Active Objects: Actions for Entity-Centric Search

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

Patrick Pantel, Michael Gamon
Microsoft Research
One Microsoft Way
Redmond, WA, USA
{ppantel,mgamon}@microsoft.com

Anitha Kannan, Ariel Fuxman
Search Labs
Microsoft Research
Mountain View, CA, USA
{ankannan,arielf}@microsoft.com

ABSTRACT

We introduce an entity-centric search experience, called *Active Objects*, in which entity-bearing queries are paired with actions that can be performed on the entities. For example, given a query for a specific flashlight, we aim to present actions such as reading reviews, watching demo videos, and finding the best price online. In an annotation study conducted over a random sample of user query sessions, we found that a large proportion of queries in query logs involve actions on entities, calling for an automatic approach to identifying relevant actions for entity-bearing queries. In this paper, we pose the problem of finding actions that can be performed on entities as the problem of probabilistic inference in a graphical model that captures how an entity bearing query is generated. We design models of increasing complexity that capture latent factors such as entity type and intended actions that determine how a user writes a query in a search box, and the URL that they click on. Given a large collection of real-world queries and clicks from a commercial search engine, the models are learned efficiently through maximum likelihood estimation using an EM algorithm. Given a new query, probabilistic inference enables recommendation of a set of pertinent actions and hosts. We propose an evaluation methodology for measuring the relevance of our recommended actions, and show empirical evidence of the quality and the diversity of the discovered actions.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Retrieval models

General Terms

Algorithms, Measurement, Experimentation, Theory

Keywords

Actions, Active Objects, Entity-Centric Search, Query Log Mining, Web Search

1. INTRODUCTION

Entities are central to a large fraction of Web search queries. Whether users seek to find information about an entity or transact on the entity (e.g., “[buy] toy story 3”, “[watch or listen to] obama weekly address”), understanding the underlying query intent is key to providing a rich search experience.

Web search today has already taken great strides away from simple query word matching. For example, popular entities in large query segments (e.g., local, entertainment, shopping) are routinely recognized in queries and rich direct displays are presented to users by filling editorially-defined templates with associated structured data. For example, a query for “lion king” on Bing yields such a direct display consisting of an image of the movie cover, showtimes at local theaters, the running time, genre, and ratings of the movie. However, since the focus is on the dominant actions, the search engine underserves, for instance, a Netflix user seeking other actions such as adding the movie to her streaming queue, or a child trying to find a toy figurine. In addition, a different movie such as Michael Moore’s most recent documentary would certainly have a different underlying intent distribution. Also, actions associated with queries for tail entities such as flashlights or small vineyards are completely ignored.

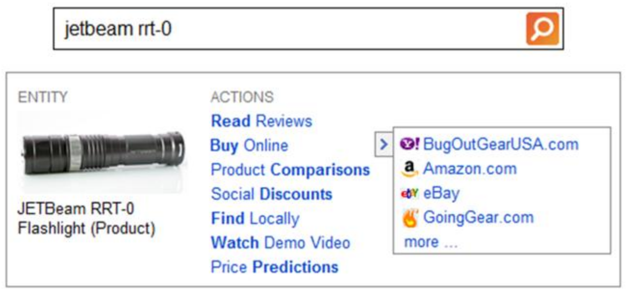

Figure 1. Search as an action broker.

Search as an action broker: A promising future search scenario involves modeling the user intents (or “verbs”) underlying the queries and brokering the webpages that accomplish the intended actions. In this vision, the broker is aware of all entities and actions of interest to its users, understands the intent of the user, ranks all providers of actions, and provides direct actionable results through APIs with the providers. For example, consider a user who queries for “jetbeam rrt-0”, a flashlight. The broker, which maintains a collection of all possible actions on flashlights and associated websites and applications that can accomplish those actions, would recognize the particular entity mentioned in the query, and would return a personalized ranked list of actions to the user. Figure 1 provides a simplistic illustration of how this user experience could look on a search results page. With actions present, users could save clicks and save time, and sometimes even discover new actions to help them toward their goals. New revenue streams open up from paid action placement, lead generation, and on-site commercial transactions.

This paper addresses several key questions that arise within this paradigm. Do Web queries tend to lend themselves to actions on entities? What does the space of actions look like? And most importantly, given a query with an entity (e.g., identified via a technique such as [18]), how can a search engine determine actions to recommend?

[*]This work was conducted at Microsoft Research.

WWW 2012, April 16–20, 2012, Lyon, France.
ACM 978-1-4503-1229-5/12/04.

We begin with an annotation study conducted over query-click logs from Bing to determine what fraction of queries contain entities, and whether these queries tend to map to particular actions that can be performed on entities.

The main problem that we address in this paper is how to *automatically* learn relevant actions for an entity-bearing query. An automated approach is necessary because there are too many possible distinct Web queries for editors to manually pair with actions. Also, manually preparing top actions for just entity types is insufficient for the following reasons. First, it would not account for context words in queries, e.g., the queries "Microsoft jobs" and "Microsoft software" should lead to different actions despite sharing the same entity (Microsoft). Second, entities of the same type can have different top actions, e.g., queries for a 2012 Ferrari may historically lead to clicks on topcarwallpapers.com while queries for a 1995 Ford historically lead to clicks on a used car value site. Last, but not least, top actions for an entity may change over time. For example, a query for the next iPhone would have different desired actions a year before launch, a week before launch, at launch, and a month after launch. The use of automated methods enables frequent re-training through a more recent data set of query-click logs. We motivate and design generative models, the most complex of which accounts for queries, entities, actions, textual query contexts, entity types, and historical click data.

We also explore a number of issues that need to be addressed in going from a theoretical generative model of actions to an actual end-to-end search engine component that is able to recommend appropriate actions when given a new query. We conclude with a user study evaluating the performance of our various models.

The major contributions of our research are:

- **Conceptual:** We introduce the *active objects* paradigm. We establish that there are specific Web actions for users to perform on named entities. We conduct an annotation study which empirically verifies that a large proportion of sampled query-click pairs reflect actions on entities.
- **Modeling:** We propose probabilistic models to generate entity bearing queries from actions, incorporating information from entity types, query words around an entity, and clicked hosts.
- **Implementation:** We train our models on three months of query data from a commercial search engine, and address the necessary end-to-end system issues for producing a system to recommend suitable actions for new queries.
- **Experimental:** We conduct a user-study to evaluate our different models, showing which model components are most important for generating actions.

2. RELATED WORK

Related work that we build upon includes entity-centric search and intents. Work that we differentiate ourselves from includes previous work on actions and topic modeling using query logs.

