Learning Joint Query Interpretation and Response Ranking

Uma Sawant
IIT Bombay, Yahoo! Labs
uma@cse.iitb.ac.in

Soumen Chakrabarti
IIT Bombay
soumen@cse.iitb.ac.in

ABSTRACT

Thanks to information extraction and semantic Web efforts, search on unstructured text is increasingly refined using semantic annotations and structured knowledge bases. However, most users cannot become familiar with the schema of knowledge bases and ask structured queries. Interpreting free-format queries into a more structured representation is of much current interest. The dominant paradigm is to segment or partition query tokens by purpose (references to types, entities, attribute names, attribute values, relations) and then launch the interpreted query on structured knowledge bases. Given that structured knowledge extraction is never complete, here we choose a less trodden path: a data representation that retains the unstructured text corpus, along with structured annotations (mentions of entities and relationships) on it. We propose two new, natural formulations for joint query interpretation and response ranking that exploit bidirectional flow of information between the knowledge base and the corpus. One, inspired by probabilistic language models, computes expected response scores over the uncertainties of query interpretation. The other is based on max-margin discriminative learning, with latent variables representing those uncertainties. In the context of typed entity search, both formulations bridge a considerable part of the accuracy gap between a generic query that does not constrain the type at all, and the upper bound where the "perfect" target entity type of each query is provided by humans. Our formulations are also superior to a two-stage approach of first choosing a target type using recent query type prediction techniques, and then launching a type-restricted entity search query.

Categories and Subject Descriptors

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

Keywords

Query interpretation; Entity search

1. INTRODUCTION

Web information representation is getting more sophisticated, thanks to information extraction and semantic Web efforts. Much structured and semistructured data now supplements unstructured, free-format textual pages. In verticals such as e-commerce, the structured data can be accessed through forms and faceted search. However, a large number of free-format queries remain outside the scope of verticals. As we shall review in Section 2, there is much recent research on analyzing and annotating them.

Here we focus on a specific kind of entity search query: some words (called *selectors*) in the query are meant to occur literally in a response document (as in traditional text search), but other words *hint* at the type of entity sought by the query. Unlike prior work on translating well-formed sentences or questions to structured queries using deep NLP, we are interested in handling "telegraphic" queries that are typically sent to search engines. Each response entity must be a member of the hinted type.

Note that this problem is quite different from finding answers to well-formed natural language questions (e.g., in Wolfram Alpha) from structured knowledge bases (perhaps curated through information extraction). Also observe that we do not restrict ourselves to queries that seek entities by attribute values or attributes of a given entity (both are valuable query templates for e-commerce and have been researched). In our setup, some responses may only be collected from diverse, open-domain, free-format text sources. E.g., typical driving *time* between Paris and Nice (the target type is time duration), or *cricketers* who scored centuries at Lords (the target type is cricketers).

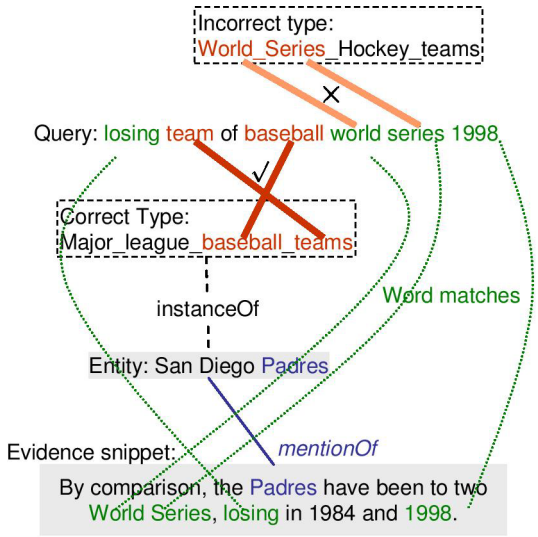

Figure 1: Example of a collective, joint query interpretation and entity ranking problem; includes a query containing different possible hint and selector words, partially matching types with member entities and corpus snippets

The target type (or a more general supertype, such as *sportsperson* in place of *cricketer*) may be instantiated in a *catalog*, but the typical user has no knowledge of the catalog or its schema. Large catalogs like Wikipedia or Freebase evolve "organically". They are not designed by linguists, and they are not minimal or canonical in any sense. Types have overlaps and redundancies. The query interpreter should take advantage of specialized types whenever available, but otherwise gracefully back off to broader types.

Figure 1 shows a query that has at least two plausible hint word sets: {team, baseball} (correct) and {world, series} (in-

WWW 2013, May 13–17, 2013, Rio de Janeiro, Brazil.
ACM 978-1-4503-2035-1/13/05.

correct). Hint words partially match descriptions of types in a catalog, which lead to member entities. Potential response entities are mentioned in document snippets (one shown), which in turn partially match selector words (world, series, losing, 1998). Given a limited number of types to choose from, a human will find it trivial to pick the best. However, a program will find it very challenging to decide *which subset* of query words are type hints, and, even after that, to select the *best type(s)* from a large type catalog. This *query interpretation* task is one part of our goal.

We posit that *corpus statistics provide critical signals* for query interpretation. For example, we might benefit from knowing that `San_Diego_Padres` rarely co-occurs with the word "hockey", which can be known only from the corpus. Query interpretation should ideally be done *jointly* with ranking entities from the corpus, because it involves a *delicate combinatorial balance* between the hint-selector split, and the (rather noisy) signals from the quality of matches between type descriptions and hint words, snippets and other words, and mentions of entities in said snippets.

Although query typing has been investigated before [38, 5], to the best of our knowledge this is the first work on combining type interpretation with learning to rank [21]. In Section 4, we present a natural, generative formulation for the task using probabilistic language models. In Section 5 we present a more flexible and powerful max-margin discriminative approach [19, 7].

In Section 6, we report on experiments involving 709 queries, over 200,000 types, 1.5 million entities, and 380 million evidence snippets collected from over 500 million Web pages. The entity ranking accuracy of a reasonable query interpreter will be between the "lower bound" of a generic system that makes no effort to identify the target type (i.e., all catalog entities are candidates), and the upper bound of an unrealistic "perfect" system that knows the target type by magic. Our salient experimental observations are:

- The generative language model approach improves entity ranking accuracy significantly beyond the lower bound wrt MAP, MRR and NDCG.
- The discriminative approach is superior to generative; e.g., it bridges 43% of the MAP gap between the lower and upper bounds.
- In fact, if we discard the entity ranks output from our system, use it only as a target type predictor, and issue a query with the predicted type, entity ranking accuracy *drops*.
- Our discriminative approach beats a recent target type prediction algorithm by significant margins.
- NLP-heavy techniques are not robust to telegraphic queries.

Our data and code will be made publicly available at `http://www.cse.iitb.ac.in/~soumen/doc/CSAW/`.

2. RELATED WORK

Interpreting a free-format query into a structured form has been explored extensively in the information retrieval (IR) and Web search communities, with several recent dedicated workshops[1]. A preliminary but critical structuring step is to demarcate phrases [6] in free-format queries. There is also a large literature on topic-independent intent discovery [10, 18] as well as topic-dependent facet [30] or template [1] inference.

The problem of *disambiguating named entities* mentioned in queries is superficially similar to ours, but is technically quite different. In Figure 2, query word *ymca* may refer to different entities, but additional query word *lyrics* hints at type *music*, whereas *address* hints at type *organization*. Note that the query text directly embeds a mention of an *entity*, not a type. Disambiguating the entity (usually) amounts to disambiguating the type—contrast Figure 2 with Figure 1. A given mention usually refers to only a few entities. In contrast, misinterpreting the hint often pollutes the entity response list beyond redemption. Delaying a hard choice of the target type, or avoiding it entirely, is likely to help.

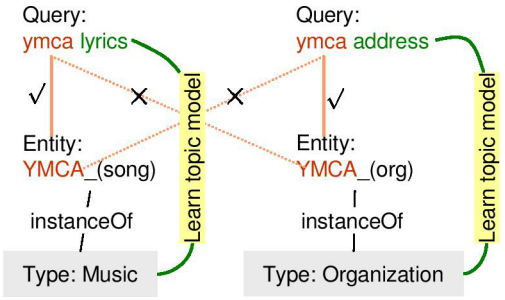

Figure 2: Disambiguating named entities in queries.

