OpenTag: Open Attribute Value Extraction from Product Profiles

Guineng Zheng[§]*, Subhabrata Mukherjee[†], Xin Luna Dong[†], Feifei Li[§]

[§]University of Utah [†]Amazon.com
{guineng, lifeifei}@cs.utah.edu, {subhomj, lunadong}@amazon.com

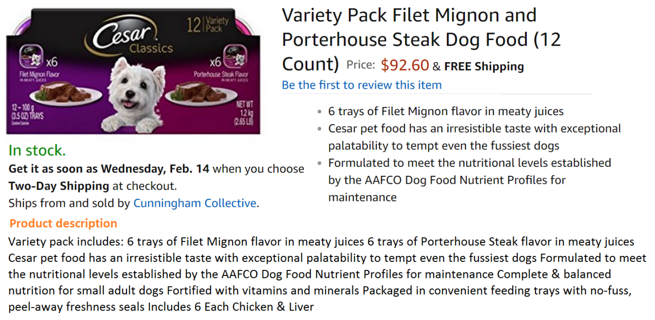
ABSTRACT

Extraction of missing attribute values is to find values describing an attribute of interest from a free text input. Most past related work on extraction of missing attribute values work with a closed world assumption with the possible set of values known beforehand, or use dictionaries of values and hand-crafted features. How can we discover new attribute values that we have never seen before? Can we do this with limited human annotation or supervision? We study this problem in the context of product catalogs that often have missing values for many attributes of interest.

In this work, we leverage product profile information such as titles and descriptions to discover missing values of product attributes. We develop a novel deep tagging model OpenTag for this extraction problem with the following contributions: (1) we formalize the problem as a sequence tagging task, and propose a joint model exploiting recurrent neural networks (specifically, bidirectional LSTM) to capture context and semantics, and Conditional Random Fields (CRF) to enforce tagging consistency; (2) we develop a novel attention mechanism to provide interpretable explanation for our model's decisions; (3) we propose a novel sampling strategy exploring active learning to reduce the burden of human annotation. OpenTag does not use any dictionary or hand-crafted features as in prior works. Extensive experiments in real-life datasets in different domains show that OpenTag with our active learning strategy discovers new attribute values from as few as 150 annotated samples (reduction in 3.3x amount of annotation effort) with a high F-score of 83%, outperforming state-of-the-art models.

ACM Reference Format:
Guineng Zheng[§]*, Subhabrata Mukherjee[†], Xin Luna Dong[†], Feifei Li[§]. 2018. OpenTag: Open Attribute Value Extraction from Product Profiles. In *KDD '18: The 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, August 19–23, 2018, London, United Kingdom.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3219819.3219839

*Work performed during internship at Amazon.



Variety Pack Filet Mignon and Porterhouse Steak Dog Food (12 Count) Price: $92.60 & **FREE Shipping**
Be the first to review this item

- 6 trays of Filet Mignon flavor in meaty juices
- Cesar pet food has an irresistible taste with exceptional palatability to tempt even the fussiest dogs
- Formulated to meet the nutritional levels established by the AAFCO Dog Food Nutrient Profiles for maintenance

In stock.
Get it as soon as Wednesday, Feb. 14 when you choose **Two-Day Shipping** at checkout.
Ships from and sold by Cunningham Collective.

Product description
Variety pack includes: 6 trays of Filet Mignon flavor in meaty juices 6 trays of Porterhouse Steak flavor in meaty juices Cesar pet food has an irresistible taste with exceptional palatability to tempt even the fussiest dogs Formulated to meet the nutritional levels established by the AAFCO Dog Food Nutrient Profiles for maintenance Complete & balanced nutrition for small adult dogs Fortified with vitamins and minerals Packaged in convenient feeding trays with no-fuss, peel-away freshness seals Includes 6 Each Chicken & Liver

Figure 1: Snapshot of a product profile.

1 INTRODUCTION

Product catalogs are a valuable resource for eCommerce retailers that allow them to organize, standardize, and publish information to customers. However, this catalog information is often noisy and incomplete with a lot of missing values for product attributes. An interesting and important challenge is to supplement the catalog with missing values for attributes of interest from product descriptions and other related product information, especially with values that we have never seen before.

Informal Problem 1. *Given a set of target attributes (e.g., brand, flavor, smell), and unstructured product profile information like titles, descriptions, and bullets: how can we extract values for the attributes from text? What if some of these values are new, like emerging brands?*

For a concrete example, refer to Figure 1 showing a snapshot of the product profile of a 'dog food' in Amazon.com with unstructured data such as title, description, and bullets. The product title "Variety Pack Fillet Mignon and Porterhouse Steak Dog Food (12 Count)" contains *two* attributes of interest namely *size* and *flavor*. We want to discover corresponding values for the attributes like "12 count" (size), "Fillet Mignon" (flavor) and "Porterhouse Steak" (flavor).

Challenges. This problem presents the following challenges.

Open World Assumption (OWA). Previous works for attribute value extraction [8, 16, 24, 25] work with a *closed world assumption* which uses a limited and pre-defined vocabulary of attribute values. Therefore, these cannot discover emerging attribute values (e.g., new brands) of newly launched products that have not been encountered before. OWA renders traditional multi-class classification techniques an unsuitable choice to model this problem.

Stacking of attributes and irregular structure. Product profile information in title and description is unstructured with tightly packed details about the product. Typically, the sellers stack several product attributes together in the title to highlight all important aspects of a product. Therefore, it is difficult to identify and segment particular attribute values — that are often multi-word phrases like "Fillet Mignon" and "Porterhouse Steak". Lack of regular grammatical structure renders NLP tools like parsers, part-of-speech (POS) taggers, and rule-based annotators [3, 18] less useful. Additionally, they also have a very sparse context. For instance, over 75% of product titles in our dataset contain fewer than 15 words whereas over 60% bullets in descriptions contain fewer than 10 words.

Limited Annotated Data. State-of-the art performance in attribute value extraction has been achieved by neural networks [11, 13, 15, 17] which are data hungry requiring several thousand annotated instances. This does not scale up with hundreds of product attributes for *every* domain, each assuming several thousand different values. This gives rise to our second problem statement.

INFORMAL PROBLEM 2. *Can we develop supervised models that require limited human annotation? Additionally, can we develop models that give intepretable explanation for its decisions, unlike black-box methods that are difficult to debug?*

Contributions. In this paper, we propose several novel techniques to address the above challenges. We formulate our problem as a sequence tagging task similar to named entity recognition (NER) [4] which has been traditionally used to identify attributes like names of persons, organizations, and locations from unstructured text.

We leverage recurrent neural networks like Long Short Term Memory Networks (LSTM) [10] to capture the *semantics and context* of attributes through distributed word representations. LSTM's are a natural fit to this problem because of their ability to handle sparse context and sequential nature of the data where different attributes and values can have inter-dependencies. Although LSTM models capture sequential nature of tokens, they overlook the sequential nature of tags. Therefore, we use another sequential model like conditional random fields (CRF) [14] to enforce tagging consistency and extract *cohesive* chunks of attribute values (e.g., multi-word phrases like 'fillet mignon') . Although state-of-the-art NER systems [11, 13, 15, 17] exploit LSTM and CRF, they essentially use them as black box techniques with no explanation. In order to address the *interpretability* challenge, we develop a novel *attention* mechanism to explain the model's decisions by highlighting importance of key concepts relative to their neighborhood context. Unlike prior works [11, 13], OpenTag does not use any dictionary or hand-crafted features.

