Query to Knowledge: Unsupervised Entity Extraction from Shopping Queries using Adaptor Grammars

Ke Zhai, Zornitsa Kozareva, Yuening Hu, Qi Li and Weiwei Guo
Yahoo! Research, 701 1st Ave, Sunnyvale, CA 94089
{kzhai,kozareva,ynhu,lqi,wguo}@yahoo-inc.com

ABSTRACT

Web search queries provide a surprisingly large amount of information, which can be potentially organized and converted into a knowledgebase. In this paper, we focus on the problem of automatically identifying *brand* and *product* entities from a large collection of web queries in online shopping domain. We propose an unsupervised approach based on adaptor grammars that does not require any human annotation efforts nor rely on any external resources. To reduce the noise and normalize the query patterns, we introduce a query standardization step, which groups multiple search patterns and word orderings together into their most frequent ones. We present three different sets of grammar rules used to infer query structures and extract *brand* and *product* entities. To give an objective assessment of the performance of our approach, we conduct experiments on a large collection of online shopping queries and intrinsically evaluate the knowledgebase generated by our method qualitatively and quantitatively. In addition, we also evaluate our framework on extrinsic tasks on query tagging and chunking. Our empirical studies show that the knowledgebase discovered by our approach is highly accurate, has good coverage and significantly improves the performance on the external tasks.

1. INTRODUCTION

Queries collected by web-scale search engines provide a large amount of useful information. This information can be organized to knowledgebase [5] which can impact and improve the performance of a wide variety of applications such as parsing, coreference resolution, entity linking. Typically, researchers focus on extracting entities for movie, music, person, location, and organization names from natural language texts [22]. Others have shown that knowledgebase can be easily built from such types by consulting external resources, such as IMDB [9], DBpedia [10], or Wikipedia [24]. However, one bigger challenge is how to learn knowledgebase for more rare yet useful entity types such as *brand* and *product*. These entities are important and crucial for commercial search engines and online advertising to function properly.

Formally, we define the *brand* and *product* entity types as:

Definition 1. A ***brand*** entity is defined as a named entity phrase whose master[1] business is to manufacture, provide or sell one or more major products or services. It also applies to subsidiaries or even divisions that operate as a separate company and master manufacturer producing or selling their own products, for instance, "*Lexus*" (owned by "*Toyota*") and "*Chevrolet*" (owned by "*GM*").

Definition 2. A ***product*** entity is defined as an entity phrase specifying generic product terms, e.g., "*jeans*", "*smartphone*", "*running shoes*". Note that a product does not include any attributes, but may include necessary generic specification. For example, "*wedding dress*" and "*dress*" are considered two different products, since they are generically different products, but "*black dress*" is *not* considered a standalone product, since it is an actual product ("*dress*") with color attributes ("*black*").[2]

To the best of our knowledge, there are no publicly available knowledgebase with updated *brand* and *product* entity lists.

Fortunately, online shopping queries collected at commercial search engines provide large amount of information, which can be organized into such knowledgebase [26]. For example, the query "*calvin klein white t shirts*" contains explicit information about the *brand* ("*calvin klein*") and *product* ("*t shirts*") with arbitrary attributes or constrains ("*white*"). Utilizing this information is able to greatly reduce human annotation time and editorial labeling efforts, which can be both expensive and time consuming to acquire.

However, extracting *brand* and *product* entities from web-scale query logs in online shopping domain is fundamentally different from classic *named entity recognition* [23, NER] frameworks. Past entity extraction works focus on natural language text [22, 30] or external web resources [20, 31]. They target on identifying entity phrases, such as person, location and organization, etc, using some indicative features (e.g., capitalization), trigger words (e.g., "*Mrs*", "*Inc*") or some grammar properties (e.g., structural patterns, like a location address usually follows the pattern of *street* +

SIGIR '16, July 17-21, 2016, Pisa, Italy

DOI: http://dx.doi.org/10.1145/2911451.2911495

[1]The definition "master" is to distinguish the concept of *brand* from other named entity types, such as *product family* of *organization*. For example, "*nexus*" (owned by "*Google*") is **not** a *brand*, but a *product family*. It does *not* apply to branding or advertisement carriers neither, e.g., "*Manchester United*" is considered *organization*.

[2]This definition can be empirically more generic and fine-grained than the extended named entity hierarchy at nlp.cs.nyu.edu/ene/.

city + *state*|*province* + *zip codes*). In addition, classic NER frameworks usually operate on a collection of sentences or paragraphs, which contains rich contextual information and syntactic structures, such as *part-of-speech* (POS) tagging, dependency parsing, etc.

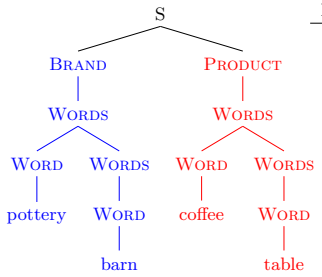
On the other hand, online shopping queries often do not have indicative features nor do shopping entity types contain any trigger words. In terms of grammar properties, majority of web search queries consist of proper nouns or noun phrases only [3, 27], which often lacks explicit syntactic dependency structure. In addition, they are usually short and noisy (e.g., misspelling, arbitrary word ordering). All of these factors rise severe difficulties in learning *brand* and *product* entities from web query logs.

In this paper, we focus on a fundamentally challenging yet interesting problem of automatic extracting *brand* and *product* entities from web-scale online shopping queries. Ideally, the extraction framework should satisfy following:

1: It is **unsupervised** and automatic, which does not require any human annotation or external resource.
2: It is **robust** to various kinds of noise and dependency structure presented in web shopping queries.
3: It is **domain-independent**, and easily generalizable to multiple domains, e.g., apparel, electronics, etc.
4: It is **data-driven**, i.e., the numbers of *brand* and *product* entities are not known a priori. Instead, the model automatically infers the number of entities appeared in the data in a completely nonparametric fashion.

In this work, we propose a novel framework which fulfills all these requirements. Our contributions in this paper lies in the following aspects. We first formulate the entity extraction problem, in its general form, as an unsupervised shallow parsing problem with *probabilistic context-free grammars* [21, PCFG] on a collection of online shopping queries. This allows our model to be **unsupervised** and **domain-independent**, and hence easily generalizable to multiple domains. We introduce a query standardization step to reduce the sparsity of the word orderings and query patterns in data. Our proposed model relies on *adaptor grammars* [15]—a nonparametric counterpart of PCFG—which is completely **data-driven** and **robust** to noise. We subsequently propose three different grammar rule sets to regulate different pattern variants in the data. We then compare our model against strong baselines and conduct comprehensive evaluations on the extracted knowledgebase both qualitatively and quantitatively. Intrinsically, we demonstrate that our framework is able to discover high quality knowledgebase. We also extrinsically evaluate the extracted knowledgebase on applications of tagging and chunking. Our result indicates that knowledgebase generated by our method significantly improves the performance.

We organize our paper as following. In Section 2, we review the adaptor grammar model, and discuss its inference procedure using *Markov chain Monte Carlo* (MCMC) method. We introduce our unsupervised entity extraction framework based on adaptor grammars in Section 3. To give an objective assessment on our proposed method, we conduct extensive experimental analysis and empirical study on web-scale query dataset collected from online shopping domain in Section 4. In Section 5, we review some past approaches on related problem. Finally, we conclude the paper and point out some possible future research directions in Section 6.



Prob	Rule
1.0	S → Brand Product
1.0	Brand → Words
1.0	Product → Words
0.7	Words → Word Words
0.3	Words → Word
0.1	Word → pottery
0.1	Word → barn
0.2	Word → coffee
0.2	Word → table
	...
?	Brand → pottery barn
?	Product → coffee table

Figure 1: An example of a phrase-structure tree and underlying probabilistic context-free grammar (PCFG). PCFGs assume the set of grammar rules are defined *a priori* and fixed. Therefore, rewriting operations are independent given the nonterminal, i.e., disregarding the yields and structure of the derivation tree. For example, in the query dataset, the highlighted derivation trees may appear several times—possibly with different *brand* or *product*—such derivation trees should be cached as a *new* grammar rule. Adaptor grammars [15, AG] break this independence assumption by jointly modeling the context and grammar, i.e., a nonparametric PCFG, allow to incorporate new context-dependent grammar rules from data.

