Knowledge Base Completion via Search-Based Question Answering

Robert West*, Evgeniy Gabrilovich†, Kevin Murphy†, Shaohua Sun†, Rahul Gupta†, Dekang Lin†

*Computer Science Department, Stanford University, Stanford, CA 94305
†Google, 1600 Amphitheatre Parkway, Mountain View, CA 94043

*west@cs.stanford.edu
†{gabr, kpmurphy, sunsh, grahul, lindek}@google.com

ABSTRACT

Over the past few years, massive amounts of world knowledge have been accumulated in publicly available knowledge bases, such as Freebase, NELL, and YAGO. Yet despite their seemingly huge size, these knowledge bases are greatly incomplete. For example, over 70% of people included in Freebase have no known place of birth, and 99% have no known ethnicity. In this paper, we propose a way to leverage existing Web-search–based question-answering technology to fill in the gaps in knowledge bases in a targeted way. In particular, for each entity attribute, we learn the best set of queries to ask, such that the answer snippets returned by the search engine are most likely to contain the correct value for that attribute. For example, if we want to find Frank Zappa's mother, we could ask the query *who is the mother of Frank Zappa*. However, this is likely to return 'The Mothers of Invention', which was the name of his band. Our system learns that it should (in this case) add disambiguating terms, such as Zappa's place of birth, in order to make it more likely that the search results contain snippets mentioning his mother. Our system also learns how many different queries to ask for each attribute, since in some cases, asking too many can hurt accuracy (by introducing false positives). We discuss how to aggregate candidate answers across multiple queries, ultimately returning probabilistic predictions for possible values for each attribute. Finally, we evaluate our system and show that it is able to extract a large number of facts with high confidence.

Categories and Subject Descriptors: H.2.8 [**Database management**]: Database applications—*Data mining*.
General Terms: Algorithms, Experimentation.
Keywords: Freebase; slot filling; information extraction.

1. INTRODUCTION

Large-scale knowledge bases (KBs)—e.g., Freebase [1], NELL [3], and YAGO [18]—contain a wealth of valuable information, stored in the form of RDF triples (subject–relation–object). However, despite their size, these knowledge bases are still woefully incomplete in many ways. For example, Table 1 shows relevant statistics for Freebase: in particular, it lists the fraction of subjects of type PERSON who have an unknown object value for 9 commonly used relations (also cf. Min et al. [12]); e.g., 71% of the roughly 3 million people in Freebase have no known place of birth, 94% have no known parents, and 99% have no known ethnicity. As Table 1 further shows, coverage is quite sparse even for the 100,000 most frequently searched-for entities. This problem is not specific to Freebase; other knowledge repositories are similarly incomplete.

The standard way to fill in missing facts in a knowledge base is to process a large number of documents in batch mode, and then to perform named-entity disambiguation followed by relation extraction (see, e.g., Ji and Grishman [8] for a recent review). We call this a 'push' model, since it pushes whatever facts it can find across all documents into the knowledge base. By contrast, in this paper, we focus on a 'pull' model, whereby we extract values for specific subject–relation pairs by making use of standard Web-search–based question-answering (QA) technology.

There are several reasons to take such an approach. First, we can leverage mature Web-search technology to find high-quality and up-to-date information sources. Second, we can rely on the returned search snippets as a mechanism for focusing attention on the parts of the documents that are most likely to contain the answer. Third, this gives us a complementary signal to the standard 'push' approach. The 'pull' paradigm enables a targeted, on-demand method for knowledge base completion; e.g., we could first run a 'push' method to collect as many facts as possible and then use our 'pull' system to retrieve facts that were not already found by the passive 'push' run. Finally, the world is constantly changing, and KBs must be kept up to date accordingly [19]; the 'pull' paradigm seems more appropriate than the 'push' paradigm for verifying whether specific previously entered facts are still valid.

The key question we address in this paper is which questions we should issue to the QA system. This is not obvious, since the QA system is expecting natural language as input, but we have no human in the loop who could formulate our queries. Furthermore, not all the queries are equally good. For example, suppose we want to determine the birthplace of the musician Frank Zappa. We could issue the search query *where does Frank Zappa come from*, but it is more effective to ask *where was Frank Zappa born*, because this query formulation will be more likely to match phrases appearing in the Web pages searched by the QA system.

As another example, consider the problem of determining Frank Zappa's mother. If we issue the query *who is the mother of Frank Zappa*, we will most likely get back snippets about 'The Mothers of Invention', which was the name of his band. In this case, we should add extra terms to the query, to try to steer the search engine to return snippets that mention his mother (cf. Collins-Thompson et al. [4]). One way to do so is to append to the query the name of the city where Zappa was born (namely, Baltimore), since the place where one was born is often mentioned in proximity to the names of one's parents.

*Research done during an internship at Google.

WWW'14, April 7–11, 2014, Seoul, Korea.
ACM 978-1-4503-2744-2/14/04.
http://dx.doi.org/10.1145/2566486.2568032.

Relation	Percentage unknown	
	All 3M	*Top 100K*
PROFESSION	68%	24%
PLACE OF BIRTH	71%	13%
NATIONALITY	75%	21%
EDUCATION	91%	63%
SPOUSES	92%	68%
PARENTS	94%	77%
CHILDREN	94%	80%
SIBLINGS	96%	83%
ETHNICITY	99%	86%

Table 1: Incompleteness of Freebase for some relations that apply to entities of type PERSON. *Left:* all 3M Freebase PERSON entities. *Right:* only the 100K most frequent PERSON entities.

The main contribution of this paper is to propose a way to learn which queries to ask the QA system for each kind of subject and relation. Our system is trained using search-query logs and existing facts in Freebase. We show that it is better to ask *multiple queries* and aggregate the results, rather than rely on the answers to a single query, since integrating several pieces of evidence allows for more robust estimates of answer correctness. At the same time, the *number of queries* to ask varies depending on the nature of the relation. On the one hand, relations that expect values from 'open' classes with large numbers of instances (e.g., CHILDREN, which expects values of type PERSON) are sensitive to the number of queries asked, and asking more than a certain number of queries decreases performance. The reason is that issuing more and more queries (of ever decreasing quality) increases the number of false positives, and if we ask too many queries, the negative impact of false positives will outweigh the positive impact of aggregating over several sources of information. On the other hand, if the relation expects values from a 'closed' class with only a limited number of instances (e.g., NATIONALITY, which expects values of type COUNTRY), the number of potential false positives is limited, and the performance will not suffer from asking more queries.

We evaluate our method by using it to fill in missing facts for 1,000 Freebase entities of type PERSON for each of the 9 relations shown in Table 1. This test set is chosen by stratified sampling from a larger pool of the 100,000 most frequently searched-for entities; thus, it contains a mix of head and tail entities. We show that we are able to reliably extract correct answers for a large number of subjects and relations, many of which cannot be extracted by conventional 'push'-type methods.

2. METHODOLOGY

In this section, we describe an end-to-end pipeline that uses a QA system in order to find new facts to add to Freebase. We first give a high-level overview—summarized in Fig. 1—before discussing each separate stage in detail.

In the knowledge base completion task [8], we are given a subject entity ID S and a relation ID R, and need to find the correct, previously unknown object entity IDs. For instance, we might be given subject ID /m/02whj (FRANK ZAPPA) and relation ID /m/01x3gb5 (PARENTS), and would be expected to return object ID /m/01xxvky (ROSE MARIE COLIMORE) or /m/01xxvkq (FRANCIS ZAPPA).

In this paper, we propose to use an existing Web-search–based QA system to perform the KB completion task. Since our QA system expects as input a query string, we need a way of lexicalizing subject–relation pairs to query strings. It is easy to look up one or more names (aliases) for the subject; the tricky issue is how to lexicalize the relation. To solve this, we mine a set of *query templates* from search query logs in an **offline training** phase (upper box of Fig. 1; Section 2.1), using a form of distant supervision [13] based on Freebase. For each relation R, this procedure constructs a set $\bar{Q}^R$ of templates. For example, *parents of* __ is a template for PARENTS; it can be instantiated for a subject S by looking up a name for S in Freebase and substituting it for the placeholder (e.g., *parents of Frank Zappa*). The same relation could also generate the template __ *mother*. We also estimate the quality of each such template using a labeled training set $\mathcal{T}^R$.

