Mining Entity Types from Query Logs via User Intent Modeling

Patrick Pantel
Microsoft Research
One Microsoft Way
Redmond, WA 98052, USA
ppantel@microsoft.com

Thomas Lin
Computer Science & Engineering
University of Washington
Seattle, WA 98195, USA
tlin@cs.washington.edu

Michael Gamon
Microsoft Research
One Microsoft Way
Redmond, WA 98052, USA
mgamon@microsoft.com

Abstract

We predict entity type distributions in Web search queries via probabilistic inference in graphical models that capture how entity-bearing queries are generated. We jointly model the interplay between latent user intents that govern queries and unobserved entity types, leveraging observed signals from query formulations and document clicks. We apply the models to resolve entity types in new queries and to assign prior type distributions over an existing knowledge base. Our models are efficiently trained using maximum likelihood estimation over millions of real-world Web search queries. We show that modeling user intent significantly improves entity type resolution for head queries over the state of the art, on several metrics, without degradation in tail query performance.

1 Introduction

Commercial search engines are providing ever-richer experiences around entities. Querying for a dish on Google yields recipe filters such as cook time, calories, and ingredients. Querying for a movie on Yahoo triggers user ratings, cast, tweets and showtimes. Bing further allows the movie to be directly added to the user's Netflix queue. Entity repositories such as Freebase, IMDB, Facebook Pages, Factual, Pricegrabber, and Wikipedia are increasingly leveraged to enable such experiences.

There are, however, inherent problems in the entity repositories: (a) coverage: although coverage of head entity types is often reliable, the tail can be sparse; (b) noise: created by spammers, extraction errors or errors in crowdsourced content; (c) ambiguity: multiple types or entity identifiers are often associated with the same surface string; and (d) over-expression: many entities have types that are never used in the context of Web search.

There is an opportunity to automatically tailor knowledge repositories to the Web search scenario. Desirable capabilities of such a system include: (a) determining the prior type distribution in Web search for each entity in the repository; (b) assigning a type distribution to new entities; (c) inferring the correct sense of an entity in a particular query context; and (d) adapting to a search engine and time period.

In this paper, we build such a system by leveraging Web search usage logs with large numbers of user sessions seeking or transacting on entities. We cast the task as performing probabilistic inference in a graphical model that captures how queries are generated, and then apply the model to contextually recognize entity types in new queries. We motivate and design several generative models based on the theory that search users' (unobserved) intents govern the types of entities, the query formulations, and the ultimate clicks on Web documents. We show that jointly modeling user intent and entity type significantly outperforms the current state of the art on the task of entity type resolution in queries. The major contributions of our research are:

- We introduce the idea that latent user intents can be an important factor in modeling type distributions over entities in Web search.
- We propose generative models and inference procedures using signals from query context, click, entity, entity type, and user intent.

- We propose an efficient learning technique and a robust implementation of our models, using real-world query data, and a realistic large set of entity types.
- We empirically show that our models outperform the state of the art and that modeling latent intent contributes significantly to these results.

2 Related Work

2.1 Finding Semantic Classes

A closely related problem is that of finding the semantic classes of entities. Automatic techniques for finding semantic classes include unsupervised clustering (Schütze, 1998; Pantel and Lin, 2002), hyponym patterns (Hearst, 1992; Pantel et al., 2004; Kozareva et al., 2008), extraction patterns (Etzioni et al., 2005), hidden Markov models (Ritter et al., 2009), classification (Rahman and Ng, 2010) and many others. These techniques typically leverage large corpora, while projects such as WordNet (Miller et al., 1990) and Freebase (Bollacker et al., 2008) have employed editors to manually enumerate words and entities with their semantic classes.

The aforementioned methods do not use query logs or explicitly determine the relative probabilities of different entity senses. A method might learn that there is independently a high chance of *eBay* being a *website* and an *employer*, but does not specify which usage is more common. This is especially problematic, for example, if one wishes to leverage Freebase but only needs the most commonly used senses (e.g., *Al Gore* is a `US Vice President`), rather than all possible obscure senses (Freebase contains 30+ senses, including ones such as `Impersonated Celebrity` and `Quotation Subject`). In scenarios such as this, our proposed method can increase the usability of systems that find semantic classes. We also expand upon text corpora methods in that the type priors can adapt to Web search signals.

2.2 Query Log Mining

Query logs have traditionally been mined to improve search (Baeza-Yates et al., 2004; Zhang and Nasraoui, 2006), but they can also be used in place of (or in addition to) text corpora for learning semantic classes. Query logs can contain billions of entries, they provide an independent signal from text corpora, their timestamps allow the learning of type priors at specific points in time, and they can contain information such as clickthroughs that are not found in text corpora. Sekine and Suzuki (2007) used frequency features on context words in query logs to learn semantic classes of entities. Paşca (2007) used extraction techniques to mine instances of semantic classes from query logs. Rüd et al. (2011) found that cross-domain generalizations learned from Web search results are applicable to NLP tasks such as NER. Alfonseca et al. (2010) mined query logs to find attributes of entity instances. However, these projects did not learn relative probabilities of different senses.