Neural network models come with an additional challenge that they require much more annotated training data than traditional machine learning techniques because of their huge parameter space. Annotation is an expensive task. Therefore, we explore active learning to reduce the burden of human annotation. Overall, we make the following novel contributions.

- **Model:** We model attribute value extraction as a sequence tagging task that supports the Open World Assumption (OWA) and works with unstructured text and sparse contexts as in product profiles. We develop a novel model OpenTag leveraging CRF, LSTM, and an *attention* mechanism to explain its predictions.
- **Learning:** We explore active learning and novel sampling strategies to reduce the burden of human annotation.
- **Experiments:** We perform extensive experiments in real-life datasets in different domains to demonstrate OpenTag's efficacy. It discovers new attribute values from as few as 150 annotated samples (reduction in 3.3x amount of annotation effort) with a high F-score of 83%, outperforming state-of-the-art models.

To the best of our knowledge, this is the first end-to-end framework for open attribute value extraction addressing key real-world challenges for modeling, inference, and learning. OpenTag does not make any assumptions about the structure of input text and could be applied to any kind of textual data like profile pages of a given product.

The rest of the paper is organized as follows: Section 2 presents a formal description and overview where we introduce sequence tagging for open attribute value extraction. Section 3 presents a detailed description of OpenTag using LSTM, CRF, and a novel attention mechanism. We discuss active learning strategies in Section 4 followed by extensive evaluations in real-life datasets in Section 5. Lastly, Section 6 presents related work followed by conclusions.

2 OVERVIEW

2.1 Problem Definition

Given a set of product profiles presented as unstructured text data (containing information like titles, descriptions, and bullets), and a set of pre-defined target attributes (e.g., *brand*, *flavor*, *size*), our objective is to extract corresponding attribute values from unstructured text. We have an OWA assumption where we want to discover new attribute values that may not have been encountered before. Note that we assume the target attributes (and not attribute-values) for *each domain* are given as input to the system. OpenTag automatically figures out the set of applicable attribute-values for each product in the domain. For example, given the inputs,

- target attributes: *brand*, *flavor*, and *size*
- product title: "PACK OF 5 - CESAR Canine Cuisine Variety Pack Fillet Mignon and Porterhouse Steak Dog Food (12 Count)"
- product description: "Variety pack includes: 6 trays of Fillet mignon flavor in meaty juices ..."

we want to extract 'Cesar' (*brand*), 'Fillet Mignon' and 'Porterhouse Steak' (*flavor*) , and '6 trays' (*size*) as the corresponding values as output from our model. Formally,

DEFINITION: OPEN ATTRIBUTE VALUE EXTRACTION. *Given a set of products I, corresponding profiles $X = \{x_i : i \in I\}$, and a set of attributes $A = \{a_1, \ldots, a_m\}$, extract all attribute-values $V_i = \langle\{v_{i,j,1}, \ldots, v_{i,j,\ell_{i,j}}\}, a_j\rangle$ for $i \in I$ and $j \in [1, m]$ with an open world assumption (OWA); we use $v_{i,j}$ to denote* the set of values *(of size $\ell_{i,j}$) for attribute a_j for the i^{th} product, and the product profile (title, description, bullets) consists of a sequence of words/tokens $x_i = \{w_{i,1}, w_{i,2}, \cdots w_{i,n_i}\}$.*

Note that we want to discover *multiple* values for a given set of attributes. For instance, in Figure 1 the target attribute is *flavor* and it assumes *two* values 'fillet mignon' and 'porterhouse steak' for the given product 'cesar canine cuisine'.

2.2 Sequence Tagging Approach

A natural approach to cast this problem into a multi-class classification problem [16] — treating any target attribute-value as a class label — suffers from the following problems. (1) *Label scaling problem:* this method does not scale well with thousands of potential values for any given attribute and will increase the volume of annotated training data; (2) *Closed world assumption:* it cannot discover any new value outside the set of labels in the training data; (3) *Label independence assumption:* it treats each attribute-value independent of the other, thereby, ignoring any dependency between them. This is problematic as many attribute-values frequently co-occur, and the presence of one of them may indicate the presence of the other. For example, the *flavor*-attribute values 'fillet mignon' and 'porterhouse steak' often co-occur. Also, the *brand*-attribute value 'cesar' often appears together with the above *flavor*-attribute values.

Based on these observations, we propose a different approach that models this problem as a *sequence tagging task*.

2.2.1 Sequence Tagging. In order to model the above dependencies between attributes and values, we adopt the sequence tagging approach. In particular, we associate a *tag* from a given tag-set to *each token* in the input sequence. The objective is to *jointly* predict all the tags in the input sequence. In case of named entity recognition (NER), the objective is to tag *entities* like (names of) persons, locations, and organizations in the given input sequence. Our problem is a specific case of NER where we want to tag attribute values given an input sequence of tokens. The idea is to exploit *distributional semantics*, where similar sequences of tags for tokens identify similar concepts.

2.2.2 Sequence Tagging Strategies. There are several different tagging strategies, where "BIOE" is the most popular one. In BIOE tagging strategy, 'B' represents the beginning of an attribute, 'I' represents the inside of an attribute, 'O' represents the outside of an attribute, and 'E' represents the end of an attribute.

Other popular tagging strategies include "UBIOE" and "IOB". "UBIOE" has an extra tag 'U' representing the unit token tag that separates one-word attributes from multi-word ones. While for "IOB" tagging, 'E' is omitted since 'B' and 'I' are sufficient to express the boundary of an attribute.

Table 1: Tagging Strategies

Sequence	duck	,	fillet	mignon	and	ranch	raised	lamb	flavor
BIOE	B	O	B	E	O	B	I	E	O
UBIOE	U	O	B	E	O	B	I	E	O
IOB	B	O	B	I	O	B	I	I	O

Table 1 shows an example of the above tagging strategies. Given a sequence *"duck, fillet mignon and ranch raised lamb flavor"* comprising of 9 words/tokens (including the comma), the BIOE tagging strategy extracts three *flavor*-attributes "duck", "fillet mignon" and "ranch raised lamb" represented by 'B', 'BE' and "BIE" respectively.

2.2.3 Advantages of Sequence Tagging. The sequence tagging approach enjoys the following benefits. (1) *OWA and label scaling.* A tag is associated to a token, and not a specific attribute-value, and, therefore scales well with new values. (2) *Discovering multi-word attribute values.* The above strategy extracts *sequence* of tokens (i.e., multi-word values) as opposed to identifying single-word values. (3) *Discovering multiple attribute values.* The tagging strategy can be extended to discover values of multiple attributes at the same time if they are tagged differently from each other. For instance, to discover two attributes 'flavor' and 'brand' *jointly*, we can tag the given sequence with tags as 'Flavor-B', 'Flavor-I' ,'Flavor-O', and 'Flavor-E' to distinguish from 'Brand-B', 'Brand-I', 'Brand-O', and 'Brand-E'.

