# Predicting Latent Structured Intents from Shopping Queries

Chao-Yuan Wu
Computer Science, UT Austin
cywu@cs.utexas.edu

Amr Ahmed
Google
amra@google.com

Gowtham Ramani Kumar
Google
gowthamku@google.com

Ritendra Datta
Google
ritendra@google.com

## ABSTRACT

In online shopping, users usually express their intent through search queries. However, these queries are often ambiguous. For example, it is more likely (and easier) for users to write a query like *"high-end bike"* than *"21 speed carbon frames jamis or giant road bike"*. It is challenging to interpret these ambiguous queries and thus search result accuracy suffers. A user oftentimes needs to go through the frustrating process of refining search queries or self-teaching from possibly unstructured information. However, shopping is indeed a structured domain, that is composed of category hierarchy, brands, product lines, features, etc. It would be much better if a shopping site could understand users' intent through this structure, present organized information, and then find the items with the right categories, brands or features.

In this paper we study the problem of inferring the latent intent from unstructured queries and mapping them to structured attributes. We present a novel framework that jointly learns this knowledge from user consumption behaviors and product metadata. We present a hybrid Long Short-term Memory (LSTM) [10] joint model that is accurate and robust, even though user queries are noisy and product catalog is rapidly growing. Our study is conducted on a large-scale dataset from Google Shopping, that is composed of millions of items and user queries along with their click responses. Extensive qualitative and quantitative evaluation shows that the proposed model is more accurate, concise, and robust than multiple possible alternatives. In terms of information retrieval (IR) performance, our model is able to improve the quality of current Google Shopping production system, which is a very strong baseline.

## 1. INTRODUCTION

Shopping is a highly structured domain where category hierarchy, brands, merchants, product lines, styles, features, etc. form a structure among items. It is through this structure that users learn, reason and make their purchase decisions. For an online shopping site, it is paramount to understand a user's intent through these structured attributes. The reason is twofold. First, it allows us to present information in a way that aligns with how users understand, reason, and make decisions, and thus ease the shopping process. For instance, given a search query *"cheap sofa"*, it is important to tell the users what brands, features, or textures are cheaper, instead of just finding sofas that have *"cheap"* in their titles —this will greatly ease the process of research and decision making. Knowledge Graph is used by many search engines to present search results as structured information and the same motivation also applies for online shopping. Secondly, and most importantly, if we can identify relevant item attributes for a given query, we can find more relevant results and thus reduce user's frustration by minimizing the number of times he/she needs to refine the search queries.

On the other hand, user queries are highly unstructured. First, they can ambiguous. For example, the meaning of *"birthday gift for daughter"*, *"camping equipments"*, or *"affordable mattress"* depends on context and differs from user to user. In addition, a query usually does not form a complete sentence, but just a list of key words in arbitrary order. Queries also often contain typos, and can be expressed in user-dependent ways. Modeling these unstructured and noisy queries in a robust and accurate way is challenging.

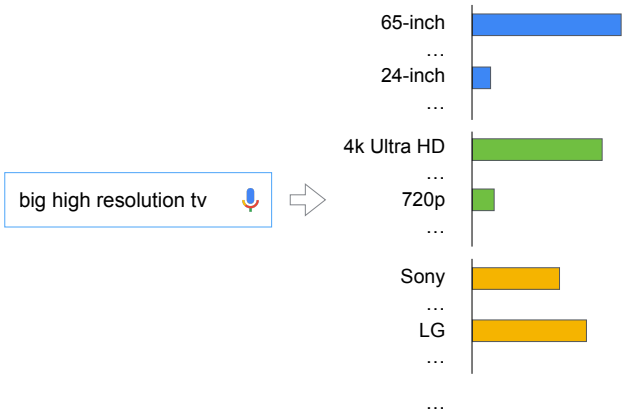



Figure 1: Given a search query, the goal is to understand the intent implied and predict a set of associated attributes. Attributes here can be brands, categories, features, etc.


*WWW 2017,* April 3–7, 2017, Perth, Australia.
ACM 978-1-4503-4913-0/17/04.
http://dx.doi.org/10.1145/3038912.3052704

In this paper we aim at *predicting latent structured intents* from shopping queries. Figure 1 presents an overview of the scenario considered. We assume that our shopping space is composed of a predefined set of *attributes*, which can be brands, merchants, features, categories, product lines, etc. Given a query, *the goal is to find the subset of attributes* that are most likely to *represent the user intent*. For example, given query *"high resolution tv"*, some implied attributes might be *"Ultra HD 4k"* (feature), *"TV"* (category), or *"LG"* (brand). Thus we can *alleviate the users* from the need to select check-boxes to specify intent.

Note that this task is different from named entity recognition (NER), where the referred entities are strictly mapped from terms in text. In our task a user does not need to mention an attribute in any way (or does not even need to be aware of the attributes), and it is the goal of this task to figure out what are the implied attributes.

The difficulty of this task does not stem just from the fact that queries are an unstructured way of representing users' intents. In addition, the model needs to generalize well to unseen queries. Usually for online shopping, the space of items is rapidly growing, and we need a system that is well-behaved even for new items or new attributes. Although one can perform frequent re-training to alleviate the problem, for a large shopping site that has a huge number of items, frequent re-training is costly. Furthermore, an online system often has a stringent latency constraints for satisfactory user experiences or hardware constraints (e.g. memory) given a production system. It is thus of great importance to build a fast and concise model.