2.3 User Intents in Search

Learning from query logs also allows us to leverage the concept of *user intents*. When users submit search queries, they often have specific intents in mind. Broder (2002) introduced 3 top level intents: *Informational* (e.g., wanting to learn), *Navigational* (wanting to visit a site), and *Transactional* (e.g., wanting to buy/sell). Rose and Levinson (2004) further divided these into finer-grained subcategories, and Yin and Shah (2010) built hierarchical taxonomies of search intents. Jansen et al. (2007), Hu et al. (2009), and Radlinski et al. (2010) examined how to infer the intent of queries. We are not aware of any other work that has leveraged user intents to learn type distributions.

2.4 Topic Modeling on Query Logs

The closest work to ours is Guo et al.'s (2009) research on Named Entity Recognition in Queries. Given an entity-bearing query, they attempt to identify the entity and determine the type posteriors. Our work significantly scales up the type posteriors component of their work. While they only have four potential types (`Movie`, `Game`, `Book`, `Music`) for each entity, we employ over 70 popular types, allowing much greater coverage of real entities and their types. Because they only had four types, they were able to hand label their training data. In contrast, our system self-labels training examples by searching query logs for high-likelihood entities, and must handle any errors introduced by this process. Our models also expand upon theirs by jointly modeling entity type with latent user intents, and by incorporating click signals.

Other projects have also demonstrated the utility of topic modeling on query logs. Carman et al. (2010) modeled users and clicked documents to personalize search results and Gao et al. (2011) applied topic models to query logs in order to improve document ranking for search.

3 Joint Model of Types and User Intents

We turn our attention now to the task of mining the type distributions of entities and of resolving the type of an entity in a particular query context. Our approach is to probabilistically describe how entity-bearing queries are generated in Web search. We theorize that search queries are governed by a latent user intent, which in turn influences the entity types, the choice of query words, and the clicked hosts. We develop inference procedures to infer the prior type distributions of entities in Web search as well as to resolve the type of an entity in a query, by maximizing the probability of observing a large collection of real-world queries and their clicked hosts.

We represent a query q by a triple $\{n_1, e, n_2\}$, where e represents the entity mentioned in the query, n_1 and n_2 are respectively the pre- and post-entity contexts (possibly empty), referred to as *refiners*. Details on how we obtain our corpus are presented in Section 4.2.

3.1 Intent-based Model (IM)

In this section we describe our main model, **IM**, illustrated in Figure 1. We derive a learning algorithm for the model in Section 3.2 and an inference procedure in Section 3.3.

Recall our discussion of intents from Section 2.3. The unobserved semantic type of an entity e in a query is strongly correlated with the unobserved user intent. For example, if a user queries for "*song*", then she is likely looking to 'listen to it', 'download it', 'buy it', or 'find lyrics' for it. Our model incorporates this user intent as a latent variable.

The choice of the query refiner words, n_1 and n_2, is also clearly influenced by the user intent. For example, refiners such as "lyrics" and "words" are more likely to be used in queries where the intent is to 'find lyrics' than in queries where the intent is to 'listen'. The same is true for clicked hosts: clicks on "lyrics.com" and "songlyrics.com" are more likely to occur when the intent is to 'find lyrics', whereas clicks on "pandora.com" and "last.fm" are more likely for a 'listen' intent.

```
For each query/click pair $\{q, c\}$
    type $t \sim Multinomial(\tau)$
    intent $i \sim Multinomial(\theta_t)$
    entity $e \sim Multinomial(\psi_t)$
    switch $s_1 \sim Bernoulli(\sigma_i)$
    switch $s_2 \sim Bernoulli(\sigma_i)$
    if $(s_1)$ l-context $n_1 \sim Multinomial(\phi_i)$
    if $(s_2)$ r-context $n_2 \sim Multinomial(\phi_i)$
    click $c \sim Multinomial(\omega_i)$
```

Table 1: Model **IM**: Generative process for entity-bearing queries.

Model **IM** leverages each of these signals: latent intent, query refiners, and clicked hosts. It generates entity-bearing queries by first generating an entity type, from which the user intent and entity is generated. In turn, the user intent is then used to generate the query refiners and the clicked host. In our data analysis, we observed that over 90% of entity-bearing queries did not contain any refiner words n_1 and n_2. In order to distribute more probability mass to non-empty context words, we explicitly represent the empty context using a switch variable that determines whether a context will be empty.