For entity disambiguation, Guo *et al.* [15] proposed a probabilistic language model through weak supervision that learns to associate, e.g., lyrics with music and address with organization. Pantel *et al.* [25, 26] pushed this farther by exploiting clicks and modeling intent. Hu *et al.* [17] addressed a similar problem. None gave a discriminative max-margin formulation, or unified the framework with learning to rank.

Given that the database community uses SQL and XQuery as unambiguous, structured representations of information needs, and that the NLP community seeks to parse sentences to a well-defined meaning, there also exists convergent database and NLP literature on interpreting free-format (source) queries into a suitable target "query language". Naturally, much of this work seeks to identify types, entities, attributes, and relations in queries. Although the theoretical problem is challenging [14], a common underlying theme is that each token in the query may be an expression of schema elements, entities, or relationships: this leads to a general assignment problem, which is solved approximately using various techniques, summarized below.

Sarkas *et al.* [33] annotated e-commerce queries using schema and data in a structured product catalog. In the context of Web-extracted knowledge bases such as YAGO [35], Pound *et al.* [29, 28] set up a collective assignment problem with a cost model that reflects syntactic similarity between query fragments and their assigned concepts, as well as semantic coherence between concepts [20]. Sarkas, Pound and others, like us, handle "telegraphic queries" that may not be well-formed sentences. DEANNA [39] solved the collective assignment problem using an integer program. It is capable of parsing queries as complex as "which director has won the Academy Award for best director and is married to an actress that has won the Academy Award for best ac-

[1]ciir.cs.umass.edu/sigir2010/qru
ciir.cs.umass.edu/sigir2011/qru
strataconf.com/stratany2011/public/schedule/detail/21413
sysrun.haifa.il.ibm.com/hrl/smer2011
km.aifb.kit.edu/ws/jiwes2012

tress?" As might be expected, DEANNA is rather sensitive to query syntax and often fails on telegraphic queries. All these systems interpret the query with the help of a fairly clean, structured knowledge base. [33, 29, 28, 39] do not give discriminative learning-to-rank algorithms that jointly disambiguate the query and ranks responses. IBM's Watson [24] identifies candidate entities first, and then scores them for compatibility with likely target types.

In this work, we do not assume that a knowledge base has been curated ahead of time from a text corpus. Instead we assume entities and types have been annotated on spans of unstructured text. Accordingly, we step back from sophisticated target schemata, settling for three basic relations (*instanceOf*, *subTypeOf*, and *mentionOf*, see Figure 1) that link a structured entity catalog with an unstructured text corpus (such as the Web). On the other hand, we take the first step toward integrating learning-to-rank [21] techniques with query interpretation.

Closest to our goal are those of Vallet and Zaragoza [38] and Balog and Neumayer (B&N) [5]. Vallet and Zaragoza first collected a ranked list of entities by launching a query without any type constraints. Each entity belongs to a hierarchy of types. They accrued a score in favor of a type from every entity as a function of its rank, and ranked types by decreasing total score. B&N investigated two techniques. In the first, descriptions of all entities e belonging to each type t were concatenated into a super document for t, and turned into a language model. In the second (similar in spirit to Vallet and Zaragoza), the score of t was calculated as a weighted average of probabilities of entity description language models generating the query, for $e \in t$.

These approaches [5, 31] use long entity descriptions, such as found on the Wikipedia page representing an entity, but not a corpus where entity mentions are annotated. The corpus documents may well not be definitional, and yet remarkably improve entity ranking accuracy, as we shall see. None of [38, 5, 31] attempt a segmentation of query words by purpose (target type vs. literal matches).

3. BACKGROUND AND NOTATION

3.1 "Telegraphic" queries

A "telegraphic" entity search query q expresses an information need that is satisfied by one or more *entities*. Query q is a sequence of $|q|$ *words*. The jth word of query q is denoted $w_{q,j}$, where $j = 1, \ldots, |q|$, and subscript q in $w_{q,j}$ is omitted if clear from context. We will interchangeably use q (as a query identifier) and $\vec{q}$ (to highlight that it is a sequence of words). Unlike full, well-formed, grammatical sentences or questions, telegraphic queries resemble short Web search queries having no clear subject-verb-object or other complex clausal structure. Some examples of natural telegraphic entity search queries and possible natural language "translations" are shown in Figure 3. $\mathcal{Q}$ denotes a set of queries.

3.2 The entity and type catalog

The *catalog* $(\mathcal{T}, \mathcal{E}, \subseteq^+, \in^+)$, is a directed acyclic graph of *type* nodes $t \in \mathcal{T}$, with edges representing the "is-subtype-of" transitive binary relation $\subseteq^+$. Each type t is described by one or more *lemmas* (descriptive phrases) $L(t)$, e.g., Austrian physicists.

Q1	Woodrow Wilson was president of which university?	woodrow wilson president university
Q2	Which Chinese cities have many international companies?	chinese city many international companies
Q3	What cathedral is in Claude Monet's paintings?	cathedral claude monet paintings
Q4	Along the banks of what river is the Hermitage Museum located?	hermitage museum banks of river
Q5	At what institute was Dolly cloned?	dolly clone institute
Q6	Who made the first airplane?	first airplane inventor

Figure 3: Natural language queries and typical telegraphic forms, with potential type description matches underlined.

Each entity e in the catalog is also represented by a node connected by "is-instance-of" edge(s) to one or more *most specific* type nodes, and transitively belongs to all supertypes; this relation is represented as $\in^+$. An entity e may be a *candidate* for a query q. The set of candidate entities for query q is called $\mathcal{E}_q \subseteq \mathcal{E}$. In training data, an entity e may be labeled relevant (denoted e_+) or irrelevant (denoted e_-) for q. $\mathcal{E}_q$ is accordingly partitioned into $\mathcal{E}_q^+, \mathcal{E}_q^-$.

3.3 Annotated corpus and snippets

The corpus is a set of free-format text documents. Each document is modeled as a sequence of words. Entity e is *mentioned* at some places in an unstructured text corpus. A "mention" is a token span (e.g., *Big Apple*) that gives evidence of reference to e (e.g., New_York_City). The mention span, together with a suitable window of context words around it, is called a *snippet*. The set of snippets mentioning e is called $\mathcal{S}_e$. $c \in \mathcal{S}_e$ is one snippet context *supporting* e.

In the Wikipedia corpus, most mentions are annotated manually as wiki hyperlinks. For Web text, statistical learning techniques [20, 16] are used for high-quality annotations. Here we assume mentions to be correct and deterministic. Extending our work to noisy mentions is left for future work.

4. GENERATIVE FORMULATION

Given the success of generative techniques in corpus modeling [8], IR [41] and entity ranking [3, 4], it is natural to propose a generative language model approach to joint query interpretation and response ranking.

As is common in generative language models, we will fix an entity e and generate the query words, by taking the following steps:

1. Choose a type from $\{t : e \in^+ t\}$;
2. Describe that type using one or more query words, which will be called *hint* words;
3. Collect snippets that mention e; and
4. Generate the remainder of the query by sampling words from these snippets.

Our goal is to rank entities by probability given the query, by taking the expectation over possible types and hints.

4.1 Choosing a type given e

Given entity e, we first pick a type t such that $e \in^+ t$, and describe t in the query (with the expectation that the system will infer t, then instantiate it to e as a response). So the basic question looks like: "if the answer is Albert

Einstein, what type (among scientist, person, organism, etc.) is likely to be mentioned in the query, *before* we inspect the query?" (*After* we see the query, our beliefs will change, e.g., depending on whether the query asks "*who* discovered general relativity?" vs. "which *physicist* discovered general relativity?") So we need to design the prior distribution $\Pr(t|e)$.

Recall that there may be hundreds of thousands of ts, and tens of millions of es, so fitting the prior for each e separately is out of the question. On the other hand, the prior is just a mild guidance mechanism to discourage obscure or low-recall types like "Austrian Physicists who died in 1972". Therefore, we propose the following crude but efficient estimate. From a query log with ground truth (i.e., each query accompanied with a t provided by a human), accumulate a hit count N_t for each type t. At query time, given a candidate e, we calculate

$$\Pr(t|e) = \begin{cases} \frac{N_t + \gamma}{\sum_{t':e\in^+t'}(N_{t'}+\gamma)}, & e \in^+ t \\ 0, & \text{otherwise} \end{cases}, \quad (1)$$

where $\gamma \in (0,1)$ is a tuned constant.