2.1 Entity-Centric Search

As proposed in Dalvi et al. [8], Entity-Centric search focuses on creating a "semantically rich aggregate view of all the information available on the Web for each concept instance". Researchers have typically focused on techniques for automatic generation of topic pages based on entities (e.g., [1] [20]), or on tailored information for particular entity classes (e.g., popular search engines displaying sports scores when given a sports team query). To the best of our knowledge we are the first to propose learning and presenting specific sets of actions for each entity.

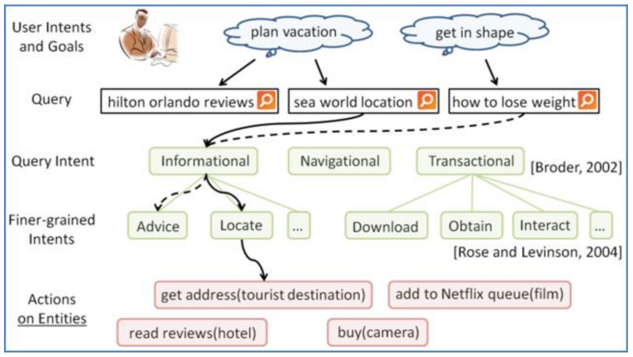

Figure 2. Actions differ from Intents: they must be performed on Entities, and are often more specific and grounded.

In recent years, many projects have aimed to enumerate named entities and their types. Some popular examples include Wikipedia (3.6M pages covering 800K categories), Freebase (20M topics covering 2K types), Facebook (85M+ fan pages covering 206 categories), Amazon.com (125M products covering 30 departments), and Schema.org (approx. 400 types). The presence of these lists facilitates the identification of entities within text [6] and queries [18].

2.2 Intents

Queries can be associated with higher-level *intents* such as "planning a vacation" or "getting in shape" (see Figure 2). Broder [3] outlined three main intents: informational, navigational, and transactional. Rose and Levinson [19] further divided informational and navigational intents into subcategories resulting in 11 finer-grained intents. Yin and Shah [22] used search logs to organize taxonomies of intent phrases, and Jansen et al. [15] studied how to classify queries into intents.

Our notion of *actions* is at an even finer level. Actions are very specific versions of intents that are performed on entities. Some actions overlap with finer-grained intents (e.g., "download"), but the majority of intents (e.g., "interact") are not concrete enough to be suggested to users. While some queries map easily into both intents and actions (e.g, "sea world location" in Figure 2), there are also queries that have a clear intent but do not contain any entity and hence cannot be associated with an action (e.g., "how to lose weight"). Task intents consisting of multiple actions (e.g., "book trip") are also out of scope.

2.3 Actions

Actions and action ontologies have been previously explored in robotics, intelligent agents, and philosophy (e.g., [16] [17]), but the primary focus in those areas was to develop a standardized set of actions (with pre-conditions and post-conditions) that could guide the planning processes of intelligent agents. In contrast, when we refer to an *action for Web search*, we refer to actions for human users to perform over the Web. Most of these actions (e.g., "read reviews" or "download") have no important prerequisites, while for those that do (e.g., "add to Netflix queue" requires a Netflix account), we assume that the preconditions can be addressed based on information known about the users.

2.4 Use of Probability Models

There has been prior work in using probability models for modeling user queries. For example, Carman et al. [5] extended Latent Dirichlet Allocation (LDA) [2] to rank documents by likelihood of the model given a particular query and user pair. Their model accounted for users, clicked documents, and query terms. Gao et al. [10] adapted statistical machine translation techniques to learn how document titles are semantic translations of queries. Guo et al. [11] used probability models to identify named entities and entity classes from query logs. Our work differs in that the primary focus of our models is on learning *actions*, a variable which other studies have not modeled.

3. ACTIONS ANNOTATION STUDY

We begin with a manual study of entities and actions in Web searches. We collect a frequency-weighted query sample of 200 query-click pairs. We examine each query to determine whether it contains an entity and whether we can infer an action that the user intends to accomplish given the query and the clicked host. Although one can only observe trends on such a small sample set, these results will serve as a guide for our automatic action induction models described in Section 4.

Throughout this paper, we define an action as follows:

Action: An empirically observable, direct manipulation or information request on an entity.

We target actions that are useful in the context of Web search. For example, "*interact*" is too coarse and "*drink*" is not an action that can be accomplished on the Web. Examples of useful actions are: "*buy*", "*add to movie queue*" and "*read reviews*".

3.1 Entities in Queries

We divided queries into four groups with respect to the presence of an entity in the query: (i) contains an entity; (ii) contains an entity category (e.g., "car battery"); (iii) contains a website entity (URL or website name); and (iv) all other queries. Figure 3 summarizes the frequency of each group, further separating out whether a query contains refiner words (e.g., "download *GoldenEye*" with the refiner "download") in addition to the entity or entity category.

43% of the queries contain an entity (29% by itself, 14% with a refiner), 14% contain an entity category (4% by itself, 10% with a refiner), 28% contain a reference to a website, and 15% do not contain any entity. Website references often occur in navigational queries where the user intends to visit a particular site, which leaves 57% of queries (43% + 14%) that have entities or entity categories. None of our annotated queries contain multiple entities. Guo et al. [11] found that 71% of search queries contained named entities, although they neither specify whether they consider frequency of individual queries, nor how they classify entity categories and website entities. Summing our entity, entity + refiner, and website entity categories, we end up with a proportion of entities in queries matching their results.

Next, we examined the *types* (or taxonomy *categories*) of the entities that we found. For entity types, we refer to the top level of the Schema.org entity taxonomy, which is a collection of schemas developed jointly by Bing, Google and Yahoo, designed explicitly with the intent of facilitating Web search over entities on the Web.

Within our sample of entities, we found that the most popular Schema.org top-level category was *CreativeWork* at 40%. This is a fairly broad category that covers all books, movies, songs, software, etc. The category *Organization* covered 37% of our entity sample. The *Organization* type covers hotels, restaurants, government organizations, local businesses, etc. There was also *Product* at 9%, and *Person* type at 8%. *Event* type covered 3% and the last 3% fell into other various types.

3.2 Actions in Queries

Next, we examined how often the queries in our sample can be associated with specific actions on entities. We also verify whether the actions in Web search are enumerable.

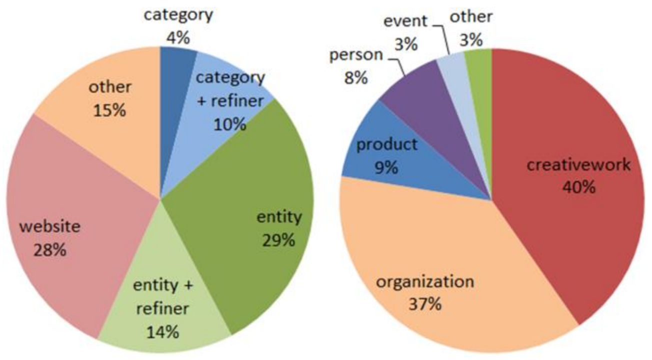

Figure 3. *Left*: Entity distribution at 200 labels. 43% of the queries contained entities, and 14% had entity categories. *Right*: Distribution of entities into Schema.org types.