## 1.1 Model overview

In this paper we propose a novel solution that is able to accurately interpret user queries while satisfying all the aforementioned requirements. A key idea of our model is to jointly train a *query network*, that learns the query-to-attributes mapping from past users' interaction responses to the presented results, with a *product network*, that learns attribute correlations from product metadata. Joint training is achieved by using a shared layer of *attribute embeddings*. To model unstructured queries in the query network, a highly flexible function class is needed. In this paper we adopt Long Short-term Memory (LSTM) bidirectional recurrent neural networks (BRNNs) [22] and to achieve both robustness and generalizability, we propose a hybrid word-level, character-level approach, that effectively ensembles a word-level model, which works well for head queries, and a character-level model, that works well for tail queries. The product network serves to learn the structure of attributes from product meta data. Here we consider this problem in an unsupervised setting where correlations are learned instead of being given via human annotations as in knowledge-graphs. Our solution is thus more general and scales better.

## 1.2 Contributions

Our contributions are as follows:

**A new framework** that predicts latent structured intents from shopping queries. It offers an interpretable and robust way to model user intents, and opens a new avenue to improve search result quality by presenting more accurate results.

**Jointly trained hybrid RNN & autoencoder.** A model that learns from the two most important sources of information available in online shopping: user interactions and product metadata. The proposed approach does *not* require additional parameters in the final model as compared to a non-joint model, yet it significantly improves the performance of the model. Further, the proposed hybrid approach is accurate and robust as it enjoys the advantages of both character-level and word-level models.

**Experiments.** We show that our model outperforms all baselines in terms of prediction accuracy. Furthermore, when plugged into an information retrieval (IR) system, our model is able to improve the quality of the search result of *Google Shopping production system*, which is a very strong baseline employing state-of-the-art IR techniques. Moreover, we demonstrate the robustness and the ability of our model to generalize to unseen new items and attributes.

# 2. ATTRIBUTES AND PROBLEM DEFINITION

We assume a set of predefined attributes that describes items the users are looking for. The attributes used in experiments contain the following types of information.

1. **Feature tags.** These describe the features, properties, or types of products. For example, for mattresses there are tags such as *king size*, *queen size*, or *twin*, and for cameras there are tags such as *waterproof*, *42.4-megapixel* or *4k video support*.
2. **Age Groups** Some products might target specific age groups. Tags of these groups are used in experiments.
3. **Categories.** Products also belong to categories, e.g. grocery, electronics, clothing, etc.
4. **Brands, product lines and merchants** are also included as attributes in experiments.

These attributes are either supplied by the merchants or extracted from product/item descriptions using information extraction (IE) techniques that are orthogonal to this paper.

For each query, we define a set of *implied (associated) attributes* to be the attributes contained in items clicked by users who issued the query. To make the data privacy-compliant, we only use head queries, i.e. queries which are issued by more than a certain number of users. While this makes our dataset skewed towards head queries, we will show how our system can learn to generalize well to tail queries that it has never seen in the training data. These query-attribute-set pairs are then considered as ground-truth examples for our model.

Furthermore, we assume that each item/product is associated with a set of attributes. Formally, given input query $q$, the goal is to predict a set of associated multi-labels, in our case attributes, $\mathbf{a}_q = (a_{q1}, \ldots, a_{qN}) \in \{0,1\}^N$, where $a_{qi}$ denotes the existence indicator of attribute $i$. Similarly, each product $p$ has attributes $\mathbf{a}_p = (a_{p1}, \ldots, a_{pN}) \in \{0,1\}^N$.

There are two sources of shopping queries in our experiments. One is from Google Search page [1]. Queries that are identified by a proprietary system to have purchase intents are included here as shopping queries. The other source is Google Shopping site[2], where all search queries are clearly shopping related.

[1] www.google.com
[2] www.google.com/shopping

## 3. RELATED WORK

To the best of our knowledge, this is the first public results on mapping shopping search queries to multi-labels that represent user intents. However, our problem formulation and techniques are related to the following lines of research.

*Entity retrieval.*

Our work is related to entity retrieval [19, 30] or entity disambiguation [31, 13]. These areas involve mapping terms from free-text to entities. The major difference between these works and this paper is that we do not assume that any terms in our query refer to any specific entities. Instead, we want to understand the latent intent of a query and find the *implied* attributes. In other words, note that in query *"high-end bike"*, none of the terms in this query refers to *"21 speed"* or *"carbon frame"* but they are the likely attributes *implied*.

*Multi-label classification.*

Multi-label classification is a well-studied problem that has been applied to a wide range of domains, such as texts [18, 12], images [3, 28], or music [24]. However, one practical issue is that the accuracy and efficiency suffers when the number of labels is huge. [2] proposes to use a subset to approximate the original space to improve efficiency. [5] and [1] utilize correlations between labels. We instead, propose to jointly train a metadata network that models the correlations between labels.

*Search engine.*

Our work is also related to studies on web search engines [4], or information retrieval (IR). To improve search result quality, popular techniques include query expansion [25, 20, 29], especially, pseudo-relevance feedback has been shown to be very effective in most cases [25]. However, note that these works have been focused on ranking of a list of documents instead of a multi-label classification problem. Furthermore, pseudo-relevance feedback focuses on extending the current query with terms from the top retrieved results, while in our approach, we seek to augment the query with terms *learnt* form user consumption behavior. [21] classifies queries into different search goals, such as "directional", "informational", "resource seeking", etc. However, their goal is understanding different types of searching behaviors (*why* they are searching) instead of the intent of each query (*what* they are searching for).

*Recurrent neural networks.*

Recurrent neural networks (RNNs) are a class of dynamic models that have demonstrated impressive results in text modeling. For example, RNNs have achieved state-of-the-art results in text generation [9], machine translation [23], and image captioning [11]. One popular use of RNNs is to summarize information in texts, e.g. [27]. In this paper we apply RNNs in similar fashions.

We also explore the idea of combining word-level and character-level RNNs. [16, 26, 17] consider similar ideas in named entity recognition (NER) and sequence tagging. However, their approaches are based on using character-level embeddings to augment word-embeddings. In this paper we consider a much simpler and efficient alternative that trains both the character-level and word-level RNNs on full queries.