In the **KB completion** phase (lower box in Fig. 1), we process each subject–relation pair (S,R) in turn. We start by selecting N^R templates $\{\bar{q}_1,\ldots,\bar{q}_{N^R}\} \subseteq \bar{Q}^R$, based on the estimate of template quality computed offline, and instantiate them for S, obtaining the queries $\{q_1,\ldots,q_{N^R}\}$ (**query template selection,** Section 2.2). (A good value for N^R is also found during offline training.)

In the subsequent **question answering** step, each query q_i is fed to the QA system, which uses Web search to produce a scored list $\mathcal{A}_i$ of answer strings (Section 2.3).

In order to deal with the answers in Freebase, we must link them to the entities they refer to. This is done in the **answer resolution** step (Section 2.4), where each list $\mathcal{A}_i$ of answer strings is converted to a list $\mathcal{E}_i$ of answer entities.

In the next phase, **answer aggregation** (Section 2.5), we merge all answer rankings $\mathcal{E}_i$—one per query—into a single ranking $\mathcal{E}$.

It is desirable to have an estimate of the probability that the answer is correct. The QA system produces quality scores, but these are real numbers that cannot be directly interpreted as probabilities. Hence, in the final **answer calibration** step (Section 2.6), we translate the output scores to probabilities using a model Θ^R that was fit in the offline training phase, via supervised machine learning.

We now describe the individual system components in detail.

2.1 Offline training

Query templates are constructed in an offline training stage. The simplest templates consist only of a *lexicalization template*, i.e., a search query in which a placeholder has been substituted for the subject, as in *parents of* __. We first describe how we construct the set of lexicalization templates from Web-search logs, and then introduce a class of slightly more complex templates that allow for appending additional terms to a query.

Mining lexicalizations from search logs. Since our search-based QA system is geared to work on search queries entered by humans, we mine the lexicalization templates from logs of such queries, using a version of distant supervision [13] on Freebase. Our goal is to count for each relation–template pair $(R,\bar{q})$ how often the relation R is expressed by the lexicalization template $\bar{q}$ in the search-engine logs. To do so, we iterate over the logs, performing the following steps for each query q (e.g., *parents of Frank Zappa*).

1. Perform named-entity recognition on q, and link the resulting mentions to entities using approximate string matching techniques. (Note that standard entity linkage methods are of limited use here, as queries have little disambiguating context.) If q does not contain exactly one entity, discard it.
2. Let S (e.g., FRANK ZAPPA) be the subject entity contained in q. The template $\bar{q}$ is obtained by replacing the name of S with a placeholder string in q (e.g., *parents of Frank Zappa* becomes *parents of* __).
3. Run the QA system on q, obtaining the top-ranked answer string a (e.g., *Francis Zappa*). Link a to Freebase to get entity A. (When matching the answer entities, we have more

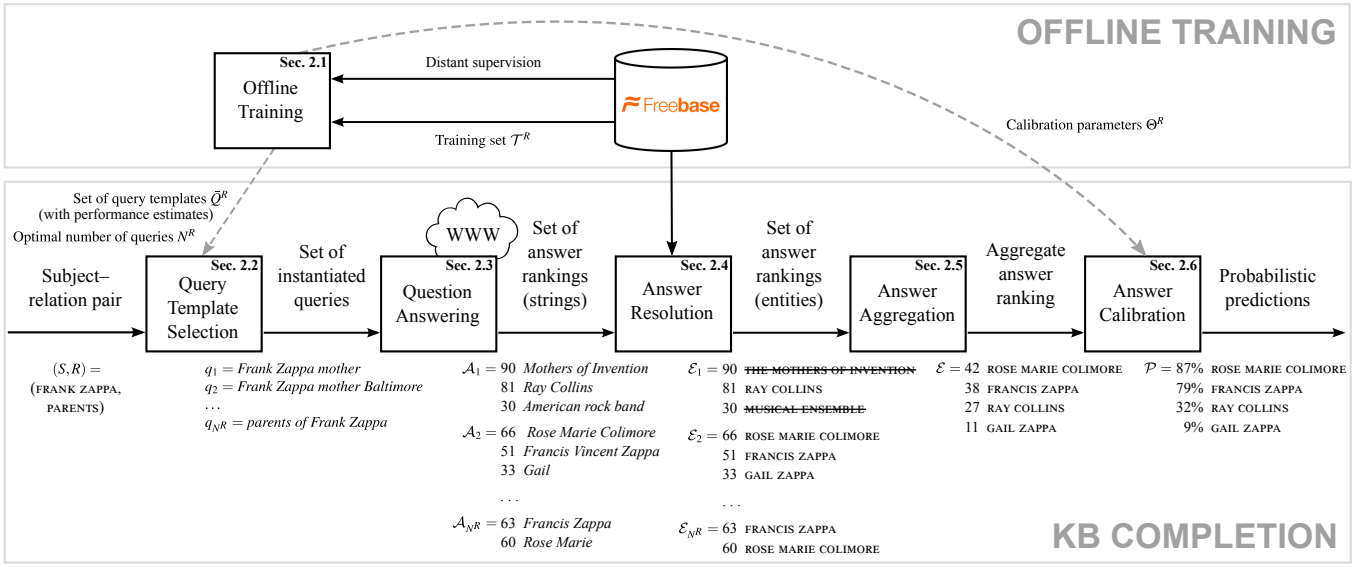


Figure 1: Overview of the pipeline for knowledge base completion, illustrated by the Frank Zappa example of Table 2. The set $\bar{Q}^R$ of query templates and the optimal number N^R of queries to use are relation-specific. The square boxes contain references to the sections in which the respective stages of the pipeline are described.

context, so we can use more sophisticated entity linkage methods, which we discuss in Section 2.4.)

4. If (S,A) is linked by a relation R (e.g., PARENTS) in Freebase, increase the count of $(R,\bar{q})$ by one.

For each relation R, we may then pick the most frequent templates as $\bar{Q}^R$, the set of lexicalization templates for R. Some of the most frequent lexicalization templates for PARENTS and PLACE OF BIRTH are listed as the horizontal axis labels of Fig. 2.

Query augmentation. The quality of such templates may vary depending on the subject. For instance, the query *Frank Zappa mother* retrieves mostly snippets that do not contain the answer, since most snippets are included because they mention Zappa's band, which was called 'The Mothers of Invention' (cf. row 1 of Table 2). However, adding more words to the query can shift the focus to more relevant documents. For example, the query *Frank Zappa mother Baltimore* will produce results of much higher quality, since passages that mention where someone was born are more likely to also contain who they were born to (cf. row 2 of Table 2; Zappa was born in Baltimore). This is also useful for disambiguation; e.g., *birthplace of Michael Jackson World Guide to Beer* is more likely to find the birthplace of the renowned late beer sommelier (and author of the 'World Guide to Beer'; cf. row 4 of Table 2) than the plain query *birthplace of Michael Jackson*, which retrieves mostly snippets about the late King of Pop (cf. row 3 of Table 2).

We refer to the process of attaching extra words to a query as *query augmentation*. Henceforth, when speaking of a query template, we mean a pair of a lexicalization template and an *augmentation template*. Augmentation templates simply specify a property (relation) for which a value is to be substituted. For instance, the query template consisting of the lexicalization template *__ mother* and the augmentation template PLACE OF BIRTH can be instantiated for the subject FRANK ZAPPA to *Frank Zappa mother Baltimore*. We also allow the empty augmentation template (i.e., no terms are appended to the lexicalization). If no value of the relation specified by the augmentation template is known for the subject, the query template cannot be instantiated.

While we currently focus on augmentations as above, where we append the name of an entity the subject is known to be in relation to, several other kinds of augmentation are conceivable. We discuss some of them in Section 5.