We manually inferred the actions that are associated with each sample query by examining the raw query strings (consisting of entities and possible refiner words), and the clicked URLs. In the majority of cases, this information clearly indicated a particular action (e.g., "yahoo messenger download" clearly indicates the action "*download*"). In the absence of refiners in the query, the clicked URL generally gives a good signal to identify the action. For example, a query for "Hobart corporation" with a click on "http://hobartcorp.com/Contact-Us/" indicates the intended action "*get contact information*". 19 of the query/URL pairs in our sample were ambiguous with respect to the intended action, e.g. "GEICO insurance"-"www.geico.com", where the specific intended action is not clear. In some of these cases we took interesting potential actions (e.g., "see menu" on a restaurant URL) and added them to our inventory of actions.

From our 200 queries, we compiled a list of 47 actions. Some of the most popular actions included "*login,*" "*play game,*" "*read news about*", and "*shop for*". Less common but still interesting actions include "*find recipe for*", "*find lyrics*", and "g*et hours of*".

Working through the 200 queries, our discovery rate of new actions dropped from over 20 distinct actions for the first 50 labels to fewer than 5 new actions for the last 50 queries. This suggests that there is an enumerable primary set of actions that users perform in the context of Web search.

4. AUTOMATIC ACTION INDUCTION

We turn our attention now to the tasks of automatically learning the underlying actions intended in Web search as well as to recommending actions given new queries. Our approach is to probabilistically describe how *actionable queries*, i.e., queries containing an entity and underlying action intent, are generated by Web search users: Our models capture the latent actions and entity types that influence the terms in the user queries and the resultant clicks on hosts. Probabilistic inference in the corresponding learned models provide actions pertinent to the queries. The models are learned by maximizing the probability of observing a large collection of real-world queries and their clicked hosts.

In this section we present two graphical models (summarized in Figure 4). To generate queries from actions, our Model 1 models query context and clicked URLs. Model 2 builds on Model 1 by also modeling entity types, and explicitly observing entities. Then, we propose an extension to each model that adds a switch variable to better handle queries with empty contexts.

Each query q is represented 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. As a running example, we consider a user who is interested in reading a review about the movie "Inception", and who issues the query "inception review" to a search engine. Here $n_1 = \varnothing$, e = "inception" and n_2 = "review". Details on how we obtain our corpus are presented in Section 5.

4.1 Model 1 (context + click)

The choice of refiner words in a query is clearly influenced by the intended action. For example, words such as "review", "ebert", and "opinion" are more likely to be used in a query if the intent is to read a review. Host clicks are also correlated with action intents. For example, clicks on "rottentomatoes.com", "epinions.com" and "dpreview.com" are more likely if the user has the intent to read reviews, whereas clicks on "bestbuy.com" and "ebay.com" are more likely for a buying intent. Broder et al. [4] also found hosts associated with queries to be useful in classifying queries.

Our first probabilistic graphical model, Model 1, leverages these signals. It generates actionable queries by first picking an action from a distribution over a set of latent actions, then choosing query context words n_1 and n_2, and then clicking on a host c. This model does not explicitly capture the entity in the query, and hence a query is represented by the pair $\{n_1, n_2\}$. The generative process below summarizes the model illustrated on the left in Figure 4:

```
Model 1: Generative model of actionable queries.
For each query $q$
    action $a$ ~ Multinomial($\theta$)
    l-context $n_1$ ~ Multinomial($\phi_a$)
    r-context $n_2$ ~ Multinomial($\phi_a$)
    click $c$ ~ Multinomial($\omega_a$)
```

In our running example for the query "*inception review*", our model first generates the action "*read reviews*", then given this action chooses the refiner words $\varnothing$ and "*review*" and then generates a click on a site such as "rottentomatoes.com".

The joint probability of the model is the product of the conditional distributions, as given by:

$$P(a, q{=}\{n_1, n_2\}, c \mid \theta, \Phi, \Omega) = P(a \mid \theta) P(n_1 \mid a, \Phi) P(n_2 \mid a, \Phi) P(c \mid a, \Omega)$$

Next, we define each of the terms in the joint distribution. Let K be the number of latent actions that govern our query log, where K is fixed in advance. Then, the probability of actions a is defined as a multinomial distribution with probability vector θ, such that the probability of a particular action is given by:

$$P(a{=}\hat{a}) = \prod_{k=1}^{K} \theta_k^{I[k=\hat{a}]} \text{ , s.t. } \sum_k \theta_k = 1$$

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

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

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

Finally, we assume there are H possible click values, corresponding to H Web hosts. A click on a host is determined by an action. Given an action a, we assume the probability of generating a click on host c is a multinomial with a probability vector ω_a such that $\Omega = [\omega_1, \ldots, \omega_K]$ captures the matrix of parameters across all K actions. In particular:

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

Inference: Given a query, we apply Bayes' rule to find the posterior distribution over the actions. In particular, the posterior distribution, $P(a|q,c)$, is directly proportional to the joint distribution. We can exactly compute this distribution by evaluating the joint for every value of a and the observed configuration of q and c.

Learning: Given a query corpus $\mathcal{Q}$ consisting of N independently and identically distributed queries (each $q^j = \{n_1^j, n_2^j\}$) and their corresponding clicked hosts, we estimate the parameters θ, Φ, and Ω that maximize the (log) probability of observing $\mathcal{Q}$. The log $P(\mathcal{Q})$ can be written as:

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

In the above equation, $P^j(a|q,c)$ is the posterior distribution over actions for the j^{th} query. We use the Expectation-Maximization (EM) algorithm to set the parameters. Starting with a random initialization of the parameters, EM iterates between the E-step in which $P^j(a|q,c)$ is computed for each query (assuming parameters are fixed as computed in the previous M-step) and the M-step in which the parameters are updated by fixing $P^j(a|q,c)$ to the values computed in the E-step.