The generative process for **IM** is described in Table 1. Consider the query "*ymca lyrics*". Our model first generates the type `song`, then given the type it generates the entity "*ymca*" and the intent 'find lyrics'. The intent is then used to generate the pre- and post-context words $\emptyset$ and "*lyrics*", respectively, and a click on a host such as "*lyrics.com*".

For mathematical convenience, we assume that the user intent is generated independently of the entity itself. Without this assumption, we would require learning a parameter for each intent-type-entity configuration, exploding the number of parameters. Instead, we choose to include these dependencies at the time of inference, as described later.

Recall that $q = \{n_1, e, n_2\}$ and let $s = \{s_1, s_2\}$, where $s_1 = 1$ if n_1 is not empty and $s_2 = 1$ if n_2 is not empty, 0 otherwise. The joint probability of the model is the product of the conditional distributions, as given by:

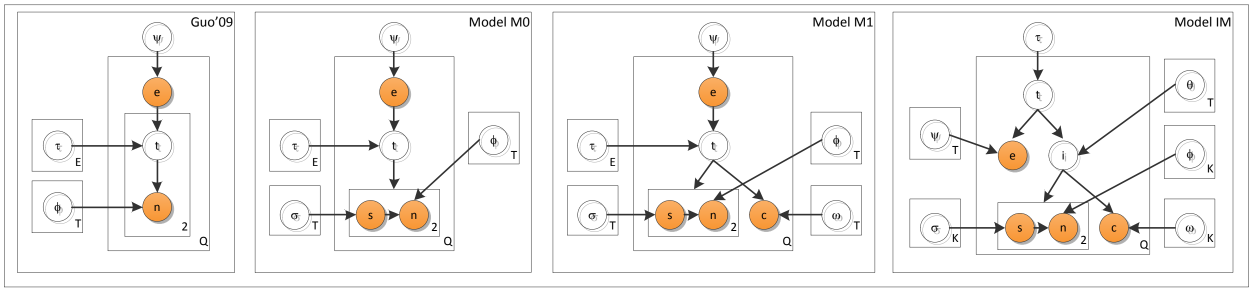

Figure 1: Graphical models for generating entity-bearing queries. $\mathbf{Guo'09}$ represents the current state of the art (Guo et al., 2009). Models $\mathbf{M0}$ and $\mathbf{M1}$ add an empty context switch and click information, respectively. Model $\mathbf{IM}$ further constrains the query by the latent user intent.

$$P(t, i, q, c \mid \tau, \Theta, \Psi, \sigma, \Phi, \Omega) = P(t \mid \tau) P(i \mid t, \Theta) P(e \mid t, \Psi) P(c \mid i, \Omega) \prod_{j=1}^{2} P(n_j \mid i, \Phi)^{I[s_j=1]} P(s_j | i, \sigma)$$

We now define each of the terms in the joint distribution. Let T be the number of entity types. The probability of generating a type t is governed by a multinomial with probability vector τ:

$$P(t=\hat{t}) = \prod_{j=1}^{T} \tau_j^{I[j=\hat{t}]} \text{, s.t. } \sum_{j=1}^{T} \tau_j = 1$$

where I is an indicator function set to 1 if its condition holds, and 0 otherwise.

Let K be the number of latent user intents that govern our query log, where K is fixed in advance. Then, the probability of intents i is defined as a multinomial distribution with probability vector θ_t such that $\Theta = [\theta_1, \theta_2, ..., \theta_T]$ captures the matrix of parameters across all T types:

$$P(i=\hat{i} \mid t=\hat{t}) = \prod_{j=1}^{K} \Theta_{\hat{t},j}^{I[j=\hat{i}]} \text{, s.t. } \forall t \sum_{j=1}^{K} \Theta_{t,j} = 1$$

Let E be the number of known entities. The probability of generating an entity e is similarly governed by a parameter Ψ across all T types:

$$P(e=\hat{e} \mid t=\hat{t}) = \prod_{j=1}^{E} \Psi_{\hat{t},j}^{I[j=\hat{e}]} \text{, s.t. } \forall t \sum_{j=1}^{E} \Psi_{t,j} = 1$$

The probability of generating an empty or non-empty context s given intent i is given by a Bernoulli with parameter σ_i:

$$P(s \mid i=\hat{i}) = \sigma_{\hat{i}}^{I[s=1]} (1 - \sigma_{\hat{i}})^{I[s=0]}$$