Formulation of our approach. Following the above discussions, we can reduce our original problem of *Open Attribute Value Extraction* to the following *Sequence Tagging Task.*

Let Y be the tag set containing all the tags decided by the tagging strategy. If we choose BIOE as our tagging strategy, then $Y = \{B, I, O, E\}$. Tag set of other strategies can be derived following a similar logic. Our objective is to learn a tagging model $F(x) \rightarrow y$ that assigns each token $w_{ij} \in W$ of the input sequence $x_i \in X$ of the i^{th} product profile with a corresponding tag $y_{ij} \in Y$. The training set for this supervised classification task is given by $\mathcal{S} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{T}$. This is a global tagging model that captures relations between tags and models the entire sequences as a whole. We denote our framework by OpenTag.

3 OPENTAG MODEL: EXTRACTION VIA SEQUENCE TAGGING

OpenTag builds upon state-of-the-art named entity recognition (NER) systems [11, 13, 15, 17] that use bidirectional LSTM and conditional random fields, but without using any dictionary or hand-crafted features as in [11, 13]. In the following section, we will first review these building blocks, and how we *adapt* them for attribute value extraction. Thereafter, we outline our novel contributions of using *Attention*, end-to-end OpenTag architecture, and active learning to reduce requirement of annotated data.

3.1 Bidirectional LSTM (BiLSTM) Model

Recurrent neural networks (RNN) capture long range dependencies between tokens in a sequence. Long Short Term Memory Networks (LSTM) were developed to address the vanishing gradient problems of RNN. A basic LSTM cell consists of various gates to control the flow of information through the LSTM connections. By construction, LSTM's are suitable for sequence tagging or classification tasks where it is insensitive to the gap length between tags unlike RNN or Hidden Markov Models (HMM).

Given an input e_t (say, the word embedding of token $x_t \in X$), an LSTM cell performs various non-linear transformations to generate a hidden vector state h_t for each token at each timestep t that can be later used for tasks such as classification.

Bidirectional LSTM's are an improvement over LSTM that capture both the previous timesteps (past features) and the future timesteps (future features) via forward and backward states respectively. In sequence tagging tasks, we often need to consider both the left and right contexts *jointly* for a better prediction model. Correspondingly, two LSTM's are used, one with the standard sequence, and the other with the sequence reversed. Correspondingly, there are two hidden states that capture past and future information that are concatenated to form the final output.

Using the hidden vector representations from *forward* and *backward* LSTM ($\overrightarrow{h_t}$ and $\overleftarrow{h_t}$ respectively) along with a non-linear transformation, we can create a new hidden vector as: $h_t = \sigma([\overrightarrow{h_t}, \overleftarrow{h_t}])$.

Finally, we add a softmax function to predict the tag for each token x_t in the input sequence $x = \langle x_t \rangle$ given hidden vector $\langle h_t \rangle$ at each timestep:

$$\Pr(y_t = k) = \text{softmax}(h_t \cdot W_h), \quad (1)$$

where W_h is the variable matrix shared across all tokens, and $k \in \{B, I, O, E\}$. For each token, the tag with the highest probability is generated as the output tag. Using the ground-labels we can train the above BiLSTM network to learn all parameters W and H using backpropagation.

Drawbacks for sequence tagging: The BiLSTM model considers *sequential nature of the given input sequence, but not the output tags.* As a result, the above model does not consider the coherency of tags during prediction. The prediction for each tag is made independent of the other tags. For example, given our set of tags $\{B, I, O, E\}$, the model may predict a mis-aligned tag sequence like $\{B, O, I, E\}$, leading to an incoherent attribute extraction. In order to avert this problem, we use Conditional Random Fields (CRF) to *also consider the sequential nature of the predicted tags.*

3.2 Tag Sequence Modeling with Conditional Random Fields and BiLSTM

3.2.1 Conditional Random Fields (CRF). For sequence labeling tasks, it is important to consider the association or correlation between labels in a neighborhood, and use this information to predict the best possible label sequence given an input sequence. For example, if we already know the starting boundary of an attribute (B), this increases the likelihood of the next token to be an intermediate (I) one or end of boundary (E), rather than being outside the scope of the attribute (O). *Conditional Random Fields (CRF) allows us to model the label sequence jointly.*

Given an input sequence $x = \{x_1, x_2, \cdots x_n\}$ and corresponding label sequence $y = \{y_1, y_2, \cdots y_n\}$, the joint probability distribution function for the CRF can be written as the conditional probability:

$$\Pr(y|x; \Psi) \propto exp\bigg(\sum_{k=1}^{K} \psi_k f_k(y, x)\bigg),$$

where $f_k(y, x)$ is the feature function, ψ_K is the corresponding weight to be learned, K is the number of features, and $\mathcal{Y}$ is the set of all possible labels. Traditional NER leverages several user defined features based on the current and previous token like the presence of determiner ('the'), presence of upper-case letter, POS tag of the current token (e.g., 'noun') and the previous (e.g., 'adjective'), etc.

Inference for general CRF is intractable with a complexity of $|\mathcal{Y}|^n$ where n is the length of the input sequence and $|\mathcal{Y}|$ is the cardinality of the label set. We use linear-chain CRF's to avoid this problem. We constrain the feature functions to depend only on the neighboring tags y_t and y_{t-1} at timestep t. This reduces the computational complexity to $|\mathcal{Y}|^2$. We can re-write the above equation as:

$$\Pr(y|x; \Psi) \propto \prod_{t=1}^{T} exp\bigg(\sum_{k=1}^{K} \psi_k f_k(y_{t-1}, y_t, x)\bigg).$$

3.2.2 Bidirectional LSTM and CRF Model. As we described above, traditional CRF models use several manually defined syntactic features for NER tasks. In this work, we combine LSTM and CRF to use semantic features like the distributed word representations. We do not use any hand-crafted features like in prior works [11, 13]. Instead, the hidden states generated by the BiLSTM model are used as input features for the CRF model. We incorporate an additional non-linear layer to weigh the hidden states that capture the importance of different states for the final tagging decision.

The BiLSTM-CRF network can use (i) features from the previous as well as future timesteps, (ii) semantic information of the given input sequence encoded in the hidden states via the BiLSTM model, and (iii) tagging consistency enforced by the CRF that captures dependency between the output tags. The objective now is to predict the best possible tag sequence of the entire input sequence given the hidden state information $\langle h_t \rangle$ as features to the CRF. The BiLSTM-CRF network forms the second component for our model.

$$\Pr(y|x; \Psi) \propto \prod_{t=1}^{T} exp\bigg(\sum_{k=1}^{K} \psi_k f_k(y_{t-1}, y_t, \langle h_t \rangle)\bigg).$$

3.3 OpenTag: Attention Mechanism

In this section, we describe our novel attention mechanism that can be used to explain the model's tagging decision unlike the prior NER systems [11, 13, 15, 17] that use BiLSTM-CRF as black-box.

In the above BiLSTM-CRF model, we consider all the hidden states generated by the BiLSTM model to be important when they are used as features for the CRF. However, not all of these states are equally important, and some mechanism to make the CRF aware of the important ones may result in a better prediction model. This is where *attention* comes into play.

The objective of the attention mechanism is to highlight important concepts, rather than focusing on all the information. Using such mechanism, we can highlight the important tokens in a given input sequence responsible for the model's predictions as well as performing feature selection. This has been widely used in the vision community to focus on a certain region of an image with "high resolution" while perceiving the surrounding image in "low resolution" and then adjusting the focal point over time.