4.2 Query word switch variables

Suppose the query is the word sequence $(w_j, j = 1, \ldots, |q|)$. For each position j, we posit a binary switch variable $z_j \in \{h, s\}$. Each z_j will be generated iid from a Bernoulli distribution with tuned parameter $\delta \in (0,1)$. If $z_j = h$, then word w_j is intended as a *hint* to the target type. Otherwise w_j is a *selector* sampled from snippets mentioning entity e. The vector of switch variables is called $\vec{z}$.

The number of possible partitions of query words into hints and selectors is $2^{|q|}$. By definition, telegraphic queries are short, so $2^{|q|}$ is manageable. One can also reduce this search space by asserting additional constraints, without compromising quality in practice. E.g., we can restrict the type hint to a contiguous span with at most three tokens.

Given $\vec{q}$ and a proposed partition $\vec{z}$, we define two helper functions, overloading symbols s and h:

$$\text{Hint words of } q\text{:} \quad h(\vec{q},\vec{z}) = \{w_{q,j} : z_j = h\} \quad (2)$$

$$\text{Selector words of } q\text{:} \quad s(\vec{q},\vec{z}) = \{w_{q,j} : z_j = s\}. \quad (3)$$

With these definitions, in the exhaustive hint-selector partition case, $\vec{z}$ is the result of $|q|$ Bernoulli trials with hint probability $\delta \in (0,1)$ for each word, so we have

$$\Pr(\vec{z}) = \delta^{|h(\vec{q},\vec{z})|}(1-\delta)^{|s(\vec{q},\vec{z})|}. \quad (4)$$

δ is tuned using training data.

In this paper we will consider strict partitions of query words between hints and selectors, but it is not difficult to generalize to words that may be both hints and selectors. Assuming each query word has a purpose, the full space grows to $3^{|q|}$, but assuming contiguity of the hint segment again reduces the space to essentially $O(|q|)$.

4.3 Type description language model

Globally across queries, the textual description of each type t induces a language model. We can define the exact form of the model in any number of ways, but, to keep implementations efficient, we will make the commonly used assumption that hint words are conditionally independent of each other given the type. Each type t is described by one or more *lemmas* (descriptive phrases) $L(t)$, e.g., Austrian physicists. Because lemmas are very short, words are rarely repeated, so we can use the multivariate Bernoulli [23] distribution derived from lemma ℓ:

$$\widehat{\Pr}(w|\ell) = \begin{cases} 1, & \text{if } w \text{ appears in } \ell, \\ 0, & \text{otherwise} \end{cases} \quad (5)$$

Following usual smoothing policies [41], we interpolate the smoothed distribution above with a background language model created from all types:

$$\widehat{\Pr}(w|\mathcal{T}) = \frac{\sum_{t\in\mathcal{T}} [\![w \text{ appears in } \ell \text{ ; } \ell \in L(t)]\!]}{|\mathcal{T}|}; \quad (6)$$

in words, the fraction of all types that contain w. $[\![B]\!]$ is 1 if Boolean condition B is true, and 0 otherwise. We splice together (5) and (6) using parameter $\beta \in (0,1)$:

$$\Pr(w|\ell) = (1-\beta)\widehat{\Pr}(w|\ell) + \beta\widehat{\Pr}(w|\mathcal{T}). \quad (7)$$

The probability of generating exactly the hint words in the query is

$$\Pr(h(\vec{q},\vec{z})|\ell) = \prod_{w\in h(\vec{q},\vec{z})} \Pr(w|\ell) \prod_{w\notin h(\vec{q},\vec{z})} (1-\Pr(w|\ell)), \quad (8)$$

where w ranges over the entire vocabulary of type descriptions. In case of multiple lemmas describing a type,

$$\Pr(\cdot|t) = \max_{\ell\in L(t)} \Pr(\cdot|\ell); \quad (9)$$

i.e., use the most favorable lemma. All fitted parameters in the distribution $\Pr(w|\ell)$ are collectively called φ.

4.4 Entity snippet language model

The selector part of the query, $s(\vec{q},\vec{z})$, is generated from a language model derived from $\mathcal{S}_e$, the set of snippets that mention candidate entity e. For simplicity we use the same kind of smoothed multivariate Bernoulli distribution to build the language model as we did for the type descriptions. Note that words that appear in snippets but not in the query are of no concern in a language model that seeks to generate the query from distributions associated with the snippets. Suppose $corpusCount(e)$ is the number of mentions of e in the corpus $\mathcal{C}$, and $corpusCount(e,w)$ be the number of mentions of e where w also occurs within a specified snippet window width. The unsmoothed probability of generating a query word w from the snippets of e is

$$\widehat{\Pr}(w|e) = \frac{corpusCount(e,w)}{corpusCount(e)} = \frac{|\{s \in \mathcal{S}_e : w \in s\}|}{corpusCount(e)}. \quad (10)$$

As before, we will smooth the above estimate using an corpus-level, entity-independent background word distribution estimate:

$$\widehat{\Pr}(w|\mathcal{C}) = \frac{1}{|\mathcal{C}|}(\text{number of documents containing } w). \quad (11)$$

And now we use the interpolation

$$\Pr(w|e) = (1-\alpha)\widehat{\Pr}(w|e) + \alpha\widehat{\Pr}(w|\mathcal{C}), \quad (12)$$

where $\alpha \in (0,1)$ is a suitable smoothing parameter. The fitted parameters of the $\Pr(w|e)$ distribution are collectively called θ. Similar to (8), the selector part of the query is

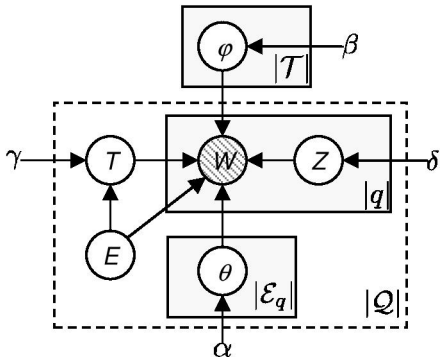

Figure 4: Plate diagram for generating a query q from a candidate entity e. Only $(w_{q,j} : j = 1, \ldots, |q|)$ are observed variables. φ represents the type description language model and θ represents the entity mention snippets language model. $(z_{q,j} : j = 1, \ldots, |q|)$ are the hidden switch variables. T is the hidden type variable.

generated with probability

$$\Pr(s(\vec{q}, \vec{z})|e) = \prod_{w \in s(\vec{q},\vec{z})} \Pr(w|e) \prod_{w \notin s(\vec{q},\vec{z})} (1 - \Pr(w|e)), \quad (13)$$

except here w ranges over all query words.

4.5 Putting the pieces together

A plate diagram for the process generating a query $\vec{q}$ is shown in Figure 4. Vertices are marked with random variables E, T, Z, W whose instantiations are specific values $e, t, \vec{z}, w \in q$.

The hidden variables of interest are the binary $Z \in \{h, s\}$, for selecting between type hint (h) and selector (s) words; and T, the type of one query. Each query picks one hidden value t, and a vector of $|q|$ size for Z, denoted $\vec{z}$. The only observed variables are the $|q|$ query words $(w_j : j = 1, \ldots, |q|)$. Also, $\alpha, \beta, \gamma, \delta$ are hyper-parameters tuned globally across queries.

In the end we are interested in $\arg\max_e \Pr(e|\vec{q})$, where

$$\begin{aligned}
\Pr(e|\vec{q}) \propto \Pr(e, \vec{q}) &= \Pr(e) \Pr(\vec{q}|e) = \Pr(e) \sum_{t,\vec{z}} \Pr(\vec{q}, t, \vec{z}|e) \\
&= \Pr(e) \sum_{t,\vec{z}} \Pr(t|e) \underline{\Pr(\vec{z}|e,t)} \Pr(\vec{q}|e,t,\vec{z}) \quad (14) \\
&\approx \Pr(e) \sum_{t,\vec{z}} \Pr(t|e) \underline{\Pr(\vec{z})} \Pr(\vec{q}|e,t,\vec{z}) \quad (15) \\
&= \Pr(e) \sum_{t,\vec{z}} \Pr(t|e) \underbrace{\Pr(\vec{z})}_{(4)} \underbrace{\Pr(h(\vec{q},\vec{z})|t)}_{(9)} \underbrace{\Pr(s(\vec{q},\vec{z})|e)}_{(13)}.
\end{aligned}$$

To get from (14) to (15) we make the simplifying assumption that the density of hint words in queries is independent of the candidate entity and type. As mentioned before, adding over $t, \vec{z}$ is feasible for telegraphic queries because they are short. The prior $\Pr(e)$ may be uninformative (i.e., uniform), or set proportional to $|\mathcal{S}_e|$ [22], or use shrunk estimates from answer types in the past. We use $\Pr(e) = |\mathcal{S}_e| / \sum_{e'} |\mathcal{S}_{e'}|$.