Manual template screening. In practice, we manually select 10 lexicalization templates from the top candidates found by the log-mining approach outlined above, and 10 augmentation templates from the relations pertaining to the subject type at hand; e.g., the augmentation templates for PARENTS and PLACE OF BIRTH are shown on the vertical axes of Fig. 2. Manual screening is not necessary but was done to reduce the number of queries during development and to facilitate ad-hoc result inspection.

2.2 Query template selection

Since query templates are defined as pairs of a lexicalization template and an augmentation template, the query space may be thought of as the Cartesian product of the set of lexicalizations and the set of augmentations. Many queries can be constructed in this two-dimensional space. We show individual examples of good vs. bad queries in Table 2. Below we discuss how to choose the good queries. But first we address the question: why not ask the QA system all the queries we can construct?

Dangers of asking too many queries. Issuing all possible queries is problematic, for two reasons. First, it is computationally challenging, since QA systems typically involve significant resources, such as CPU time, database lookups, or even, as in our case, Web-search queries. Second, it may be detrimental statistically: not all queries are equally good, so by asking all possible queries, we are likely to also ask many poor queries, which may dilute the result with false positives.[1] It is worth pointing out that the notion of a query's being poor is conditioned on the QA system's quality. By construction (cf. Section 2.1), all queries are good from the per-

[1]We use the term 'false positive' to denote a bad answer candidate that gets a high score from the QA system.

Query specification	Top result snippets (candidate answer strings in bold)
Subject–relation pair: (FRANK ZAPPA, PARENTS) **True answer:** ROSE MARIE COLIMORE **Template:** (__ *mother*, [no augmentation]) **Query:** *Frank Zappa mother*	**[1]** *The **Mothers of Invention** – Wikipedia, the free encyclopedia* The **Mothers of Invention** were an **American rock band** from California that served as the backing musicians for Frank Zappa, a self-taught composer and performer [. . .] **[2]** ***Ray Collins** of Frank Zappa's **Mothers of Invention** Dies | Billboard* **Ray Collins**, a singer who co-founded the **Mothers of Invention** with Frank Zappa but left when "too much comedy" started appearing in the band's songs, died on Monday [. . .]
Subject–relation pair: (FRANK ZAPPA, PARENTS) **True answer:** ROSE MARIE COLIMORE **Template:** (__ *mother*, PLACE OF BIRTH) **Query:** *Frank Zappa mother Baltimore*	**[1]** *Frank Zappa – Wikipedia, the free encyclopedia* Frank Vincent Zappa was born in Baltimore, Maryland, on December 21, 1940. His mother, **Rose Marie Colimore** [. . .]; his father, **Francis Vincent Zappa** [. . .] **[2]** *Frank Zappa statue to be dedicated in September – The Baltimore Sun* Frank Zappa statue to be dedicated in September. [. . .] His mother, **Rose Marie Colimore**, was a librarian, and his widow, **Gail**, lobbied to have the bust placed near a city library.
Subject–relation pair: (MICHAEL JACKSON (WRITER), PLACE OF BIRTH) **True answer:** LEEDS **Template:** (*birthplace of* __, [no augmentation]) **Query:** *birthplace of Michael Jackson*	**[1]** *Michael Jackson – Wikipedia, the free encyclopedia* Michael Jackson was born on August 29, 1958, in **Gary, Indiana**. He was the eighth of ten children in an African-American working-class family [. . .] in Gary, an industrial city near **Chicago**. **[2]** *Michael Jackson's House – **Gary, IN** – Yelp* 8 Reviews of Michael Jackson's House "WTF. this place is kind of a bummer. Streets aren't labeled, potholes aren't filled. The house is the only place on the block that was properly painted.
Subject–relation pair: (MICHAEL JACKSON (WRITER), PLACE OF BIRTH) **True answer:** LEEDS **Template:** (*birthplace of* __, WORKS WRITTEN) **Query:** *birthplace of Michael Jackson World Guide to Beer*	**[1]** *Michael Jackson (writer) – Wikipedia, the free encyclopedia* Jackson was born in **Leeds, West Yorkshire** and spent his early years in nearby **Wetherby**. [. . .] Jackson, Michael (1977). The World Guide to Beer [. . .] **[2]** *The Unique Michael Jackson | Philly Beer Scene* He was born in **Wetherby** in the city of **Leeds**. [. . .] Compensation eventually was awarded in 1988, when his agent [. . .] negotiated a fee on the re-write as The New World Guide to Beer.

Table 2: Example queries for two subject–relation pairs, alongside top result snippets retrieved by the search-based QA system (candidate answer strings in bold). The FRANK ZAPPA queries demonstrate how augmentation can shift the focus to more relevant snippets, the MICHAEL JACKSON (WRITER) queries, how augmentation can be useful for disambiguation.

spective of human users, but some of them are poor for our purposes because the QA system does not do well on them.

Heatmap representation of template quality. A compact way of visualizing query quality is afforded by the heatmaps in Fig. 2, which shows the query space for the PARENTS and PLACE OF BIRTH relations. Lexicalization templates are shown on the horizontal axes, while augmentation templates span the vertical axes. (Note that we only show the manually selected subsets of lexicalization and augmentation templates, but many more are possible.) The color encodes the average quality of queries instantiating the respective template, computed on the supervised training set $\mathcal{T}^R$. Quality is measured in terms of mean reciprocal rank (MRR, cf. Section 3.1.2) of the true answer in the answer ranking (after the answer resolution phase, cf. Section 2.4), i.e., larger values (brighter colors) are better. We see that, on average, some lexicalizations are better than others (e.g., the colloquial __ *mom* performs worst for PARENTS) and that some augmentations increase query quality over augmentationless queries (e.g., PLACE OF BIRTH helps for PARENTS, as in the Frank Zappa example from Section 2.1), whereas others decrease it (e.g., CHILDREN hurts for PARENTS).

Query selection strategies. A heatmap as in Fig. 2 may be computed for every relation (based on a training set of subjects for which the ground-truth answers are known in Freebase) and may subsequently serve as the basis for deciding which queries to send to the QA system. Given a heatmap of query quality, the exact choice of queries is determined by two factors.

First, we can decide how to pick templates from the heatmap. One option is to act greedily, always picking the templates with the largest values in the heatmap. Another option would be to add some diversity to the queries. A simple way to do this is to sample (without replacement) from the heatmap, by converting it to a probability distribution. A standard way to obtain such a distribution is to pass the values through the softmax function:

$$\Pr(\bar{q}) \propto \exp\left(\gamma\, \mathrm{MRR}(\bar{q})\right). \qquad (1)$$

In the above equation, γ is like an inverse 'temperature' parameter, which controls the degree of greediness. To set $\gamma = 0$ is to choose templates uniformly at random, and as γ is increased, ever more probability mass is shifted onto the highest-valued template.

Second, given γ, we can choose how many queries to pick. We are interested in finding the number N^R that optimally trades off the advantages of many queries (more pieces of evidence) against those of few queries (fewer false positives) when aggregation is done. In practice, we run the full pipeline on the training set $\mathcal{T}^R$ for a wide range of values and choose as N^R the value that yields the aggregated answer rankings with the highest MRR (cf. Section 3.1.2).

An exploration of the effects of varying the degree of greediness and the number of queries is presented in Section 3.

2.3 Question answering

In this paper, we use an in-house natural-language QA system. Since the system is proprietary, we cannot give all the details, but we outline the basic approach below (see also Paşca [14]).