In the Natural Language Processing domain, attention mechanism has been used with great success in Neural Machine Translation (NMT) [1]. NMT systems comprise of a sequence-to-sequence encoder and decoder. Semantics of a sentence is mapped into a fixed-length vector representation by an encoder, and then the translation is generated based on that vector by a decoder. In the original NMT model, the decoder generates a translation solely based on the last hidden state. But it is somewhat unreasonable to assume all information about a potentially very long sentence can be encoded into a single vector, and that the decoder will produce a good translation solely based on that. With an attention mechanism, instead of encoding the full source sequence into a fixed-length vector, we allow the decoder to *attend* to different parts of the source sentence at each step of the output generation. Importantly, we let the model learn what to attend to based on the input sentence and what it has produced so far.

We follow a similar idea. In our setting, the encoder is the underlying BiLSTM model generating the hidden state representation $\langle h_t \rangle$. We introduce an attention layer with an attention matrix A to capture the similarity of any token with respect to all the neighboring tokens in an input sequence. The element $\alpha_{t,t'} \in A$ captures the similarity between the hidden state representations h_t and $h_{t'}$ of tokens x_t and $x_{t'}$ at timesteps t and t' respectively. The attention mechanism is implemented similar to an LSTM cell as follows:

$$g_{t,t'} = tanh(W_g h_t + W_{g'} h_{t'} + b_g), \tag{2}$$

$$\alpha_{t,t'} = \sigma(W_a g_{t,t'} + b_a), \tag{3}$$

where, σ is the element-wise sigmoid function, W_g and $W_{g'}$ are the weight matrices corresponding to the hidden states h_t and $h_{t'}$; W_a is the weight matrix corresponding to their non-linear combination; b_g and b_a are the bias vectors.

The attention-focused hidden state representation l_t of a token at timestep t is given by the weighted summation of the hidden state representation $h_{t'}$ of all other tokens at timesteps t', and their similarity $\alpha_{t,t'}$ to the hidden state representation h_t of the current token. Essentially, l_t dictates how much to *attend* to a token at any timestep *conditioned on their neighborhood context*. This can be used to highlight the model's final tagging decision based on token importance.

$$l_t = \sum_{t'=1}^{n} \alpha_{t,t'} \cdot h_{t'}. \tag{4}$$

In Section 5.4, we discuss how OpenTag generates interpretable explanations of its tagging decision using this attention matrix.

3.4 Word Embeddings

Neural word embeddings map words that co-occur in a similar context to nearby points in the embedding space [19]. This forms the first layer of our architecture. Compared to bag-of-words (BOW) features, word embeddings capture both syntactic and semantic information with low-dimensional and dense word representations. The most popular tools for this purpose are Word2Vec [19] and GloVe [22], which are trained over large unlabeled corpus. Pre-trained embeddings have a single representation for each token. This does not serve our purpose as the same word can have a different representation in different contexts. For instance, 'duck' (bird) as a *flavor*-attribute value should have a different representation than 'duck' as a *brand*-attribute value. Therefore, we learn the word representations conditioned on the attribute tag (e.g., 'flavor'), and generate different representations for different attributes. In our setting, each token at time t is associated with a vector $e_t \in R^d$, where d is the embedding dimension. The elements in the vector are latent, and considered as parameters to be learned.

3.5 OpenTag Architecture: Putting All Together

Figure 2 shows the overall architecture of OpenTag. The first layer is the word embedding layer that generates an embedding vector e_t for each token x_t in the input sequence x. This vector is used as an input to the bidirectional LSTM layer that generates its hidden state representation h_t as a concatenation of the forward and backward LSTM states. This representation captures its future and previous timestep features.

The output of BiLSTM goes as input to the attention layer that learns which states to focus or *attend* to in particular — generating the attention-focused hidden state representation $\langle l_t \rangle$ for the input

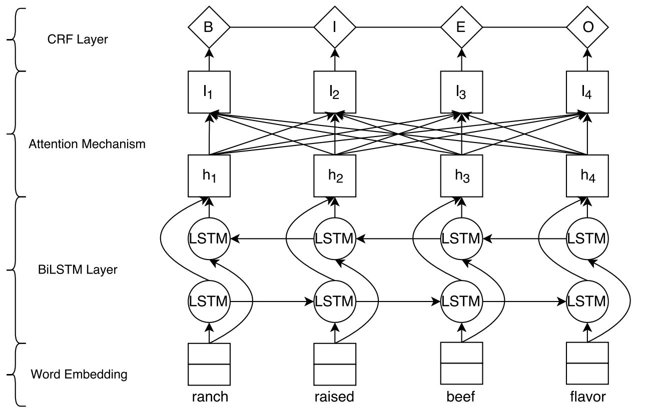


Figure 2: OpenTag **Architecture: BiLSTM-CRF with Attention.**

sequence $\langle x_t \rangle$. These representations are used as input features in the CRF that enforces tagging consistency — considering dependency between output tags and the hidden state representation of tokens at each timestep. The joint probability distribution of the tag sequence is given by:

$$\Pr(y|x;\Psi) \propto \prod_{t=1}^{T} exp\Bigg(\sum_{k=1}^{K} \psi_k f_k(y_{t-1}, y_t, \langle l_t \rangle)\Bigg). \tag{5}$$

For training this network, we use the maximum conditional likelihood estimation, where we maximize the log-likelihood of the above joint distribution with respect to all the parameters Ψ over m training instances $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^m$:

$$L(\Psi) = \sum_{i=1}^{m} \log \Pr(\mathbf{y}_i|\mathbf{x}_i;\Psi). \tag{6}$$

The final output is the best possible tag sequence y^* with the highest conditional probability given by:

$$y^* = \text{argmax}_y \Pr(y|x;\Psi). \tag{7}$$

4 OPENTAG: ACTIVE LEARNING

In this section, we present our novel active learning framework for OpenTag to reduce the burden of human annotation.

An essential requirement of supervised machine learning algorithms is annotated data. However, manual annotation is expensive and time consuming. In many scenarios, we have access to a lot of unlabeled data. Active learning is useful in these scenarios, where we can allow the learner to select samples from the un-labeled pool of data, and request for labeling.

Starting with a *small set of labeled instances* as an initial training set L, the learner iteratively requests labels for one or more instances from a *large unlabeled pool* of instances U using some query strategy Q. These instances are labeled, and added to the base set L, and the process is repeated till some stopping criterion is reached. The challenge is to design a good query strategy Q that selects the most informative samples from U given the learner's hypothesis space. This aims to improve the learner's performance with *as little annotation effort as possible*. This is particularly useful for sequence labeling tasks, where the annotation effort is proportional to the length of a sequence, in contrast to instance classification

Table 2: Sampling Strategies. LC: Least confidence. TF: Tag flip.

	duck	,	fillet	mignon	and	ranch	raised	lamb	flavor
Gold-Label (G)	B	O	B	E	O	B	I	E	O
Strategy: LC (S1)	O	O	B	E	O	B	I	E	O
Strategy: TF (S2)	B	O	B	O	O	O	O	B	O

tasks. OpenTag employs active learning with a similar objective to reduce manual annotation efforts, while making judicious use of the large number of unlabeled product profiles. There are many different approaches to formulate a query strategy to select the most informative instances to improve the active learner.