If we allow a query word to represent both a type hint and a selector, the clean separation after (15) no longer works, but it is possible to extend the framework using a soft-OR expression. We omit details owing to space constraints.

4.6 Explaining a top-ranking entity

In standard text search, top-ranking URLs are accompanied by a summary with matching query words highlighted. In our system, top-ranking entities need to be justified by explaining to the user how the query was interpreted. Specifically, we need to show the user the inferred type, and the inferred purpose (hint or selector) of each query word.

$$\begin{aligned}
\Pr(t, \vec{z}|e, \vec{q}) &\propto \Pr(e, t, \vec{q}, \vec{z}) \\
&= \Pr(e) \Pr(t|e) \underline{\Pr(\vec{z}|e,t)} \Pr(\vec{q}|e,t,\vec{z}) \\
&\approx \Pr(e) \Pr(t|e) \underline{\Pr(\vec{z})} \Pr(\vec{q}|e,t,\vec{z}) \quad (16)
\end{aligned}$$

approximating $\Pr(\vec{z}|e,t) \approx \Pr(\vec{z})$ as before. Now we can report $\arg\max_{t,\vec{z}} \Pr(t, \vec{z}|e, \vec{q})$ as the explanation for e. It is also possible to report marginals such as $\Pr(t|e, \vec{q})$ or $\Pr(z_j|e, \vec{q})$ this way.

4.7 Potential pitfalls

As often happens, a generative formulation starts out feeling natural, but is soon mired in a number of questionable assumptions and tuned hyper parameters. In recent times, this story has played out in many problems, such as information extraction [32] and learning to rank [21], where generative language models were proposed earlier, but the latest algorithms are all discriminatively trained. The above formulation has several potential shortcomings:

- The modeling of $\Pr(t|e)$ is necessarily a compromise.
- $\Pr(z_j)$ is assumed to be independent of q and e, and iid. These assumptions may not be the best.
- In the interest of computational feasibility, the language models for both types and snippets are simplistic. Phrase and exact matches are difficult to capture.
- Hyper parameters $\alpha, \beta, \gamma, \delta$ can only be tuned by sweeping ranges; no effective learning technique is obvious.
- As often happens with complex generative models, the scales of probabilities being multiplied (15) are diverse and hard to balance.

5. DISCRIMINATIVE FORMULATION

Instead of designing conditional distributions as in Section 4, here we will design feature functions, and learn weights corresponding to them by using relevant and (samples of) irrelevant entity sets $\mathcal{E}_q^+, \mathcal{E}_q^-$ associated with each query q, as is standard in learning to rank [21]. The benefit is that it is much safer to incrementally add highly informative but strongly correlated features (such as exact phrase match, match with and without stemming, etc.) to discriminative formulations.

Standard notation used in structured max-margin learning uses $\phi(x, y) \in \mathbb{R}^d$ as the feature map, where x is an observation and y is the label to be predicted. A *model* $\lambda \in \mathbb{R}^d$ is fitted so that $\lambda \cdot \phi(x, y_{\text{correct}}) > \lambda \cdot \phi(x, y_{\text{incorrect}})$. Once λ gets fixed via training, given a new text instance x_{test}, *inference* is the process of finding $\arg\max_y \lambda \cdot \phi(x_{\text{test}}, y)$.

In our case, we use the notation $\phi(q, e, t, \vec{z})$ for the feature map. q gives us access to the sequence of words in the query, and is the analog of x above. e gives us access to the snippets $\mathcal{S}_e$ that support e, and is the analog of y above. t and $\vec{z}$ are *latent variable* [40] inputs to the feature map whose role will be explained shortly.

Guided by the generative formulation in Section 4, we partition the feature vector as follows:

$$\phi(q, e, t, \vec{z}) = \big(\phi_1(q, e), \phi_2(t, e), \phi_3(q, \vec{z}, t), \phi_4(q, \vec{z}, e)\big), \quad (17)$$

where

- $\phi_1(q, e)$ models the prior for e.
- $\phi_2(t, e)$ models the prior $\Pr(t|e)$.
- $\phi_3(q, \vec{z}, t)$ models the compatibility between the type hint part of query words and the proposed type t.
- $\phi_4(q, \vec{z}, e)$ models the compatibility between the selector part of query words and $\mathcal{S}_e$.

5.1 Features ϕ_1 modeling entity prior

In Section 4.5 we used $\Pr(e) = |\mathcal{S}_e| / \sum_{e'} |\mathcal{S}_{e'}|$ as a prior probability for e. It is natural to make this one element in ϕ_1. But the discriminative setup allows us to introduce other powerful features.

$|\mathcal{S}_e|$ does not distinguish between snippets that match the query well vs. poorly. Let $\text{IDF}(w)$ be the inverse document frequency [2] of query word w, and $\text{IDF}(q) = \sum_{w \in q} \text{IDF}(w)$. $c \cap q$ is the set of query words found in snippet c, with total $\text{IDF}(c \cap q) = \sum_{w \in c \cap q} \text{IDF}(w)$. Then the match-quality-weighted snippet support for e is characterized as

$$\phi_1(q, e)[\cdot] = \frac{1}{2^{|q|}\ \text{IDF}(q)} \sum_{c \in \mathcal{S}_e} \text{IDF}(c \cap q), \quad (18)$$

where $2^{|q|}\ \text{IDF}(q)$ normalizes the feature across diverse queries.

Another feature in ϕ_1 relates to negative evidence. If there are other words present, a query that directly mentions an entity is hardly ever answered correctly by that entity; Tom_Cruise could not be the answer for the query `tom cruise wife`. Another (0/1) element in ϕ_1 is whether a description ("lemma") of e is contained in the query. In our experiments, the model element in λ corresponding to this feature turns out a negative number, as expected.

5.2 Features ϕ_2 modeling type prior

We have already proposed one way to estimate $\Pr(t|e)$ in Section 4.1. This estimate a natural element in ϕ_2. We can also help the learner use the generality or specificity of types, measured as this feature: $\big|\{e : e \in^+ t\}\big| / |\mathcal{E}|$. In our experiments, the element of λ corresponding to this feature also got negative values, indicating preference of specific types over generic ones. This corroborates earlier observation regarding the depth of desired types in a hierarchy [5].

5.3 Hint-type compatibility features ϕ_3

Given the input parameters of $\phi_3(\vec{q}, \vec{z}, t)$, we compute the hint word subsequence $h(\vec{q}, \vec{z})$ as in (2). Now we can define any number of features between these hint words and the given type t, which has lemma set $L(t)$.

- A standard feature borrowed from (9) is $\Pr(h(\vec{q}, \vec{z})|t)$.
- Unlike in the generative formulation, we can add synthetic features. E.g., a feature that has value 1 if ℓ matches the subsequence $h(\vec{q}, \vec{z})$ *exactly*.
- In Section 4, the size of $h(\vec{q}, \vec{z})$ was drawn from a binomial distribution controlled by hyper parameter δ. To model more general distributions, we use binary features of the form

$$\begin{cases} 1, & |h(\vec{q}, \vec{z})| < k \\ 0, & \text{otherwise} \end{cases}$$

for $k = 1, \ldots$, to capture the belief that smaller number of hint words is preferable.

5.4 Selector-snippets compatibility features ϕ_4

Now consider q and its selectors $s(\vec{q}, \vec{z}) \subseteq q$ as word sets (no duplicates), and the snippets $\mathcal{S}_e$ supporting candidate entity e. $\phi_4(q, \vec{z}, e)$ will include feature/s that express the extent of match or compatibility between the selector words and the snippets. We need to characterize and then combine two kinds of signals here:

- The rarity (hence, informativeness) of a subset of $s(\vec{q}, \vec{z})$ that match in snippets, and
- The number of supporting snippets [22] that match a given word set.

(A third kind of signal, proximity [27, 37, 36], is favored indirectly, because snippets have limited width. A more refined treatment of proximity is left for future work.)