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 θ is given by the average of the posterior distributions over the actions:

$$\theta_{\hat{a}} = \frac{\sum_{j=1}^{N} P^j(a{=}\hat{a} \mid q, c)}{\sum_{j=1}^{N} \sum_a P^j(a \mid q, c)}$$

For a fixed a, the update for ϕ_a is given by the weighted average of the context words, where the weights are the posterior distributions over the actions, for each query. In particular:

$$\Phi_{\hat{a},\hat{n}} = \frac{\sum_{j=1}^{N} P^j(a{=}\hat{a} \mid q, c)\left[I[n_1^j{=}\hat{n}] + I[n_2^j{=}\hat{n}]\right]}{2\sum_{j=1}^{N} P^j(a{=}\hat{a} \mid q, c)}$$

Similarly, we can update Ω, the parameters that govern the distribution over clicked hosts for each action. For a fixed a, it is updated by taking the weighted average of the clicked hosts, with weights provided by the posterior distribution over the actions:

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

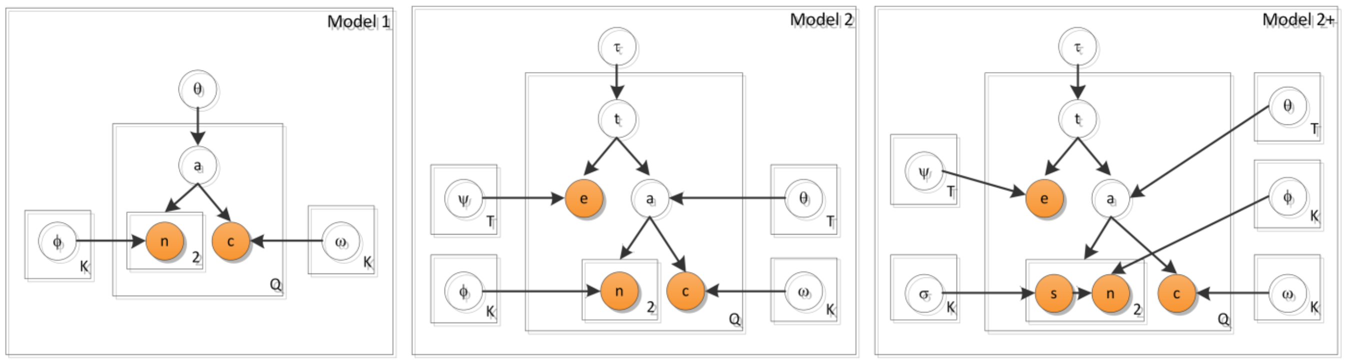

Figure 4. Generative models for actionable queries. Model 1 includes query context words (n) and host clicks (c), Model 2 adds the entity type (t) and the entity (e), and Model 2$^+$ adds an empty context switch (s). Shaded circles are observed variables.

4.2 Model 2 (context + click + type + entity)

The semantic type of the entity mentioned in the query is often strongly correlated with the intended action. For example, if the queried entity is a movie, the user is likely to be looking to buy it, rent it, view local showtimes, or buy theater tickets. It is unlikely however that the user is interested in hacking it, getting its address, or connecting to it. Similarly, a "read biography" action is more likely for a *person* entity and a "view stock price" action is more likely for a *corporation* entity. By accounting for types, the model can avoid recommending incorrect typed actions, such as "view stock price" on a person entity.

In addition, entities themselves are instances of very few types and hence we expect them to be helpful in disambiguating the types. Therefore, in this model, we explicitly model the entities and their types. The middle diagram of Figure 4 illustrates the graphical model. The generative process for Model 2 is as follows:

```
Model 2: Generative model of actionable queries.
For each query $q$
    type $t$ ~ Multinomial($\tau$)
    action $a$ ~ Multinomial($\theta_t$)
    entity $e$ ~ Multinomial($\psi_t$)
    l-context $n_1$ ~ Multinomial($\phi_a$)
    r-context $n_2$ ~ Multinomial($\phi_a$)
    click $c$ ~ Multinomial($\omega_a$)
```

Note that in our generative model, we are assuming that the action is generated independently of the entity itself. However, the choice of the entity also influences the subset of actions that are possible for a particular choice of the type. The independence assumption between actions and entities is a matter of mathematical convenience. Otherwise, we require learning a parameter for each action-type-entity configuration, giving rise to a huge number of parameters. Instead, we choose to include these dependencies at the time of inference, as described later.

For our running example, Model 2 first generates the type "*film*", then given the type, it generates the entity "*inception*" and then generates the action "*read reviews*". The action is used to generate the pre- and post- context words ∅ and "*review*", and then the click on a site such as "rottentomatoes.com".

The joint probability over the model variables is:

$$P(t, a, q{=}\{n_1, e, n_2\}, c \mid \Theta, \Phi, \Omega, \tau, \Psi) = \\ P(t \mid \tau)P(a \mid t, \Theta)P(e \mid t, \Psi) \\ P(c \mid a, \Omega)P(n_1 \mid a, \Phi)P(n_2 \mid a, \Phi)$$

Next, we describe each term in the joint probability. Let T be the number of entity types. The probability of generating a type t is governed by a multinomial with probability vector τ. In particular:

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

Let E be the number of known entities. The probability of generating an entity e given type t is a multinomial with a probability vector ψ_t such that $\Psi = [\psi_1, \ldots, \psi_T]$ captures the matrix of parameters across all T types. In particular:

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

Since actions are now conditioned on types, for every value of type, it is a multinomial distribution with probability vector θ_t such that $\Theta = [\theta_1, \ldots, \theta_T]$ represents parameters across types:

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

Prior distributions over the context words and clicked host remain unchanged as in Model 1.

Inference*:* Given a query, and the learned model, we can apply Bayes' rule to find the posterior distribution, $P(a,t|q,c)$, over the actions, as it is proportional to $P(a,t,q,c)$. We compute this quantity exactly by evaluating the joint for each combination of a and t, and the observed values of q and c.

During inference, we also enforce that for an entity, there are only certain admissible types. As an example, if the entity is *Inception*, valid types include *film* and *book*. We set the posterior probability of invalid types (and hence the relevant type-action configurations) to zero. We obtain the set of admissible types for every entity using an external knowledge base. In this paper, we use Freebase (see Section 5.1). A desirable side effect of this strategy is that only valid ambiguities are captured in the posterior distribution. Thus the model can focus on capturing the actions for multiple of its *valid possible* senses (types).

Learning*:* We omit the log probability of the query corpus for brevity. As in the previous model, we perform maximum likelihood estimation of the parameters using the EM algorithm. Below, in the interest of space, we only present M-step update equations for some of the parameters that are unique to this model. Other parameter updates are similar in spirit to Model 1.