As our baseline strategy, we consider the method of least confidence (LC) [6] which is shown to perform quite well in practice [28]. It selects the sample for which the classifier is least confident. In our sequence tagging task, the confidence of the CRF in tagging an input sequence is given by the conditional probability in Equation 5. Therefore, the query strategy selects the sample x with maximum uncertainty given by:

$$Q^{lc}(x) = 1 - \Pr(y^*|x; \Psi), \tag{8}$$

where y^* is the best possible tag sequence for x.

However, this strategy has the following drawbacks: (1) The conditional probability of the entire sequence is proportional to the product of (potential of) successive tag ($\langle y_{t-1}, y_t \rangle$) transition scores. Therefore, a false certainty about any token's tag y_t can pull down the probability of the entire sequence — leading to missing a valuable query. (2) When the oracle reveals the tag of a token, this may impact only a few other tags, having a relatively low impact on the entire sequence.

4.1 Method of Tag Flips

In order to address these limitations, we formulate a new query strategy to identify informative sequences *based on how difficult it is to assign tags to various tokens in a sequence.*

In this setting, we simulate a committee of OpenTag learners $C = \{\Psi^{(1)}, \Psi^{(2)}, \cdots \Psi^{(E)}\}$ to represent different hypotheses that are consistent with labeled set L. The most informative sample is the one for which there is major disagreement among committee members.

We train OpenTag for a preset number of epochs E using dropout [29] regularization technique. Dropout prevents overfiting during network training by randomly dropping units in the network with their connections. Therefore, for each epoch e, OpenTag learns a different set of models and parameters $\Psi^{(e)}$ — thereby simulating a committee of learners due to the dropout mechanism.

After each epoch, we apply $\Psi^{(e)}$ to the unlabeled pool of samples and record the best possible tag sequence $y^*(\Psi^{(e)})$ assigned by the learner to each sample.

We define a *flip* to be a change in the tag of a token of a given sequence across *successive* epochs, i.e., the learners $\Psi^{(e-1)}$ and $\Psi^{(e)}$ assign different tags $y^*_t(\Psi^{(e-1)}) \neq y^*_t(\Psi^{(e)})$ to token $x_t \in x$. If the tokens of a given sample sequence frequently change tags across successive epochs, this indicates OpenTag is uncertain about the sample, and *not stable.* Therefore, we consider tag flips (TF) to be a measure of uncertainty for a sample and model stability, and *query for labels for the samples with the highest number of tag flips.*

Illustrative example. Consider the snippet in Table 2 and the tag sequences assigned by two different sampling strategies $S1$ and $S2$ corresponding to least confidence and tag flip respectively. Assume that the gold-label sequence (G) is known. In practice, we do *not* use the ground labels for computing tag-flips during learning; instead we use predictions from OpenTag from the previous epoch.

Algorithm 1: Active learning with tag flips as query strategy.

```
Given: Labeled set $L$, unlabeled pool $U$, query strategy $Q$, query batch
  size $B$
repeat
    for each epoch $e \in E$ do
        // simulate a committee of learners using current $L$
        $\Psi^{(e)}$ = train($L$)
        Apply $\Psi^{(e)}$ to unlabeled pool $U$ and record tag flips
    for each query $b \in B$ do
        // find the instances with most tag flips over $E$ epochs
        $x^* = \text{argmax}_{x \in U}\, Q^{tf}(x)$
        // label query and move from unlabeled pool to labeled set
        $L = L \cup \{x^*, label(x^*)\}$
        $U = U - x^*$
until some stopping criterion
```

Contrasting the tag sequence $S2$ with the gold sequence G, we observe 4 flips corresponding to mismatch in tags of 'mignon' and 'ranch raise lamb'.

Given an unlabeled pool of instances, the strategy for least confidence may pick sequence $S1$ that the learner is most uncertain about in terms of the overall probability of the entire sequence. We observe this may be due to mis-classifying 'duck' that is an important concept at the start of the sequence. However, the learner gets the remaining tags correct. In this case, if the oracle assigns the tag for 'duck', it does not affect any other tags of the sequence. Therefore, this is not an informative query to be given to the oracle for labeling.

On the other hand, the tag flip strategy selects sequence $S2$ based on the number of flips of token-tags that the model has grossly mis-tagged. Labeling this query has much more impact on the learner to tune its parameters than the other sequence $S1$.

The flip based sampling strategy is given by:

$$Q^{tf}(x) = \sum_{e=1}^{E} \sum_{t=1}^{n} \mathcal{I}(y^*_t(\Psi^{(e-1)}) \neq y^*_t(\Psi^{(e)})), \tag{9}$$

where $y^*_t(\Psi^{(e)})$ is the best possible tag sequence for x assigned by the learner $\Psi^{(e)}$ in epoch e and $\mathcal{I}(\cdot)$ is an indicator function that assumes the value 1 when the argument is true, and 0 otherwise. Q^{tf} computes the number of tag flips of tokens of a sequence across successive epochs.

Algorithm 1 outlines our active learning process. The batch-size indicates how many samples we want to query for labels. Given a batch-size of B, the top B samples with the highest number of flips are manually annotated with tags. We continue the active learning process until the validation loss converges within a threshold.

Table 3: Data sets.

Domain	Profile	Attribute	Training		Testing	
			Samples	Extractions	Samples	Extractions
Dog Food (DS)	Title	Flavor	470	876	493	602
Dog Food	Title	Flavor	470	716	493	762
	Desc	Flavor	450	569	377	354
	Bullet	Flavor	800	1481	627	1179
	Title	Brand	470	480	497	607
	Title	Capacity	470	428	497	433
	Title	Multi	470	1775	497	1632
Camera	Title	Brand	210	210	211	211
Detergent	Title	Scent	500	487	500	484

5 EXPERIMENTS

5.1 OpenTag: Training

We implemented OpenTag using Tensorflow, where some basic layers are brought from Keras.[1]. We run our experiments within docker containers on a 72-core machine powered by Ubuntu Linux.

We use 100-dimensional pre-trained word vectors from GloVe [22] for initializing our word embeddings that are optimized during training. Embeddings for words not in GloVe are randomly initialized and re-trained. Masking is adopted to support variable length input. We set the hidden size of LSTM to 100, which generates a 200 dimensional output vector for BiLSTM after concatenation. The dropout rate is set to 0.4. We use Adam [12] for parameter optimization with a batch size of 32. We train the models for 500 epochs, and report the averaged evaluation measures for the last 20 epochs.

5.2 Data Sets

We perform experiments in 3 domains, namely, (i) dog food, (ii) detergents, and (iii) camera. For each domain, we use the product profiles (like titles, descriptions, and bullets) from `Amazon.com` public pages. The set of applicable attributes are defined per-domain. For each product in a domain, OpenTag figures out the set of applicable attribute values. We perform experiments with different configurations to validate the robustness of our model.

Table 3 gives the description of different data sets and experimental settings. It shows the (i) domain, (ii) type of profile, (iii) target attribute, (iv) number of samples or products we consider, and (v) the number of extractions in terms of attribute values. 'Desc' denotes description whereas 'Multi' refers to multiple attributes (e.g., flavor, capacity, and brand). 'DS' represents a disjoint training and test set with no overlapping attribute values; for all other data sets, we randomly split them into training and test instances.

Evaluation measure. We evaluate the *precision*, *recall*, and *f-score* of all models. In contrast to prior works evaluating tag-level measures — we evaluate extraction quality of our model with either full or no credit. In other words, given a target *flavor*-extraction of "ranch raised lamb", a model gets credit *only* when it extracts the full sequence. After a model assigns the best possible tag decision, attribute values are extracted and compared with ground truth.