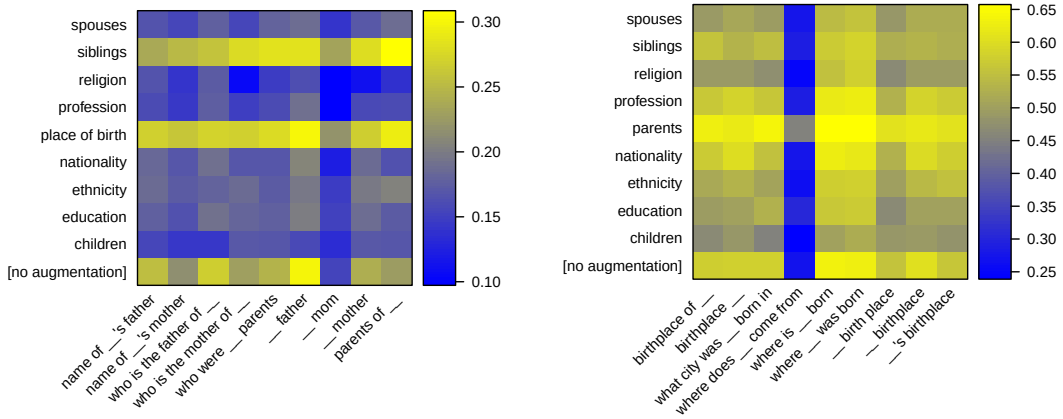
Input. A search query that can be answered by short phrases. It may be a natural-language question, such as *who was Frank Zappa's mother*, or a terser query, as in *Frank Zappa mother Baltimore* (the latter is used as an example in this section).

Output. A list of candidate answer strings, ranked according to an internally computed answer quality score.

Step 1: Query analysis. Find the head phrase of the query (*mother*). When applying the QA system in our pipeline, we can set the head phrase explicitly, as we generate the queries given a relation.

Step 2: Web search. Issue the input query to the search engine, retrieving the top n result snippets, where n is a tuneable parameter (we choose $n = 50$). Two snippets for the query *Frank Zappa mother Baltimore* are reproduced in row 2 of Table 2.

Step 3: Snippet analysis. Score each phrase in the result snippets with respect to how good an answer it is to the input query.



(a) PARENTS (b) PLACE OF BIRTH

Figure 2: Heatmaps for the (a) PARENTS and (b) PLACE OF BIRTH relations, capturing the performance (mean reciprocal rank; cf. Section 3.1.2) of a number of query templates on the training set. Each combination of lexicalization template *(horizontal axes)* and augmentation template *(vertical axes)* defines a query template. Brighter colors signify higher MRR, i.e., better performance.

Each phrase is represented as a vector of features, and the score is computed as a weighted sum of these features, with weights fitted ahead of time via supervised machine learning. For instance, *Rose Marie Colimore* is a good candidate because it is contained in a highly ranked snippet, is a noun phrase, has high inverse document frequency, appears close to the query term *mother*, and is highly related to the head phrase *mother* of the query (since both typically appear in person-related contexts in large text corpora).

Step 4: Phrase aggregation. The same phrase may appear several times across all snippets (e.g., *Rose Marie Colimore* appears twice in row 2 of Table 2), and each instance is scored separately in step 3. This step computes an aggregate score for each distinct phrase, again via machine learning, based on features such as the number of times the phrase appears and the average and maximum values (over all instances of the phrase) of the features from step 3.

2.4 Answer resolution

For each query q_i, the QA system returns a list $\mathcal{A}_i$ of answer strings, but what we want is a list of entities $\mathcal{E}_i$. For this, we use standard entity linkage techniques, such as [6], which takes into account the lexical context of each mention, and [7], which takes into account other entities near the given mention, using joint inference. For example, if we see the string *Gail*, it could refer to GAIL, a river in Austria, but if the context is *Zappa married his wife Gail in 1967*, it is more likely to be referring to the person GAIL ZAPPA (cf. Fig. 1).

Since we know the type of answer we are looking for, we can use this as an additional constraint, by discarding all incorrectly typed answer entities (e.g., THE MOTHERS OF INVENTION and MUSICAL ENSEMBLE in Fig. 1).

2.5 Answer aggregation

After the answer resolution step, we have one ranking of correctly typed answer entities for each query. But since, in general, we issue several queries per subject–relation pair to the QA system, we need to merge all of their rankings into a single answer ranking.

We adopt a simple yet effective approach, computing an entity's aggregate score as the mean of its ranking-specific scores. Assume we asked the QA system N^R queries $q_1,\ldots,q_{N^R}$ for the subject–relation pair (S,R), resulting in N^R rankings $\mathcal{E}_1,\ldots,\mathcal{E}_{N^R}$. Let Ω be the set of entities occurring across all these rankings. Each entity $E \in \Omega$ has a score in each ranking $\mathcal{E}_i$, referred to as $s_i(E)$; if E does not appear in $\mathcal{E}_i$, we set $s_i(E) = 0$. Now, E's overall score $s(E)$ is computed as its average score across all rankings, i.e., $s(E) = \frac{1}{N^R}\sum_{i=1}^{N^R} s_i(E)$. The answer entities Ω alongside the scores s define the aggregated ranking $\mathcal{E}$ for (S,R).

This eliminates false positives that are ranked high in a single ranking (e.g., RAY COLLINS in Fig. 1), possibly because the respective query was of low quality. On the contrary, entities appearing in many rankings, but not necessarily on top, are generally ranked high in the aggregate ranking, as they contribute fewer ranking-specific scores of zero (e.g., ROSE MARIE COLIMORE in Fig. 1).

2.6 Answer calibration

The goal of the answer calibration step is to turn the scores attached to entities in the aggregate ranking into probabilities that tell us how likely an entity is to be the true object. Such interpretable scores are important if we want to make informed decisions on how to act upon a proposed answer: whether we want to discard it immediately; how we should prioritize it for validation by humans; or what weight to give it in a knowledge fusion algorithm for merging evidence from different fact extraction methods.

To map QA scores to probabilities, we apply logistic regression to the QA scores (a standard technique called *Platt scaling* [16]); the model was trained on an independent development set. We investigated more sophisticated features, such as the number of times each entity appeared across multiple query responses, but this did not seem to help. Note that multiple answers can be correct (e.g., people can have multiple parents and spouses), so the probabilities do not sum to 1 across answers; rather, each individual calibrated answer is a number between 0 and 1.

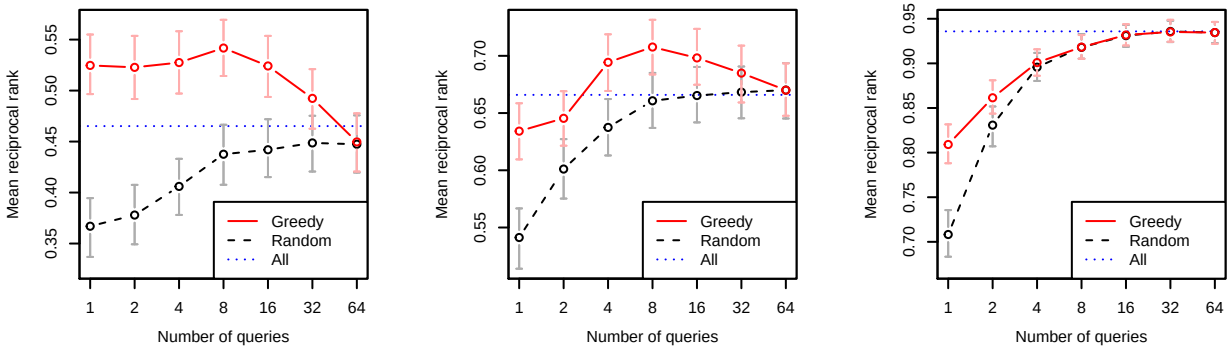
3. EMPIRICAL EVALUATION

Having described the full pipeline in Section 2, we now evaluate it. We proceed by first introducing our test data and quality metrics. Then, we evaluate our answer rankings, and last, we investigate the quality and quantity of our final, probabilistic predictions.

3.1 Experimental setup

3.1.1 Training and testing data

For testing our method, we consider the 9 relations of Table 1. To be able to train and test our method, we need to have, for each rela-



(a) SPOUSES (b) PLACE OF BIRTH (c) NATIONALITY

Figure 3: Performance in terms of MRR for three representative relations (with bootstrapped 95% confidence intervals). For performance and curve shape of the remaining test relations, cf. Table 3. The (logarithmic) x-axes show the number of queries fed to the QA system. *Solid red:* Greedily selecting the query templates that perform best on the training set. *Dashed black:* Selecting queries uniformly at random. *Dotted blue:* Selecting all available queries.

tion R, a number of subjects alongside the ground-truth objects they are connected to by R. We obtain this ground truth from Freebase by sampling subjects with known values for R.

