider only a single e-commerce domain– fashion. The large vocabulary represents the unstructured and unorganized nature of the data where different sellers describe fashion with different semantics. In addition to these 25K labeled listings, we crawl 2M fashion e-commerce listings for the purpose of analysis and pretraining of models.

label	examples
apparel (western)	top, short, shirt, skirt, dress
apparel (ethnic)	lehenga, saree, anarkali, kurti
footwear	pump, wedges, shoes, flat
shape	shirt, wrap, bodycon, spathetti
sleeve	sleeveless, short, full, cap
pattern	cartoon, animal, polka-dot, plaid
fabric	lace, crochet, fur, satin
color	indigo, lime, silver, orchi
color-type	soft, dark, cool, warm
length	mini, short, maxi, knee-length
occasion	cocktail party, wedding, formal
brand	cocktail house, lime road, bodycon

Table 5: Sample labels and values

3.1.2 Labels and unseen data. We consider 23 keyword labels to tag words and phrases. In the cases of key phrases such as brand names that often span multiple words, we tag each of the words by the same associated labels. This is a weaker form of labeling as compared to BIO scheme that has start and end labels for key phrases but a stringent case for evaluating label accuracy. A subset of these labels and some of the values they take is shown in table 5. It is important to note that fashion (and most of the online retail domain such as consumer electronics) is a time-variant domain; hence, we cannot rely on all values to be known apriori for each label. Only a few popular values, (for instance, bodycon, spaghetti etc. for shape) remain static and appear frequently. For instance, four new shape values are added for fashion listings across two fashion seasons within a year. The discovery problem becomes even more important towards identifying brand names that are added and deleted quite frequently.

For apparel-type and footwear-type as labels, we consider 18 fashion items (such as tops, dress, jeans, shorts, pumps, wedges) and select 750 to 1000 tagged samples for each of the items. We have total 5311 unique label values in 24890 fashion listings and more than 110K total label values tagged with labels. For each of the 23 keyword labels, we split respective label values across train, validation and test in approximately 70%, 10% and 20% proportion. Thus, about 20% of the label values for each of the labels are never seen by the machine learned models during the training phase.

3.1.3 Dataset preparation. To label the above data at word level by keyword labels and affiliation labels, we employed two curators aided by automation. We first crawled the most popular values for each of labels from the Web and used those values to build a temporary attribute and attribute-values schema. Given a listing, we used a simple string-match approach to tag every possible occurrence of label-values in words or phrases by respective labels. Often a word received multiple labels (e.g., indigo as a color as well as brand in 'indigo fashion') due to semantic disambiguity (precision error). Furthermore, since a web crawl may not fetch all possible attribute-values, the above procedure missed to assign a label to a few words or phrases which were valid label values (recall error).

At this point, we asked the curators to independently correct the labels or add new labels to prepare a tagged listing. We selected only those samples which had a consensus on all labels by both the curators. We logged correction actions by both the curators and found that both of them corrected at least 32% of the tags from the ones output by string-matching automation software. This includes words or phrases that are marked incorrectly (for instance, although 'high' is a neck type, high in 'this is high class dress' is used in a general sense) as well as words or phrases with at least two labels (semantic disambiguation). Among all the tags eventually labeled, we found that the curators discovered 12% new label-values.

3.1.4 Multi-product listings and semantic ambiguity. Out of 25K tagged listings, more than 35% of the listings have at least two fashion items, apparel or footwear or both. More than 63% of listings have at least 4 attribute values associated with at least one fashion item. Among the 5311 labels across all listings, 1063 have at least two labels conflicting between colors, brands, merchants, shape, pattern and neck types. While 32% tags were corrected among total tags for all listings, about 20% label-values have semantic ambiguity with other label-values. Identifying all fashion items and associated attributes from a product listing with high precision and high recall thus becomes quite challenging.

We assume that a product in a listing can have multiple occurrences of its constituent items and attributes; however all occurrences of the same fashion item refer to a single product. For instance, `[red colored top with blue color jeans, the top has printed pattern]` has multiple occurrences of words 'top'; we assume that there is only one underlying product with its constituent item as 'top'. As mentioned earlier, less than 0.05% of cases in our tagged dataset violate this assumption.

3.2 Setup details

We used Theano library with Lasagne package to build and train neural networks on AWS g2 instances that have NVIDIA GRID GPUs, each with 1,536 CUDA cores and 4 GB of video memory. Each word and special character is treated as a separate input token. We do not preprocess data and do not hand-craft any features.