5.3 Performance: Attribute Value Extraction

Baselines. The first baseline we consider is the BiLSTM model [10]. The second one is the state-of-the-art sequence tagging model for named entity recognition (NER) tasks using BiLSTM and CRF [11, 13, 15, 17] but without using any dictionary or hand-crafted features as in [11, 13] . We adopt these models for attribute value extraction. Training and test data are the same for all the models.

[1]Note that we do not do any hyper-parameter tuning. Most default parameter values come from Keras. It may be possible to boost OpenTag performance by careful tuning.

Table 4: Performance comparison of different models on attribute value extraction for different product profiles and datasets. OpenTag outperforms other state-of-the-art NER systems [11, 13, 15, 17] based on BiLSTM-CRF.

Datasets/Attribute	Models	Precision	Recall	Fscore
Dog Food: Title	BiLSTM	83.5	85.4	84.5
Attribute: Flavor	BiLSTM-CRF	83.8	85.0	84.4
	OpenTag	**86.6**	**85.9**	**86.3**
Camera: Title	BiLSTM	94.7	88.8	91.8
Attribute: Brand	BiLSTM-CRF	91.9	**93.8**	92.9
	OpenTag	**94.9**	93.4	**94.1**
Detergent: Title	BiLSTM	81.3	82.2	81.7
Attribute: Scent	BiLSTM-CRF	**85.1**	82.6	83.8
	OpenTag	84.5	**88.2**	**86.4**
Dog Food: Description	BiLSTM	57.3	58.6	58
Attribute: Flavor	BiLSTM-CRF	62.4	51.5	56.9
	OpenTag	**64.2**	**60.2**	**62.2**
Dog Food: Bullet	BiLSTM	93.2	94.2	93.7
Attribute: Flavor	BiLSTM-CRF	94.3	94.6	94.5
	OpenTag	**95.7**	**95.7**	**95.7**
Dog Food: Title	BiLSTM	71.2	67.4	69.3
Multi Attribute:	BiLSTM-CRF	72.9	67.3	70.1
Brand, Flavor, Capacity	OpenTag	**76.0**	**68.1**	**72.1**

Table 5: OpenTag results on disjoint split; where it discovers new attribute values never seen before with 82.4% **f-score.**

Train-Test Framework	Precision	Recall	F-score
Disjoint Split (DS)	83.6	81.2	82.4
Random Split	86.6	85.9	86.3

Tagging strategy. Similar to previous works, we also adopt $\{B, I, O, E\}$ tagging strategy. We experimented with other tagging strategies, where $\{B, I, O, E\}$ performed marginally better than the others.

Attribute value extraction results. We compare the performance of OpenTag with the aforementioned baselines for identifying attribute values from different product profiles (like title, description and bullets) and different sets of attributes (like brand, flavor and capacity) in different domains (like dog food, detergent and camera). Table 4 summarizes the results, where all experiments were performed on random train-test splits. The first column in the table shows the domain, profile type, and attribute we are interested in. We observe that OpenTag consistently outperforms competing methods with a high overall f-score of 82.8%.

We also observe the highest performance improvement (5.3%) of OpenTag over state-of-the-art BiLSTM-CRF model on product descriptions, which are more structured and provide more context than either titles or bullets. However, overall performance of OpenTag for product descriptions is much worse. Although descriptions are richer in context, the information present is also quite diverse in contrast to titles or bullets that are short, crisp and focused.

Table 6: OpenTag has improved performance on extracting values of multiple attributes jointly vs. single extraction.

Attribute	Precision	Recall	F-Score
Brand: Single	52.6	42.6	47.1
Brand: Multi	**58.4**	**44.7**	**50.6**
Flavor: Single	83.6	**81.2**	**82.4**
Flavor: Multi	**83.7**	77.5	80.5
Capacity: Single	81.5	86.4	83.9
Capacity: Multi	**87.0**	**87.2**	**87.1**

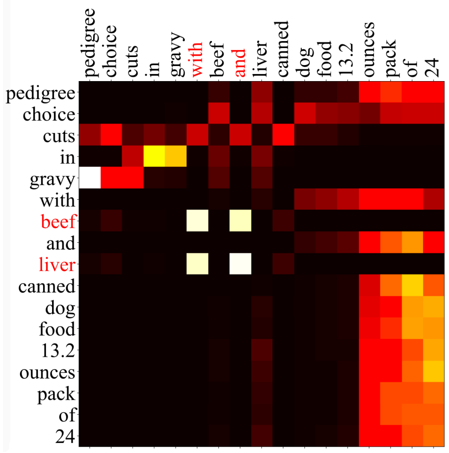


Figure 3: OpenTag **shows interpretable explanation for its tagging decision as shown by this heat map of *learned* attention matrix A for a product title. Each map element highlights importance (indicated by light color) of a word with respect to neighboring context.**

Discovering new attribute values with open world assumption (OWA). In this experiment (see Table 5), we want to find the performance of OpenTag in discovering new attribute values it has *never seen* before. Therefore, we make a clear separation between training and test data such that they do not share any attribute value. Compared with the earlier random split setting, OpenTag still performs well in the disjoint setting with a f-score of 82.4% in discovering new attribute values for *flavors* from dog food titles. However, it is worse than random split – where it has the chance to see some attribute values during training, leading to better learning.

Joint extraction of multi-attribute values. As we discussed in Section 2.2.2, OpenTag is able to extract values of multiple attributes jointly by modifying the tagging strategy. In this experiment, we extract values for *brand, flavor* and *capacity* from titles of dogfood data *jointly* on a disjoint split of data. Using $\{B, I, O, E\}$ as the tagging strategy, each attribute a has its own $\{B_a, I_a, E_a\}$ tag with O shared among them – with a total of 10 tags for three attributes. From Table 4, we observe OpenTag to have a 2% f-score improvement over our strongest BiLSTM-CRF baseline.

As we previously argued, *joint* extraction of multiple attributes can help leverage their distributional semantics together, thereby, improving the extraction of *individual* ones as shown in Table 6. Although the performance in extracting *brand* and *capacity* values improve in the joint setting, the one for *flavor* marginally degrades.

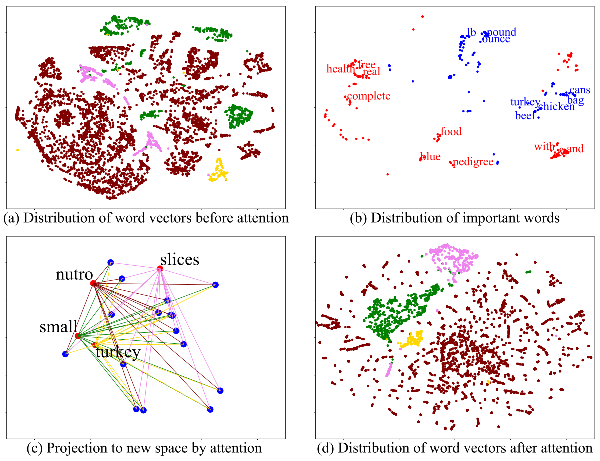


Figure 4: Sub-figures (in order) show how OpenTag **uses *attention* to cluster concepts and tags similar to each other in embedding space. Color scheme for tags. B: Green, I: Gold, O: Maroon, E: Violet.**

5.4 OpenTag: Interpretability via Attention

Interpretable explanation using attention. Figure 3 shows the heat map of the attention matrix A — as learned by OpenTag during training — of a product title. Each element of the heat map highlights the importance (represented by a lighter color) of a word with respect to its neighboring context, and, therefore how it affects the tagging decision. To give an example, consider the four white boxes located in the center of the figure. They demonstrate that the two corresponding words "with" and "and" (in columns) are important for deciding the tags of the tokens "beef" and "liver" (in rows) which are potential values for the target *flavor*-attribute. It makes sense since these are conjunctions connecting two neighboring *flavor* segments. This concrete example shows that our model has learned the semantics of conjunctions and their importance for attribute value extraction. This is interesting since we do not use any part-of-speech tag, parsing, or rule-based annotation as commonly used in NER tasks.