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

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

4.3 Empty Contexts

Generally in Web search, most query contexts are left empty. For example, users tend to query for "obama" far more frequently than by adding refiners such as "support obama" or "obama schedule". In fact, upon inspection of the Φ table for Models 1-2, we noticed that over 90% of the probability mass is covered by the empty context. In order to spread that mass to useful context words, we explicitly represent the empty context using a switch variable that determines whether a context will be empty. The rightmost diagram in Figure 4 illustrates how we model the switch in Model 2, called Model 2^+. The generative story for both Models 1 and 2 can be augmented as follows:

```
Model X + Switch:
For each query q
        ...
        ~~l-context $n_1$ ~ Multinomial($\phi_a$)~~
        ~~r-context $n_2$ ~ Multinomial($\phi_a$)~~
        switch $s_1$ ~ Multinomial($\sigma_a$)
        switch $s_2$ ~ Multinomial($\sigma_a$)
        if ($s_1$) l-context $n_1$ ~ Multinomial($\phi_a$)
        if ($s_2$) r-context $n_2$ ~ Multinomial($\phi_a$)
        …
```

Incorporating the switch into the joint probability of each model is straightforward. Below we show it for Model 2:

$$P(t, a, q{=}\{n_1, e, n_2\}, c, s{=}\{s_1, s_2\} \mid \Theta, \Phi, \Omega, \tau, \Psi, \sigma) = P(t \mid \tau) P(a \mid t, \Theta) P(e \mid t, \Psi) P(c \mid a, \Omega) \prod_{i=1}^{2} P(n_i \mid a, \Phi)^{I[s_i=1]} P(s_i \mid a, \sigma)$$

The probability of generating an empty or non-empty context s given action a is given by a Bernoulli with parameter σ_a:

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

The *M*-step update function for the switch parameter σ is:

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

In the above models, we learned point estimates for the parameters (Φ,Θ,Ω,τ,Ψ,σ) that govern the variables of interest, including type, actions, context, entities and clicks. 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 the parameters well. In addition, our approach enabled us to learn (and perform inference in) the model with large amounts of data with reasonable computing time.

4.4 Enforcing Action Diversity in Learning

In training Model 2 using the EM algorithm, we found that the local optimal solutions often amounted to action clusters that were tied very strongly to specific types. For instance, the *athlete* entity type had a *P*(Action | Type) of 95% into an action cluster that focuses on sports statistics. While it is desirable that the model learns a good top-ranked action (e.g., "Retrieve Sports Statistics"), we also want to be able to recommend a full range of actions for queries (e.g., for the *athlete* type we would also want to see the next top actions, such as "Follow on Social Networks", "Read Biography", "View Pictures" and "Buy Tickets to see"). If one top action absorbs too much probability mass, we often observe empirically that the lower-ranked actions do not gain sufficient probability mass. This is clearly an artifact of the EM algorithm-based learning paradigm.

We resolve this through a two-step procedure for learning. In the first step, we run EM iterations to learn only the parameters that do not involve the entity type (i.e., by freezing the Θ parameter). This allows Model 2 to learn action clusters tied more closely to query contexts and clicked hosts. In a second step, we continue learning with additional EM iterations, now also letting the algorithm learn the Θ parameter. We found that this strategy reduces the average amount of mass for the top-ranking action clusters, which in turn leads to probability mass being more evenly distributed across actions and ultimately to better ranking of the action clusters. In one experiment, we found that this two-step learning reduced the average top P(Action | Type) value from 48% to 28%, distributing the mass more evenly across other actions.

4.5 Decoding

Consider a runtime scenario where a new search query q = "*new york city hotels*" is received. Decoding is accomplished as follows. First, we run a named-entity recognizer (e.g., from [11] or [18]) to identify the entity e = *"new york city*". This leaves the query contexts as $n_1 = \varnothing$ and n_2 = "*hotels*" (and switch values s_1 = *true* and s_2 = *false*). We use historical search query data to identify a distribution $P(c \mid q)$ over all hosts $c \in H$ that received a click for this query in the past. The recommendation score (probability) of an action a is then:

$$P(a \mid q{=}\{n_1, e, n_2\}, c, s) = \sum_t \sum_{c \in H} P(a, t \mid q, c, s) P(c \mid q)$$

The parameter Ω can be directly looked up to rank hosts given each action a. Note that if no click history is available, for instance if observing a query with a never before seen entity, the model can still recommend actions using its other parameters. Also, if the candidate types of an ambiguous entity are known, then we can return an action distribution given each type. If the types are unknown, then we can return an action distribution over each latent type. In both cases, we can marginalize the types to get an action distribution for the query.

4.6 Cluster Labeling: Web Action Phrases

The action clusters discovered by our models are clusters of words defined by the Φ parameter. We need to "translate" each action into action recommendation phrases that can be presented to the user (e.g., "read reviews" or "download").

We begin by examining the most probable context words for each action. The leftmost word cloud in Figure 5 illustrates this

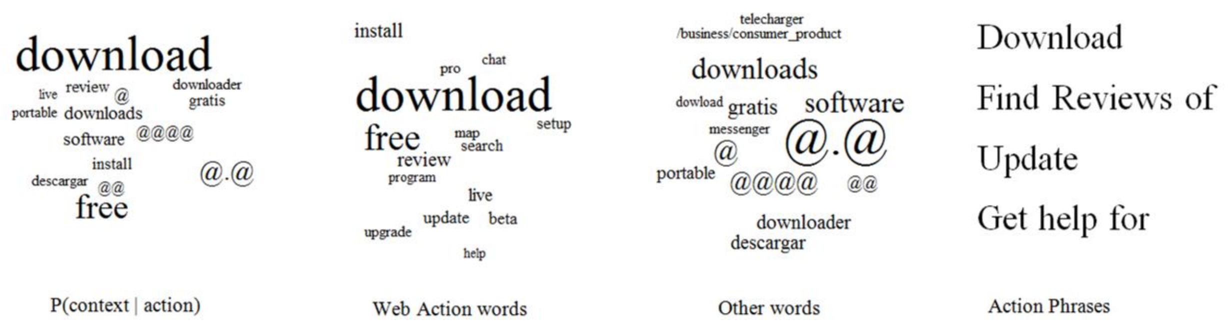

Figure 5. To obtain Action Phrases we first identify top Web Action words from the action's most likely context words.

for one of our discovered actions. Clearly this is a cluster that relates to downloading free software.[1] We then tease out the "actions" by obtaining a list of verbs/action words, and then intersecting this list against the context words in the clusters.

Using a generic verb list is not ideal here because we are restricted to actions that users can perform on the Web, many verbs do not take people in the agent role (e.g., "merge"), and generic verb lists often do not contain words that can be used as Web-based actions such as "blog", "podcast" or "torrent". To obtain a list of appropriate actions, we defined a few key lexical patterns (similar to Hearst [13]) that generally contain action words, such as:

"want to (x)"   "have to (x)"   "you can (x)"   "I can (x)"

We then obtain the most frequent instances of (x) by applying these patterns against a large Web body-text trigram corpus. After filtering out adverbs (using 21 additional patterns, designed to catch adverbs in this corpus) and noise (the 25% of actions with the lowest frequency / unigram count, e.g., "a" and "boy"), this leaves us with a list of 13,417 action words. This list still contains a number of actions (e.g., "shock" or "kill") that users cannot perform over the Web, so we filtered it down to the 1,279 *Web actions* that also occurred with the pattern "(x) at (y)" in our trigrams, where (y) takes the form of a website URL (e.g., "Amazon.com"). Examples of the most popular Web actions include: "buy", "review", "shop" and "unsubscribe".