Let V be the shared vocabulary size of all query refiner words n_1 and n_2. Given an intent, i, the probability of generating a refiner n is given by a multinomial distribution with probability vector ϕ_i such that $\Phi = [\phi_1, \phi_2, ..., \phi_K]$ represents parameters across intents:

$$P(n=\hat{n} \mid i=\hat{i}) = \prod_{v=1}^{V} \Phi_{\hat{i},v}^{I[v=\hat{n}]} \text{, s.t. } \forall i \sum_{v=1}^{V} \Phi_{i,v} = 1$$

Finally, we assume there are H possible click values, corresponding to H Web hosts. A click on a host is similarly determined by an intent i and is governed by parameter Ω across all K intents:

$$P(c=\hat{c} \mid i=\hat{i}) = \prod_{h=1}^{H} \Omega_{\hat{i},h}^{I[h=\hat{c}]} \text{, s.t. } \forall i \sum_{h=1}^{H} \Omega_{i,h} = 1$$

3.2 Learning

Given a query corpus $\mathcal{Q}$ consisting of N independently and identically distributed queries $q^j = \{n_1^j, e^j, n_2^j\}$ and their corresponding clicked hosts c^j, we estimate the parameters τ, Θ, Ψ, σ, Φ, and Ω by maximizing the (log) probability of observing $\mathcal{Q}$. The $\log P(\mathcal{Q})$ can be written as:

$$\log P(\mathcal{Q}) = \sum_{j=1}^{N} \sum_{t,i} P^j(t, i \mid q, c) \log P^j(q, c, t, i)$$

In the above equation, $P^j(t, i \mid q, c)$ is the posterior distribution over types and user intents for the j^{th} query. We use the Expectation-Maximization (EM) algorithm to estimate the parameters. The parameter updates are obtained by computing the derivative of $\log P(\mathcal{Q})$ with respect to each parameter, and setting the resultant to 0.

The update for τ is given by the average of the posterior distributions over the types:

$$\tau_{\hat{t}} = \frac{\sum_{j=1}^{N} \sum_i P^j(t{=}\hat{t}, i \mid q, c)}{\sum_{j=1}^{N} \sum_{t,i} P^j(t, i \mid q, c)}$$

For a fixed type t, the update for θ_t is given by the weighted average of the latent intents, where the weights are the posterior distributions over the types, for each query:

$$\Theta_{\hat{t},\hat{i}} = \frac{\sum_{j=1}^{N} P^j(t{=}\hat{t}, i{=}\hat{i} \mid q, c)}{\sum_{j=1}^{N} \sum_i P^j(t{=}\hat{t}, i \mid q, c)}$$

Similarly, we can update Ψ, the parameters that govern the distribution over entities for each type:

$$\Psi_{\hat{t},\hat{e}} = \frac{\sum_{j=1}^{N} \sum_i P^j(t{=}\hat{t}, i \mid q, c) I[e^j{=}\hat{e}]}{\sum_{j=1}^{N} \sum_i P^j(t{=}\hat{t}, i \mid q, c)}$$

Now, for a fixed user intent i, the update for ω_i is given by the weighted average of the clicked hosts, where the weights are the posterior distributions over the intents, for each query:

$$\Omega_{\hat{i},\hat{c}} = \frac{\sum_{j=1}^{N} \sum_t P^j(t, i{=}\hat{i} \mid q, c) I[c^j{=}\hat{c}]}{\sum_{j=1}^{N} \sum_t P^j(t, i{=}\hat{i} \mid q, c)}$$

Similarly, we can update Φ and σ, the parameters that govern the distribution over query refiners and empty contexts for each intent, as:

$$\Phi_{\hat{i},\hat{n}} = \frac{\sum_{j=1}^{N} \sum_t P^j(t, i{=}\hat{i} | q, c) \Big[I[n_1^j{=}\hat{n}] I[s_1^j{=}1] + I[n_2^j{=}\hat{n}] I[s_2^j{=}1] \Big]}{\sum_{j=1}^{N} \sum_t P^j(t, i{=}\hat{i} | q, c) \Big[I[s_1^j{=}1] + I[s_2^j{=}1] \Big]}$$

and

$$\sigma_{\hat{i}} = \frac{\sum_{j=1}^{N} \sum_t P^j(t, i{=}\hat{i} \mid q, c) \Big[I[s_1{=}1] + I[s_2{=}1] \Big]}{2 \sum_{j=1}^{N} \sum_t P^j(t, i{=}\hat{i} \mid q, c)}$$

3.3 Decoding

Given a query/click pair $\{q, c\}$, and the learned **IM** model, we can apply Bayes' rule to find the posterior distribution, $P(t, i \mid q, c)$, over the types and intents, as it is proportional to $P(t, i, q, c)$. We compute this quantity exactly by evaluating the joint for each combination of t and i, and the observed values of q and c.