A snippet $c \in \mathcal{S}_e$, interpreted as a subset of query words q, *covers* $s(\vec{q}, \vec{z})$ if $c \supseteq s(\vec{q}, \vec{z})$. Otherwise $c \subset s(\vec{q}, \vec{z})$. Recall every snippet c has an $\text{IDF}(c) = \sum_{w \in c \cap q} \text{IDF}(w)$. We propose two features:

$$\frac{1}{2^{|q|}\ \text{IDF}(q)} \sum_{c \supseteq s(\vec{q}, \vec{z})} \text{IDF}(s(\vec{q}, \vec{z})) = \frac{\text{IDF}(s(\vec{q}, \vec{z}))\ |\{c : c \supseteq s(\vec{q}, \vec{z})\}|}{2^{|q|}\ \text{IDF}(q)} \quad (19)$$

and

$$\frac{1}{2^{|q|}\ \text{IDF}(q)} \sum_{c \subset s(\vec{q}, \vec{z})} \text{IDF}(c). \quad (20)$$

We found the separation above to be superior to collapsing covering and non-covering snippets into one sum. Another useful feature was the fraction of snippets c such that $c = q$ (exactly matching all query words).

5.5 Inference and training

With a wrong choice of hint-selector partition $\vec{z}$, or a wrong choice of type t, even a highly relevant response e could score very poorly. Therefore, any reasonable scoring scheme should evaluate e under the *best* choice of $t, \vec{z}$. I.e., the score of e should be

$$\max_{t: e \in^+ t, \vec{z}} \lambda \cdot \phi(q, e, t, \vec{z}). \quad (21)$$

(Note that t ranges over only those types to which e belongs.) In learning to rank [21], three training paradigms are commonly used: itemwise, pairwise and listwise. Because of the added complexity from the latent variables $t, \vec{z}$, here we discuss itemwise and pairwise training. Pairwise linear discrimination [19] remains an effective approach for learning to rank. Listwise training is left for future work, as is the use of nonlinear models like boosted regression trees.

In itemwise training, each response entity e is one item, which can be good (relevant, denoted e_+) or bad (irrelevant, denoted e_-). Following standard max-margin methodology, we want

$$\forall q, e_+ : \quad \max_{t, \vec{z}} \lambda \cdot \phi(q, e_+, t, \vec{z}) \geq 1 - \xi_{q, e_+}, \text{ and} \quad (22)$$

$$\forall q, e_- : \quad \max_{t, \vec{z}} \lambda \cdot \phi(q, e_-, t, \vec{z}) \leq 1 + \xi_{q, e_-}, \quad (23)$$

where $\xi_{q, e_+}, \xi_{q, e_-} \geq 0$ are the usual SVM-style slack variables. Constraint (23) is easy to handle by breaking it up

into the conjunct:

$$\forall q, e_-, \forall t, \forall \vec{z}: \qquad \lambda \cdot \phi(q, e, t, \vec{z}) \leq 1 + \xi_{q,e_-}. \tag{24}$$

However, (22) is a *disjunctive* constraint, as also arises in multiple instance classification or ranking [7]. A common way of dealing with this is to modify constraint (22) into

$$\forall q, e_+: \quad \sum_{t,\vec{z}} u(q, e_+, t, \vec{z}) \lambda \cdot \phi(q, e_+, t, \vec{z}) \geq 1 - \xi_{q,e_+} \tag{25}$$

where $u(q, e, t, \vec{z}) \in \{0, 1\}$ and

$$\forall q, e_+: \qquad \sum_{t,\vec{z}} u(q, e_+, t, \vec{z}) = 1.$$

This is an integer program, so the next step is to relax the new variables to $0 \leq u(q, e, t, \vec{z}) \leq 1$ (i.e., the $(t, \vec{z})$-simplex). Unfortunately, owing to the introduction of new variables $u(\cdots)$ and multiplication with old variables λ, the optimization is no longer convex.

Bergeron *et al.* [7] propose an alternating optimization: holding one of u and λ fixed, optimize the other, and repeat (there are no theoretical guarantees). Note that if λ is fixed, the optimization of u is a simple linear program. If u is fixed, the optimization of λ is comparable to training a standard SVM. The objective would then take the form

$$\tfrac{1}{2}\|\lambda\|^2 + \frac{C}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} \frac{\sum_{e_+ \in \mathcal{E}_q^+} \xi_{q,e_+} + \sum_{e_- \in \mathcal{E}_q^-} \xi_{q,e_-}}{|\mathcal{E}_q^+| + |\mathcal{E}_q^-|} \tag{26}$$

Here $C > 0$ is the usual SVM parameter trading off training loss against model complexity. Note that u does not appear in the objective.

In our application, $\phi(q, e, t, \vec{z}) \geq \vec{0}$. Suppose $\lambda \geq \vec{0}$ in some iteration (which easily happens in our application). In that case, to satisfy constraint (25), it suffices to set only one element in u to 1, corresponding to $\arg\max_{t,\vec{z}} \lambda \cdot \phi(q, e, t, \vec{z})$, and the rest to 0s. In other words, a particular $(t, \vec{z})$ is chosen ignoring all others. This severely restricts the search space over u, λ in subsequent iterations and has greater chance of getting stuck in a local minima.

To mitigate this problem, we propose the following annealing protocol. The u distribution collapse reduces entropy suddenly. The remedy is to subtract from the objective (to be minimized) a term related to the entropy of the u distribution:

$$(26) + D \sum_{q, e_+} \sum_{t, \vec{z}} u(q, e_+, t, \vec{z}) \log u(q, e_+, t, \vec{z}). \tag{27}$$

Here $D \geq 0$ is a temperature parameter that is gradually reduced in powers of 10 toward zero with the alternative iterations optimizing u and λ. Note that the objective (27) is convex in u, λ and ξ_*. Moreover, with either u or λ fixed, all constraints are linear inequalities.

```
1: initialize u to random values on the simplex
2: initialize D to some positive value
3: while not reached local optimum do
4:     fix u and solve quadratic program to get next λ
5:     reduce D geometrically
6:     fix λ and solve convex program for next u
```

Figure 5: Pseudocode for discriminative training.

Very little changes if we extend from itemwise to pairwise training, except the optimization gets slower, because of the sheer number of pair constraints of the form:

$$\forall q, e_+, e_-: \quad \max_{t,\vec{z}} \lambda \cdot \phi(q, e_+, t, \vec{z}) - \max_{t,\vec{z}} \lambda \cdot \phi(q, e_-, t, \vec{z}) \geq 1 - \xi_{q,e_+,e_-}. \tag{28}$$

The itemwise objective in (26) changes to the pairwise objectice

$$\tfrac{1}{2}\|\lambda\|^2 + \frac{C}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} \frac{1}{|\mathcal{E}_q^+|\,|\mathcal{E}_q^-|} \sum_{e_+ \in \mathcal{E}_q^+, e^- \in \mathcal{E}_q^-} \xi_{q,e_+,e_-}. \tag{29}$$

For clarity, first we rewrite (28) as

$$\forall q, e_+, e_-: \quad \max_{t,\vec{z}} \lambda \cdot \phi(q, e_+, t, \vec{z}) \geq 1 - \xi_{q,e_+,e_-} + \max_{t',\vec{z}'} \lambda \cdot \phi(q, e_-, t', \vec{z}').$$

Then we pull out $t', \vec{z}'$:

$$\forall q, e_+, e_-, t', \vec{z}': \quad \max_{t,\vec{z}} \lambda \cdot \phi(q, e_+, t, \vec{z}) \geq 1 - \xi_{q,e_+,e_-} + \lambda \cdot \phi(q, e_-, t', \vec{z}').$$

Finally, we use a new set of u variables to convert this to an alternating optimization as before:

$$\forall q, e_+, e_-, t', \vec{z}': \quad \sum_{t,\vec{z}} u(q, e_+, t, \vec{z}) \lambda \cdot \phi(q, e_+, t, \vec{z}) \geq 1 - \xi_{q,e_+,e_-} + \lambda \cdot \phi(q, e_-, t', \vec{z}'). \tag{30}$$

These enhancements do not change the basic nature of the optimization.

5.6 Implementation details

5.6.1 Reducing computational requirements

The space of $(q, e, t, \vec{z})$ and especially their discriminative constraints can become prohibitively large. To keep RAM and CPU needs practical, we used the following policies; our experimental results are insensitive to them.