2. ADAPTOR GRAMMARS

In this section, we discuss probabilistic context-free grammars and adaptor grammars.

2.1 Probabilistic Context-free Grammars

Probabilistic context-free grammars (PCFG) define probability distributions over derivations of a context-free grammar. A PCFG $\boldsymbol{\mathcal{G}}$ is defined as a set of variables $\langle \boldsymbol{W}, \boldsymbol{N}, \boldsymbol{R}, \boldsymbol{\theta} \rangle$. Given a collection of terminals $\boldsymbol{W}$ and a collection of nonterminals $\boldsymbol{N}$, $\boldsymbol{\mathcal{G}}$ is described by a set of probabilistic grammar rules $\langle \boldsymbol{R}, \boldsymbol{\theta} \rangle$. Let us denote the collection of all the rules rewriting a nonterminal c is $\boldsymbol{R}(c)$. For a nonterminal c, each of the grammar rules $r \mapsto \beta \in \boldsymbol{R}(c)$—commonly referred to as a *production*—is associated with a probability $\theta_{r \mapsto \beta}$, i.e., $\sum_{r \mapsto \beta \in \boldsymbol{R}(c)} \theta_{r \mapsto \beta} = 1$.

For any given sentence, a PCFG starts with the unique *start symbol* $S \in \boldsymbol{N}$, recursively rewrites a nonterminal into its derivations according to the probabilistic grammar rules $\langle \boldsymbol{R}, \boldsymbol{\theta} \rangle$. This builds a hierarchical *derivation tree structure*, starting from a nonterminal to a sequence of terminals, i.e., leaf nodes. This sequence of terminals often referred to as the *yield* of the derivation tree. Figure 1 illustrates an example of a derivation tree, its yields and underlying probabilistic context-free grammar (PCFG). Interested readers may refer to [21] for a more detailed description of PCFGs.

2.2 Adaptor Grammars

PCFGs assume the set of grammar rules are defined *a priori* and fixed. Therefore, rewriting operations are independent given the nonterminal, i.e., disregarding the yields and structure of the derivation tree. This context-freeness assumption often can be too strong for modeling natural language. Referring to Figure 1, in the query dataset, the derivation tree Brand $\mapsto$ pottery barn may appear over and over again, possibly with different products. If every time, such tree is constructed from raw PCFG rules, it is both time consuming and possibly error prone.

Adaptor grammars [15] break this independence assumption by jointly modeling the context and the grammar. It

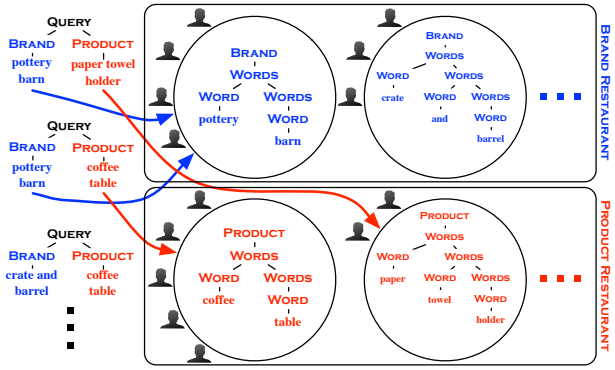


Figure 2: Illustration of the Chinese restaurant franchise.

is a nonparametric version of PCFG, which allows the model to incorporate new context-dependent grammar rules in a completely data-driven fashion. It specifies a set of adapted nonterminals, often called the *adaptors*. For each adaptor c, the model subsequently imposes a nonparametric prior on the distribution over its parse trees. This allows the model to dynamically learn more meaningful derivation trees and expand the rule set according to data. In general form, adaptor grammars use *Pitman-Yor process* [29, PYP] prior, which is also called the *Pitman-Yor adaptor grammar* (PYAG).

One realization of PYP is via the view of the *Chinese restaurant process* [13, CRP], parametrized by **scale parameter** a, **discount factor** b and **base distribution** G_c. The CRP assumes a set of tables, each of which serves a dish (a derivation tree in our context) randomly drawn from the base distribution, i.e., $z_c \sim G_c$. Each customer (an adapted nonterminal c in our context) entering the restaurant chooses a table to sit based on the choice of all previous customers (Figure 2). Let us denote K as the number of tables occupied by all past $n-1$ customers, and the sequence $x_1, \ldots, x_{n-1}$ represents the table indices that they sit at, i.e., $x_i \in \{1, \ldots, K\}$. The n-th customer choose to sit at table x_n

$$x_n | x_1, \ldots, x_{n-1} \sim \tfrac{Kb+a}{n-1+a}\delta_{K+1} + \textstyle\sum_{k=1}^{K} \tfrac{m_k - b}{n-1+a}\delta_k, \qquad (1)$$

where m_k is the number of customers sit at table k, i.e., $\sum_{k=1}^{K} m_k = n-1$. Variable δ_{K+1} stands for the case that a new table is chosen, i.e., a new instance of derivation tree is sampled. Therefore, the n-th customer chooses to sit at a new table $K+1$ with probability $\frac{Kb+a}{n-1+a}$ and an existing table k with probability $\frac{m_k - b}{n-1+a}$.

According to exchangability and de Finetti's theorem, all customers in CRP are mutually exchangeable, and do not alter the distribution, i.e., all m_k's remain the same. Therefore, for a given distribution over number of customers per table $\boldsymbol{n} = \{n_1, \ldots, n_K\}$, its probability is governed by PYP as

$$p_{\text{PYP}}(\boldsymbol{n} | a, b) = \frac{\prod_{k=1}^{K} (b(k-1)+a) \prod_{j=1}^{m_k - 1} (j-b)}{\prod_{i=0}^{n-1} (i+a)}, \qquad (2)$$

where K is the number of tables occupied and n is the number of observed samples drawn from CRP, i.e., total number of customers. The CRP formulation of PYP enables us to infer the latent variables in PYAG using *Markov chain Monte Carlo* (MCMC) method, which we will discuss in Section 2.3.

Formally, a Pitman-Yor adaptor grammar PYAG $\boldsymbol{\mathcal{A}}$ extends a PCFG and defined as following variables:

1: a collection of terminals $\boldsymbol{W}$, nonterminals $\boldsymbol{N}$ and context-free grammar rule set $\boldsymbol{R}$;
2: Dirichlet prior $\boldsymbol{\alpha}_c$ for the PCFG production probabilities $\boldsymbol{\theta}_c$ of each nonterminal $c \in \boldsymbol{N}$, i.e., $\boldsymbol{\theta}_c \sim \text{Dir}(\boldsymbol{\alpha}_c)$;
3: a set of non-recursive adapted nonterminals $\boldsymbol{M} \subseteq \boldsymbol{N}$;
4: PYP parameters a_c, b_c for each adaptor $c \in \boldsymbol{M}$.

In following section, we discuss in details about the inference process using a Metropolis-Hastings sampler.