It is important to note that at runtime when a new query is issued, we have to resolve the entity in the absence of any observed click. However, we do have access to historical click probabilities, $P(c \mid q)$. We use this information to compute $P(t \mid q)$ by marginalizing over i as follows:

$$P(t \mid q) = \sum_i \sum_{j=1}^{H} P(t, i \mid q, c_j) P(c_j \mid q) \qquad (1)$$

3.4 Comparative Models

Figure 1 also illustrates the current state-of-the-art model **Guo′09** (Guo et al., 2009), described in Section 2.4, which utilizes only query refinement words to infer entity type distributions. Two extensions to this model that we further study in this paper are also shown: Model **M0** adds the empty context switch parameter and Model **M1** further adds click information. In the interest of space, we omit the update equations for these models, however they are trivial to adapt from the derivations of Model **IM** presented in Sections 3.1 and 3.2.

3.5 Discussion

Full Bayesian Treatment: In the above models, we learn point estimates for the parameters $(\tau, \Theta, \Psi, \sigma, \Phi, \Omega)$. One can take a Bayesian approach and treat these parameters as variables (for instance, with Dirichlet and Beta prior distributions), and perform Bayesian inference. However, exact inference will become intractable and we would need to resort to methods such as variational inference or sampling. We found this extension unnecessary, as we had a sufficient amount of training data to estimate all parameters reliably. In addition, our approach enabled us to learn (and perform inference in) the model with large amounts of data with reasonable computing time.

Fitting to an existing Knowledge Base: Although in general our model decodes type distributions for arbitrary entities, in many practical cases it is beneficial to constrain the types to those admissible in a fixed knowledge base (such as Freebase). As an example, if the entity is "ymca", admissible types may include `song`, `place`, and `educational_institution`. When resolving types, during inference, one can restrict the search space to only these admissible types. A desirable side effect of this strategy is that only valid ambiguities are captured in the posterior distribution.

4 Evaluation Methodology

We refer to *QL* as a set of English Web search queries issued to a commercial search engine over a period of several months.

4.1 Entity Inventory

Although our models generalize to any entity repository, we experiment in this paper with entities covering a wide range of web search queries, coming from 73 types in Freebase. We arrived at these types by grepping for all entities in Freebase within *QL*, following the procedure described in Section 4.2, and then choosing the top most frequent types such that 50% of the queries are covered by an entity of one of these types[1].

4.2 Training Data Construction

In order to learn type distributions by jointly modeling user intents and a large number of types, we require a large set of training examples containing tagged entities and their potential types. Unlike in Guo et al. (2009), we need a method to automatically label *QL* to produce these training cases since manual annotation is impossible for the range of entities and types that we consider. Reliably recognizing entities in queries is not a solved problem. However, for training we do not require high coverage of entities in *QL*, so high precision on a sizeable set of query instances can be a proper proxy.

To this end, we collect candidate entities in *QL* via simple string matching on Freebase entity strings within our preselected 73 types. To achieve high precision from this initial (high-recall, low-precision) candidate set we use a number of heuristics to only retain highly likely entities. The heuristics include retaining only matches on entities that appear capitalized more than 50% in their occurrences in Wikipedia. Also, a standalone score filter (Jain and Pennacchiotti, 2011) of 0.9 is used, which is based on the ratio of string occurrence as an exact match in queries to how often it occurs as a partial match.

The resulting queries are further filtered by keeping only those where the pre- and post-entity contexts (n_1 and n_2) were empty or a single word (accounting for a very large fraction of the queries). We also eliminate entries with clicked hosts that have been clicked fewer than 100 times over the entire *QL*. Finally, for training we filter out any query with an entity that has more than two potential types[2]. This step is performed to reduce recognition errors by limiting the number of potential ambiguous matches. We experimented with various thresholds on allowable types and settled on the value two.

The resulting training data consists of several million queries, 73 different entity types, and approximately 135K different entities, 100K different refiner words, and 40K clicked hosts.

4.3 Test Set Annotation

We sampled two datasets, *HEAD* and *TAIL*, each consisting of 500 queries containing an entity belonging to one of the 73 types in our inventory, from a frequency-weighted random sample and a uniform random sample of *QL*, respectively.

We conducted a user study to establish a gold standard of the correct entity types in each query. A total of seven different independent and paid professional annotators participated in the study. For each query in our test sets, we displayed the query, associated clicked host, and entity to the annotator, along with a list of permissible types from our type inventory. The annotator is tasked with identifying all applicable types from that list, or marking the test case as faulty because of an error in entity identification, bad click host (e.g. dead link) or bad query (e.g. non-English). This resulted in 2,092 test cases ({query, entity, type}-tuples). Each test case was annotated by two annotators. Inter-annotator agreement as measured by Fleiss' κ was 0.445 (0.498 on *HEAD* and 0.386 on *TAIL*), considered moderate agreement.