*Autoencoder.*

Autoencoder is an unsupervised learning approach that finds low-dimensional representation of data automatically. For example, [7] uses autoencoded deep-learning representation for speech compression and recognition, and [15] uses autoencoders for image retrieval. The product attribute network in our model is one form of autoencoder. However, our goal is to jointly train a better attribute embedding, instead of obtaining the representation as in traditional settings.

## 4. PROPOSED MODELS

In this section, we present a detailed description of a series of proposed models. Following the problem formulation in Section 2, the goal of these models is to learn a function that maps a query to a the set of attributes relevant to the intent of the user who issued this query. Our models vary in terms of how they represent queries in order generalize to unseen words/queries.

### 4.1 Multi-layer perceptron (MLP)

Probably one of the simplest models for our purpose is a multi-layer perception (MLP) that takes query word counts as input and predicts attributes. Formally, in a $L$-layer multi-label classification MLP, we have hidden factors

$$h^{(i)} = \sigma^{(i)}(W^{(i)}h^{(i-1)} + b^{(i)}), i = 1, \dots L, \quad (1)$$

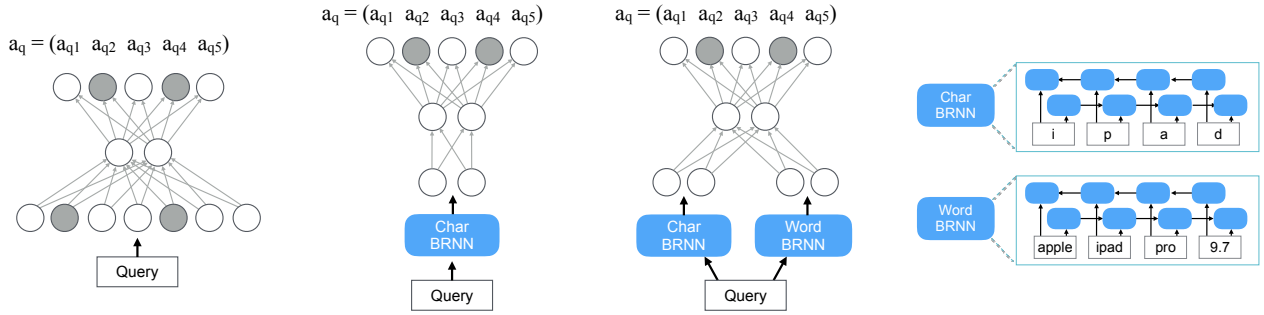
where $h^{(0)}$ is the input, $\sigma^{(i)}$ denotes non-linear activation function at layer $i$, and $\sigma^{(L)}$ is the sigmoid output predictions. $W^{(i)}$ and $b^{(i)}$ denote the weight and bias term respectively. Figure 2a illustrates a MLP model in which we see that the hidden layer $h$ represents the query embedding, which is then fed via a fully-connected layer to predict a set of binary attribute indicators. The system is trained in a standard way to minimize the logistic loss of the query-attributes training data. However, in a standard implementation where input queries are encoded as word count vectors, word order information is lost (*"milk chocolate"* is indistinguishable from *"chocolate milk"*). This is thus not an ideal option for shopping query understanding.

### 4.2 LSTM-based networks: Char-BRNN and Word-BRNN

Another way to summarize the semantics of a query in a low-dimensional vector is through recurrent neural networks (RNNs). Given an input sequence $x_1, \dots, x_T$, a RNN performs

$$h_i = f(h_{i-1}, x_i), i = 1, \dots, T$$

for some function $f$, where $h_i$ denotes the hidden state of the sequence after observing $x_i$. That is, it is a network that learns how to update the state given input at each time step. For query understanding, we can train a RNN that encodes the information of a query in $h_T$, which is then followed by a classifier for attribute prediction. In this paper we adopt Long Short-term Memory (LSTM) [10] RNNs, as it is able to encode long-range context, address the vanishing gradient problem, and is slightly more general than alternatives such as Gated Recurrent Units (GRUs) [6]. An LSTM state




(a) MLP. (b) Character-level model. (c) Hybrid model. (d) Bidirectional RNNs.

Figure 2: (a) (b) (c): Three possible choices of query networks. (d): Character-level and word-level bidirectional recurrent neural networks (BRNN).

update is composed of the following operations

$$[f_t, i_t, o_t] = \sigma\left(W\left[h_{t-1}, x_t\right] + b\right) \tag{2}$$

$$l_t = \tanh\left(V\left[h_{t-1}, x_t\right] + d\right) \tag{3}$$

$$c_t = f_t c_{t-1} + i_t l_t \tag{4}$$

$$h_t = o_t \tanh\left(c_t\right), \tag{5}$$

where $c_t$ denotes the cell state, and $f_t$, $i_t$, $o_t$ are the forget gate, input gate, and output gate respectively. These gates control how information is added to or removed from cell states along the sequence of state updates. For simplicity, we denote this set of operations by $h_t = \text{LSTM}(h_{t-1}, x_t)$.

A simple generalization of RNN is to construct a **bidirectional RNN (BRNN)** that summarizes information of a sequence from both directions [22]. Formally, we construct

$$\overrightarrow{h_t} = \text{LSTM}(\overrightarrow{h_{t-1}}, x_t) \quad \text{and} \quad \overleftarrow{h_t} = \text{LSTM}(\overleftarrow{h_{t+1}}, x_t). \tag{6}$$

With this BRNN, we can define **a character-level or a word-level BRNN**-based attribute prediction model, with the final layer defined as

$$\hat{\mathbf{a}}_q = \sigma\left(W_{\text{dec}} \cdot \phi(W_{\text{fusion}}\left[\overrightarrow{h}_T, \overleftarrow{h}_0\right])\right) \tag{7}$$

The bias term for each affine transformation and $h$s' dependency on query $q$ are skipped for clarity. The first affine transformation, given by $W_{\text{fusion}}$, serves as a fusion layer that fuses the two directions of RNN states to realize the query representation. $\sigma$ denotes element-wise sigmoid function $\sigma(x) = (1 + \exp(-x))^{-1}$, $\phi$ is some non-linear function, and $\hat{\mathbf{a}}_q$ is the final sigmoid prediction output given query $q$. This model is illustrated in Figure 2b for the case of **char-BRNN** (**word-BRNN** follows in a similar fashion). We note here that *char-BRNN* and *word-BRNN* are technically identical, with the difference being that in char-BRNN the model operates on the sequence of characters in the queries (thus can generalizes to unseen words) while in word-BRNN the model operates on the sequence of words in the query thus can better model head words.