- We sampled down bad (irrelevant) entities e_- that were allowed to generate constraint (28).
- For empty $h(\vec{q}, \vec{z}) = \varnothing$, $\phi_3(q, \vec{z}, t)$ provides no signal. In such cases, we allow t to take only one value: the most generic type `Entity`.

5.6.2 Explaining a top-ranking entity

This is even simpler in the discriminative setting than in the generative setting; we can simply use (21) to report $\arg\max_{t,\vec{z}} \lambda \cdot \phi(q, e, t, \vec{z})$.

5.6.3 Implementing a target type predictor

Extending the above scheme, each entity e scores each candidate types t as $score(t|e) = \max_{\vec{z}} \lambda \cdot \phi(\cdot, e, t, \vec{z})$. This induces a ranking over types for each entity. We can choose the overall type predicted by the query as the one whose sum of ranks among the top-k entities is smallest. An apparently crude approximation would be to predict the best type for the single top-ranked entity. But $k > 1$ can stabilize the predicted type, in case the top entity is incorrect. (We may want to predict a single type as a feedback to the user, or to compare with other type prediction systems, but, as we shall see, not for the best quality of entity ranking, which is best done collectively.)

6. EXPERIMENTS

6.1 Testbed

6.1.1 Catalog and annotated corpus

Our type and entity catalog was YAGO [35], with about 200,000 types and 1.9 million entities. An annotator trained on mentions of these entities in Wikipedia[2] was applied [12] over a Web corpus from a commercial search engine, having 500 million spam-free Web pages. This resulted in about 8 billion entity annotations, average 16 annotations per page. These were then indexed [13].

6.1.2 Type constrained entity search

The index supports semistructured queries specified by:

- an answer type t from among the 200,000 YAGO types,
- a bag of words and phrases in a IDF-WAND (weak-and) operator [11], and
- a snippet window width.

A DAAT [11] query processor returns a stream of snippets at most as wide as the given window width limit, that contain a mention of some entity $e \in^+ t$ and satisfies the WAND predicate. In case of phrases in the query, the WAND threshold is computed by adding the IDF of constituent words.

Our query processor is implemented using MG4J [9] in Java, with no index caching. Basic keyword WAND queries take a few seconds over 500 million documents. Setting $t =$ `Entity`, the root type, and asking for a stream of all entities in qualifying snippets, slows down the query by a small factor. A discriminative snippet scoring and aggregation technique [34] achieves entity ranking accuracy superior to recent approaches.

6.2 Queries with ground truth

We use 709 entity search queries collected from many years of TREC and INEX competitions, along with relevant and irrelevant entities. Two paid masters students, familiar with Web search engines, read the full TREC/INEX description of entity search queries and wrote out queries they would naturally issue to a commercial search engine. They also selected the best (as per their judgment) type from YAGO for each query, as ground truth. The distribution of types is heavy-tailed, with 69% of the atypes in this list occurring only once and top four atypes accounting for one third of queries. The atypes towards top are mostly generic (*location*, *person*, etc.), while those toward the bottom are more specific (*Brooklyn_Dodgers_players*, *Dilbert_characters* etc.). This data is publicly available at bit.ly/WSpxvr. Launching the queries with the known types resulted in 380 million snippets supporting candidate entities; these are also available on request. We also performed type prediction (Section 5.6.3) on dataset provided in [5]. Since this dataset does not contain ground truth of relevant entities for each query, we did not test entity ranking.

6.3 Generic and "perfect" baselines

The ranking accuracy of a reasonable query interpreter algorithm in our framework will lie between two baselines:

Generic: The generic baseline assumes zero knowledge of query types, instead using $t =$ `Entity`, the root/s of the type hierarchy in the catalog.

"Perfect": The "perfect" baseline assumes complete (human-provided) knowledge of the type and uses it in the semistructured query launched over the catalog and annotated corpus.

Of course, even "perfect" may perform poorly in some queries, because of lack of support for relevant entities in the corpus, snippets incorrectly or not annotated (both false positive and negative), incorrect absence of paths between types and entities in the catalog, or some inadequacy of the type-constrained entity ranker. It is also possible for an algorithm (including ours) to perform worse than generic on some queries, by choosing a particularly unfortunate type, but obviously it should do better than generic on average, to be useful.

6.4 Measurements and results

As is standard in entity ranking research, we report NDCG at various ranks, mean reciprocal rank (MRR, not truncated) and mean average precision (MAP) at the entity (not document) level. Space constraints prevent us from defining these; see Liu [21] for details. For Discriminative, C is tuned by 5-fold cross validation at the query level. For Generative, we swept over $\alpha, \beta, \gamma, \delta$ in powers of 10 (e.g. $10^{-5}, 10^{-4}, \ldots, 1$).

	Generic	Generative	Discriminative	Perfect
MAP	$0.323^{\downarrow\downarrow}$	$0.414^{\downarrow}$	**0.462**	0.644
MRR	$0.332^{\downarrow\downarrow}$	$0.432^{\downarrow}$	**0.481**	0.664

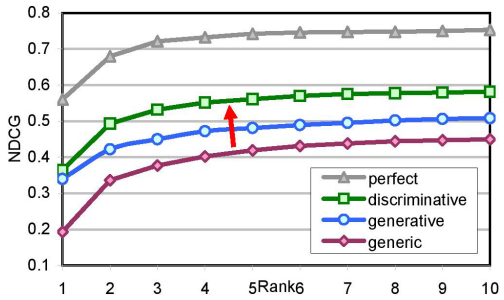

Figure 6: Generic, generative, discriminative and "perfect" accuracies.

6.4.1 Our algorithms vs. generic and perfect

For our techniques to be useful, they must bridge a substantial part of the gap between the generic lower bound and the perfect upper bound. Figure 6 confirms that Generative bridges 28% of the MAP gap between generic and perfect, whereas discriminative is significantly better at 43%. MRR and NDCG follow similar trends. All gaps are statistically significant at 95% confidence level (indicated by $\downarrow$).

Figure 6 is aggregated over all queries. Figure 7 focuses on average precision disaggregated into queries, comparing discriminative against generic. While some queries are damaged by discriminative, many more are improved.

Failure analysis revealed residual $(t, \vec{z})$ ambiguity, coupled with lack of $\in^+$ or $\subseteq^+$ paths in an incomplete catalog to be the major reasons for losses on some queries. Even though there is some ground yet to cover to reach "perfect" levels, these results show there is much hope for automatically interpreting even telegraphic queries.

6.4.2 Benefits of annealing optimization

Figure 8 shows that discriminative with our entropy-based annealing protocol performs significantly (marked with "$\downarrow$")

[2]Cross validated accuracy on Wikipedia was about 90%.

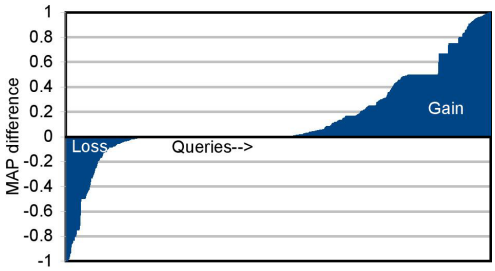

Figure 7: MAP of discriminative minus map of generic, compared query-wise between generic and discriminative. Below zero means discriminative did worse than generic on that query. Queries in (arbitrary) order of discriminative AP gain.

better than the scheme proposed by Bergeron *et al.*[7]. This may be of independent interest in multiple instance ranking and max-margin learning with latent variables.

	Bergeron (26)	Entropy (27)
MAP	0.416$^{\downarrow}$	**0.462**
MRR	0.432$^{\downarrow}$	**0.481**

Figure 8: Benefits of annealing protocol.

6.4.3 Comparison with B&N's type prediction

B&N [5] proposed two models, of which the "entity-centric" model was generally superior. Each entity e was associated with a textual description (e.g., Wikipedia page) which induced a smoothed language model θ_e. B&N estimate the score of type t as

$$\Pr(q|t) = \sum_{e \in +t} \Pr(q|\theta_e) \Pr(e|t), \qquad (31)$$

where $\Pr(e|t)$ was set to uniform. Note that no corpus (apart from the one of entity descriptions) was used. The output of B&N's algorithm (hereafter, "B&N") is a ranked list of types, not entities. We implemented B&N, and obtained accuracy closely matching their published numbers, using the DBpedia catalog with 358 types, and 258 queries (different from our main query set and testbed).