From *HEAD* and *TAIL*, we eliminated three categories of queries that did not offer any interesting type disambiguation opportunities:

- queries that contained entities with only one

[1]In this process, we omitted any non-core Freebase type (e.g., `/user/*` and `/base/*`), types used for representation (e.g., `/common/*` and `/type/*`), and too general types (e.g., `/people/person` and `/location/location`) identified by if a type contains multiple other prominent subtypes. Finally, we conflated seven of the types that overlapped with each other into four types (such as `/book/written_work` and `/book/book`).

[2]For testing we did not omit any entity or type.

	HEAD				TAIL			
	nDCG	**MAP**	$\mathbf{MAP_W}$	**Prec@1**	**nDCG**	**MAP**	$\mathbf{MAP_W}$	**Prec@1**
$\mathbf{B_{FB}}$	0.71	0.60	0.45	0.30	0.73	0.64	0.49	0.35
$\mathbf{Guo'09}$	$0.79^\dagger$	$0.71^\dagger$	$0.62^\dagger$	$0.51^\dagger$	$\mathbf{0.80}^\dagger$	$\mathbf{0.73}^\dagger$	$\mathbf{0.66}^\dagger$	$\mathbf{0.52}^\dagger$
M0	$0.79^\dagger$	$0.72^\dagger$	$0.65^\dagger$	$0.52^\dagger$	$\mathbf{0.82}^\dagger$	$\mathbf{0.75}^\dagger$	$\mathbf{0.67}^\dagger$	$\mathbf{0.57}^\dagger$
M1	$0.83^\ddagger$	$0.76^\ddagger$	$0.72^\ddagger$	$0.61^\ddagger$	$\mathbf{0.81}^\dagger$	$\mathbf{0.74}^\dagger$	$\mathbf{0.67}^\dagger$	$\mathbf{0.55}^\dagger$
IM	$\mathbf{0.87}^\ddagger$	$\mathbf{0.82}^\ddagger$	$\mathbf{0.77}^\ddagger$	$\mathbf{0.73}^\ddagger$	$\mathbf{0.80}^\dagger$	$\mathbf{0.72}^\dagger$	$\mathbf{0.66}^\dagger$	$\mathbf{0.52}^\dagger$

Table 2: Model analysis on HEAD and TAIL. † indicates statistical significance over $\mathbf{B_{FB}}$, and ‡ over both $\mathbf{B_{FB}}$ and $\mathbf{Guo'09}$. Bold indicates statistical significance over all non-bold models in the column. Significance is measured using the Student's t-test at 95% confidence.

potential type from our inventory;

- queries where the annotators rated all potential types as good; and
- queries where judges rated none of the potential types as good

The final test sets consist of 105 head queries with 359 judged entity types and 98 tail queries with 343 judged entity types.

4.4 Metrics

Our task is a ranking task and therefore the classic IR metrics **nDCG** (normalized discounted cumulative gain) and **MAP** (mean average precision) are applicable (Manning et al., 2008).

Both nDCG and MAP are sensitive to the rank position, but not the score (probability of a type) associated with each rank, $S(r)$. We therefore also evaluate a weighted mean average precision score $\mathbf{MAP_W}$, which replaces the precision component of **MAP**, $P(r)$, for the r^{th} ranked type by:

$$P(r) = \frac{\sum_{\hat{r}=1}^{r} I(\hat{r})S(\hat{r})}{\sum_{\hat{r}=1}^{r} S(\hat{r})} \qquad (2)$$

where $I(r)$ indicates if the type at rank r is judged correct.

Our fourth metric is **Prec@1**, i.e. the precision of only the top-ranked type of each query. This is especially suitable for applications where a single sense must be determined.

4.5 Model Settings

We trained all models in Figure 1 using the training data from Section 4.2 over 100 EM iterations, with two folds per model. For Model **IM**, we varied the number of user intents (K) in intervals from 100 to 400 (see Figure 3), under the assumption that multiple intents would exist per entity type.

We compare our results against two baselines. The first baseline is an assignment of Freebase types according to their frequency in our query set $\mathbf{B_{FB}}$, and the second is Model $\mathbf{Guo'09}$ (Guo et al., 2009) illustrated in Figure 1.

5 Experimental Results

Table 2 lists the performance of each model on the *HEAD* and *TAIL* sets over each metric defined in Section 4.4. On head queries, the addition of the empty context parameter σ and click signal Ω together (Model **M1**) significantly outperforms both the baseline and the state-of-the-art model $\mathbf{Guo'09}$. Further modeling the user intent in Model **IM** results in significantly better performance over all models and across all metrics. Model **IM** shows its biggest gains in the first position of its ranking as evidenced by the **Prec@1** metric.