To optimize either network, we minimize logistic loss,

$$L_{\text{query}} := \sum_{q \in \mathcal{Q}} \sum_{i \in [1,N]} \mathbf{a}_{qi} \log(\hat{\mathbf{a}}_{qi}) + (1 - \mathbf{a}_{qi}) \log(1 - \hat{\mathbf{a}}_{qi}), \tag{8}$$

where $\mathcal{Q}$ denotes the set of training queries.

## 4.3 Hybrid network

Char-BRNN and word-BRNN are both powerful models yet each of them has its own drawback. In word-BRNN the model that takes a sequence of words as input and statistical strength is not shared between words. This is not ideal since for an online shopping site, new model names will appear in queries, and those names usually have some patterns (e.g. *i5-6200u* and *i7-6500u* are CPUs of the same brand, and *D3200* and *D3300* are cameras in the same product line). It is desirable to build a model that learns the underlying patterns. One way to alleviate this issue is using prefixes or suffixes as features, or perform stemming. However, these approaches only address a few special cases instead of providing a general solution. In addition, in practice the number of possible words is huge (consider all possible typos in queries). Modeling all of them is infeasible, so one might end up discarding infrequent words.

On the other hand, char-BRNN can solve this generalization problem, by taking a sequence of *characters* as inputs. This greatly reduces the model size, as the number of possible characters is much smaller than the number of possible words. Words with similar forms or patterns share similar character sequences, and the resulting hidden state updates would thus also be similar. Character-level RNNs have shown promising results in multiple domains [27, 9]. However, as a character-level model relaxes the natural word boundaries, it suffers at rare words where we do not have enough data to understand a specific arrangement of characters.

It is thus tempting to design a model that enjoys the strength of both word-level RNNs and character-level RNNs. [16, 26, 17] propose to concatenate the word-level embedding and the character-level embedding, which is extracted from either another RNN or a convolutional neural network. Note that in these approaches the character-level embedding of one word does not depend on previous words. In fact, one can deem these approaches as extracting features for each word by character-level models.

In this paper we instead, consider jointly training a character-level and a word-level RNN, both of which are constructed on a full query. The output of the two are then concatenated to form the *query representation*. We call this a "hybrid" network, **Hybrid-BRNN**. This allows us to model cross-word dependency among characters, and the implementation and

training is much simpler than those in [16, 26, 17]. In other words, we can build a hybrid query network:

$$h^{\text{hybrid}} := \left[\overrightarrow{h}_T^{\text{char}}, \overleftarrow{h}_0^{\text{char}}, \overrightarrow{h}_T^{\text{word}}, \overleftarrow{h}_0^{\text{word}}\right] \quad (9)$$

$$\hat{\mathbf{a}}_q = \sigma\left(W_{\text{dec}} \cdot \phi(W_{\text{fusion}} \cdot h^{\text{hybrid}})\right), \quad (10)$$

as shown in Figure 2c. As before, the query representation $h^{\text{hybrid}}$ is fed via a fully-connected layer to predict attributes $\mathbf{a}_q$ and the model is trained end-to-end to minimize the logistic loss similar to Equation (8).

## 4.4 Joint Networks

In our query networks (MLP, char-BRNN, word-BRNN and hybrid-BRNN) described in the previous subsections, the correlation between attributes is ignored (not explicitly modeled). As the number of attributes becomes huge, predictions can be inaccurate. To address this issue we propose to jointly train an autoencoder on product metadata that learns the correlation. The key insight is that co-occurrence pattern of product attributes should be very similar to the co-occurrence pattern of latent attributes implied by queries. This enables us to use the information from millions of products available in the online shopping site (such as Google Shopping) database to more accurately capture the attribute semantics. We first begin by defining the product network then explain the joint training strategy.

*Product Network.*

Given a product $p$ annotated by a set of attributes $\mathbf{a}_p = (a_{p1}, \ldots, a_{pN}) \in \{0,1\}^N$. We train an autoencoder that minimizes

$$\hat{\mathbf{a}_p} := \sigma\left(W_{\text{dec}} \cdot \phi\left(W_{\text{enc}}\ \mathbf{a}_p\right)\right) \quad (11)$$

$$L_{\text{product}} := \sum_{p \in \mathcal{P}} \sum_{i \in [1,N]} \mathbf{a}_{pi} \log(\hat{\mathbf{a}}_{pi}) + (1 - \mathbf{a}_{pi}) \log(1 - \hat{\mathbf{a}}_{pi}), \quad (12)$$

where $W_{\text{enc}} \in \mathbb{R}^{D \times N}, W_{\text{dec}} \in \mathbb{R}^{N \times D}$, and $D < N$. $\mathcal{P}$ denotes the set of training products. Bias terms are omitted for clarity. Right hand side of Figure 3 shows the architecture of this network. Intuitively it learns an encoder that *encodes* the input attributes into a $D$-dimensional space, and a decoder that is able to recover the original input from this lower dimensional vector. We call the rows in $W_{\text{dec}}$ attribute embeddings. If $D$ is small, this formulation forces the embeddings of frequent co-occurring attributes to be similar.