	B&N	Discr($k=1$)	Discr($k=5$)	Discr($k=10$)
MAP	0.33	0.33	0.384	0.390

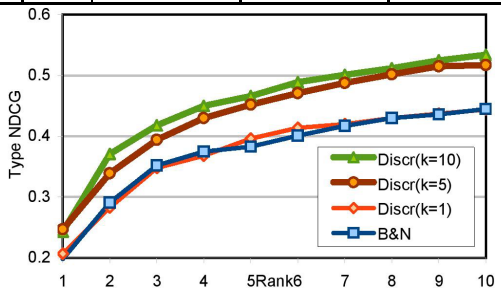

Figure 9: Type prediction by B&N vs. discriminative.

We turned our system into a type predictor (Section 5.6.3), and also used DBpedia like B&N and compared type prediction accuracy on dataset provided in [5]. Results are shown in Figure 9 after including the top k returned types. At $k = 1$, our discriminative type prediction matches B&N, and larger k performs better, owing to stabilizing consensus from lower-ranked entities. Coupled with the results in Section 6.4.6, this is strong evidence that our unified formulation is superior, even if the goal is type prediction.

6.4.4 Comparison with B&N-based entity ranking

A type prediction may be less than ideal, and yet entity prediction may be fine. One can take the top type predicted by B&N, and launch an entity query (see Section 6.1.2) with that type restriction. To improve recall, we can also take the union of the top k predicted types. The result is a ranked list of entities, on which we can compute entity-level MAP, MRR, NDCG, as usual. In this setting, both B&N and our algorithm (discriminative) used YAGO as the catalog. Results for our dataset (Section 6.2) are shown in Figure 10.

k	MAP	MRR	%$\mathcal{Q}$ better	%$\mathcal{Q}$ worse
1	0.066	0.068	5.50	88.58
5	0.137	0.144	15.80	76.30
10	0.171	0.180	20.73	69.53
15	0.201	0.211	24.54	63.47
20	0.204	0.215	26.80	60.51
25	0.222	0.233	29.34	56.84
30	0.232	0.244	29.76	55.01
Generic	0.323	0.432	—	—

Figure 10: B&N-driven entity ranking accuracy.

We were surprised to see the low entity ranking accuracy of B&N (which is why we recreated very closely their reported type ranking accuracy on DBpedia). Closer scrutiny revealed that the main reason for lower accuracy was changing the type catalog from DBpedia (358 types) to YAGO (over 200,000 types). Entity ranking accuracy is low because B&N's type prediction accuracy is very low on YAGO: 0.04 MRR, 0.04 MAP, and 0.058 NDCG@10. For comparison, our type prediction accuracy is 0.348 MRR, 0.348 MAP, and 0.475 NDCG@10. This is entirely because of corpus/snippet signal: if we switch off snippet-based features ϕ_4, our accuracy also plummets. The moral seems to be, large organic type catalogs provide enough partial and spurious matches for *any* choice of hints, so it is essential (and rewarding) to exploit corpus signals.

6.4.5 Role of the corpus

A minimally modified B&N that uses the corpus may replace Wikipedia entity descriptions with corpus-driven descriptions, i.e., a pseudo-document made up of all snippets retrieved for a particular entity from the corpus. As we see in Figure 11, ranking accuracy improves marginally. This indicates that in the case of Web-scale entity search, an imperfectly annotated corpus can prove to be more useful than a small human-curated information source.

k	MAP	MRR	%$\mathcal{Q}$ better	%$\mathcal{Q}$ worse
1	0.070	0.078	5.08	88.01
5	0.163	0.170	15.94	73.77
10	0.213	0.222	22.28	63.47
15	0.237	0.246	26.66	55.99
20	0.270	0.279	29.34	49.65
25	0.277	0.287	30.89	45.98
30	0.287	0.299	32.16	42.45
Generic	0.323	0.432	—	—

Figure 11: B&N-driven entity ranking accuracy with corpus-driven entity description.

On an average, B&N type prediction, followed by query launch, seems worse than generic. This is almost entirely because of choosing bad types for many, but not all queries. There *are* queries where B&N shows a (e.g., MAP) lift beyond generic, but they are just too few (Figure 12).

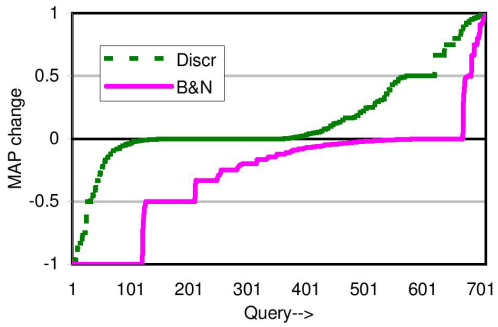

Figure 12: 2-stage entity ranking via B&N does boost accuracy for some queries, but the overall effect is negative. Joint interpretation and ranking also damages some queries but improves many more.

6.4.6 *Benefits of joint inference*

The beneficial role of the corpus is now established, but is *joint* inference really necessary, if a good query type interpreter were available? To test this in a controlled setting, we run our system, throw away the ranked entity list, and only retain the predicted type (Section 5.6.3), then launch a query restricted to this type (Section 6.1.2) and measure entity ranking accuracy.

	Joint	2-stage ($k=1$)	2-stage ($k=5$)	2-stage ($k=10$)
MAP	**0.462**	$0.370^{\downarrow}$	$0.361^{\downarrow}$	$0.365^{\downarrow}$
MRR	**0.481**	$0.384^{\downarrow}$	$0.375^{\downarrow}$	$0.377^{\downarrow}$

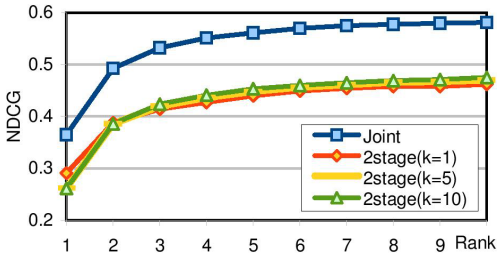

Figure 13: Joint inference improves entity ranking quality compared to 2-stage.

Figure 13 shows that the result is significantly (shown by "↓") less accurate than via joint inference, even after tuning k, which indicates that no *single* inferred type may retain enough information for the best entity ranking, and that *joint* inference is indeed vital.

6.4.7 *Coarse DBpedia types with Web corpus*

A plausible counter-argument to the above experiments is that, by moving from only 358 DBpedia types to over 20,000 YAGO types, we are making the type prediction problem hopelessly difficult for B&N, and that this level of type refinement is unnecessary for high accuracy in entity search. We modified our system to use types from DBpedia, and correspondingly re-indexed our Web corpus annotations using DBpedia types. As partial confirmation of the above hypothesis, the entity ranking accuracy using B&N did increase substantially. However, as shown in Figure 14, the entity ranking accuracy achieved by our discriminative algorithm remains unbeaten. Also compare with Figure 6 — whereas B&N improves by coarsening the type system, our discriminative algorithm seems to be degraded by this move.

k	MAP	MRR
1	0.135	0.145
5	0.240	0.250
10	0.295	0.305
Discr	0.422	0.437

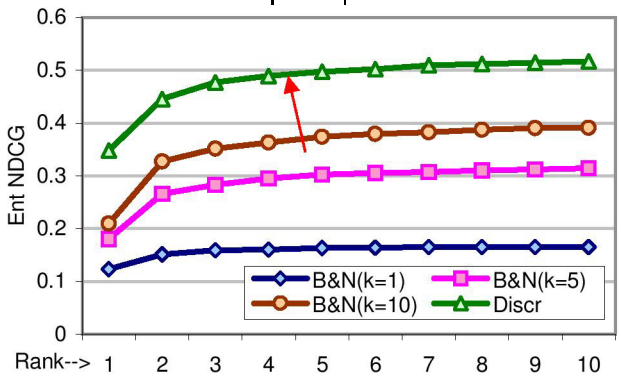

Figure 14: Entity ranking accuracy using DBpedia types.

6.4.8 *DEANNA on telegraphic queries*

We also tried to use the Web interface to send a sample of our telegraphic queries and their well-formed sentence counterparts to DEANNA [39] and receive back the interpretation. We manually inspected their output. Some anecdotes are shown in Figure 15. The queries are from Figure 3. None of the telegraphic queries was successfully interpreted. The well-formed questions saw partial success.