Hereby, we make what we call the 'local closed-world assumption': Assume Freebase has a non-empty set of objects $\mathcal{O}$ for a given subject–relation pair (S,R). The local closed-world assumption then posits that $\mathcal{O}$ contains *all* ground-truth objects for (S,R).

In selecting subjects, we restrict ourselves to the 100,000 most frequently searched-for persons. We repeat the following stratified sampling procedure twice, to construct (1) the training sets $\mathcal{T}^R$ and (2) test sets for each R: For each relation, consider only the subjects (from the base set of 100,000 persons) for which the objects are known. Divide this subject set into 100 percentiles (with respect to frequency) and randomly sample 10 subjects per percentile, for a total of 1,000 subjects per relation.

The rationale for restricting ourselves to the top 100,000 persons was that such frequent entities tend to be of higher interest to the general user, while at the same time, Freebase is still rather incomplete even in this regime (cf. Table 1). Also, it is important to note that, although our base set encompasses only about 3% of Freebase's roughly 3 million person entities, most of them are likely to be unknown to most users (for example, the tail of the top 100,000 contains persons such as BIRTHE KJÆR, a Danish singer, or MOHAMMAD-REZĀ LOTFI, a Persian classical musician).

We manually select 10 lexicalization templates for each relation. As augmentation templates, we use 10 relations: our 9 test relations (see above) plus RELIGION. Of course, when testing on relation R, we are not allowed to use R itself for query augmentation, so there are $10\times(10-1)=90$ candidate templates per subject–relation pair (not all of which can be necessarily instantiated for every subject, since the relation specified by an augmentation template might not be known for every subject).

3.1.2 *Ranking metrics*

Next, we introduce the ranking metrics used to quantify performance. Consider a subject–relation pair (S,R) with the set $\mathcal{O}=\{O_1,\ldots,O_n\}$ of ground-truth objects, and assume we want to evaluate an entity ranking $\mathcal{E}$. Let $r_1<\ldots<r_n$ be the ranks of the elements of $\mathcal{O}$ in $\mathcal{E}$, in ascending order. The rank of elements of $\mathcal{O}$ not appearing in $\mathcal{E}$ is defined as infinity. Then, the *reciprocal rank (RR)* of $\mathcal{E}$ is defined as the reciprocal of the rank of the highest-ranked true answer, i.e., as $1/r_1$. Averaging over several rankings yields the *mean reciprocal rank (MRR)*. The reciprocal of the MRR is the harmonic mean rank of the highest-ranked true answers.

If the emphasis is on retrieving each, rather than any, true answer from $\mathcal{O}$, another useful metric is *average precision (AP)*, defined as $\frac{1}{n}\sum_{i=1}^{n} i/r_i$. Averaging over several rankings yields the *mean average precision (MAP)*.

For both RR and AP, the best possible value is 1, and the worst possible, 0. RR upper-bounds AP, and if $n=1$ (e.g., because R is a functional relation), RR equals AP.

3.2 Quality of answer rankings

We previously stated the intuition that issuing too many queries to the QA system may be harmful because of the negative impact of false positives (answers that get ranked unduly high), and that we might counteract this effect by asking a smaller set of well selected queries. The goal of the first part of this section is to show that this is indeed the case, by evaluating different query selection methods. In the second part of this section, we illustrate the effects of query subselection in more detail by performing a more fine-grained analysis on a per-subject basis.

3.2.1 *Subselecting queries for aggregation*

We now perform an evaluation of the effects of subselecting queries for aggregation. Recall that query selection is based on heatmaps as in Fig. 2 (one per relation), which are computed in an offline training stage and which quantify, for each template, how well it performs on our set of 1,000 training subjects. Also recall from Section 2.2 that we can act at different degrees of greediness when selecting templates according to these heatmaps. Further, for each greediness level, we can ask the QA system any number of queries, up to the number of queries available for the input subject–relation pair (around 90, cf. Section 3.1.1).

Fig. 3 explores these combinations of greediness level and number of queries asked. Each panel pertains to one relation and contains one curve for each of two greediness levels: random in dashed black ($\gamma=0$ in (1)) and greedy in solid red ($\gamma\to\infty$). The x-axes show the number N^R of queries, the y-axes, performance for the respective combination of greediness and number of queries (measured as the MRR across all aggregate rankings, one ranking for each of the 1,000 test subjects). Finally, the blue dotted horizontal lines indicate the MRR achieved when aggregating over all available queries; i.e., if we extended the x-axes as far to the right as

Relation R	MRR (N^R)	MRR (all)	MAP (N^R)	MAP (all)	N^R	Greedy-curve shape	Closedness
SPOUSES	0.54	0.47	0.50	0.43	8	inverted U	0.010
PARENTS	0.33	0.28	0.25	0.22	8	inverted U	0.013
SIBLINGS	0.30	0.27	0.24	0.23	8	inverted U	0.015
CHILDREN	0.25	0.20	0.18	0.14	8	inverted U	0.018
PLACE OF BIRTH	0.71	0.67	0.71	0.67	8	inverted U	0.026
EDUCATION	0.83	0.82	0.78	0.77	32	diminishing returns	0.063
PROFESSION	0.58	0.58	0.47	0.46	16	diminishing returns	0.21
NATIONALITY	0.94	0.94	0.93	0.93	32	diminishing returns	0.24
ETHNICITY	0.78	0.77	0.76	0.76	32	diminishing returns	0.28

Table 3: Performance of our system on 9 relations. We show MRR and MAP for two query selection strategies: (1) greedily selecting the optimal number N^R of queries (corresponding to the highest values of the red curves in Fig. 3); (2) selecting all available queries (corresponding to the horizontal lines in Fig. 3). We also show the closedness for all relations (cf. Section 3.2.1 for a definition).

possible, the curves for all greediness levels would necessarily converge to the horizontal lines. For space reasons, we show plots for three representative relations only, but the observations that follow apply equally to the relations not shown in Fig. 3. The results for all relations are summarized in Table 3.

Greedy is best. The reason we restrict the plots to the two extreme greediness levels is that we found that intermediate levels lie strictly in between: the more we explore, the more we approach the performance of random selection. So the first observation is that greedy query selection works best for all relations (for performance metrics, cf. the MRR and MAP columns in Table 3).

Asking too many queries can hurt. As a second point, the greedy (red) curves also reveal that performance depends on the number of queries asked. In all cases, we do better by asking the QA system more than one query. In some cases, it is best not to ask too many queries, manifest in an inverted-U shape (SPOUSES and PLACE OF BIRTH in Fig. 3, but also PARENTS, SIBLINGS, CHILDREN). In these cases, we achieve the best performance by asking 8 queries. In other cases, asking more queries is always better, manifest in a diminishing-returns shape (NATIONALITY in Fig. 3, but also ETHNICITY, EDUCATION, PROFESSION). The columns 'N^R' and 'Greedy-curve shape' of Table 3 summarize the shapes of the greedy (red) curves for all test relations. While this table indicates that N^R is optimally chosen as 16 or 32 for the relations with diminishing-returns curves, the MRR and MAP achieved for those values is only marginally better than for $N^R = 8$ (cf. Fig. 3(c)), so we conclude that issuing $N^R = 8$ queries is a good choice for all R.

Open vs. closed relations. Whether a relation exposes an inverted-U or a diminishing-returns shape has to do with the answer type it expects; e.g., SPOUSES expects an object of type PERSON, an 'open' type with a large number of instances. This means that there are many potential false positives, and by asking more and more queries of ever poorer quality, we introduce ever more of them into the aggregate answer ranking, which makes the greedy (red) curve decrease. On the other extreme, NATIONALITY expects objects of type COUNTRY, a 'closed' type with only around 200 instances, such that the number of potential false positives is very limited.