We observe a different behavior on tail queries where all models significantly outperform the baseline $\mathbf{B_{FB}}$, but are not significantly different from each other. In short, the strength of our proposed model is in improving performance on the head at no noticeable cost in the tail.

We separately tested the effect of adding the empty context parameter σ. Figure 2 illustrates the result on the *HEAD* data. Across all metrics, σ improved performance over all models[3]. The more expressive models benefitted more than the less expressive ones.

Table 2 reports results for Model **IM** using $K = 200$ user intents. This was determined by varying K and selecting the top-performing value. Figure 3 illustrates the performance of Model **IM** with different values of K on the *HEAD*.

[3]Note that model **M0** is just the addition of the σ parameter over $\mathbf{Guo'09}$.

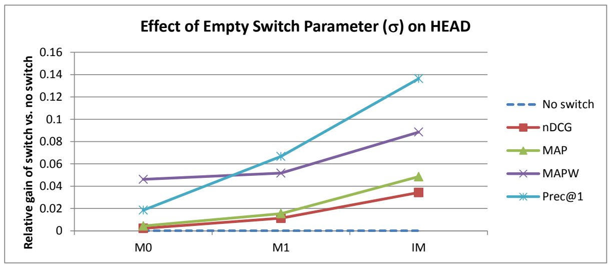

Figure 2: The switch parameter σ improves performance of every model and metric.

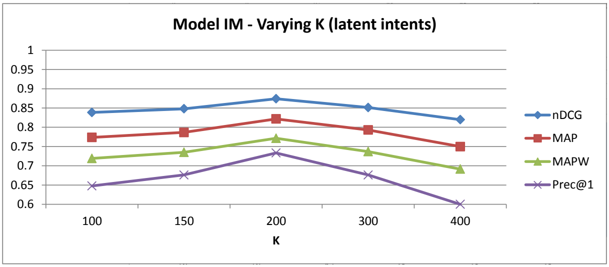

Figure 3: Model performance vs. the number of latent intents (K).

Our models can also assign a prior type distribution to each entity by further marginalizing Eq. 1 over query contexts n_1 and n_2. We measured the quality of our learned type priors using the subset of queries in our *HEAD* test set that consisted of only an entity without any refiners. The results for Model **IM** were: $nDCG = 0.86$, $MAP = 0.80$, $MAP_W = 0.75$, and $Prec@1 = 0.70$. All metrics are statistically significantly better than $\mathbf{B_{FB}}$, $\mathbf{Guo'09}$ and **M0**, with 95% confidence. Compared to Model **M1**, Model **IM** is statistically significantly better on $Prec@1$ and not significantly different on the other metrics.

Discussion and Error Analysis: Contrary to our results, we had expected improvements for both *HEAD* and *TAIL*. Inspection of the *TAIL* queries revealed that entities were greatly skewed towards people (e.g., `actor`, `author`, and `politician`). Analysis of the latent user intent parameter Θ in Model **IM** showed that most `people` types had most of their probability mass assigned to the same three generic and common intents for people types: 'see pictures of', 'find biographical information about', and 'see video of'. In other words, latent intents in Model **IM** are over-expressive and they do not help in differentiating people types.

The largest class of errors came from queries bearing an entity with semantically very similar types where our highest ranked type was not judged correct by the annotators. For example, for the query "*philippine daily inquirer*" our system ranked `newspaper` ahead of `periodical` but a judge rejected the former and approved the latter. For "*ikea catalogue*", our system ranked `magazine` ahead of `periodical`, but again a judge rejected `magazine` in favor of `periodical`.

An interesting success case in the *TAIL* is highlighted by two queries involving the entity "*ymca*", which in our data can either be a `song`, `place`, or `educational_institution`. Our system learns the following priors: 0.63, 0.29, and 0.08, respectively. For the query "*jamestown ymca ny*", **IM** correctly classified "*ymca*" as a `place` and for the query "*ymca palomar*" it correctly classified it as an `educational_institution`. We further issued the query "*ymca lyrics*" and the type `song` was then highest ranked.

Our method is generalizable to any entity collection. Since our evaluation focused on the Freebase collection, it remains an open question how noise level, coverage, and breadth in a collection will affect our model performance. Finally, although we do not formally evaluate it, it is clear that training our model on different time spans of queries should lead to type distributions adapted to that time period.

6 Conclusion

Jointly modeling the interplay between the underlying user intents and entity types in web search queries shows significant improvements over the current state of the art on the task of resolving entity types in head queries. At the same time, no degradation in tail queries is observed. Our proposed models can be efficiently trained using an EM algorithm and can be further used to assign prior type distributions to entities in an existing knowledge base and to insert new entities into it.