3.2.1 Model parameters. For initializing model parameters, we follow procedure in [11] and report results for the parameters that give us the best performance.

layer	hyper-parameters	value
CNN	window size	4
	number of filters	28
LSTM	state size	240
	initial state	0
Dropout	rate	0.5
Other	batch size	16
	initial learning rate	0.01
	decay rate	0.06
	momentum	0.9
	gradient clipping	5.0
	epochs	35

Table 6: Hyper parameters for all experiments

We initialize character embeddings as uniform samples from $[-\sqrt{\frac{4}{d}}, +\sqrt{\frac{4}{d}}]$. Here d indicates embedding dimension, which we set to 28. As observed in [11], original word2vec embeddings are fundamentally trained without considering common symbols, punctuations and digits. Since we do not use any data pre-processing, instead of word2vec, we choose GloVe [4] embeddings with embedding dimension 100. We fine tune GloVe embeddings on 2M fashion listings crawled separately. Furthermore, we fine tune character-level and GloVe embeddings during gradient updates of the neural network model. Furthermore, We tune the hyper-parameters on the development sets by grid search and arrive at parameters shown in table 6. We observe that drop out achieves significant performance boost in accuracy (thus we corroborate the observation in [11]) and apply dropout on character embeddings before inputting to CNN, and on both the input and output vectors of BLSTM. Model training is then relatively straightforward and we follow the standard applications of gradient based learning and backpropagation. We apply early stopping procedure as mentioned in [11] and observe that we achieve best performance at 35 epochs.

Towards a fair comparison, we pre-train neural networks for all approaches on 2M listings for the task of predicting next word in the text. Note that, for approaches in prior work, we set the sizes of hidden vectors and other hyper-parameters as mentioned in the respective works.

3.2.2 Metrics. We evaluate all approaches on weighted precision, weighted recall, weighted F1 and weighted accuracy for attribute as well as product-attribute affiliation predictions. For instance, we compute weighted F1 for labels as follows: $\sum_{i \in \mathcal{L}_a} P_i \times F1_i$ where P_i is the fraction of all tags to predict for label i. We evaluate labeling accuracy for two tasks separately: labeling of keywords by product-types and attribute-types and labeling of association of keywords to products. We also compute the joint accuracy (instances where we get both the tasks right) and consider it as the accuracy for product discovery task. Furthermore, we consider two cases for the evaluation: keyword values that are present in train and validation set (in-vocabulary) and keyword values that are not present in train or validation set (out-of-vocabulary).

3.2.3 Evaluation. We compare our proposals discussed in Section 2. For our baseline we train two separate end-to-end neural networks each for the individual tasks of keyword prediction and product-attribute affiliation prediction.

Now, instead of having two separate networks, neural architecture in [15] shares network parameters to solve product discovery. The shared part of the network contains word-level embedding as the input layer along with bidirectional LSTM as the feature formation layer. The network then has two separate CRFs for each of the tasks. In comparison, we evaluate our proposed approaches: **joint-CRF** where we consider a single end-to-end network that outputs a label as an item-attribute pair in a joint fashion and **cascading-CRF** where we output keyword label as well as associated product with two CRFs working in tandem while sharing input and feature formation layers. For joint-CRF approach, we consider *number of fashion items* × *number of attribute types* as labels. In our dataset, we have 18 apparel and other fashion items and 21 attribute types, thus total 23 labels.

3.3 Comparison of multi-task approaches

Results in Table 7 indicate that joint-CRF approach performs significantly better than having two separate CRFs for each task. This corroborates our hypothesis that joint prediction of attribute and product for each input word is a better design approach than traditional multi-task approach of two separate CRFs. Furthermore, our cascading-CRF approach gives even greater accuracy and F1 score than joint-CRF approach. This validates our design approach that is based on the dependency between attribute prediction and product association in product discovery: if the underlying context and semantics is captured correctly, only a few attributes can be associated with a given product and vice-versa. For instance, while color applies to most of the products, sleeve type applies only to full-body or upper-body clothing. Cascading-CRF approach exploits such dependency and hence it results in superior performance.

criteria	approach	(acc, f1)	(prec, recall)
in-vocab label prediction	baseline	(84.6%, 85.2%)	(86.4%, 84.2%)
	prior-work [15]	(89.4%, 90.2%)	(89.4%, 91.1%)
	joint	(92%, 92.8%)	(93.4%, 92.4%)
	cascading	(93.4%, 94.1%)	(94.1%, 94.3%)
product attribute affiliation	baseline	(89.5%, 89.8%)	(89.4%, 90.3%)
	prior-work [15]	(93%, 92.6%)	(92.4%, 92.9%)
	joint	(95.7%, 95.8%)	(96.3%, 95.4%)
	cascading	(97.7%, 98.6%)	(98.4%, 98.9%)
product discovery	prior-work [15]	(81%, 80.84%)	(81.4%, 80.3%)
	joint	(90.5%, 90.9%)	(90.7%, 91.3%)
	cascading	(92.1%, 92.2%)	(92.7%, 91.8%)