2.3 MCMC Inference

The inference process is to learn the latent variables from observed data, i.e., $p(\boldsymbol{T}|\boldsymbol{X})$. To accomplish this, we first write down the joint distribution over their derivation trees $\boldsymbol{T}$ given a collection of sentences $\boldsymbol{X}$, is

$$\begin{aligned} p(\boldsymbol{T}|\boldsymbol{\alpha}, \boldsymbol{a}, \boldsymbol{b}) &= \textstyle\int_{\boldsymbol{\theta}, \boldsymbol{\pi}, \boldsymbol{z}} p(\boldsymbol{T}|\boldsymbol{\theta}, \boldsymbol{\pi}, \boldsymbol{z}) \cdot p(\boldsymbol{\theta}, \boldsymbol{\pi}, \boldsymbol{z}|\boldsymbol{\alpha}, \boldsymbol{a}, \boldsymbol{b}) \delta_{\boldsymbol{\theta}, \boldsymbol{\pi}, \boldsymbol{z}} \\ &= \textstyle\prod_{c \in \boldsymbol{N} \setminus \boldsymbol{M}} p_{\text{DIR}}(\boldsymbol{f}_c(\boldsymbol{T}) | \boldsymbol{\alpha}_c) \prod_{c \in \boldsymbol{M}} p_{\text{PYP}}(\boldsymbol{n}_c(\boldsymbol{T}) | a_c, b_c), \end{aligned}$$

where $\boldsymbol{n}_c(\boldsymbol{T})$ represents the frequency vector of all adapted rules for nonterminal c being observed in $\boldsymbol{T}$, and $\boldsymbol{f}_c(\boldsymbol{T})$ represents the frequency vector of all PCFG rules for nonterminal c being observed in $\boldsymbol{T}$. The posterior probability for Dirichlet $p_{\text{DIR}}(\boldsymbol{f}|\boldsymbol{\alpha})$ is

$$p_{\text{DIR}}(\boldsymbol{f}_c | \boldsymbol{\alpha}_c) = \tfrac{\Gamma(\sum_{k=1}^{K} \alpha_k)}{\Gamma(\sum_{k=1}^{K} f_k + \alpha_k)} \textstyle\prod_{k=1}^{K} \tfrac{\Gamma(f_k + \alpha_k)}{\Gamma(\alpha_k)},$$

where $K = |\boldsymbol{R}(c)|$ is the number of PCFG rules associated with c, and variables $\boldsymbol{f}$ and $\boldsymbol{\alpha}$ are both vectors of size K.

Given an observation string $\boldsymbol{x}_i$, in order to compute the posterior distribution over its derivation trees, we need to normalize $p(\boldsymbol{T}|\boldsymbol{\alpha}, \boldsymbol{a}, \boldsymbol{b})$ over all derivation trees that has yields $\boldsymbol{x}_i$. This probability is, unfortunately, intractable to compute, so we turn to *Markov chain Monte Carlo* (MCMC) method and iteratively sample $\boldsymbol{t}_i \sim p(\boldsymbol{t}_i | \boldsymbol{x}_i, \boldsymbol{T}_{-i})$. We construct an "auxiliary" PCFG variable $\boldsymbol{\mathcal{G}'}$ that emulates the PYAG behavior, and approximate the conditional distribution with it. The PCFG approximation $\boldsymbol{\mathcal{G}'} \equiv \langle \boldsymbol{W}, \boldsymbol{N}, \boldsymbol{R}', \boldsymbol{\theta}' \rangle$ can be viewed as a static snapshot of the PYAG given all the derivation trees $\boldsymbol{T}_{-i}$. Let us assume $\boldsymbol{T}_{-i}$ are observations from a PYAG $\boldsymbol{\mathcal{A}} \equiv \langle \boldsymbol{W}, \boldsymbol{N}, \boldsymbol{R}, \boldsymbol{\alpha}, \boldsymbol{M}, \boldsymbol{a}, \boldsymbol{b} \rangle$, the rule set $\boldsymbol{R}'$ in PCFG approximation $\boldsymbol{\mathcal{G}'}$ is defined as

$$\boldsymbol{R}' = \boldsymbol{R} \cup \{c \mapsto \mathsf{yields}(z) : c \in \boldsymbol{M}, z \in \boldsymbol{z}_c\}, \qquad (3)$$

where $\boldsymbol{z}_c$ represents the set of all derivation subtrees observed in $\boldsymbol{T}_{-i}$ that rooted at c, and $\mathsf{yields}(z)$ represents the yields of derivation tree z. Their corresponding rule probability is

$$\begin{aligned} \theta'_{c \mapsto \beta} = &\left(\tfrac{m_c b_c + a_c}{n_c + a_c} \right) \left(\tfrac{f_{c \mapsto \beta}(\boldsymbol{z}_c) + \alpha_{c \mapsto \beta}}{m_c + \sum_{c \mapsto \beta \in \boldsymbol{R}(c)} \alpha_{c \mapsto \beta}} \right) \\ &+ \textstyle\sum_{z \in \boldsymbol{z}_c : \mathsf{yield}(z) = \beta} \left(\tfrac{n_z - b_c}{n_c + a_c} \right), \end{aligned} \qquad (4)$$

where $f_{c \mapsto \beta}(\boldsymbol{z}_c)$ is the frequency count of observing production $c \mapsto \beta$ in all derivation tree set $\boldsymbol{z}_c$, and n_z is the frequency count of observing a particular tree z in all $\boldsymbol{z}_c$. Variable m_c is the number of unique derivation tree, i.e., $m_c = |\boldsymbol{z}_c|$, and n_c is the frequency count of observing all derivation trees rooted at c, i.e., $n_c = \sum_{z \in \boldsymbol{z}_c} n_z$.

The approximation PCFG $\boldsymbol{\mathcal{G}'}$ offers an efficient sampling alternative using Metropolis-Hastings to generate draws from the conditional $p(\boldsymbol{t}_i | \boldsymbol{x}_i, \boldsymbol{T}_{-i})$ [11]. We use it as our proposal distribution in a Metropolis-Hastings algorithm. Let us assume $\boldsymbol{t}_i$ is the current derivation tree, and $\boldsymbol{t}'_i$ is a sample from $p(\boldsymbol{t}_i | \boldsymbol{x}_i, \boldsymbol{T}_{-i})$, we accept the proposal with probability

$$A(\boldsymbol{t}_i \to \boldsymbol{t}'_i) = \min \left(1, \tfrac{p(\boldsymbol{t}'_i, \boldsymbol{T}_{-i} | \boldsymbol{\alpha}, \boldsymbol{a}, \boldsymbol{b})}{p(\boldsymbol{t}_i, \boldsymbol{T}_{-i} | \boldsymbol{\alpha}, \boldsymbol{a}, \boldsymbol{b})} \cdot \tfrac{p(\boldsymbol{t}_i | \boldsymbol{x}_i, \boldsymbol{\mathcal{G}'})}{p(\boldsymbol{t}'_i | \boldsymbol{x}_i, \boldsymbol{\mathcal{G}'})} \right). \qquad (5)$$

informational	where, what, which, who, why, when, can, could, shall, should, will, would, how, am, is, are, do, does, ...
navigational	http, www, com, inc, org, company brand, web, site, website, ...
transactional	# best, top #, rating, cost, verse, vs, diff(erence), comparison, compare, instruction(s), before and after, ...

Table 1: Some sample patterns and words for general domain-independent query filtering and preprocessing. We ignore all queries mapping to any of these pattern to reduce sparsity and modeling noises in query dataset.

The overall inference procedure iteratively updates $\boldsymbol{\mathcal{G}'}$, $\boldsymbol{\theta'}$ and $\boldsymbol{t}_i$ until convergence, which follows the standard paradigm:

1: Randomly initialize parse trees for all observations.
2: **while** model not converged **do**
3: Randomly choose observation $\boldsymbol{x}_i$ and its derivation tree $\boldsymbol{t}_i$.
4: Construct PCFG approximation $\boldsymbol{\mathcal{G}'}$ from $\boldsymbol{T}_{-i}$, i.e., update the rule set and probabilities according to Equation 3 and 4.
5: Sample a parse tree $\boldsymbol{t}'_i$ from $\boldsymbol{\mathcal{G}'}$ and accept the proposal according to Equation 5.
6: Adjust the associated counts with rules and grammars.

3. UNSUPERVISED ENTITY EXTRACTION

Our approach consists of three different steps. The first *preprocessing* step (Section 3.1) focuses on cleaning "out-of-domain" queries in different taxonomies (i.e., informational, navigational, and transactional queries) and regularizing the "in-domain" queries (filtering out domain-dependent stopwords). After that, we apply a *query standardization* step—which we discuss in Section 3.2—on all queries to further decrease their sparsity. Finally, in Section 3.3, we describe the grammar rules used for the adaptor grammar.