QID	Well-formed	Telegraphic
Q1	Missing target type	Empty
Q2	Incorrect, missed Wikipedia type "list of cities in China"	Incorrect fragments
Q3	Incorrect target type (painting)	Empty
Q4	Incorrect fragments	Incorrect fragments
Q5	Incorrect fragments	Empty
Q6	No target type	Empty

Figure 15: DEANNA interpretations of some of our queries.

7. CONCLUSION

We initiated a study of generative and discriminative formulations for joint query interpretation and response ranking, in the context of targeted-type entity search needs expressed in a natural "telegraphic" Web query style. Using 380 million snippets from a Web-scale corpus with 500 million documents annotated at 8 billion places with over 1.5 million entities and 200,000 types from YAGO, we showed experimentally that jointly interpreting target type and ranking responses is superior to a two-phase interpret-then-execute paradigm.

Our work opens up several directions for further research. Our notion of selectors can be readily generalized to allow mentions of entities as literals [15, 26] in the query. More sophisticated training using bundle methods may further improve the discriminative formulation. Finally, modeling listwise [21] losses, and/or exploring more powerful non-linear scoring functions (e.g., via boosting) may also help.

8. REFERENCES

[1] G. Agarwal, G. Kabra, and K. C.-C. Chang. Towards rich query interpretation: walking back and forth for mining query templates. In *WWW Conference*, pages 1–10. ACM, 2010.

[2] R. A. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

[3] K. Balog, L. Azzopardi, and M. de Rijke. Formal models for expert finding in enterprise corpora. In *SIGIR Conference*, pages 43–50, 2006.

[4] K. Balog, L. Azzopardi, and M. de Rijke. A language modeling framework for expert finding. *Information Processing and Management*, 45(1):1–19, 2009.

[5] K. Balog and R. Neumayer. Hierarchical target type identification for entity-oriented queries. In *CIKM*, pages 2391–2394. ACM, 2012.

[6] M. Bendersky, W. Croft, and D. Smith. Two-stage query segmentation for information retrieval. In *SIGIR Conference*, pages 810–811. ACM, 2009.

[7] C. Bergeron, J. Zaretzki, C. Breneman, and K. P. Bennett. Multiple instance ranking. In *ICML*, pages 48–55. ACM, 2008.

[8] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[9] P. Boldi and S. Vigna. MG4J at TREC 2005. In E. M. Voorhees and L. P. Buckland, editors, *TREC*, number SP 500-266 in Special Publications. NIST, 2005.

[10] A. Broder. A taxonomy of web search. *SIGIR Forum*, 36(2):3–10, 2002.

[11] A. Z. Broder, D. Carmel, M. Herscovici, A. Soffer, and J. Zien. Efficient query evaluation using a two-level retrieval process. In *CIKM*, pages 426–434. ACM, 2003.

[12] S. Chakrabarti, S. Kasturi, B. Balakrishnan, G. Ramakrishnan, and R. Saraf. Compressed data structures for annotated web search. In *WWW Conference*, pages 121–130, 2012.

[13] S. Chakrabarti, D. Sane, and G. Ramakrishnan. Web-scale entity-relation search architecture (poster). In *WWW Conference*, pages 21–22, 2011.

[14] R. Fagin, B. Kimelfeld, Y. Li, S. Raghavan, and S. Vaithyanathan. Understanding queries in a search database system. In *PODS Conference*, pages 273–284. ACM, 2010.

[15] J. Guo, G. Xu, X. Cheng, and H. Li. Named entity recognition in query. In *SIGIR Conference*, pages 267–274. ACM, 2009.

[16] J. Hoffart et al. Robust disambiguation of named entities in text. In *EMNLP Conference*, pages 782–792, Edinburgh, Scotland, UK, July 2011. SIGDAT.

[17] J. Hu, G. Wang, F. Lochovsky, J.-t. Sun, and Z. Chen. Understanding user's query intent with Wikipedia. In *WWW Conference*, pages 471–480. ACM, 2009.

[18] B. J. Jansen, D. L. Booth, and A. Spink. Determining the informational, navigational, and transactional intent of Web queries. *Information Processing and Management*, 44(3):1251–1266, May 2008.

[19] T. Joachims. Optimizing search engines using clickthrough data. In *SIGKDD Conference*, pages 133–142. ACM, 2002.

[20] S. Kulkarni, A. Singh, G. Ramakrishnan, and S. Chakrabarti. Collective annotation of Wikipedia entities in Web text. In *SIGKDD Conference*, pages 457–466, 2009.

[21] T.-Y. Liu. Learning to rank for information retrieval. In *Foundations and Trends in Information Retrieval*, volume 3, pages 225–331. Now Publishers, 2009.

[22] C. Macdonald and I. Ounis. Learning models for ranking aggregates. In *Advances in Information Retrieval*, volume 6611 of *LNCS*, pages 517–529. 2011.

[23] A. McCallum and K. Nigam. A comparison of event models for naive Bayes text classification. In *AAAI/ICML-98 Workshop on Learning for Text Categorization*, pages 41–48. AAAI Press, 1998.

[24] J. W. Murdock, A. Kalyanpur, C. Welty, J. Fan, D. A. Ferrucci, D. C. Gondek, L. Zhang, and H. Kanayama. Typing candidate answers using type coercion. *IBM Journal of Research and Development*, 56(3/4):7:1–7:13, 2012.

[25] P. Pantel and A. Fuxman. Jigs and lures: Associating web queries with structured entities. In *ACL Conference*, pages 83–92, Portland, Oregon, USA, June 2011.

[26] P. Pantel, T. Lin, and M. Gamon. Mining entity types from query logs via user intent modeling. In *ACL Conference*, pages 563–571, Jeju Island, Korea, July 2012.

[27] D. Petkova and W. B. Croft. Proximity-based document representation for named entity retrieval. In *CIKM*, pages 731–740. ACM, 2007.

[28] J. Pound, A. K. Hudek, I. F. Ilyas, and G. Weddell. Interpreting keyword queries over Web knowledge bases. In *CIKM*, 2012.

[29] J. Pound, I. F. Ilyas, and G. Weddell. Expressive and flexible access to Web-extracted data: a keyword-based structured query language. In *SIGMOD Conference*, pages 423–434. ACM, 2010.

[30] J. Pound, S. Paparizos, and P. Tsaparas. Facet discovery for structured Web search: a query-log mining approach. In *SIGMOD Conference*, pages 169–180, 2011.

[31] H. Raviv, D. Carmel, and O. Kurland. A ranking framework for entity oriented search using Markov random fields. In *Joint International Workshop on Entity-Oriented and Semantic Search*, pages 1:1–1:6, Portland, OR, 2012. ACM. Located with SIGIR Conference.

[32] S. Sarawagi. Information extraction. *FnT Databases*, 1(3), 2008.

[33] N. Sarkas, S. Paparizos, and P. Tsaparas. Structured annotations of Web queries. In *SIGMOD Conference*, 2010.

[34] U. Sawant and S. Chakrabarti. Features and aggregators for web-scale entity search. arXiv 1303.3164, 2013.

[35] F. M. Suchanek, G. Kasneci, and G. Weikum. YAGO: A core of semantic knowledge unifying WordNet and Wikipedia. In *WWW Conference*, pages 697–706. ACM Press, 2007.

[36] K. M. Svore, P. H. Kanani, and N. Khan. How good is a span of terms? exploiting proximity to improve Web retrieval. In *SIGIR Conference*, pages 154–161. ACM, 2010.

[37] T. Tao and C. Zhai. An exploration of proximity measures in information retrieval. In *SIGIR Conference*, pages 295–302. ACM, 2007.

[38] D. Vallet and H. Zaragoza. Inferring the most important types of a query: a semantic approach. In *SIGIR Conference*, pages 857–858. ACM, 2008.

[39] M. Yahya, K. Berberich, S. Elbassuoni, M. Ramanath, V. Tresp, and G. Weikum. Natural language questions for the Web of data. In *EMNLP Conference*, pages 379–390, Jeju Island, Korea, July 2012.

[40] C.-N. J. Yu and T. Joachims. Learning structural SVMs with latent variables. In *ICML*, pages 1169–1176. ACM, 2009.

[41] C. Zhai. Statistical language models for information retrieval: A critical review. *Foundations and Trends in Information Retrieval*, 2(3):137–213, Mar. 2008.