Table 7: Comparison of multi-task neural architectures for product discovery

Regarding the evaluation of prior work, it is worth noting a couple of observations: (1) The fashion-commerce listing data that we have, as mentioned earlier, is highly unstructured, and the approach in prior work [15] yields considerably lower values for accuracy and f1 score as compared to the application of the same approach on the native data in [15]. (2) The approach in [15] uses 40K sentences for training, 4K sentences for validation and 4K sentences for test which is an order of magnitude lesser than the size of our dataset; hence, the presence of variance in the validation and test dataset is not the cause for the worse performance of prior work compared to our approach. We thus choose cascaded-CRF as our approach to compare against prior work and report the final accuracy and F1 numbers.

criteria	(acc, f1)	(prec, recall)
4K	(61%, 60.9%)	(61.6%, 60.3%)
7K	(82.2%, 82.9%)	(82.6%, 83.3%)
10.5K	(89.5%, 90.04%)	(89.7%, 90.4%)
12K	(91.3%, 91.6%)	(91.3%, 92%)
14K	(90.4%, 91.7%)	(91.5%, 92.1%)

Table 8: Dataset requirement for task at hand

We further evaluate our approach of cascaded CRF on different sizes of training datasets, see table 8. We observe that for the problem at hand, performance of our model, for the best set of hyper-parameters, approaches a saturation point at 12K examples. We also crawled additional 1.5M fashion e-commerce listings to evaluate our approach in the wild. Our trained model discovered and tagged more than 2.6M fashion products with more than 9.5M fashion attributes. Thus, we believe that by tagging only a few thousand examples at word level, one can expect end-to-end trainable neural networks to achieve high performance on tasks as complex as product discovery on e-fashion listings. Furthermore, we believe that our approach is quite generic and can be utilized to discover multiple items and respective attributes from an unstructured text in any other domain with a small set of samples tagged at word level.

criteria	approach	(acc, f1)	(prec, recall)
in-vocab label prediction	baseline	(87.1%, 87.7%)	(88.3%, 87.2%)
	prior-work	(91.1%, 91.5%)	(90.8%, 92.3%)
	joint	(92.8%, 93.9%)	(94.6%, 93.4%)
	cascading	(95.2%, 95%)	(94.9%, 95.2%)
out-vocab label prediction	baseline	(80.5%, 80.79%)	(80.4%, 81.2%)
	prior-work	(85.9%, 86.59%)	(85.9%, 87.3%)
	joint	(91.3%, 91.24%)	(91.2%, 91.4%)
	cascading	(92.2%, 92.29%)	(92.5%, 92.1%)

Table 9: Comparison of each approach on out-of-vocabulary data

3.4 Performance on out-of-vocabulary attribute values

Table 9 shows comparison of these approaches on in-vocabulary label-values and out-of-vocabulary label-values. As we mentioned earlier, about 20% of the label-values among all label-values are not seen by the model; that we evaluate as out-of-vocabulary label predictions. We observe very little drop in performance by joint and cascading prediction models while it affects baseline and prior approaches significantly. This demonstrates that our proposals are able to capture label dependencies effectively, leading to a good performance on out-of-vocabulary labels.

3.5 Comparison with non-multi-task product extraction prior work

Prior work in [5] proposes a word2vec [18] and CRF based NER approach to tag attributes present in titles (i.e., one sentence) of e-commerce catalogs. To understand performance improvement by our approach to solve NER on an e-commerce dataset, we compare effectiveness of our approach on product discovery and as a consequence on NER against the technique in [5]. Our approach utilizes character-embedding along with BLSTM, a composition that we expect to capture dependency and long-term context present in the listing text. We note that work in [5] processes only titles and does not consider parsing description along with titles, and hence does not evaluate NER thoroughly on paragraphs in e-commerce listings. As described in [5], we implement the design using word2vec approach [18] and use skip-gram, context window size of 2, down-sampling parameter as 1e-3, number of negative samples as 10 and set vector dimension to 300.