3.1 Preprocessing

One universal problem of query processing and understanding is the diversity of the queries, which introduces large amount of noise to modeling. The noise is due to possibly multiple reasons, for example, misspelling, tokenization. In addition to misspelling noise, following the general web search taxonomy [4], queries are usually of different forms, including informational (e.g., "what is gnc whey protein"), navigational (e.g., "www bobbi brown com"), and transactional (e.g., "loreal eye serum versus estee lauder eye cream" or "10 best man cologne"). Table 1 shows some of the general patterns and words for each category. In our settings, we ignore any queries matching these forms.

To reduce the noise during our modeling, we also collect some non-conventional stopwords commonly used in shopping queries. For example, marketing events (e.g., "sale", "promo", "deal"), shopping fashion (e.g., "online", "shipping"), price range (e.g., "discount", "cheap"), working condition (e.g., "used", "refurbished"), popularity (e.g., "popular", "hot", "cool"), quality control(e.g., "good", "nice"), etc. These stopwords can be applied universally on all domains, for example—in the scope of this paper—*apparel*, *electronics*, *health and beauty*, *home and garden*, and *sporting goods*. In practice, these non-conventional stopwords can be extracted using statistical methods, for instance, inverse document frequency. Besides such preprocessing step, to further reduce the sparsity over search patterns and word orderings, we subsequently perform a query standardization step.

3.2 Query Standardization

We collect a set of online shopping queries sampled during a 9-months interval. There are large number of queries that have different word orderings despite the fact that people were searching for the same brand and/or product. For example, the queries "*florajen probiotics*" and "*probiotics florajen*" lead to the same product ("*probiotics*") from the brand ("*florajen*"). We refer to the set of queries containing exactly the same bag of words as one unique ***query family***.

Theoretically, given n words, there can be $n!$ number of unique word orderings available. However, in practice, we do not observe all possible word sequences in the search log for that particular query. In fact, we discover that the distribution over all possible word sequences for any particular query family is very sparse, regardless of different query length n. In practice, we find out that approximately more than 90% of the query families appear in exact one word ordering in our search log, and less than 2% exhibit more than two word orderings, regardless of the query length. In addition to being sparse, the distribution over the word orderings tends to be highly skewed, regardless of the query length or the number of word orderings. Namely, for a query of length n, the number of word orderings actually being observed is significantly smaller than the number of all of its possible word orderings, i.e., $n!$. For instance, significant fraction (approximately 70% to 90%) of them have one predominant word ordering which accounts for more than 60% of the total traffic against all possible word orderings. This well suggests that, in online shopping domain, one query family usually searches for one particular brand and/or product. [3]

Often, users also keep entities as continuous phrases during search, i.e., do *not* interleaving across each other. For instance, in our random sample, we observe following different word orderings for "*miracle gel*" from "*sally hansen*":[4]

sally hansen miracle gel; (96.60%)
miracle gel sally hansen; (3.37%)
sally hansen gel miracle. (0.03%)

where all other possible sequence combinations of these words do not appear in the collected search queries at all. Note that, in this case, the *product* entity "*miracle gel*" gets rewritten to "*gel miracle*" in a small fraction of queries, but it remains continuous as one product entity. Therefore, it is fairly unlikely to see queries or word orderings like "*sally miracle gel hansen*" or "*miracle sally hansen gel*", etc. The sparsity property in word orderings reveals substantial information about online search queries, and essentially enables us to model them using adaptor grammar.

However, these less frequent word orderings certainly introduce a lot of noise during modeling. To address this problem, we perform a query standardization step to reduce the sparsity and group all the word orderings in a query family to the most frequent one. The idea of query standardization is

[3] This phenomena is frequently hold within some particular domain, but not necessarily across different domains. For instance, the query "*paris hilton*" is often referred to the person in **celebrity** domain, whereas "*hilton paris*" often searches for *hilton* hotel in the city of *paris* in **travel** domain.
[4] The number after shows the probability of observing the corresponding word ordering in search logs.

summarized as following:

1: **for** every query family **do**
2: Collect distribution $\{\mathbf{w}_i : c_i\}$, where c_i is the frequency associated with word ordering $\mathbf{w}_i$.
3: Find the word ordering with highest frequency, i.e., $\hat{\mathbf{w}} = \arg\max_{w_i} c_i$, and group the query family to word ordering $\hat{\mathbf{w}}$ with frequency $\sum_i c_i$.

Take the above examples as illustrations, query standardization step reduces multiple search patterns and word orderings to their most frequent ones like following:

sally hansen miracle gel (96.60%)
miracle gel sally hansen (3.37%)
sally hansen gel miracle (0.03%)
$\Rightarrow$ *sally hansen miracle gel*.

It greatly reduces the sparsity in search patterns and word orderings, hence, significantly improve the parsing performance. In our experiments, our study shows that on average about half of the unique word orderings are aggregated to their predominant patterns.

3.3 Grammar Rules

As previously discussed, users tend to construct their queries in online shopping domain with continuous entity phrases (e.g., *brand*, *product*), and do not interleave them across each other. This allows us to formulate the problem into a *chunking* problem, which is also commonly referred to as the *shallow parsing* of a sentence [1]. However, there are still challenges remaining on identifying *brand* and *product* entities from online shopping queries. One particular challenge in this case, is that, unlike sentences or paragraphs, they often lack of the grammatical information or syntactic structure, since majority of them consist of proper nouns or noun phrases [3]. For instance, *part-of-speech* (POS) tagging information, in this case, is often less accurate than structured sentences or documents. Therefore, classical chunking models do not often work very well.

In addition to word orderings being sparse and highly skewed for each query family, queries often adhere to some generic patterns in online shopping domain, for example, "BRAND PRODUCT". These query search patterns are preserved from aggregated query families. Similar to the distribution over word orderings, the distribution over these patterns is also sparse and highly skewed. Out of the queries searching for the same brand and/or product, we find that significant amount of them follow the pattern of "BRAND PRODUCT", despite the variants of word ordering in each entity type. In fact, for all search queries in shopping domain, pattern "BRAND PRODUCT" is more predominant than any other patterns, such as "PRODUCT BRAND". Such a property inspires us to solve the problem with an approach akin to shallow grammar parsing, and we want our chunking model to be as "context-free" as possible.

However, as discussed in Section 2, vanilla *probabilistic context-free grammars* (PCFG) model imposes strong independence assumption between grammar rules and internal structures of derivation trees, such context-freeness hypothesis may often lead to less accurate language modeling. Unlike PCFG—its parametric counterpart—adaptor grammars constantly cache the derivation trees and dynamically expands the grammar rule set in a completely data-driven fashion. Therefore, it provides a more flexible modeling option than PCFG. In addition, it is an unsupervised method, and hence does not require any human annotation efforts to extract the *brand* and *product* entities from a large collection of online search queries.

We start with a simple grammar that decomposes a query into a BRAND, a PRODUCT or a simple combination of both:

QUERY → BRAND	PRODUCT → WORDS
QUERY → PRODUCT	WORDS → WORD WORDS
QUERY → BRAND PRODUCT	WORDS → WORD
BRAND → WORDS	WORD → ...

The underlined nonterminals refer to the adaptors, i.e., the adapted nonterminal nodes, such that derivation trees rewriting them will be cached by the framework, and subsequently added to the grammar rule set. We refer this grammar as the **chunk** grammar.

However, in practice, we found that it is not enough to capture some other common patterns with prepositional phrases, for example, "*sephora shea butter for lips*", "*chin strap to stop snoring*", "*massage oils at walgreens*", "*nail polish bottles with brush*", etc. Therefore, we propose the **chunk+prep** grammar to explicitly modeling prepositional phrases:

QUERY → BRAND	BRAND → WORDS
QUERY → PRODUCT	PRODUCT → WORDS
QUERY → BRAND PRODUCT	WORDS → WORD WORDS
QUERY → QUERY PREPPHRASE	WORDS → WORD
PREPPHRASE → PREP WORDS	WORD → ...
PREP → for|to|by|with|at|...	