*Joint Training.*

In order for the two networks (query and product networks) to communicate information, we make the two networks share parameters in the attribute embedding layer $W_{\text{dec}}$. Figure 3 illustrates this idea. Here we show the hybrid query network (hybrid-BRNN) as an example, but other query networks (such as char-BRNN or word-BRNN) can be jointly trained in the same manner. Training is performed through minimizing the joint loss,

$$L_{\text{joint}} := L_{\text{query}} + \lambda L_{\text{product}}, \quad (13)$$

where $\lambda$ controls the trade-off between learning from product structure and learning from user responses. Note that the product model is not needed during serving. After all the two models only share parameters, but do not mix inputs and outputs. Thus the model size and inference efficiency stays the same as the models without joint training. In Section 5.1, we show that this joint training approach consistently provides performance improvements across different variants of the model.

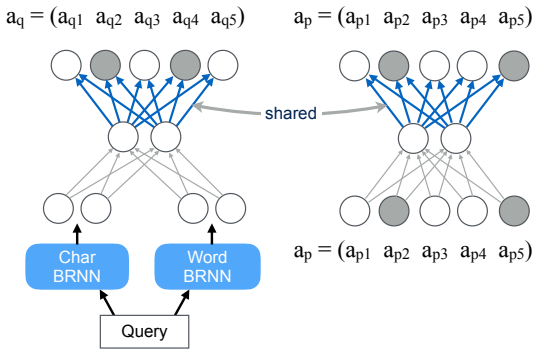



Figure 3: Full model. Parameters in the final fully-connected layer (in blue) are shared.

# 5. EXPERIMENTS

Our experiments are performed on a proprietary dataset collected at Google Shopping. The dataset is composed of more than **150 million queries** with associated attributes selected as described in Section 2. We split the data into 80%, 10%, and 10% for training, validation, and testing sets respectively. The three sets have **no overlapping** queries. The queries contain more than 100,000 unigrams. In our experiments, about 30,000 attributes are considered.

Our evaluation contains both intrinsic evaluation (attribute prediction accuracy with ablation studies) and extrinsic evaluation based on information retrieval results.

*Training details.*

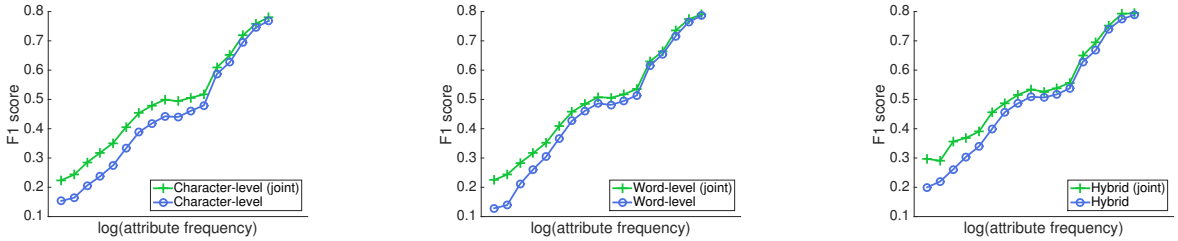
All models are trained by backpropagation. We use ADAM [14] with learning rate 0.001 and a mini-batch size of 128. Gradients are clipped to a maximum norm of 5.0. For RNNs, we use LSTM with one hidden layer with 512 units. We initialize the neural networks with $\mathcal{N}(0, \sigma_{init}^2)$ with $\sigma_{init}^2 = \frac{2}{(\text{fan in})+(\text{fan out})}$, following [8]. Hyperparameters, learning algorithm parameters as well as *embedding sizes for words , characters and final prediction layer* are selected by cross-validation over the validation set.

## 5.1 Intrinsic Evaluation

We first evaluate our model in terms of attribute prediction accuracy. Evaluation metrics are standard precision, recall, and F1 scores (2·precision · recall/(precision+recall)).

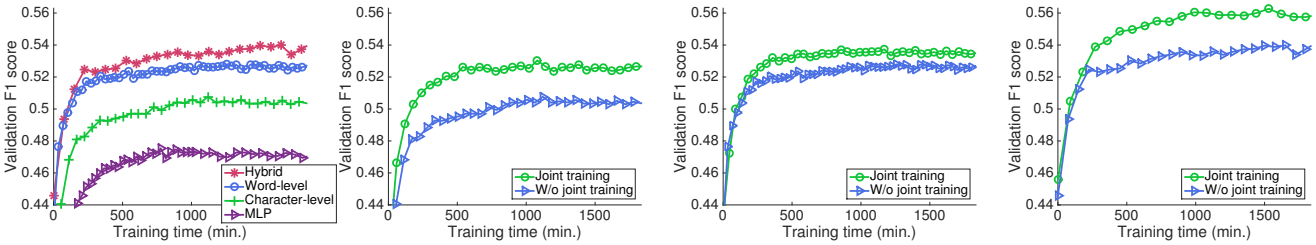
### 5.1.1 Prediction accuracy and size

We evaluate all three types of query models (character-level, word-level, and hybrid), and compare them with the baseline of MLP. The results are summarized in Table 1. We see that all variants of RNN-based models outperform MLP. The hybrid approach that combines character-level and word-level models outperforms individual ones significantly in F1 scores. We also observe a trade-off between size and accuracy here: the hybrid model achieves the best performance at the cost of a bigger model size. On the other




(a) Character-level model. (b) Word-level model. (c) Hybrid model.

Figure 4: For all variants of our model, the benefit gained from joint training is the greatest for rare attributes. Actual attribute frequencies are hidden to protect proprietary data.




(a) Without joint-training. (b) Character-level model. (c) Word-level model. (d) Hybrid model.