Towards a fair comparison, we fine-tune word2vec feature vectors by pretraining the model on an unlabeled dataset consisting of 2M listings. Table 10 shows the experimental results. By analyzing logs obtained during label curation, we observe frequent conflicts of apparel type against its general use and color type against brand name. Hence, we specifically compare prior work on such labels (or entity-types) and evaluate NER on our dataset. We observe that our approach that utilizes character-embedding, CNN, BLSTM and

criteria	approach	(acc, f1)	(prec, recall)
apparel	prior-work [5]	(83%, 82.69%)	(82%, 83.4%)
	cascaded	(95.9%, 95.34%)	(94.4%, 96.3%)
color	prior-work [5]	(80%, 82.3%)	(82.3%, 82.5%)
	cascaded	(93.1%, 92.8%)	(92.5%, 93.3%)
brand	prior-work [5]	(80%, 79.4%)	(79.1%, 79.9%)
	cascaded	(92.4%, 91.7%)	(92%, 91.6%)

Table 10: Results for extracting keyword labels

cascaded-CRF significantly outperforms prior work to solve NER on e-commerce dataset. Furthermore, as shown in table 11, prior work fails to capture dependency and context across multiple sentences and performs poorly in doing semantic disambiguation.

apparel FP	color FP	brand FN
wear on top of	dress up to be golden actress	kurti from RED
this shirt dress	matches with blue sky	only silver lining
dress up your summer	pattern with yellow flowers	fab dress at diamond blue

Table 11: False positive and false negative for prior work

3.6 Dominant product

More than 35% of listings in our database contain at least two products. However, to index and rank a listing, it is important for a search engine to identify the dominant product from it. To identify the dominant product, we take a simplified yet effective approach. We extend the neural network design of product discovery and train a softmax based classifier on top of hidden state output by the last state of BLSTM. Based on statistical evidence, we observe that a listing contains at most 5 products and hence we keep fanout of softmax to 5 classes. We take neural network trained for product discovery and fine tune the network on 4K listings for the task of predicting dominant category. We evaluate the fine tuned model on 1.5K listings consisting of at least two products and achieve accuracy of 98.7%.

4 RELATED WORK

Transforming unstructured text into structured products is a well-researched area; however, to the best of our knowledge, there is no prior work that tackles the multi-product discovery problem, especially using deep neural networks. Importantly, most of the prior work on multi-task learning employs two different datasets for two different types of labels or tasks, we experiment and evaluate product discovery for multi-task learning while employing a single dataset tagged for two different tasks.

4.1 Attribute extraction

In one of the earlier works on product extraction, [3] used semi-supervised learning algorithms using word context as features to extract product attributes and values from text. [16] presented a bootstrapping approach for extracting four types of product attributes and their values from product listing titles while relied on HTML page structure [17]. In other work, the focus of [12] was only on product attribute extraction, without considering multiple products in a listing text. [5, 16] modeled shopping domain attribute extraction as a Named Entity Recognition (NER) problem while considering on the title text of a shopping listing. [5] used distributed word representations to extract attributes from a title text. All the above approaches assume that a given listing text describes only a single product. This is not true in practice, and the existence of multiple products gives rise to the product-attribute affiliation constraint. We handle this constraint using a multi-task learning framework based on a deep neural network that is end-to-end trainable. Furthermore, we work with a total of 23 attributes and the complete listing text, which makes our task more complex.

4.2 Sequence labeling

There has been a great deal of progress recently in the area of sequence labeling[9, 11] through deep learning. We draw inspirations the above works and extend it to the task of product discovery interpreted as a multi-task learning problem. Multi-task learning has been successfully shown to improve natural language processing [2], Chinese word segmentation [15], text classification [10] and collaborative filtering [1] tasks. Our work is in parallel with the above research.

4.3 Text to product matching

A different line of research [6, 8, 13] assumes that structured product specifications exist in a database and focuses on matching text offers to appropriate product specifications. This research does not apply for our product discovery task as we work on unstructured (as well as semi-structured) text listings. Moreover, in our case, the schema is known only partially in terms of category and attribute labels, not values.

5 FUTURE WORK

Due to the unorganized nature of sellers who upload product listings in a distributed fashion, e-commerce catalogs often have noisy text with spelling mistakes or spaces omitted between two keywords. While contextual spelling mistakes can be taken care of in our current approach, chunking of text by inserting spaces and other punctuations at appropriate places needs further work. A possible approach to do contextual chunking is to predict labels at character level. We are experimenting on extending our framework for such character-level label prediction. Furthermore, we wish to apply the existing framework to unstructured text on web pages such as fashion and lifestyle blogs and evaluate product discovery. Discovery of products from such text paragraphs where the context is spread at even longer distances is a challenging problem. Parsing such text may necessitate a new design of an LSTM cell, or multiple-layers of BLSTM stacked on top of each other.