These prepositional phrases are sometimes informative, such that they reveal possible *brand* entity, e.g., "*at walgreens*", "*at cvs*". On the other hand, they can also be noisy, e.g., "*at home*", "*at night*". This grammar explicitly models all prepositional phrases. We leave the study of utilizing prepositional phrases to identify *brand* and *product* entities as future work.

This grammar, again, overlooks another important information, i.e., it does not capture the modifier information about the product. These modifiers could be descriptions, but more likely are model names or product families. For example, the word "*professional*" in the query "*andis professional hair clippers*" is a general description. The phrase "*despicable me minion*" in the query "*despicable me minion hoodie*" modifies a product. The word "*hdmi*" specifies a product family in the query "*iogear wireless hdmi transmitter and receiver*". The phrase "*galaxy s1*" in the query "*samsung galaxy s1 cell phone*" is a model number. The word "*tv*" specifies the purpose of the product in the query "*ge coaxial tv cable*". To address this, we further extend it to the **chunk+prep+mod** grammar:

QUERY → BRAND	BRAND → WORDS
QUERY → MODPROD	PRODUCT → WORDS
QUERY → BRAND MODPROD	MODIFIER → WORDS
QUERY → QUERY PREPPHRASE	WORDS → WORD WORDS
MODPROD → MODIFIER PRODUCT	WORDS → WORD
PREPPHRASE → PREP WORDS	WORD → ...
PREP → for|to|by|with|at|...	

We found this grammar works best in practice. In Section 4, we conduct experimental study on the performance under different grammars, and show empirical evaluation of our approach against the vanilla PCFG approach.

4. EXPERIMENTS

In this section, we report the empirical result of our approach on a large collection of web shopping queries. To the best of our knowledge, there are no off-the-shelf unsupervised methods available to automatically identify *brand* and *product* entities from online shopping search queries. We compare our approach against the vanilla probabilistic context-free

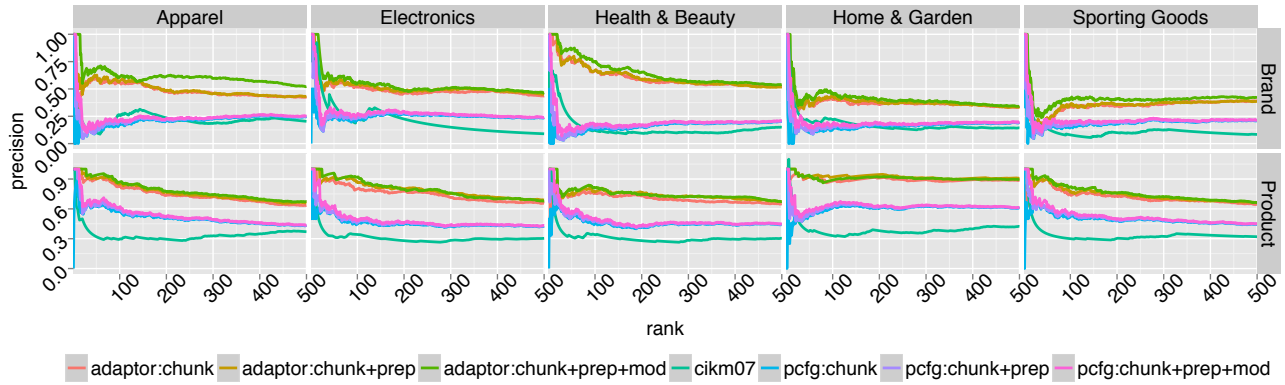


Figure 3: Precision against entity rank discovered by different approaches with different grammars regarding to 5 domains and 2 entity types. Precision is evaluated by professional editors. Our approach constantly achieves better performance.

grammar [21, PCFG] with the same parsing grammar. Different than adaptor grammars, which imposes a Pitman-Yor process prior on the distributions over grammar rules, we assume a Dirichlet distribution as the prior for PCFG. We learn the PCFG using Bayesian inference with a MCMC sampler [16]. This PCFG approach is akin to the shallow parsing approach [1] as discussed in Section 3.3, so serves as a comparable unsupervised baseline for our model. In addition, we also compare against a semi-supervised approach [25, CIKM07]. For every domain and entity type, we seed the model with a collection of 5 popular entities as suggested in [25]. The seeded list for corresponding *brand* and *product* entities for each domain are shown as below:

Apparel	*nike*, *michael kors*, *ugg*, *ray ban*, *kate spade* *shoe*, *dress*, *handbag*, *engagement ring*, *hat*
Electronics	*samsung*, *sony*, *dell*, *apple*, *canon* *printer*, *tablet*, *cell phone*, *remote control*, *laptop*
Health & Beauty	*estee lauder*, *gnc*, *urban decay*, *mac*, *opi* *supplement*, *shampoo*, *nail polish*, *soap*, *lipstick*
Home & Garden	*home depot*, *john deere*, *sears*, *target*, *lowes* *lights*, *chairs*, *curtains*, *lamps*, *pillows*
Sporting Goods	*under armour*, *nike*, *adidas*, *puma*, *kappa* *jacket*, *bike*, *pants*, *socks*, *helmet*

We first conduct intrinsic evaluation of our model against both PCFG and CIKM07 on knowledge discovery. Then, we assess the quality of the inferred knowledgebase extrinsically by using them as features on two external tasks.

4.1 Intrinsic Evaluation

The data we use are raw queries in shopping domain collected from a commercial search engine during a 9-month time interval. We retrieve all queries that have at least one online ads click, and randomly sample them. Based on the most frequent product category of their clicked ads, these queries are classified into following 5 different domains: *Apparel*, *Electronics*, *Health & Beauty*, *Home & Garden*, and *Sporting Goods*, which contains $906K$, $444K$, $625K$, $1.1M$ and $412K$ unique queries respectively. For both PCFG and our unsupervised approaches, we train them based on adaptor grammars on all available data. During our experiments, we let all MCMC samplers running for 2000 iterations to make sure all models are fully converged. For our unsupervised approach, we examine hyperparameter settings of $a = \{0.01, 0.05, 0.1\}$ and $b = \{100, 500, 1000\}$ for PYP. For the PCFG approach, recall that Dirichlet distribution is defined by its scaling parameter α, we examine different settings of $\{0.01, 0.05, 0.1\}$ in our experiments. For both unsupervised approaches, we report the performance on the model with highest data likelihood. We collect the distributions over all discovered entities and rank them according to their probabilities. For semi-supervised approach CIKM07, we rank all entities by Jensen-Shannon as suggested. We evaluate all approaches on the precision measure at different rank of the retrieved entity types.

Precision @ Rank. In order to give a comprehensive study of our approach, we looked into the precision of the retrieved entities at different rank for different approaches with different grammars. Precision measures the percentage of retrieved entities which correctly belong to an entity type. The precision is evaluated by professional editors judging if a retrieved entity belongs to the corresponding type. Figure 3 captures the performance on precision at different ranks of both *brand* and *product* entities discovered in dataset of each domain. Our approach constantly achieves better performance than CIKM07 and PCFG under all grammars. For grammars, we notice that **chunk+prep+mod** grammar yields better or comparable performance than other two grammars. We find that the semi-supervised approach CIKM07 often discovers entities in a greedy manner. For instance, in addition to "*apple*", it also ranks high on "*apple mac*" and "*hdmi cable for apple*" as *brand* names. Besides "*dress*", it also identifies "*calvin klein dress*" or "*black dress*" as *product*. In addition, we also find CIKM07 model takes much more memory resource and longer computation time to train, partly due to the large variance in search patterns. Subsampling the dataset increases the speed, but would downgrade the performance. In contrast, our proposed approach (and all PCFG-based approaches, in general) utilizes the context-free grammar rules, which are more compact, efficient and memory-friendly than flat representations. For *sporting goods* domain, all approaches yield low precision in identifying *brand* entity types. This is mainly due to the reason that our models discover a lot of sports team and club names (refer to Table 2 for examples). By our guidelines in Definition 1, even though these sport clubs are the branding or advertisement carriers for many products, they are considered *organizations*, rather than the actual *brand*. Instead, their respective sponsors—such as "*nike*", "*under armour*", etc—are.