The second word cloud in Figure 5 shows $P(n \mid a)$ for those contexts n that passed our filter. The third word cloud shows the remaining words when Web action words are removed. The resulting three word cloud types, illustrated in Figure 5, are used as a tool for a human-annotation task to specify the appropriate action phrases for each cluster. From our automatically generated word clouds of action words, non-action words, and the popular hosts for each action cluster, we found it easy for annotators to specify these action phrases. In future work we will explore techniques for fully automating this process of learning action phrases from action words.

5. EXPERIMENTAL RESULTS

5.1 Data

We collected several months of queries issued to Bing and filtered them to retain only those that contain a signal for learning actions, by (i) removing any query that did not lead to a click and (ii) removing any query that did not contain an entity.

We cover a large number of oft-queried entities by focusing on the most important entity types discovered in our query analysis from Section 3 (see Figure 3). Note that Schema.org does not provide actual instances for their entity taxonomy, so we rely instead on Freebase for instances. We chose types from Freebase that correspond to the most often queried types in Schema.org such as *films, business operations, product lines* and *people*. Since Freebase is a fine-grained knowledge base, we also included subtypes such as *athletes*, *actors* and *politicians*, for a total of 21 total types (Table 1). The resulting sets account for approximately 3.4 million entity instances after de-duplication.

Accurate entity recognition is a difficult problem and at model application time one needs high precision and high recall entity recognition and entity to type mappings (e.g., using methods such as described in [7] and [21]). For our model training, given the large amount of available queries, we require only high precision entity recognition, so we turn to the following simple but effective method. We start by matching our query log with all our Freebase entity instances. To avoid problems like a query for "nice pants" getting matched to the city "Nice" in France, we apply an ambiguity filter on the capitalization ratio of our instances and allow matches on only the entities that appear capitalized at least 50% of the time in Wikipedia. To ensure that we do not match on substrings within entities (e.g., if "Harry Potter" is the correct entity but not in our database of entities, we do not want to match on "Harry" or "Potter" separately), we also apply a standalone score filter [14] at 0.9, which calculates how often a string occurs as an exact match in queries relative to how often it occurs as a partial match.

Table 1: 21 Freebase types used in our experiments.

website	product line	digital camera
consumer product	software	film
comp/video game	person	athlete
politician	actor	artist
employer	business operation	restaurant
location	travel destination	tourist attraction
sports facility	university	road

For query contexts n_1 and n_2 defined in Section 4, although one could potentially use arbitrary n-gram context sizes, we keep only queries where the contexts are empty or consist of single words (accounting for a very large fraction of the queries).

We define a navigational query as one where the user only wants to navigate to a specific site and is unlikely to be interested in any other action presented to her. We automatically eliminate such queries from the training set, where a query is considered navigational if in our logs it is associated with >1,000 clicks where >98% of clicks were to the same host (~2% of our data points). Finally, we eliminate entries with clicked hosts that have been clicked fewer than 100 times over our entire query log.

After applying the filters described above, this yielded several million data points for training our models. Our data covers 235K

[1] Note that '@' is a wildcard for any digit. Thus "@.@" is a placeholder for software versions such as "3.1" or "2.0."

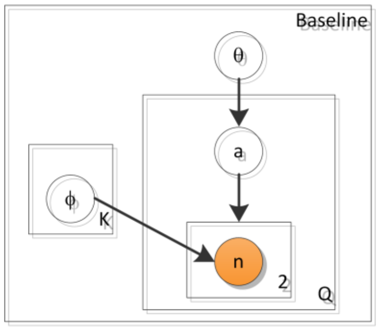

Figure 6. Baseline Model.

distinct Freebase entities, 129K distinct context words, and 58K distinct click hosts. We refer to the resulting queries as *actionable queries* and denote the resultant query set as $\mathcal{Q}$ according to Section 4.

5.2 Model Settings

We trained our models with 50 action clusters, set according to our earlier annotation study in Section 3.2, which found that this would give us good coverage over the main actions in Web search. Alternatively, the constraint could be alleviated by analyzing the semantic similarity between context words in the resulting clusters, or by using techniques similar to those for finding the optimal *k* in *k*-means [12], or by other methods such as those discussed by Blei et al. [2]. We conducted our two-step learning over 100 total EM iterations, running 2 folds per model.

5.3 Experimental Configurations

We used three test sets for our study:

- **HEAD**: 100 queries from a frequency-weighted random query sample of $\mathcal{Q}$.
- **TAIL**: 100 queries from a uniform random sample of $\mathcal{Q}$.
- **Type-Balanced**: 16 queries obtained as follows: Sampling starts from a frequency-weighted sample of $\mathcal{Q}$, but during sampling, we only admit new queries to the test set if they cover a type that has not been covered yet.

The HEAD sample was used to test expected user impact in a Web search scenario whereas the TAIL sample tests how our method applies to rare entities. Whereas manually curated models could potentially address a large portion of head queries, only an automated method can model the tail. In our TAIL sample, we noticed that the entities were skewed towards the *person* type. We introduced the Type-Balanced set to test our model performance over a broad set of entity types, including less common types such as *university* and *tourist attraction*.

Finally, we report our results against the following models:

- **Baseline**: Simpler version of Model 1 that uses only query context words as observed variables, illustrated in Figure 6.
- **Models 1, 2**: As described in Section 4.
- **Model 2$^+$**: Model 2 with the Empty Switch as described in Section 4.3 and illustrated on the right in Figure 4.

There are 12 resulting experimental configurations.

5.4 User Study

We conducted a user study for each experimental configuration to determine relative effectiveness at discovering and suggesting actions. The goals of the study are to assess the following:

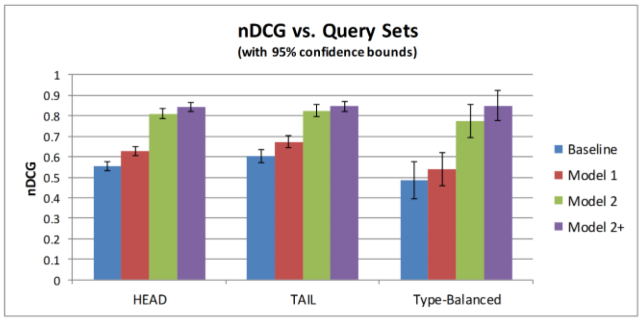

Figure 7. Normalized Discounted Cumulative Gain (nDCG) for each experimental configuration from Section 5.3, with 95% confidence bounds. The addition of types and entities (Model 2) had the largest effect, followed by clicked hosts (Model 1) and then empty switch (Model 2$^+$).

- End-to-end application results: Given a new query, the model should be able to recommend actions that are of interest to users.
- Diversity: The model should learn a comprehensive set of user intended actions, not just a few common actions.