6 CONCLUSIONS

In this work, we described the previously untackled problem of product discovery in e-commerce: identifying multiple items along with their associated attributes from an e-commerce test listing. Product discovery from unstructured paragraphs is important for e-commerce search engines to index product listings appropriately. We modeled product discovery as a multi-task learning problem and proposed a cascaded-CRF based neural network architecture. Our architecture captured keyword ambiguity and product-attribute attachment ambiguity that is widely prevalent e-fashion listings. We evaluated our approach on a complex dataset of 25K tagged e-fashion listings. We showed that our approach outperforms state-of-the-art approach in multi-task learning to result in 92.2% F1 score and surpasses neural architecture in prior work to solve NER for e-commerce by 13% for F1 score. We also observed that our multi-task learning framework could be trained end-to-end with only a small sample of 12K tagged text listings. To our knowledge, this is the first work to show effectiveness of neural architectures to solve product discovery on a large and practical dataset that goes beyond popular POS and NER datasets. We believe that this approach can be utilized to discover items and attributes in other domains, e.g., drug discovery from the vast amount text that is present in health-care domain. Such text especially contains multiple items each having its associated attributes.

REFERENCES

[1] Trapit Bansal, David Belanger, and Andrew McCallum. 2016. Ask the GRU: Multi-task Learning for Deep Text Recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys '16)*. ACM, New York, NY, USA, 107–114. https://doi.org/10.1145/2959100.2959180

[2] R. Collobert and J. Weston. 2008. A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning. In *International Conference on Machine Learning, ICML.*

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

[4] Pennington Jeffrey, Socher Richard, and Manning Christopher. 2014. Glove: Global vectors for word representation.. In *In Proceedings of EMNLP.*

[5] Mahesh Joshi, Ethan Hart, Mirko Vogel, and Jean-David Ruvini. 2015. Distributed Word Representations Improve NER for e-Commerce. In *Proceedings of NAACL-HLT.* 160–167.

[6] Anitha Kannan, Inmar E Givoni, Rakesh Agrawal, and Ariel Fuxman. 2011. Matching unstructured product offers to structured product specifications. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining.* ACM, 404–412.

[7] Yun Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2015. Character-aware neural language models. *Twenty-Ninth AAAI Conference on Artificial Intelligence* (2015).

[8] Hanna Köpcke, Andreas Thor, Stefan Thomas, and Erhard Rahm. 2012. Tailoring entity resolution for matching product offers. In *Proceedings of the 15th International Conference on Extending Database Technology.* ACM, 545–550.

[9] Guillaume Lample, Miguel Ballesteros, Kazuya Kawakami, Sandeep Subramanian, and Chris Dyer. 2016. Neural Architectures for Named Entity Recognition. In *In proceedings of NAACL-HLT (NAACL 2016).* San Diego, US.

[10] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Recurrent Neural Network for Text Classification with Multi-Task Learning. *CoRR* abs/1605.05101 (2016). http://arxiv.org/abs/1605.05101

[11] Xuezhe Ma and Eduard Hovy. 2016. End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers).* Association for Computational Linguistics, Berlin, Germany, 1064–1074. http://www.aclweb.org/anthology/P16-1101

[12] Karin Mauge, Khash Rohanimanesh, and Jean-David Ruvini. 2012. Structuring e-commerce inventory. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1.* Association for Computational Linguistics, 805–814.

[13] Gabor Melli. 2014. Shallow semantic parsing of product offering titles (for better automatic hyperlink insertion). In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining.* ACM, 1670–1678.

[14] Kanani Pallika, Wick Michael, and Pocock Adam. 2015. Attribute Extraction from Noisy Text Using Character-based Sequence Tagging Models. In *Machine Learning for eCommerce workshop, NIPS.*

[15] Nanyun Peng and Mark Dredze. 2016. Multi-task Multi-domain Representation Learning for Sequence Tagging. *CoRR* abs/1608.02689 (2016). http://arxiv.org/abs/1608.02689

[16] Duangmanee Pew Putthividhya and Junling Hu. 2011. Bootstrapped named entity recognition for product attribute extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing.* Association for Computational Linguistics, 1557–1567.

[17] Disheng Qiu, Luciano Barbosa, Xin Luna Dong, Yanyan Shen, and Divesh Srivastava. 2015. Dexter: large-scale discovery and extraction of product specifications on the web. *Proceedings of the VLDB Endowment* 8, 13 (2015), 2194–2205.

[18] Mikolov Tomas, Chen Kai, Corrado Greg, and Dean. Jeffrey. [n. d.]. Efficient estimation of word representations in vector space. In *Proceedings of Workshop at ICLR.*