Although this paper leverages latent intents in search queries, it stops short of understanding the nature of the intents. It remains an open problem to characterize and enumerate intents and to identify the types of queries that benefit most from intent models.

References

Enrique Alfonseca, Marius Pasca, and Enrique Robledo-Arnuncio. 2010. Acquisition of instance attributes via labeled and related instances. In *Proceedings of SIGIR-10*, pages 58–65, New York, NY, USA.

Ricardo Baeza-Yates, Carlos Hurtado, and Marcelo Mendoza. 2004. Query recommendation using query logs in search engines. In *EDBT Workshops*, Lecture Notes in Computer Science, pages 588–596. Springer.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of SIGMOD '08*, pages 1247–1250, New York, NY, USA.

Andrei Broder. 2002. A taxonomy of web search. *SIGIR Forum*, 36:3–10.

Mark James Carman, Fabio Crestani, Morgan Harvey, and Mark Baillie. 2010. Towards query log based personalization using topic models. In *CIKM'10*, pages 1849–1852.

Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. 2005. Unsupervised named-entity extraction from the web: An experimental study. volume 165, pages 91–134.

Jianfeng Gao, Kristina Toutanova, and Wen-tau Yih. 2011. Clickthrough-based latent semantic models for web search. In *Proceedings of SIGIR '11*, pages 675–684, New York, NY, USA. ACM.

Jiafeng Guo, Gu Xu, Xueqi Cheng, and Hang Li. 2009. Named entity recognition in query. In *Proceedings of SIGIR-09*, pages 267–274, New York, NY, USA. ACM.

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th International Conference on Computational Linguistics*, pages 539–545.

Jian Hu, Gang Wang, Frederick H. Lochovsky, Jian tao Sun, and Zheng Chen. 2009. Understanding user's query intent with wikipedia. In *WWW*, pages 471–480.

Alpa Jain and Marco Pennacchiotti. 2011. Domain-independent entity extraction from web search query logs. In *Proceedings of WWW '11*, pages 63–64, New York, NY, USA. ACM.

Bernard J. Jansen, Danielle L. Booth, and Amanda Spink. 2007. Determining the user intent of web search engine queries. In *Proceedings of WWW '07*, pages 1149–1150, New York, NY, USA. ACM.

Zornitsa Kozareva, Ellen Riloff, and Eduard Hovy. 2008. Semantic class learning from the web with hyponym pattern linkage graphs. In *Proceedings of ACL.*

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine Miller. 1990. Wordnet: An on-line lexical database. volume 3, pages 235–244.

Marius Paşca. 2007. Weakly-supervised discovery of named entities using web search queries. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, CIKM '07, pages 683–690, New York, NY, USA. ACM.

Patrick Pantel and Dekang Lin. 2002. Discovering word senses from text. In *SIGKDD*, pages 613–619, Edmonton, Canada.

Patrick Pantel, Deepak Ravichandran, and Eduard Hovy. 2004. Towards terascale knowledge acquisition. In *COLING*, pages 771–777.

Filip Radlinski, Martin Szummer, and Nick Craswell. 2010. Inferring query intent from reformulations and clicks. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 1171–1172, New York, NY, USA. ACM.

Altaf Rahman and Vincent Ng. 2010. Inducing fine-grained semantic classes via hierarchical and collective classification. In *Proceedings of COLING*, pages 931–939.

Alan Ritter, Stephen Soderland, and Oren Etzioni. 2009. What is this, anyway: Automatic hypernym discovery. In *Proceedings of AAAI-09 Spring Symposium on Learning by Reading and Learning to Read*, pages 88–93.

Daniel E. Rose and Danny Levinson. 2004. Understanding user goals in web search. In *Proceedings of the 13th international conference on World Wide Web*, WWW '04, pages 13–19, New York, NY, USA. ACM.

Stefan Rüd, Massimiliano Ciaramita, Jens Müller, and Hinrich Schütze. 2011. Piggyback: Using search engines for robust cross-domain named entity recognition. In *Proceedings of ACL '11*, pages 965–975, Portland, Oregon, USA, June. Association for Computational Linguistics.

Hinrich Schütze. 1998. Automatic word sense discrimination. *Computational Linguistics*, 24:97–123, March.

Satoshi Sekine and Hisami Suzuki. 2007. Acquiring ontological knowledge from query logs. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, pages 1223–1224, New York, NY, USA. ACM.

Xiaoxin Yin and Sarthak Shah. 2010. Building taxonomy of web search intents for name entity queries. In *WWW*, pages 1001–1010.

Z. Zhang and O. Nasraoui. 2006. Mining search engine query logs for query recommendation. In *WWW*, pages 1039–1040.