To put this intuition in numbers, we compute, for each relation, the number of unique answer entities contained in all rankings across all subjects and queries. Similarly, we compute the number of all unique *ground-truth* answers across all subjects and queries. Dividing the second by the first number yields the fraction of all distinct answers that are ever true answers (akin to the notion of precision), which we refer to as 'closedness'. The results of this calculation are displayed in the 'Closedness' column of Table 3. We see that the closedness is lowest for person-typed relations and highest for the predicates ETHNICITY and NATIONALITY. The values are significantly larger for relations with a diminishing-returns shape than for those with an inverted-U shape.

In conclusion, the answer to the question whether we can profit from the robustness of aggregation without injecting too many false positives is, Yes: by asking a small, well chosen fraction of all available queries, we do better than by asking a single query and at least as well as by asking all available queries.

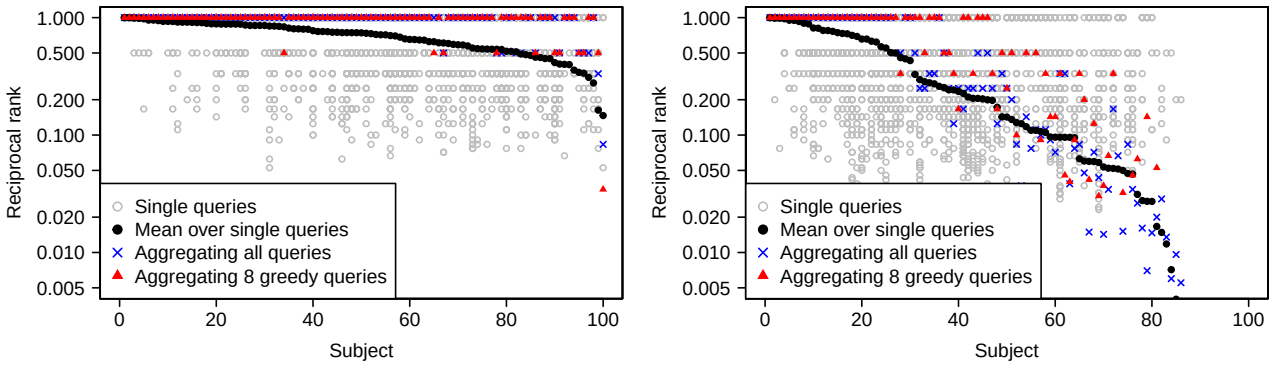
3.2.2 Subject-level analysis

To better understand the effects of aggregating the rankings resulting from multiple queries, let us consider Fig. 4. In these plots, each column (x-value) represents one of 100 randomly sampled test subjects. Within each column, there is one gray circle per query, with the y-axis showing the corresponding RR (on a logarithmic scale, i.e., values of 0 do not appear). The per-subject MRR is obtained by taking column-wise averages, plotted as black dots. Subjects are sorted on the x-axis in order of increasing MRR (such that the black curve is descending by design). The blue crosses show the RR when aggregating over *all* queries available for the respective subject (around 90, cf. Section 3.1.1), while the red triangles show the RR when aggregating 8 greedily chosen queries (since $N^R = 8$ was found to be a good value in Section 3.2.1). That is, the average of all black dots equals the value of the corresponding black curve in Fig. 3 at $x = 1$; the average of all blue crosses equals the value of the corresponding blue horizontal line; and the average of all red triangles equals the value of the corresponding red curve at $x = 8$.

We investigate two representative relations. Fig. 4(a) shows the results for NATIONALITY, a 'closed' relation (diminishing-returns shape in Fig. 3) on which we do nearly perfectly (MRR 0.94, or harmonic mean rank 1.1). Fig. 4(b) visualizes performance for SPOUSES, an 'open' relation (inverted-U shape in Fig. 3) on which performance, while still good (MRR 0.54, or harmonic mean rank 1.9), is well inferior to that on NATIONALITY.

We see that, in the case of NATIONALITY, aggregating all available queries (blue crosses, often occluded by the red triangles) is very effective, to the extent that for nearly all subjects, the aggregate ranking has an RR of 1 (for an MRR of 0.94 across all subjects). That is, aggregating over all available queries achieves virtually the same performance as if we chose the single best query for the respective test input—which is, of course, impossible, since we cannot know ahead of time which query will perform best (we only have estimates from the training phase).

As Fig. 4(b) demonstrates, the SPOUSES relation is considerably harder. While for about 40% of subjects, aggregating over all available queries (blue crosses) places a true answer at rank 1, there also is a considerable number of subjects for which aggregating does



(a) NATIONALITY (b) SPOUSES

Figure 4: Subject-level performance analysis (cf. Section 3.2.2).

not outperform random query selection (where blue crosses lie beneath black dots). Although for many of these subjects there is at least one query for which a true answer gets rank 1 (gray circles), blindly aggregating all queries (blue crosses) often cannot recover it. The reason is that, among all available queries, there are many of poor quality, which overwhelm the aggregate ranking with false positives. In this case, more careful query selection helps: on average, the red triangles lie significantly above the blue crosses (MRR 0.54 vs. 0.46). In particular, note that several red triangles achieve an RR of 1, while their blue-cross counterparts lie further below.

3.3 Quality of calibrated predictions

As motivated in Section 2.6, it is desirable to know for each answer candidate how likely it is to be correct. Computing this probability (also called *confidence*) is the goal of the answer calibration step. In this section, we evaluate this step, followed by an analysis of the number of high-confidence predictions our system makes.

Quality of answer calibration. We proceed as follows, for each relation R separately. For each of the 1,000 test subjects for R, run the full pipeline (using greedy selection of $N^R = 8$ queries, which was found to be near-optimal in Section 3.2.1), resulting in one aggregate ranking with calibrated scores per subject. Consider the set of all answer entities, across all subjects, and partition it according to the calibrated scores. For partitioning, we divide the range $[0\%, 100\%]$ into 20 buckets spanning 5% each. Under perfect calibration, the fraction of true answers in each bucket should lie within the range that defines the bucket.

Graphically, this translates to the following requirement. If we plot the 20 probability buckets on the x-axis and the fraction of true answers per bucket on the y-axis, we want the resulting curve to lie as close to the diagonal running through the origin as possible. Fig. 5 visualizes the results of this graphical test for the same three relations depicted in Fig. 3. For these relations (and equally for the ones not plotted) the diagonal is followed closely, which implies that calibration works well.

Number of high-quality answers. Eventually, we are interested in making a large number of high-quality predictions, since those are the best candidates to be suggested for Freebase.[2] We can get an idea of the number of high-quality predictions by counting how many predictions we make with high confidence. The results for all 9 test relations are summarized in Table 4, which contains the

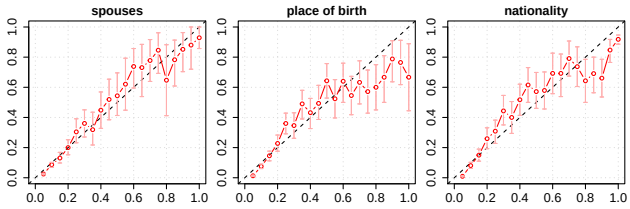


Figure 5: Calibration results (with bootstrapped 95% confidence intervals). *Horizontal axes:* Predicted probability, binned in 20 buckets of width 5%. *Vertical axes:* Fraction of positive examples in bucket.

numbers of facts extracted above different confidence thresholds. Since our test set contains 1,000 subjects for each relation, a value of 1,000 means that we predict one fact per subject on average with a confidence above the respective threshold, a value of 100 implies we predict one fact per 10 subjects, etc.

When evaluating the quality of answer rankings returned by our method (cf. Table 3), we found performance to be lowest on CHILDREN (MRR 0.25; for an explanation, cf. the discussion in Section 5). However, we also only extract 8 facts with a confidence above 50% for CHILDREN, so our system knows that its answers are poor in this case, which is crucial for making the output actionable. Our answer rankings are best for NATIONALITY (MRR 0.94), which is reflected in high confidence values: we extract 366 facts with a confidence over 90%, i.e., over one per three subjects.