Figure 5: (a) Even though the hybrid model is more flexible and powerful, it converges at a similar speed as other models. (b) (c) (d): Speed of convergence stays similar even with joint-training.

| Model | Size | Precision | Recall | F1 |
|---|---|---|---|---|
| MLP | 1.04 GB | 0.694 | 0.359 | 0.473 |
| Char-BRNN | **254 MB** | 0.720 | 0.386 | 0.503 |
| Word-BRNN | 334 MB | 0.731 | 0.408 | 0.524 |
| **Hybrid-BRNN** | 588 MB | **0.737** | **0.431** | **0.544** |

Table 1: Prediction accuracy and size. All RNN-based models outperform MLP significantly. The hybrid approach outperforms the character-level-only and the word-level-only models.

hand, the character-level model, albeit much smaller, still outperforms MLP by a large margin and provides competitive performance.

### 5.1.2 *Joint training*

| | Char-BRNN | Word-BRNN | Hybrid-BRNN |
|---|---|---|---|
| Query only | 0.503 | 0.524 | 0.544 |
| Joint-training | **0.524** | **0.537** | **0.561** |

Table 2: F1 scores of all variants. Joint training improves F1 scores for all models.

Here we study the benefits provided by joint training. Table 2 summarizes the results. We focus on RNN-based models (character-level, word-level, and hybrid BRNNs), as they outperform MLP significantly. We see that joint-training consistently boosts performance across all variants. We further examine the prediction accuracy on attributes with different frequencies. Results are summarized in Figure 4. We see that the benefit gained from joint training is the greatest for rare attributes. This confirms our motivation of using joint training to defeat inaccuracies at the tail when we have a large number of attributes.

### 5.1.3 *Training time*

In Figure 5, we plot the learning curves of all the models. Interestingly, we found that even though the hybrid model is more flexible and powerful, it converges at a similar speed as other models, as shown in Figure 5a. In Figure 5b, 5c, and 5d, we compare the convergence of models with and without joint training. We see that the convergence rate stays similar even with joint-training for all models.

### 5.1.4 *Qualitative analysis*

| *"MDRZX390MC"* (real headphone model) | *"MDRZX709MK"* (*made-up* headphone model) |
|---|---|
| Headphones brand: Sony | Headphones brand: Sony |
| Brand: Sony | Brand: Sony |
| Connectivity: Wired | Headphones type: Headphones |
| Headphones type: Headphones | Headphones use: Cell Phone |

Table 3: Example queries (top row) and the predicted top attributes (the following rows). The left query is a headphone model name and the right query is a *made-up* model name that simulates a future unseen model. Our model is able to generalize to unseen model names based on the character pattern of a word.

| *"yoga 3"* | *"yoga 5"* (made-up model) | *"yoga mat"* | *"yoga mattt"* (typo) |
|---|---|---|---|
| Brand: Lenovo | Laptops: brand: Lenovo | Yoga & Pilates Mats | Yoga & Pilates Mats |
| Laptops: Touchscreen | Brand: Lenovo | Brand:Yoga Direct | Gender: Unisex |
| Laptops brand: Lenovo | Lenovo product line: IdeaPad | Brand:Ggi | Merchant: YogaOutlet.com |

Table 4: Example queries (top row) and the predicted top attributes (the following rows). Our model is able to generalize to different unseen words in differently reasonable ways.

| *"gift for girlfriend"* | *"gift for 10-year-old girl"* |
|---|---|
| Age group: adult | Gender: female |
| Gender: female | Gender: unisex |
| Brand: Alex And Ani | Merchant: Fat Brain Toys |
| Bracelets: age group: adult | Merchant: Target |
| Bracelets: material: silver | Age group: adult |
| Bracelets: brand: Alex and Ani | Merchant: Walmart |
| Bracelets: silhouette: bangle | Brand: Rose Art |
| merchant: Alex and Ani | Brand: Klutz |
| Bracelets: department: women | Merchant: Jet.com |
| Gender: unisex | Age group: children |
| Bracelets: gender: female | Merchant: Toys R Us |

Table 5: Our model is able to model complex abstract queries. Top row shows queries (in quotes) and the following gives the predicted top attributes.

We first study our model's ability to generalize to unseen queries. This is extremely important for online shopping, as the space of items is rapidly growing. We want our system to be robust even for unseen words. Table 3 shows an example illustrating how the model generalizes. The query on the left is a headphone model name, and the query on the right is a *made-up* model name that simulates a future unseen query. We see that even though the made-up name is out-of-vocabulary, our model is still able to predict reasonable headphone-related attributes based on the character pattern. Similarly, Table 4 shows another example on a different set of queries. Again, we see that our model is robust to typo and can generalize to different unseen queries in differently reasonable ways.

Table 5 shows an example illustrating how our model handles abstract complex queries. We see that even the left query, *"gift for girlfriend"*, does not contain *"bracelet"* as a keyword, our model predicts bracelets and some related brands to be likely attributes. These are arguably reasonable guesses without further information. Similarly, even the right query, *"gift for 10-year-old girl"*, does not contain *"toy"* as a keyword, our model reasonably retrieves some toy brands and related merchants[3].

## 5.2 Extrinsic evaluation on IR

In this section we present extrinsic evaluation results on information retrieval (IR) performance. Specifically, we study whether the predicted user-intent attributes are able to improve the search result quality.

The dataset over which we evaluate our model consists of about 200,000 queries that are **non-overlapping** with the training queries. Each query is associated with a list of search results, each of which has a human-annotated rating. The human ratings are in a 10-point scale from "very irrelevant" to "very relevant". In addition to human rating, each query result has an IR score, which is generated from the Google IR system. The system is a very strong baseline that is used in production at Google. During retrieval, the results for a query are ranked in descending order by the IR score.

We use the predicted attributes to generate new rankings by boosting the original IR scores based on the attributes predicted. Specifically, for each query-result pair $(q, d)$, we boost the IR score $s_{qd}$ and obtain a new score $s^{\star}_{qd} := s_{qd} \cdot (1 + \alpha \frac{|A_q \cap A_d|}{|A_q|})$, where $A_q$ is the set of top $k$ attributes predicted for $q$ and $A_d$ is the set of attributes of $d$ respectively. $\alpha$ is some positive constant that is selected by cross-validation for this experiment. Incorporating confidence scores of the predicted attributes (soft predictions) into boosting is left as future work.