The latter goal is interesting because it deepens our understanding of the actions that Web search users most commonly perform, and a diverse set of actions internally could also be indicative of the ability to perform well on less common queries and on queries whose entities belong to less popular types.

Annotation Guidelines: To measure whether the recommended actions are of interest to users, we adopt a PEGFB graded relevance scale similar to Web search [9]. In our case, we define the grades as:

- ***P****erfect* action: Exactly the explicit intent of the user as stated in the query. (only used for queries with context)
- ***E****xcellent* action: The presumed likely intent of the user as stated in the query.
- ***G****ood* action: Likely to be interesting to the user, although not the stated intent.
- ***F****air* action: Possibly of interest to some users who issue the query.
- ***B****ad* action: Unlikely to be of interest to any user who issues this query.

We employed a total of seven paid independent annotators for grading the actions suggested in each configuration. For each action, two annotations were obtained. Inter-rater agreement using Fleiss' κ was 0.28 (fair agreement) when the P, E, G relevance judgments were collapsed. Note that there is some amount of subjectivity in ratings, especially for queries with no context. For example, on a query for "Obama", some annotators felt that the "Watch videos about" action is *Good*, while others felt it is *Fair*. When exact ratings differed, they still tended to be close in rank. Annotators were also allowed to specify and skip labeling any test query that was judged navigational or that contained entity recognition errors. This occurred in 16.5% of the test cases.

For each query set, each model configuration was set to return up to seven actions to be judged according to our PEGFB scale.

5.5 Experimental Results

The results (using P=5, E=4, G=3, F=2, B=1) from our model configurations are summarized in Figure 7. The evaluation measure is Normalized Discounted Cumulative Gain (nDCG) on the top-7 suggested actions per model.

Table 2. Actions recommended by the various models for the query "Webster University"
Entity: "Webster University" Context: (∅, ∅) Types: employer, university, and location

Baseline (context)	Model 1 (+click)	Model 2 (+type, +entity)	Model 2^+ (+switch)
1.Torrent	1.Torrent	1.Read reviews of	1.Find address
2.Read biography	2.Read biography	2.See map of	2.See pictures of
3.Find adult pictures of	3.Read news about	3.Follow sports teams of	3.Find map of
4.Watch videos	4.See pictures of	4.Get weather in	4.Read news about
5.See picture of	5.Apply for jobs at	5.Apply for jobs at	5.Apply for jobs at
6.Get quotes from	6.Get quotes from	6.Find address of	6.See cost of
7.Apply for jobs at	7.See videos with	7.See tuition of	7.See ranking of

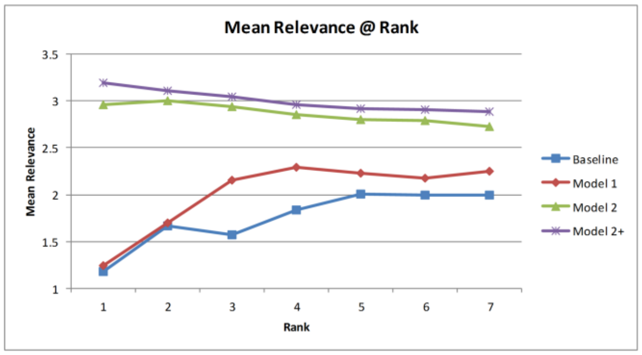

Figure 8. Mean relevance at action rank for our models.

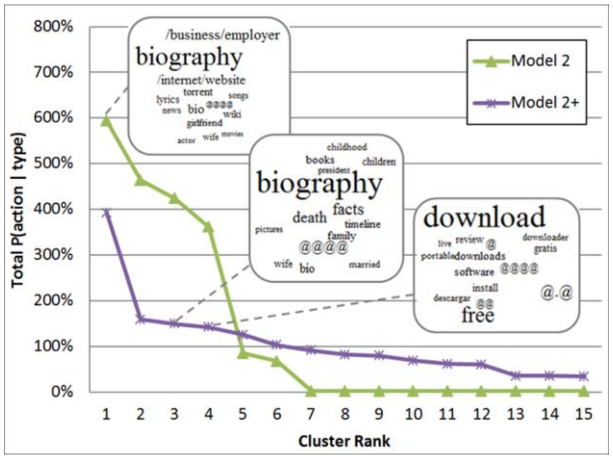

Figure 9. Model 2^+ distributes probability of action given type more evenly across actions than Model 2.

For all query sets, addition of the click host (Model 1) improves over the baseline because it provides an additional useful signal for learning accurate clusters. Adding entity type and the entities (Model 2) proves to be the most important signal in terms of significant relevance improvement across evaluation sets. Adding the empty switch (Model 2^+) does not significantly impact overall relevance, however in the Type-Balanced set we see a tendency for this model to perform better. In Section 5.5.2, we show that Model 2^+ learns a more diverse set of clusters than other models.

Figure 8 shows Mean Relevance as a function of the rank of an action for head queries. Somewhat surprisingly, the Baseline model and Model 1 show an increase in mean relevance as the rank increases (indicating a ranking deficiency), while the other models show a steady but slight decline. Upon inspection, we believe that since many queries do not contain surrounding context words, the Baseline model reverts to the prior that it learned over each action cluster (the θ parameter) in those cases. This manifests itself as a static list of actions that is impervious to the entity being queried. Model 1 does have access to a click information parameter Ω, but the prior on the first two actions in θ are too high to overcome. Support from multiple constraint sources is necessary to overcome the prior.

5.5.1 Error Analysis

Table 2 illustrates action recommendations from our models for the random query "Webster University", which has empty contexts. The Baseline model (which only models action and context) has no information to use for recommending an action other than its action priors, and therefore recommends the most popular general actions it learned from its training set.[2] These tend to fit the more common types, so baseline scores are lower on the type-balanced set, which contains fewer common types. Model 1 does a little better by incorporating prior click information, but still recommends actions that do not apply to the entity's type (e.g., "read biography") because the model does not account for type. Models 2 and 2^+ recommend reasonable sets of actions.

One source of error we noticed arose from how the 21 types that we used did not include the primary types of a number of the entities in the data. For example, for the query "Jefferson High School", interesting actions would mostly be centered on a *high school* type. However, because *high school* is not among our 21 modeled types, our models recognize "Jefferson High School" only as an *employer* and a *location*. As a result, the recommended actions are more general. It should be possible to alleviate this problem by expanding the number of modeled types.

Note also that among the 21 Freebase entity types we use, some of the types have higher query log frequency than others. For example, the *person* type has many more entries in the data than the *tourist destination* type. This leads to our models learning action clusters optimized more toward the popular types than the sparser types. We did explore balancing the training data by only keeping elements of the people subtype (*artist*, *politician*, *actor* and *athlete*) with the types fairly equally represented, and found that this led to each of those types having more action diversity. This suggests that to address sparser types, we may want to discover actions based on type-balanced subsets of the data first, and then either use those actions to initialize clusters in the full training with those actions, or devise a hierarchical setup that incorporates type-subtype information.