For completeness, each number of extracted facts in Table 4 is followed (in parentheses) by the fraction of facts that are correct, such that multiplying the two numbers in each cell yields the total number of *correct* facts for the respective confidence threshold.[3]

Precison and recall. Fig. 6 shows precision–recall curves (interpolated [10]) for the three representative example relations (those also shown in Figs. 3 and 5). These curves were computed for a single ranking per relation, formed by listing all predictions for the relation (across subjects) in order of confidence. As expected, the curve for NATIONALITY looks best: since it is a closed relation (cf. Section 3.2), the impact of false positives is limited, and precision stays high even as recall is increased. Further, PLACE OF BIRTH is more

[2] In practice, all automatically extracted facts are screened by human raters before they are added to Freebase.

[3] Sometimes we are too confident in our top predictions (e.g., for PROFESSION, of the facts with a confidence above 90%, only 65% are correct). But since human raters verify all facts before they are added to Freebase, perfect precision is not our main concern.

Relation	> 10%	> 30%	> 50%	> 70%	> 90%	Novel
SPOUSES	1,395 (0.37)	518 (0.64)	293 (0.79)	160 (0.84)	67 (0.91)	14%
PARENTS	1,278 (0.21)	213 (0.48)	78 (0.63)	35 (0.63)	7 (0.57)	38%
SIBLINGS	958 (0.21)	168 (0.50)	66 (0.65)	22 (0.73)	2 (1.00)	19%
CHILDREN	753 (0.20)	62 (0.48)	8 (0.62)	0 (—)	0 (—)	—
PLACE OF BIRTH	1,723 (0.38)	766 (0.57)	426 (0.62)	209 (0.67)	52 (0.73)	15%
EDUCATION	2,400 (0.44)	1,222 (0.66)	857 (0.74)	535 (0.78)	173 (0.82)	19%
PROFESSION	2,405 (0.31)	719 (0.53)	388 (0.62)	202 (0.65)	68 (0.65)	30%
NATIONALITY	1,747 (0.53)	1,061 (0.71)	748 (0.79)	557 (0.83)	366 (0.90)	15%
ETHNICITY	1,805 (0.44)	909 (0.63)	601 (0.70)	408 (0.75)	175 (0.85)	31%

Table 4: Numbers of facts extracted above different confidence thresholds, for 1,000 subjects per relation. *Parentheses:* Precision, i.e., fraction of correct facts. The column labeled 'Novel' contains the percentage of facts extracted with a confidence above 70% that are found by none of a collection of complementary methods (cf. Section 3.3).

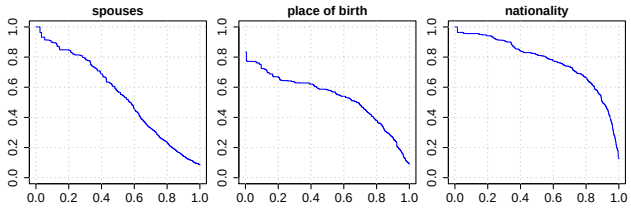


Figure 6: Precision–recall curves. *Horizontal axes:* Recall. *Vertical axes:* Precision.

closed than SPOUSES and achieves higher MRR and MAP (cf. Table 3). Therefore, it is not surprising that the precision–recall curve for PLACE OF BIRTH is more concave than for SPOUSES. Note, however, that at low levels of recall, SPOUSES achieves higher precision than PLACE OF BIRTH, i.e., our most confident predictions for SPOUSES are better than for PLACE OF BIRTH. This demonstrates that, even if the quality of the *uncalibrated* answer rankings for the average subject (which is what MRR and MAP capture) is worse, there still is value in our overall top predictions across all subjects when considering the *calibrated* answer scores.

Novelty of extracted facts. Finally, we compare the overlap of the facts extracted by the present system with facts extracted by our in-house state-of-the-art 'push' system [5] (which is similar to Ji and Grishman [8]). Concretely, we consider the predictions we make with a confidence above 70% and compute the fraction that are not found by the conventional 'push' methods (also with a confidence above 70%). The values range from 14% (SPOUSES) to 38% (PARENTS), with a mean of 23% over all 9 relations. (All values are listed in the right column of Table 4.) We conclude that our 'pull' method adds substantial value over existing 'push' methods.

4. RELATED WORK

Related work can be divided into three main areas: papers about question answering (QA), papers about knowledge base completion (KBC), and papers about using QA to solve the KBC task. We briefly review each of these below.

The field of general QA has been popular for a long time. A milestone was the introduction, in 1999, of a specialized track related to QA into the annual competition held at the Text Retrieval Conference [20]. Many systems in this competition, as well as our own system, are based on the approach outlined by Paşca [14]. However, our focus is not developing better QA technology, but rather addressing the issue of how to use such systems for KBC.

The KBC task has grown in popularity as a research topic after being introduced as an annual competition in 2008 to the Text Analysis Conference [11]. Good summaries of the standard approaches to this task are given by Ji and Grishman [8] and Weikum and Theobald [21]. Most of these methods process each document in turn according to a 'push' model (cf. Section 1), extracting as many facts as possible by using named-entity linkage and (supervised) relation extraction methods.

In this paper, we focus on a 'pull' model, whereby we try to retrieve individual documents to fill in specific facts, using QA technology. While this is a relatively new approach, there are some related works. The most similar is perhaps Kanani and McCallum's [9] work on using reinforcement learning to learn an optimal policy for efficiently filling in missing values in a KB (they focus on filling in the email address, job title, and department affiliation of 100 professors at UMass Amherst). The actions available are to perform one of 20 possible types of query (e.g., name, name + "CV", name + "Amherst"), to download one of the n resulting Web pages, or to extract one of the three relations from the page. By contrast, we learn the value of each possible query formulation using a myopic strategy; we always process $n = 50$ snippets resulting from search; and we extract the values from each snippet independently.

OpenEval [17] focuses on classifying if a given subject–relation–object triple is true or not, based on retrieved Web pages, whereas we focus on returning all high-confidence object values for a given subject–relation pair based on snippets. A further difference is that OpenEval glosses over the distinction between entities and their names, or mentions, which can cause problems due to synonymy.

Another related approach is 'Conversing Learning' [15]. Here, the goal is to formulate natural-language questions about inference rules (e.g., 'Is it true that, if X and Y have children in common, then they must be married?') used by the NELL system [3], and to pose these questions to Twitter and Yahoo! Answers, hoping that humans will answer 'yes' or 'no' to the questions. By contrast, we do not ask humans, but instead perform targeted Web searches, and our questions are about specific facts rather than inference rules.

Finally, Byrne and Dunnion [2] formulate one query per subject–relation pair, using manually constructed templates, and search a small collection of documents to retrieve answers. By contrast, we learn how to formulate the queries, and we search the entire Web.

5. DISCUSSION

The main goal of this paper is to present and evaluate an end-to-end pipeline for knowledge base completion based on search-based question answering. While it is fully functional and works well on our evaluation data, many more improvements can be made. The purpose of this section is to discuss the separate parts of the pipeline, pointing out common failure modes and highlighting potential directions for future work.

Query construction (Section 2.1). Several further kinds of augmentation beyond appending known properties are conceivable. For instance, we could add phrases that tend to co-occur with the correct answer on Web pages (e.g., the strings *hospital* or *was born in* could help in queries for PLACE OF BIRTH). Also, when choosing which properties to augment with, we could attempt to pick ones that disambiguate between entities with similar names; e.g., when the subject is called Michael Jackson (as in Table 2), appending the value of PROFESSION is better than appending the value of GENDER, since the latter is shared by the two ambiguous subjects, while the former distinguishes them.

Another interesting idea for query augmentation would be to admit the exclusion operator when constructing queries. This could provide a tool for explicitly reducing the number of bad snippets retrieved by the QA system. For instance, snippets containing the word *music*—most likely about Michael Jackson the singer rather than the beer sommelier—would be avoided by the query *Michael Jackson birthplace –music*.

Query selection (Section 2.2). Currently, query choice is done in batch mode: for each relation R, we first choose a predetermined number N^R of queries and then feed them to the QA system all at once. An alternative approach could follow a sequential rather than a batch paradigm, asking one query at a time, and inspecting the aggregated and calibrated ranking after each query. This process could continue until the calibrated probabilities of the top-ranked answers are high enough or the ranking has stayed stable for a while. Such a setup would be more adaptive with respect to the number of queries asked and could thus be potentially more effective at avoiding to ask too many queries (cf. [9]).