The boosted scores $s^{\star}$s will induce a re-ranking of results for each query. We measure the quality of a ranking based on discounted cumulative gain (DCG) , which is calculated based on human-ratings: $\text{DCG@}n := r_1 + \sum_{i=2}^{n} \frac{r_i}{\log_2(i)}$, where $r_i$ is the human rating of the result at rank $i$.

### 5.2.1 Overall results

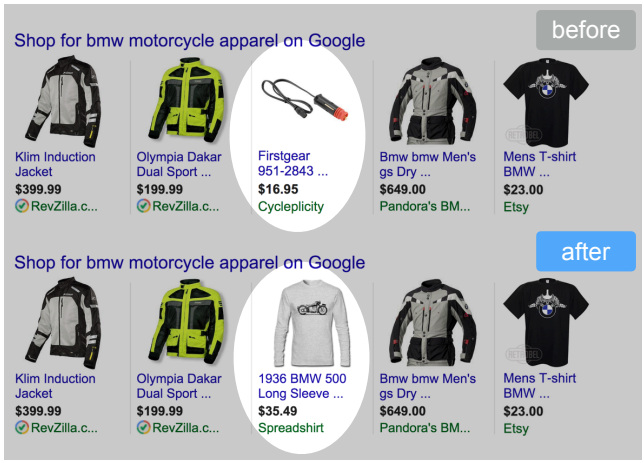
| | Gains in DCG (%) | | | |
|---|---|---|---|---|
| k | @1 | @3 | @5 | @10 |
| MLP | −0.221* | 1.202 | 1.432 | 0.336 |
| Char-BRNN | 3.577 | 1.895 | 1.580 | 0.593 |
| Word-BRNN | 7.874 | 3.117 | 1.627 | 0.685 |
| **Hybrid-BRNN** | **11.027** | **4.592** | **1.823** | **0.914** |

Table 6: Gains in DCG over current production system. All BRNN models here are jointly trained. The number of attributes used ($k$) is selected by cross-validation for each model. The number with a star is not statistically significant.

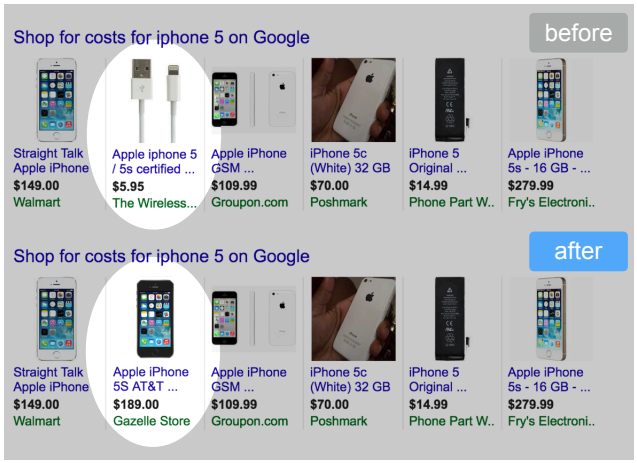
The IR performance is compared with current Google production system, which is a very strong baseline that incorporates many powerful techniques such as pseudo-relevance feedback and query expansion [25, 20], to name a few.

Table 6 shows the improvements in terms of DCG given by jointly-trained RNN-based models. We see that all RNN-based models significantly improve DCGs. Especially, we see a great improvement in DCG@1, which represents the quality of the most important top position. Improvements on later positions confirm the effectiveness of our approach: after all, pseudo-relevance feedback is limited to extracting attributes from only top few results, while our approach directly predicts the most relevant attributes among all. In ad-

[3] Predicted *"Age group: adult"* attribute in the right column could be due to the fact that some gifts for kids, such as LEGO or soccer balls, are not kids-exclusive.




(a) Query: *"bmw motorcycle apparel"*




(b) Query: *"costs for iphone 5"*

Figure 6: Examples of re-ranking of results (before and after). (a) Our model correctly predicts the category *"Shirts and Tops"*, so our boosting promotes and shows a T-shirt instead of an adaptor at position 3. (b) Production incorrectly shows an iPhone charger at position 2. However, after boosting we demote that result and instead show an iPhone 5S.

dition, our hybrid model consistently outperforms character-level-only and word-level-only models.

### 5.2.2 *Examples of re-ranking*

Figure 6 presents 2 examples of how our attribute-prediction-based boosting is able to correct some mistakes made by current Google Shopping production system. In Figure 6a, given query *"bmw motorcycle apparel"*, our model correctly predicts the category *"Shirts and Tops"*, so our boosting promotes and shows a T-shirt instead of an (incorrect) adaptor at position 3. Similarly, as shown in Figure 6b, with query *"costs for iphone 5"*, production incorrectly shows an iPhone charger at position 2. However, after boosting we demote that result and instead show an iPhone 5S.

### 5.2.3 *Ablation studies*

| | Gains in DCG (%) | | | |
|---|---|---|---|---|
| k | @1 | @3 | @5 | @all |
| 2 | −11.492 | 2.036* | 1.502* | 0.621 |
| 3 | −5.337 | 1.245* | 1.661 | 0.709 |
| 4 | 1.626* | 2.370* | 1.714 | 0.809 |
| 5 | 6.334 | 2.245* | 1.541 | 0.747 |
| 6 | 7.333 | 2.856* | 1.397* | 0.725 |
| 7 | 8.282 | 3.442 | 1.713 | 0.813 |
| 8 | 6.753 | 4.521 | 1.805 | 0.852 |
| 9 | 11.027 | 4.592 | 1.823 | 0.914 |
| 12 | −1.710* | 2.196* | 1.876* | 0.547* |
| 15 | −15.560 | 0.243* | 0.559* | 0.175* |
| 20 | −21.916 | 0.271* | 0.556* | 0.021* |

Table 7: Gains in DCG with different number of top attributes used per query ($k$) for hybrid model. A $k$ at around 9 gives best results for all models. Numbers with a star are not statistically significant.

| | Gains in DCG (%) | | | |
|---|---|---|---|---|
| k | @1 | @3 | @5 | @10 |
| Hybrid | 8.326 | 3.071 | 1.638 | 0.665 |
| **Joint hybrid** | **11.027** | **4.592** | **1.823** | **0.914** |

Table 8: Joint training brings greater gains in DCG.