Discovered Entities. Table 2 lists the most frequent *brands*, *products*, prepositional and modifier phrases identified by our approach based on adaptor grammars. Our approach is able to correctly identify complicated *brand* entities (e.g., "*tiffany and co*", "*at t*", "*bath and body works*", "*bed bath and beyond*"

	Apparel	Electronics	Health & Beauty	Home & Garden	Sporting Goods
Brand	michael kors	samsung	estee lauder	home depot	under armour
	north face	sony	gnc	walmart	dallas cowboys
	under armour	dell	walmart	john deere	nike
	vera bradley	samsung galaxy	urban decay	sears	green bay packers
	kate spade	apple	mac	target	denver broncos
	alex and ani	walmart	opi	lowes	chicago bears
	tiffany and co	bose	neutrogena	craftsman	seahawks
	columbia	rca	lancome	hamilton beach	notre dame
	oakley	at t	bath and body works	battery operated	orleans saints
	dooney and bourke	straight talk	too faced	bed bath and beyond	adidas
Product	running shoes	printer	supplement	lights	apparel
	wedding dresses	phone	hair products	chairs	jacket
	handbags	tablet	nail polish	curtains	accessories
	engagement ring	cell phone	soap	lamps	bike
	hats	remote control	lipstick	pillows	hoodie
	jeans	phone cases	eyeglass frames	cabinets	backpack
	necklace	cable	essential oil	bed	knife
	watch	ink cartridges	contact lenses	coffee table	pants
	rain boots	printers	protein powder	lighting fixtures	helmet
	sunglasses	adapter	shampoo and conditioner	shower curtain	socks
PrepPhrase	for toddlers	at walmart	for weight loss	at home depot	for football
	with sleeves	for laptop	for natural hair	for the home	for hunting
	for babies	with microphone	for thinning hair	for bathroom	for the money
	for teens	for mac	for plantar fasciitis	with lights	for baseball
	with pockets	with answering machine	for adults	with storage	for fishing
	for little girls	for the money	for fine hair	for living room	with wheels
	with hood	for ipad mini	at home	with wheels	for home
	with diamonds	for cars	for oily skin	with drawers	for bikes
	for school	for windows	for curly hair	by the yard	for beginners
	with rhinestones	for seniors	for babies	for crafts	for guns
Modifier	halloween	cell phone	blood pressure	dining room	football
	girls	wireless	hair	outdoor	golf
	leather	mini	natural	kitchen	pro
	gold	all in one	gel	stainless steel	fishing
	sexy	tv	weight loss	christmas	super bowl
	toddler	pro	liquid	electric	youth
	baby	phone	nail polish	christmas tree	baseball
	boys	portable	anti aging	cast iron	elite
	vintage	wifi	plus	furniture	training
	ladies	smart	pro	glass	marine

Table 2: Most frequent *brand* entities, *product* entities, prepositional and modifier phrases discovered by our approach based on adaptor grammars using ***chunk+prep+mod*** grammar for each shopping domain.

and "*green bay packers*") and *product* entities (e.g., "*engagement ring*", "*remote control*", "*shampoo and conditioner*", "*coffee table*" and "*hoodie*") of arbitrary length in a completely unsupervised data-driven fashion. One interesting observations is that our model discovers some joint phrases, such as "*samsung galaxy*", which is more like a combination of a *brand* entity ("*samsung*") with a model modifier "*galaxy*". This again is due to the nonparametric nature of the modeling framework, i.e., if the model sees a large amount of such phrases appear in the dataset, it would identify the entire phrase as a segment. On the other hand, PCFG is less capable in discovering such phrases, simply because it imposes too much independent assumption on the underlying grammar. One thing worth to note is, for the *sporting goods* category, we discover a lot of sports team names, these are due to the reason that queries fall into this domain are often looking for team jerseys, jackets, hats, etc.

In addition, our approach also discovers a lot of popular *prepositional phrases* for each shopping domain, e.g., "*for little girls*", "*with answering machine*", "*for oily skin*", "*by the yard*" and "*for beginners*", as well as some common modifier under each domain, e.g., "*halloween*", "*all in one*", "*anti aging*", "*stainless steel*" and "*super bowl*". These prepositional and modifiers phrases are modeled jointly with the *brand* and *product* entities, and provides additional information or constraints to the products.

4.2 Extrinsic Evaluation

In addition to intrinsic evaluation, we also conduct extrinsic empirical study on the quality of our discovered knowledgebase to external tasks. We use knowledgebase extracted from different approaches as features to external tasks and evaluate its effectiveness. For the purpose, we build, train and test two supervised algorithms—a query tagging model based on *conditional random fields* [18, CRF], and a query chunking model based on *maximum entropy* method [17, MAXENT].

For every category, we reserve a collection of $3K$ random samples, and manually annotate them with professional editors. In addition to *brand* and *product*, the label space also includes other entity types, such as model number, product family, attribute specification, etc. We split the labeled dataset by 80% for training and 20% for testing. For both CRF and MAXENT models, we use contextual features including words and their lemmas in a 5-word surrounding window; and lexical features including surface, pattern, shape and length of previous, current and next word. In addition, we also use the knowledgebase lookup features by examining if we find a match of the target word in any entity inside our extracted knowledgebase. We gradually increase the size of the knowledgebase, and evaluate the performance of CRF and MAXENT at different rank. We choose all parameters using 5-fold cross validation on the training data, and report the performance of the best settings.