Question answering (Section 2.3). It is a strength of our method that it leverages powerful Web-search machinery for retrieving relevant and up-to-date information that is independent of Freebase. Nonetheless, in the evaluation (cf. Tables 3 and 4) it became clear that our system works better on some relations than others. We have already discussed the different properties of 'open' vs. 'closed' relations (Section 3.2.1). Still, there remain effects that are not explained by this distinction. Consider, e.g., the relations SPOUSES and CHILDREN, both expecting objects of type PERSON. Although SPOUSES is arguably even more 'open' than CHILDREN (cf. Table 3, where SPOUSES has the lowest closedness), our performance is considerably better for SPOUSES than for CHILDREN (MRR 0.54 vs. 0.25). Error analysis led us to conclude that the effect is due to the QA system: result snippets that mention the subject's children often also mention their spouse, to the extent that, in some cases, the spouse appears more often in the snippets than the children themselves, so our QA system, which uses frequency of occurrence among its main features, may return the spouse in place of the children. It is also problematic that children are less frequently mentioned by name than other people the subject is related to.

We emphasize that we do not rely on the internal details of the QA system, but merely require that it take a query string as input and return a scored list of answer strings as output. Treating the QA system in this black-box fashion means we can in principle replace it with any QA system with the same input–output signature.

Answer calibration (Section 2.6). A fruitful addition, which could also help mitigate the problem of SPOUSES vs. CHILDREN from the previous paragraph, could be to inject world knowledge into the answer calibration step; e.g., if we know from Freebase that Y is subject X's husband, we would want the calibration model to learn that Y is unlikely to also be X's child. One way to enable this would be to augment the feature vectors that serve as input to the calibration step by binary features indicating all known relations between the subject and the candidate object. Then, the logistic regression used for calibration could learn which relations are mutually exclusive in Freebase (e.g., it could learn a large negative weight for the SPOUSES feature of the model for the CHILDREN relation).

Head vs. tail entities. Our test subjects were carefully sampled in a stratified manner, such that we are covering entities at all levels of popularity (from our base set of the 100,000 most frequently searched-for people; cf. Section 3.1.1). We originally hypothesized that performance would be better for more popular subjects. However, we could not confirm this intuition in our experiments: when ordering subjects according to popularity rather than MRR on the x-axis in Fig. 4, no correlation between MRR and popularity could be discerned. This is important for the following reason.

Recall that, to allow for automated evaluation, we sampled a test set of subjects for which the ground-truth objects are known in Freebase. Of course, for the system to be truly useful, it must be run on subjects for which the object is presently unknown. However, we found that Freebase is less complete for unpopular than for popular entities. Thus, and since there are more unpopular than popular entities in Freebase, it is important that our method works well on the less popular entities, too.

This being said, there are fundamental limits to any method for automated knowledge base completion—including ours—, stemming from the fact that many true facts are hard, or even impossible, to find on the Web. For instance, Freebase lists ROSE MARIE COLIMORE as one of FRANK ZAPPA's parents, and our Web-search–based QA system successfully retrieves many documents that mention this fact. But what if we chose ROSE MARIE COLIMORE as the subject whose parents we want to find? Not only is the answer unknown in Freebase, there currently are not even any Freebase entities for Colimore's parents. The reason is that the vast majority of information on Colimore—whether in Freebase or on the Web in general—deals with her exclusively in her role as Zappa's mother and rarely discusses any other aspects of her life. As a consequence, it is very challenging even for humans—let alone for automated knowledge base completion methods—to answer the question who Colimore's parents were. This means that, beyond finding the answer, having it verified by humans is a difficult task, too.

6. CONCLUSIONS

This paper presents a method for filling gaps in a knowledge base. Our approach is different from a number of prior projects in that it follows a 'pull' model that attempts to find the missing objects for a given subject–relation pair *on demand,* rather than as one of many facts discovered during a full pass over a large corpus (which we call a 'push' model).

Our system uses a question-answering system (as a black box), which in turn takes advantage of mature Web-search technology (also as a black box) to retrieve relevant and up-to-date text passages to extract answer candidates from. We propose an end-to-end pipeline that lexicalizes subject–relation pairs to Web-search queries, chooses a good subset of queries, performs Web-search–based question answering, links candidate answer strings to Freebase entities, aggregates the results from all queries, and finally produces probabilistically scored rankings of answer entities.

We show empirically that choosing the right queries—without choosing too many—is crucial, especially for relations with objects from 'open' types with many instances (such as PERSON). Finally, we demonstrate that, for several relations, our system makes a large number of high-confidence predictions; e.g., we predict a nationality with a confidence above 90% for one in three test subjects.

7. REFERENCES

[1] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data (SIGMOD)*, 2008.

[2] L. Byrne and J. Dunnion. UCD IIRG at TAC 2010 KBP slot filling task. In *Proceedings of the 3rd Text Analysis Conference (TAC)*, 2010.

[3] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. Hruschka, and T. Mitchell. Toward an architecture for never-ending language learning. In *Proceedings of the 24th Conference on Artificial Intelligence (AAAI)*, 2010.

[4] K. Collins-Thompson, J. Callan, E. L. Terra, and C. L. A. Clarke. The effect of document retrieval quality on factoid question answering performance. In *Proceedings of the 27th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 2004.

[5] X. L. Dong, K. Murphy, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, T. Strohmann, S. Sun, and W. Zhang. Knowledge Vault: A Web-scale approach to probabilistic knowledge fusion. In submission, 2014.

[6] M. Dredze, P. McNamee, D. Rao, A. Gerber, and T. Finin. Entity disambiguation for knowledge base population. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING)*, 2010.

[7] X. Han, L. Sun, and J. Zhao. Collective entity linking in Web text: A graph-based method. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 2011.

[8] H. Ji and R. Grishman. Knowledge base population: Successful approaches and challenges. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2011.

[9] P. Kanani and A. McCallum. Selecting actions for resource-bounded information extraction using reinforcement learning. In *Proceedings of the 5th ACM International Conference on Web Search and Data Mining (WSDM)*, 2012.

[10] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[11] J. Mayfield, J. Artiles, and H. T. Dang. Overview of the TAC 2012 knowledge base population track. In *Proceedings of the 5th Text Analysis Conference (TAC)*, 2012.

[12] B. Min, R. Grishman, L. Wan, C. Wang, and D. Gondek. Distant supervision for relation extraction with an incomplete knowledge base. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2013.

[13] M. Mintz, S. Bills, R. Snow, and D. Jurafsky. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP (ACL–IJCNLP)*, 2009.

[14] M. Paşca. *Open Domain Question Answering from Large Text Collections*. CSLI Publications, 2003.

[15] S. Pedro and E. Hruschka. Conversing learning: Active learning and active social interaction for human supervision in never-ending learning systems. In *Advances in Artificial Intelligence–IBERAMIA*. 2012.

[16] J. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In A. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*. MIT Press, 1999.

[17] M. Samadi, M. Veloso, and M. Blum. OpenEval: Web information query evaluation. In *Proceedings of the 27th Conference on Artificial Intelligence (AAAI)*, 2013.

[18] F. Suchanek, G. Kasneci, and G. Weikum. YAGO: A core of semantic knowledge. In *Proceedings of the 16th International World Wide Web Conference (WWW)*, 2007.

[19] Y. Takaku, N. Kaji, N. Yoshinaga, and M. Toyoda. Identifying constant and unique relations by using time-series text. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP–CoNLL)*, 2012.

[20] E. Voorhees and D. Tice. The TREC-8 question answering track report. In *Proceedings of the 8th Text Retrieval Conference (TREC)*, 1999.

[21] G. Weikum and M. Theobald. From information to knowledge: Harvesting entities and relationships from Web sources. In *Proceedings of the 29th ACM SIGMOD–SIGACT–SIGART Symposium on Principles of Database Systems (PODS)*, 2010.