OpenTag **achieves better concept clustering.** The following discussions refer to Figure 4. The sub-figures in order show how attention operates on the input data to generate better concept clustering. The corresponding experiments were performed on extracting flavors from dog food titles. For visualizing high-dimensional word embeddings in a 2-d plane, we use t-SNE to reduce the dimensionality of the BiLSTM hidden vector (of size 200) to 2.

Figure 4 (a) shows the distribution of word embeddings *before* they are operated by *attention* — where each dot represents a word token and its color represents a tag ($\{B, I, O, E\}$). We use four different colors to distinguish the tags. We observe that words with different tags are initially spread out in the embedding space.

We calculate two importance measures for each word by aggregating corresponding attention weights: (i) its importance to the *attribute words* (that assume any of the $\{B, I, E\}$ tags for tokens located within an attribute-value), and (ii) its importance to the *outer words* (that assume the tag O for tokens located outside the attribute-value). For each measure, we sample top 200 important words and plot them in Figure 4 (b). We observe that all semantically related words are located close to each other. For instance, conjunctions like “with”, “and” and “&” are located together in the right bottom; whereas quantifiers like “pound”, “ounce” and “lb” are located together in the top.

We observe that the red dots — representing the most important words to the *attribute words* — are placed at the boundary of the embedding space by the attention mechanism. It indicates that the attention mechanism is smart enough to make use of the most distinguishable words located in the boundary. On the other hand, the blue dots — denoting the most important words to the *outer words* — are clustered within the vector space. We observe that quantifiers like “pound", “ounce" and “lb" help to locate the outer words. It is also interesting that attribute words like “turkey", “chicken" and “beef” that are extracted as values of the *flavor*-attribute assume an important role in tagging outer words, where these tokens typically signal the boundary of the attribute.

Figure 4 (c) shows how attention mechanism projects the hidden vectors into a new space. Consider the following example: “nutro natural choice small breed turkey slices canned dog food, 3.5 oz. by nutro", where “small breed turkey slices" is the extracted value of the *flavor*-attribute. Each blue dot in the figure represents a word of the example in the original hidden space. Red dots denote the word being projected into a new space by the attention mechanism. Again, we observe that similar concepts (red dots corresponding to four sample words) come closer to each other *after* projection.

Figure 4 (d) shows the distribution of word vectors *after* being operated by the attention mechanism. Comparing this with Figure 4 (a), we observe that similar concepts (tags) now show a better grouping and separability from different ones after using attention.

5.5 OpenTag **with Active Learning: Results**

5.5.1 Active Learning with Held-Out Test Set. In order to have a strict evaluation for the active learning framework: we use a blind held-out test set H that OpenTag cannot access during training. The original test set in Table 3 is randomly split into unlabeled pool U and held-out test set H with the ratio of $2:1$. We start with a very small number of labeled instances, namely 50 randomly sampled instances, as our initial labeled set L. We employ 20 rounds of active learning for this experiment. Figure 5 shows the results on two tasks: (i) extracting values for *scent*-attribute from titles of detergent products, and (ii) extracting values for multiple attributes *brand, capacity* and *flavor* from titles of dog food products.

OpenTag with tag flip sampling strategy for single attribute value extraction improves the precision from 59.5% (on our initial labeled set of 50 instances) to 91.7% and recall from 70.7% to 91.5%. This is also better than the results reported in Table 4, where OpenTag obtained 84.5% precision and 88.2% recall – trained on the entire data set. Similar results hold true for multi-attribute value extraction.

We also observe that the tag flip strategy (TF) outperforms least confidence (LC) [6] strategy by 5.6% in f-score for single attribute value extraction and by 2.2% for multi-attribute value extraction.

5.5.2 Active Learning without Held-out Data. Next we explore to what extent can active learning reduce the burden of human annotation. As before, we start with a very small number (50) of labeled instances as initial training set L. We want to find: how many

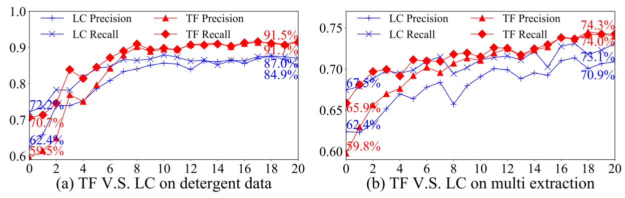


Figure 5: OpenTag **active learning results on held-out test set.** OpenTag ***with* tag flip (TF) outperforms least confidence (LC) [6] strategy, as well as** OpenTag ***without* active learning. X-axis shows the number of epochs; Y-axis shows corresponding precision and recall values.**

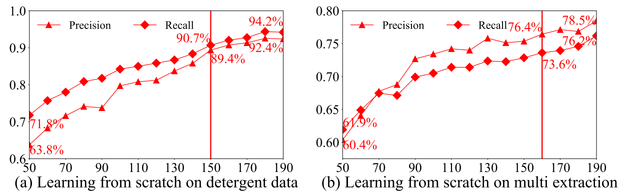


Figure 6: Results of active learning with tag flip. OpenTag **reduces burden of human annotation by** $3.3x$**. X-axis shows number of annotations; Y-axis shows corresponding precision and recall values.**

rounds of active learning are required to match the performance of OpenTag as in original training data of 500 labeled instances. In contrast to the previous setting with a held-out test set, in this experiment OpenTag can access all of the unlabeled data to query for their labels. Figure 6 shows its results.

For this setting, we use the best performing query strategy from the last section using tag flips (TF). We achieve almost the same level of performance with only 150 training instances that we had initially obtained with 500 training instances in the previous section. Figure 6 (b) shows a similar result for the second task as well. This shows that OpenTag with TF query strategy for active learning can drastically cut down on the requirement of labeled training data.

6 RELATED WORK

Rule-based extraction techniques [21] make use of domain-specific vocabulary or dictionary to spot key phrases and attributes. These suffer from limited coverage and closed world assumptions. Similarly, rule-based and linguistic approaches [3, 18] leveraging syntactic structure of sentences to extract dependency relations do not work well on irregular structures like titles.

An NER system was built [25] to annotate brands in product listings of apparel products. Comparing results of SVM, MaxEnt, and CRF, they found CRF to perform the best. They used seed dictionaries containing over 6,000 known brands for bootstrapping. A similar NER system was built [20] to tag brands in product titles leveraging existing brand values. In contrast to these, we do not use any dictionaries for bootstrapping, and can discover new values.