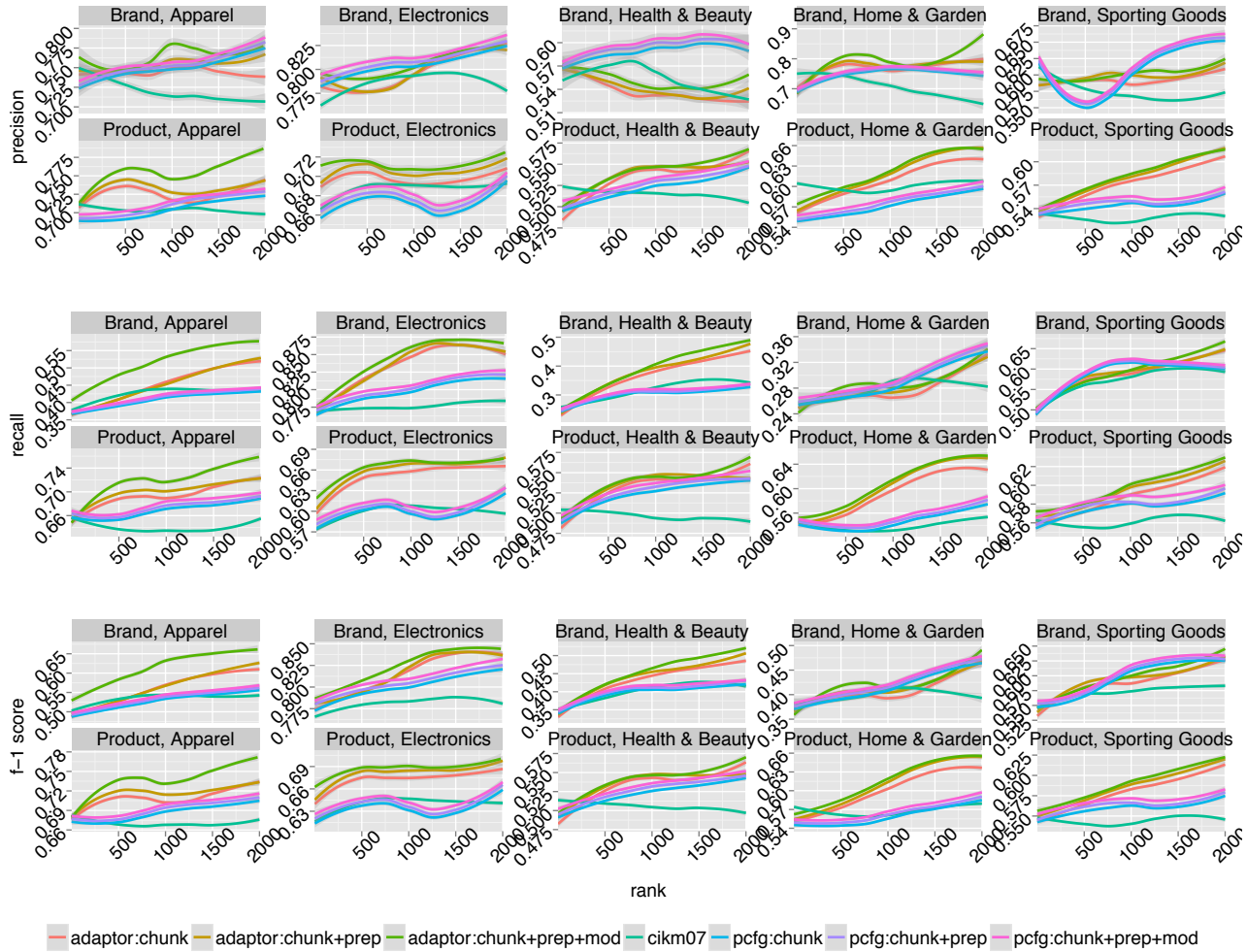


Figure 4: Precision, recall, and F_1 score of CRF tagging model on different entity types in 5 domains against the rank (size) of the knowledgebase discovered by different approaches under different grammars. Plots are smoothed to reflect clear trends. The sequence tagger constantly improves recall when we reveal more entries from the discovered knowledgebase to the model. This implies the knowledgebase discovered by our model is of good quality and provides better coverage to this tasks.

Query Tagging Model. Figure 4 shows the precision, recall, and F_1 score of CRF model on different entity types in 5 domains against the rank (size) of the knowledgebase used to generate features. These knowledgebases are extracted by different approaches with different grammars. We gradually increase the knowledgebase by the step size 50. The precision measure drops initially, but after we increase the knowledgebase to a certain threshold, which has sufficient coverage over all entity types, it starts to improve. On the recall measure, the performance of CRF tagger constantly improves when we reveal more entries from the discovered knowledgebase to the model. This implies the knowledgebase discovered by our model provides a good coverage to this tasks. The overall F_1 score demonstrates consistent improvements against entity rank. In addition, we show that the CRF sequence tagger constantly yields better performance when using knowledgebase extracted by our model based on adaptor grammar, as oppose to using the one discovered by the PCFG or CIKM07. This well indicates the knowledgebase discovered by our model is of good quality and hence can be used to improve the performance of external models.

Query Chunking Model. Figure 5 shows the precision, recall, and F_1 score of MAXENT model in 5 domains against the rank (size) of the knowledgebase used to generate features. We observe a similar behavior with previous query tagging task, such that the performance on recall and F_1 measure of MAXENT model constantly improves. This is possibly due to the reason that, as we reveal more entities from the knowledgebase discovered by our approach to the model, it establishes more sufficient coverage to the chunker. In contrast, the knowledgebases extracted by PCFG and CIKM07 models are not effective enough, and the respective performance on recall and F_1 measure does not change much. This again well implies the knowledgebase discovered by our model provides much better coverage than the ones extracted by PCFG or CIKM07 approach.

5. RELATED WORK

The objective of this work is to automatically identify *brand* and *product* entities from query logs in a complete unsupervised way. This is very different from classic *named entity recognition* (NER) problem, and can be significantly more challenging. First, unlike person names, organization titles or location labels, almost all of the products are *not*

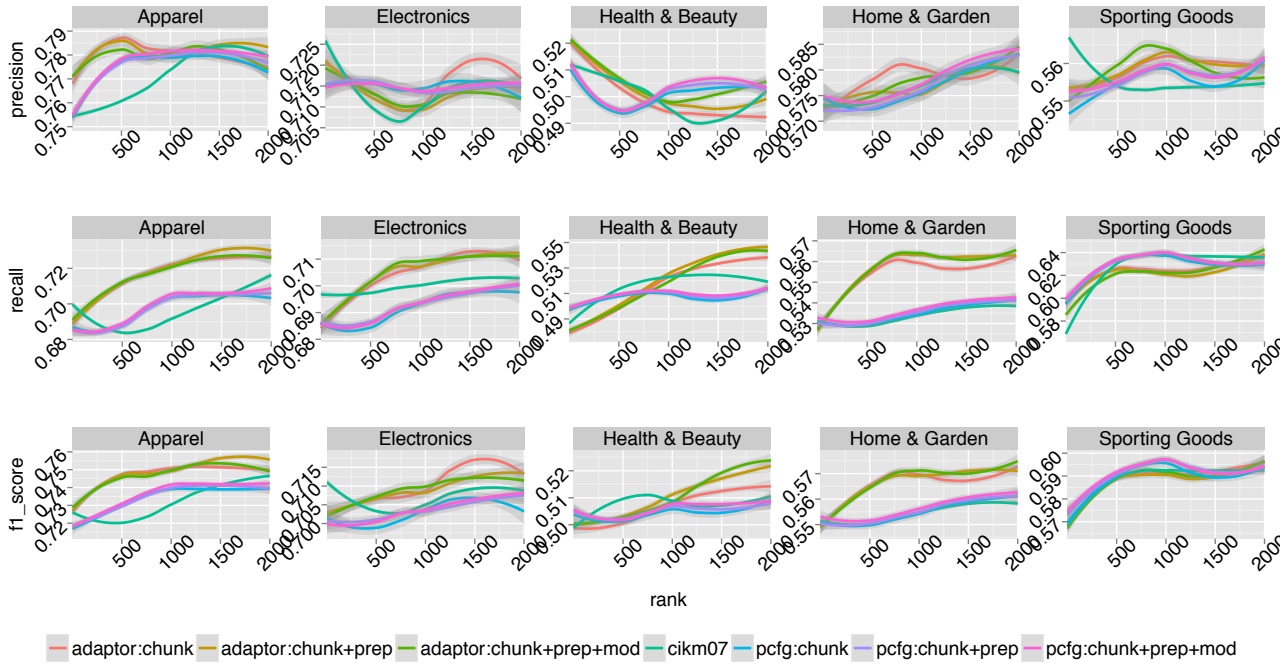


Figure 5: Precision, recall, and F_1 score of MAXENT chunking model in 5 domains against the rank (size) of the knowledgebase discovered by different approaches with different grammars. Plots are smoothed to reflect clear trends. As we reveal more entities from the discovered knowledgebase to the model, the performance over recall of MAXENT chunking model constantly improves. This indicates the knowledgebase discovered by our model is of good quality.

notable entity names. They often do not carry very indicative surface features (e.g., capitalization) nor grammar properties (e.g., structural patterns). For example, "Mr." and "Dr." may very well indicate person names; "Inc" and "Ltd" are strong indicators for organization titles; a location address usually follows the pattern of *street* + *city* + *state* or *province* + *zip codes*. Second, classic NER frameworks usually works on a collection of sentences or paragraphs, which contains rich contextual information and semantic structures. In our case, we operate the framework on a collection of web search queries, which are usually short and noisy (e.g., misspelling, arbitrary ordering, etc).

To the best of our knowledge, there have been very limited research efforts attempting to extract *brand* and *product* entities from query logs automatically. We summarize a few past approaches for NER in queries, either in a supervised or semi-supervised way.