Here we give ablation analysis of the IR performance. We present detailed results only for the best-performing hybrid model due to space constraints, but the same conclusion holds for all variants. First of all, we examine the impact of different numbers of predicted attributes used per query, i.e. $k$. The results are summarized in Figure 7. We found that for all models, a $k$ at around 9 gives the best results overall. We conjecture that for small $k$s we might "overfit" by being very specific, and for very large $k$s some of the predicted attributes become less accurate, so it would hurt performance. We hypothesize that in future work where we use soft predictions for score-boosting, the need to find an optimal $k$ can be eliminated. In addition, we found that all jointly trained models outperform the ones without joint training at all positions. Table 8 presents the results for hybrid model with $k = 9$.

## 6. CONCLUSION

In this paper we study a new framework that predicts latent structured intents from shopping queries. We propose a jointly trained hybrid RNN-autoencoder that learns from user responses and product metadata simultaneously. Our model is more accurate and robust than all baselines, and is able to generalize to unseen queries better. High-quality human-ratings are used to evaluate IR performance. We show that our model can significantly improve the quality of current Google Shopping production system.

## 7. REFERENCES


[1] W. Bi and J. T. Kwok. Multi-label classification on tree-and dag-structured hierarchies. In *ICML*, pages 17–24, 2011.

[2] W. Bi and J. T.-Y. Kwok. Efficient multi-label classification with many labels. In *ICML*, pages 405–413, 2013.

[3] M. R. Boutell, J. Luo, X. Shen, and C. M. Brown. Learning multi-label scene classification. *Pattern recognition*, 37(9):1757–1771, 2004.

[4] S. Brin and L. Page. Reprint of: The anatomy of a large-scale hypertextual web search engine. *Computer networks*, 56(18):3825–3833, 2012.

[5] W. Cheng, E. Hüllermeier, and K. J. Dembczynski. Bayes optimal multilabel classification via probabilistic classifier chains. In *ICML*, pages 279–286, 2010.

[6] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[7] L. Deng, M. L. Seltzer, D. Yu, A. Acero, A.-r. Mohamed, and G. E. Hinton. Binary coding of speech spectrograms using a deep auto-encoder. In *Interspeech*, pages 1692–1695. Citeseer, 2010.

[8] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, volume 9, pages 249–256, 2010.

[9] A. Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

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

[11] A. Karpathy and L. Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *CVPR*, pages 3128–3137, 2015.

[12] I. Katakis, G. Tsoumakas, and I. Vlahavas. Multilabel text classification for automated tag suggestion. *ECML PKDD discovery challenge*, 75, 2008.

[13] S. S. Kataria, K. S. Kumar, R. R. Rastogi, P. Sen, and S. H. Sengamedu. Entity disambiguation with hierarchical topic models. In *SIGKDD*, pages 1037–1045. ACM, 2011.

[14] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[15] A. Krizhevsky and G. E. Hinton. Using very deep autoencoders for content-based image retrieval. In *ESANN*, 2011.

[16] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*, 2016.

[17] X. Ma and E. Hovy. End-to-end sequence labeling via bi-directional lstm-cnns-crf. *arXiv preprint arXiv:1603.01354*, 2016.

[18] A. McCallum. Multi-label text classification with a mixture model trained by em. In *AAAI workshop on text learning*, pages 1–7, 1999.

[19] F. Nikolaev, A. Kotov, and N. Zhiltsov. Parameterized fielded term dependence models for ad-hoc entity retrieval from knowledge graph. In *SIGIR*, pages 435–444. ACM, 2016.

[20] J. J. Rocchio. Relevance feedback in information retrieval. *SMART Retrieval System–Experiments in Automatic Document Processing*, 1971.

[21] D. E. Rose and D. Levinson. Understanding user goals in web search. In *WWW*, pages 13–19. ACM, 2004.

[22] M. Schuster and K. K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

[23] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *NIPS*, pages 3104–3112, 2014.

[24] K. Trohidis, G. Tsoumakas, G. Kalliris, and I. P. Vlahavas. Multi-label classification of music into emotions. In *ISMIR*, volume 8, pages 325–330, 2008.

[25] J. Xu and W. B. Croft. Query expansion using local and global document analysis. In *SIGIR*, pages 4–11. ACM, 1996.

[26] Z. Yang, R. Salakhutdinov, and W. Cohen. Multi-task cross-lingual sequence tagging from scratch. *arXiv preprint arXiv:1603.06270*, 2016.

[27] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy. Hierarchical attention networks for document classification. In *NAACL*, 2016.

[28] Z.-J. Zha, X.-S. Hua, T. Mei, J. Wang, G.-J. Qi, and Z. Wang. Joint multi-label multi-instance learning for image classification. In *CVPR*, pages 1–8. IEEE, 2008.

[29] C. Zhai and J. Lafferty. Model-based feedback in the kl-divergence retrieval model. In *CIKM*, pages 403–410, 2001.

[30] N. Zhiltsov, A. Kotov, and F. Nikolaev. Fielded sequential dependence model for ad-hoc entity retrieval in the web of data. In *SIGIR*, pages 253–262. ACM, 2015.

[31] S. Zwicklbauer, C. Seifert, and M. Granitzer. Robust and collective entity disambiguation through semantic embeddings. In *SIGIR*, pages 425–434. ACM, 2016.