There has been quite a few works on applying neural networks for sequence tagging. A multi-label multi-class Perceptron classifier for NER is used by [16]. They used linear chain CRF to segment text with BIO tagging. An LSTM-CRF model is used [13] for product attribute tagging for brands and models with a lot of hand-crafted features. They used 37,000 manually labeled search queries to train their model. In contrast, OpenTag does not use hand-crafted features, and uses active learning to reduce burden of annotation.

Early attempts include [9, 23], which apply feed-forward neural networks (FFNN) and LSTM to NER tasks. Collobert et al. [5] combine deep FFNN and word embedding [19] to explore many NLP tasks including POS tagging, chunking and NER. Character-level CNNs were integrated [26] to augment feature representation, and their model was later enhanced by LSTM [4]. Huang et al. [11] adopts CRF with BiLSTM for jointly modeling sequence tagging. However they use heavy feature engineering. Lample et al. [15] use BiLSTM to encode both character-level and word-level feature, thus constructing an end-to-end BiLSTM-CRF solution for sequence tagging. Ma et al. [17] replace the character-level model with CNNs. Currently, BiLSTM-CRF models as above is state-of-the-art for NER. Unlike prior works, OpenTag uses attention to improve feature representation and gives interpretable explanation of its decisions. Bahdanau et al. [1] successfully applied attention for alignment in NMT systems. Similar mechanisms have recently been applied in other NLP tasks like machine reading and parsing [2, 30].

Early active learning for sequence labeling research [7, 27] employ least confidence (LC) sampling strategies. Settles and Craven made a thorough analysis over other strategies and proposed their entropy based strategies in [28]. However, the sampling strategy of OpenTag is different from them.

7 CONCLUSIONS

We presented OpenTag — an end-to-end tagging model leveraging BiLSTM, CRF and Attention — for imputation of missing attribute values from product profile information like titles, descriptions and bullets. OpenTag does not use any dictionary or hand-crafted features for learning. It also does not make any assumption about the structure of the input data, and, therefore, could be applied to any kind of textual data. The other advantages of OpenTag are: (1) *Open World Assumption (OWA):* It can discover new attribute values (e.g., emerging brands) that it has never seen before, as well as multi-word attribute values and multiple attributes. (2) *Irregular structure and sparse context:* It can handle unstructured text like profile information that lacks regular grammatical structure with stacking of several attributes, and a sparse context. (3) *Limited annotated data:* Unlike other supervised models and neural networks, OpenTag requires less training data. It exploits *active learning* to reduce the burden of human annotation. (4) *Interpretability:* OpenTag exploits an *attention* mechanism to generate explanations for its verdicts that makes it easier to debug. We presented experiments in real-life datasets in different domains where OpenTag discovers *new* attribute values from as few as 150 annotated samples (reduction in 3.3x amount of annotation effort) with a high F-score of 83%, outperforming state-of-the-art models.

ACKNOWLEDGMENTS

Guineng Zheng and Feifei Li were partially supported by NSF grants 1443046 and 1619287. Feifei Li was also supported in part by NSFC grant 61428204. The authors would like to sincerely thank Christos Faloutsos, Kevin Small and Ron Benson for their insightful and constructive comments on the paper, and Arash Einolghozati for his contributions to the project. We would also like to thank the following students for helping with annotation: Min Du, Neeka Ebrahimi, Richie Frost, Dyllon Gagnier, Debjyoti Paul, Yuwei Wang, Zhuoyue Zhao, Dong Xie, Yanqing Peng, Ya Gao, Madeline MacDonald.

REFERENCES

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural Machine Translation by Jointly Learning to Align and Translate. *CoRR* abs/1409.0473 (2014).

[2] Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733* (2016).

[3] Laura Chiticariu, Rajasekar Krishnamurthy, Yunyao Li, Frederick Reiss, and Shivakumar Vaithyanathan. 2010. Domain Adaptation of Rule-based Annotators for Named-entity Recognition Tasks *(EMNLP '10)*. 1002–1012.

[4] Jason PC Chiu and Eric Nichols. 2015. Named entity recognition with bidirectional LSTM-CNNs. *arXiv preprint arXiv:1511.08308* (2015).

[5] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *JMLR* 12, Aug (2011), 2493–2537.

[6] Aron Culotta and Andrew McCallum. 2005. Reducing Labeling Effort for Structured Prediction Tasks *(AAAI'05)*. 746–751.

[7] Aron Culotta and Andrew McCallum. 2005. Reducing labeling effort for structured prediction tasks. In *AAAI*, Vol. 5. 746–751.

[8] Rayid Ghani, Katharina Probst, Yan Liu, Marko Krema, and Andrew Fano. 2006. Text Mining for Product Attribute Extraction. *SIGKDD Explor. Newsl.* (2006).

[9] James Hammerton. 2003. Named entity recognition with long short-term memory *(HLT-NAACL '03)*. 172–175.

[10] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[11] Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF Models for Sequence Tagging. *CoRR* abs/1508.01991 (2015).

[12] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980 (2014).

[13] Zornitsa Kozareva, Qi Li, Ke Zhai, and Weiwei Guo. 2016. Recognizing Salient Entities in Shopping Queries *(ACL '16)*. 107–111.

[14] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data *(ICML '01)*. 282–289.

[15] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural Architectures for Named Entity Recognition.. In *HLT-NAACL*. 260–270.

[16] Xiao Ling and Daniel S. Weld. 2012. Fine-grained Entity Recognition *(AAAI'12)*.

[17] Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. *arXiv preprint arXiv:1603.01354* (2016).

[18] Andrei Mikheev, Marc Moens, and Claire Grover. 1999. Named Entity Recognition Without Gazetteers *(EACL '99)*. 1–8.

[19] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and Their Compositionality *(NIPS'13)*. 3111–3119.

[20] Ajinkya More. 2016. Attribute Extraction from Product Titles in eCommerce. *CoRR* abs/1608.04670 (2016).

[21] David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification. *Linguisticae Investigationes* 30, 1 (2007), 3–26.

[22] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation *(EMNLP '14)*. 1532–1543.

[23] G Petasis, S Petridis, G Paliouras, V Karkaletsis, SJ Perantonis, and CD Spyropoulos. 2000. Symbolic and neural learning for named-entity recognition. In *Proceedings of the Symposium on Computational Intelligence and Learning*. 58–66.

[24] Petar Petrovski and Christian Bizer. 2017. Extracting Attribute-value Pairs from Product Specifications on the Web *(WI '17)*. 558–565.

[25] Duangmanee (Pew) Putthividhya and Junling Hu. 2011. Bootstrapped Named Entity Recognition for Product Attribute Extraction *(EMNLP '11)*. 1557–1567.

[26] Cicero Santos and Victor Guimaraes. 2015. Boosting named entity recognition with neural character embeddings. *arXiv preprint arXiv:1505.05008* (2015).

[27] Tobias Scheffer, Christian Decomain, and Stefan Wrobel. 2001. Active hidden markov models for information extraction. In *ISIDA*. Springer, 309–318.

[28] Burr Settles and Mark Craven. 2008. An Analysis of Active Learning Strategies for Sequence Labeling Tasks *(EMNLP '08)*. 1070–1079.

[29] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *JMLR* 15 (2014), 1929–1958.

[30] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, and Kaiser. 2017. Attention is all you need. In *NIPS*.