The problem of open-domain entity recognition and extraction from web search queries is originally proposed in [25], which they rely on a seed-based weakly-supervised method. The method starts with a set of human labeled named entities—commonly referred to as seeds—and iteratively acquires new named entities explicitly from web queries. Li et al. [19] propose a semi-supervised approach based on conditional random fields method to extract entities from user queries. Guo et al. [12] introduce another weakly-supervised method, using partially labeled named entities as seeds, and train topic models on different domains. Pantel et al. [28] extend this work to jointly model user intent as latent variables in the framework. However, both of these methods are limited to the queries containing only one named entity.

Later, [14] propose a multi-stage method on different domains. The method starts with extracting named-entity candidates from query logs using surface-level properties, e.g., capitalization, etc. It then filters out the uncertain candidates and applies clustering based on different feature space. It works considerably well on domains like celebrities, cities, diseases, movies, etc, partly because queries of these domains usually carry along very indicative surface features.

In another study, Du et al. [7] focus on the domain of car models and use the entire search session as additional contextual information. They train both conditional random fields and topic models with new contextual features and demonstrate significant improvement. More recently, Alasiry et al. [2] determine named entities using grammar annotation and query segmentation with the help of additional snippets from web resources. Eiselt and Figueroa [8] propose a supervised two-step approach to identify named entities from open-domain search queries.

Unfortunately, all these works rely on semi-supervised methods with additional human annotations, which can be costly to acquire. To address this problem, we propose a completely unsupervised method based on adaptor grammar [15]. It does not require any human annotation efforts (e.g., named entity seeds or labeled data) which typically can be expensive to acquire. Our method also does not rely on additional web resources, such as query sequences in search sessions or web snippets.

6. CONCLUSION AND FUTURE WORK

Adaptor grammars [15] offer a great flexibility and modeling advantage in probabilistic context-free grammar parsing. In this paper, we apply the adaptor grammar model on a collection of queries in online shopping domain to extract *brand* and *product* entities efficiently and effectively. We propose a three step approach. The first step preprocesses the noisy query data and cleans up all stopwords. We then conduct a query standardization step, which aggregates and groups less

frequent word orderings, to further reduce the noise and regularize the queries. Finally, we propose three different sets of grammar rules to infer the query structures and extract the entities in a completely data-driven fashion. We compare our model against the vanilla PCFG approach—a variant akin to the shallow parsing approach—and demonstrate significant better performance on precision measure on retrieved *brand* and *product* entities. We also evaluate the effectiveness of the discovered knowledgebase on two external supervised tasks—a query tagging model based on conditional random fields and a query chunking model based on maximum entropy model. We show that the knowledgebase discovered by our framework is of high quality and significantly improves the overall performance of both models.

We would also like to point out some possible future directions of our work. One possible extension is to use the variational Bayesian inference [6] as oppose to the MCMC method in this paper. It offers a more scalable alternative and easier amendable to online learning [33] and parallelization [32]. One limitation of our work is that we do not explicitly model the *product families* and *model numbers* during parsing. Instead, we subsume them all into the adaptor nonterminal MODIFIER. Such information can be critical for a query understanding system to better understand the constraints and prompt more desirable results to users during search and ranking. In the future, we would like to explore along this direction to jointly model these entities in adaptor grammar framework.

7. REFERENCES

[1] S. P. Abney. *Parsing by chunks*. Springer, 1992.

[2] A. Alasiry, M. Levene, and A. Poulovassilis. Detecting candidate named entities in search queries. In *SIGIR*, 2012.

[3] C. Barr, R. Jones, and M. Regelson. The linguistic structure of english web-search queries. In *EMNLP*, 2008.

[4] A. Broder. A taxonomy of web search. *SIGIR Forum*, 36(2), Sept. 2002.

[5] M. J. Cafarella, D. Downey, S. Soderland, and O. Etzioni. Knowitnow: Fast, scalable information extraction from the web. In *EMNLP*, 2005.

[6] S. B. Cohen, D. M. Blei, and N. A. Smith. Variational inference for adaptor grammars. In *NAACL*, 2010.

[7] J. Du, Z. Zhang, J. Yan, Y. Cui, and Z. Chen. Using search session context for named entity recognition in query. In *SIGIR*, 2010.

[8] A. Eiselt and A. Figueroa. A two-step named entity recognizer for open-domain search queries. In *IJCNLP*, 2013.

[9] O. Etzioni, M. Cafarella, D. Downey, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates. Unsupervised named-entity extraction from the web: An experimental study. *Artificial intelligence*, 165(1), 2005.

[10] P. Exner and P. Nugues. Entity extraction: From unstructured text to dbpedia rdf triples. In *The Web of Linked Entities Workshop (WoLE 2012)*, 2012.

[11] J. T. Goodman. *Parsing Inside-out*. PhD thesis, Cambridge, MA, USA, 1998.

[12] J. Guo, G. Xu, X. Cheng, and H. Li. Named entity recognition in query. In *SIGIR*, 2009.

[13] H. Ishwaran and L. F. James. Generalized weighted chinese restaurant processes for species sampling mixture models. *Statistica Sinica*, 13(4), 2003.

[14] A. Jain and M. Pennacchiotti. Open entity extraction from web search query logs. In *COLING*, 2010.

[15] M. Johnson, T. L. Griffiths, and S. Goldwater. Adaptor grammars: A framework for specifying compositional nonparametric Bayesian models. In *NIPS*, 2007.

[16] M. Johnson, T. L. Griffiths, and S. Goldwater. Bayesian inference for PCFGs via Markov chain Monte Carlo. In *NAACL*, 2007.

[17] R. Koeling. Chunking with maximum entropy models. In *Proceedings of the 2nd workshop on Learning language in logic and the 4th conference on Computational natural language learning-Volume 7*, 2000.

[18] J. Lafferty, A. McCallum, and F. C. Pereira. Conditional random fields: probabilistic models for segmenting and labeling sequence data. In *ICML*, 2001.

[19] X. Li, Y.-Y. Wang, and A. Acero. Extracting structured information from user queries with semi-supervised conditional random fields. In *SIGIR*, 2009.

[20] B. Liu, R. Grossman, and Y. Zhai. Mining data records in web pages. In *KDD*, 2003.

[21] C. D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, MA, 1999.

[22] D. Nadeau. *Semi-supervised named entity recognition: learning to recognize 100 entity types with little supervision*. PhD thesis, 2007.

[23] D. Nadeau and S. Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1), 2007.

[24] J. Nothman, J. R. Curran, and T. Murphy. Transforming wikipedia into named entity training data. In *Proceedings of the Australian Language Technology Workshop*, 2008.

[25] M. Paşca. Weakly-supervised discovery of named entities using web search queries. In *CIKM*, 2007.

[26] M. Paşca. Queries as a source of lexicalized commonsense knowledge. In *EMNLP*, October 2014.

[27] M. Paşca. Interpreting compound noun phrases using web search queries. In *NAACL*, 2015.

[28] P. Pantel, T. Lin, and M. Gamon. Mining entity types from query logs via user intent modeling. In *ACL*, 2012.

[29] J. Pitman and M. Yor. The two-parameter Poisson-Dirichlet distribution derived from a stable subordinator. *Annals of Probability*, 25(2), 1997.

[30] L. Ratinov and D. Roth. Design challenges and misconceptions in named entity recognition. In *CoNLL*, 2009.

[31] T.-L. Wong, W. Lam, and T.-S. Wong. An unsupervised framework for extracting and normalizing product attributes from multiple web sites. In *SIGIR*, 2008.

[32] K. Zhai, J. Boyd-Graber, N. Asadi, and M. L. Alkhouja. Mr. lda: A flexible large scale topic modeling package using variational inference in mapreduce. In *WWW*, 2012.

[33] K. Zhai, J. Boyd-Graber, and S. Cohen. Online adaptor grammars with hybrid inference. *TACL*, 2, 2014.