5.5.2 Action Cluster Quality

In addition to the end-to-end application goal, it is also desirable for a model to learn a good, diverse set of actions. One

[2] For the query with context, "download Skype," the baseline model is able to recommend actions "download" and "login to."

metric for visualizing this is to graph “Total P(Action | Type)” as a function of “Cluster Rank,” as in Figure 9. This illustrates the distribution of probability mass across the cluster ranks. Here we only compare Models 2 and 2^+, because Model 1 does not model entity type. Given that we used 21 total types, the maximum value would be 2100% (if all 21 types mapped 100% to one cluster). Model 2 appears to have six primary action clusters that receive the majority of the probability from types, while Model 2^+ learns a much more diverse set of actions clusters, which we also observed by inspecting the word clouds in the Φ parameter.

Note that only learning 6 primary action clusters does not mean that Model 2 can only recommend up to 6 distinct action phrases. First, the remaining clusters do have nonzero weight and can contribute action phrases. Second, individual action clusters may contain a mixture of action phrases. For example, one of the Model 2 clusters contains actions for “read biography”, “find lyrics” and “download file” all within the same cluster. This does not cause type mismatches at decoding time because action phrases are typed (e.g., “download file” will only be recommended when the entity is of a type it applies to, such as *software* type), but it does limit the ability of the models to discover and refine good action clusters specifically around the less common actions. The lower ranked clusters within Model 2^+ do look very coherent around specific actions, for example, “read biography” is in a cluster only with related terms such as “facts”, “childhood” and “timeline” while “download” is in a cluster with related terms like “software”, “install” and “free”.

6. CONCLUSIONS

We proposed the notion of *actions* in Entity-Centric Search. We conducted an annotation study on query log data to gauge the prevalence of entities and associated actions in Web search. We developed generative models to learn latent actions from queries, and we implemented them over large real-world query logs. We experimentally showed that modeling click hosts and entity types, along with query context words, yields high relevance on the task of action recommendation, and that explicitly representing empty contexts greatly improves action diversity. Finally, we addressed various issues for developing an end-to-end system for actions, and we are now able to automatically recommend good sets of actions for users issuing new queries.

Future research directions include expanding the number of entity types and modeling actions for “entity category” queries (e.g., “shoes”). Additionally, we believe that our current random initialization of action clusters can be improved upon by seeding the clusters with some prior knowledge. We are also considering adding a user model to our approach in order to better target user-specific actions. For the “Webster University” query in Table 2, for example, actions such as “read reviews of” and “see rankings of” are more suited for prospective students, while “see map of” and “follow sports teams of” are a better fit for current students.

This work takes first steps towards the larger vision of search as an action broker outlined in the introduction. We envision a world where publishers can tag (automatically or manually) their Web pages and native applications with the actions that they can accomplish; a world where users’ intended actions can be inferred and executed seamlessly via connections to these providers. Only then will entities become, truly, *active objects*.

7. ACKNOWLEDGMENTS

The authors thank Omar Alonso for implementing the user study for our evaluation as well as Oren Etzioni, Larry Heck, Rico Malvar, and Jan Pedersen for valuable discussions.

8. REFERENCES

[1] Balasubramanian, N. and Cucerzan, S. Topic Pages: An Alternative to the Ten Blue Links. In *IEEE-ICSC* (2010).

[2] Blei, D.M., Ng, A. and Jordan, M. Latent Dirichlet Allocation. In *Journal of Machine Learning Research*, 3:993-1022, (2003).

[3] Broder, A. A Taxonomy of Web Search. *SIGIR Forum*, volume 36 number 2 pages 3-10 (2002).

[4] Broder, A., Fontoura, M., Gabrilovich, E., Joshi, A., Josifovski, V., Zhang, T. Robust Classification of Rare Queries Using Web Knowledge. In *SIGIR* (2007).

[5] Carman, M.J., Crestani, F., Harvey, M., and Baillie, M. Towards Query Log Based Personalization using Topic Models. In *Proceedings of CIKM* (2010).

[6] Cucerzan, S. Large-Scale Named Entity Disambiguation Based on Wikipedia Data. In *Proceedings of EMNLP* (2007).

[7] Curran, J. R. and Clark, S. Language independent NER using a maximum entropy tagger. In *CoNLL*, pp. 164-167 (2003).

[8] Dalvi, N., Kumar, R., Pang, B., Ramakrishnan, R., Tomkins, A., Bohannon, P., Keerthi, S., Merugu, S., A Web of Concepts. In *Proceedings of PODS* (2009).

[9] Dupret, G. and Piwowarski, B. A User Behavior Model for Average Precision and its Generalization to Graded Judgments. In *Proceedings of SIGIR*, pages 531-538 (2010).

[10] Gao, J., Toutanova, K. and Yih, W. Clickthrough-Based Latent Semantic Models for Web Search. In *Proceedings of SIGIR* (2011).

[11] Guo, J., Xu, G., Cheng, X. and Li, H. Named Entity Recognition in Query. In *Proceedings of SIGIR*, pages 267-274 (2009).

[12] Hamerly, G. and Elkan, C. Learning the K in K-Means. In *Proceedings of the 7th Annual Conference on Neural Information Processing Systems (NIPS)* (2003).

[13] Hearst, M. Automatic Acquisition of Hyponyms from Large Text Corpora. In *Proceedings of COLING* (1992).

[14] Jain, A. and Pennacchiotti, M. Domain-Independent Entity Extraction from Web Search Query Logs. In *WWW* (2011).

[15] Jansen, B.J., Booth, D. and Spink, A. Determining the User Intent of Web Search Engine Queries. In *WWW* (2007).

[16] Kemke, C. and Walker, E. Planning with Action Abstraction and Plan Decomposition Hierarchies. In *IAT* (2006).

[17] Metzinger, T. and Gallese, V. The Emergence of a Shared Action Ontology: Building Blocks for a Theory. In *Consciousness and Cognition*, 12, 549-571 (2003).

[18] Pantel, P. and Fuxman, A. Jigs and Lures: Associating Web Queries with Structured Entities. In *ACL* (2011).

[19] Rose, D. E. and Levinson, D. Understanding User Goals in Web Search. In *Proceedings of WWW* (2004).

[20] Sauper, C. and Barzilay, R. Automatically Generating Wikipedia Articles: A Structure-Aware Approach. In *Proceedings of ACL* (2009).

[21] Sekine, S. and Suzuki, H. Acquiring Ontological Knowledge from Query Logs. In *Proceedings of WWW* (2007).

[22] Yin, X. and Shah, S. Building Taxonomy of Web Search Intents for Name Entity Queries. In *WWW* (2010).