Knowledge Vault: A Web-Scale Approach to Probabilistic Knowledge Fusion

Xin Luna Dong[*], Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao,
Kevin Murphy[†], Thomas Strohmann, Shaohua Sun, Wei Zhang
Google, 1600 Amphitheatre Parkway, Mountain View, CA 94043
{lunadong|gabr|geremy|wilko|nlao|kpmurphy|tstrohmann|sunsh|weizh}@google.com

ABSTRACT

Recent years have witnessed a proliferation of large-scale knowledge bases, including Wikipedia, Freebase, YAGO, Microsoft's Satori, and Google's Knowledge Graph. To increase the scale even further, we need to explore automatic methods for constructing knowledge bases. Previous approaches have primarily focused on text-based extraction, which can be very noisy. Here we introduce Knowledge Vault, a Web-scale probabilistic knowledge base that combines extractions from Web content (obtained via analysis of text, tabular data, page structure, and human annotations) with prior knowledge derived from existing knowledge repositories. We employ supervised machine learning methods for fusing these distinct information sources. The Knowledge Vault is substantially bigger than any previously published structured knowledge repository, and features a probabilistic inference system that computes calibrated probabilities of fact correctness. We report the results of multiple studies that explore the relative utility of the different information sources and extraction methods.

Keywords

Knowledge bases; information extraction; probabilistic models; machine learning

1. INTRODUCTION

"The acquisition of knowledge is always of use to the intellect, because it may thus drive out useless things and retain the good. For nothing can be loved or hated unless it is first known."

– Leonardo da Vinci.

[*]Authors are listed alphabetically

[†]Corresponding author

In recent years, several large-scale knowledge bases (KBs) have been constructed, including academic projects such as YAGO [39], NELL [8], DBpedia [3], and Elementary/ DeepDive [32], as well as commercial projects, such as those by Microsoft[1], Google[2], Facebook[3], Walmart [9], and others. (See Section 7 for a detailed discussion of related work.) These knowledge repositories store millions of facts about the world, such as information about people, places and things (generically referred to as entities).

Despite their seemingly large size, these repositories are still far from complete. For example, consider Freebase, the largest open-source knowledge base [4]. 71% of people in Freebase have no known place of birth, and 75% have no known nationality[4]. Furthermore, coverage for less common relations/predicates can be even lower.

Previous approaches for building knowledge bases primarily relied on direct contributions from human volunteers as well as integration of existing repositories of structured knowledge (e.g., Wikipedia infoboxes). However, these methods are more likely to yield head content, namely, frequently mentioned properties of frequently mentioned entities. Suh et al. [41] also observed that Wikipedia growth has essentially plateaued, hence unsolicited contributions from human volunteers may yield a limited amount of knowledge going forward. Therefore, we believe a new approach is necessary to further scale up knowledge base construction. Such an approach should automatically extract facts from the whole Web, to augment the knowledge we collect from human input and structured data sources. Unfortunately, standard methods for this task (cf. [44]) often produce very noisy, unreliable facts. To alleviate the amount of noise in the automatically extracted data, the new approach should automatically leverage already-cataloged knowledge to build prior models of fact correctness.

In this paper, we propose a new way of automatically constructing a Web-scale probabilistic knowledge base, which we call the *Knowledge Vault*, or KV for short. Like many other knowledge bases, KV stores information in the form

[1]`http://www.bing.com/blogs/site_blogs/b/search/archive/2013/03/21/satorii.aspx`

[2]`http://www.google.com/insidesearch/features/search/knowledge.html`

[3]`http://www.insidefacebook.com/2013/01/14/facebook-builds-knowledge-graph-with-info-modules-on-community-pages/`

[4]Numbers current as of October 2013, cf. [28]. Freebase data is publicly available at `https://developers.google.com/freebase/data`.

of RDF triples (subject, predicate, object).An example is `</m/02mjmr, /people/person/place_of_birth /m/02hrh0_>`, where `/m/02mjmr` is the Freebase id for Barack Obama, and `/m/02hrh0_` is the id for Honolulu. Associated with each such triple is a confidence score, representing the probability that KV "believes" the triple is correct.

Entity types and predicates come from a fixed ontology, which is similar to that used in other systems, such as YAGO [39], NELL [8], DeepDive [32], and various systems participating in the TAC-KBP slot-filling competition [22]. Knowledge bases that use a fixed ontology should be contrasted with open information extraction (Open IE) approaches, such as Reverb [12], which work at the lexical level. Open IE systems usually have multiple redundant facts that are worded differently, such as `<Barack Obama, was born in, Honolulu>` and `<Obama, place of birth, Honolulu>`. In contrast, KV separates facts about the world from their lexical representation. This makes KV a structured repository of knowledge that is language independent.

The contributions of this paper are threefold. First, the Knowledge Vault is different from previous works on automatic knowledge base construction as it combines noisy extractions from the Web together with prior knowledge, which is derived from existing knowledge bases (in this paper, we use Freebase as our source of prior data). This approach is analogous to techniques used in speech recognition, which combine noisy acoustic signals with priors derived from a language model. KV's prior model can help overcome errors due to the extraction process, as well as errors in the sources themselves. For example, suppose an extractor returns a fact claiming that Barack Obama was born in Kenya, and suppose (for illustration purposes) that the true place of birth of Obama was not already known in Freebase. Our prior model can use related facts about Obama (such as his profession being US President) to infer that this new fact is unlikely to be true. The error could be due to mistaking Barack Obama for his father (entity resolution or co-reference resolution error), or it could be due to an erroneous statement on a spammy Web site (source error).

Second, KV is much bigger than other comparable KBs (see Table 1). In particular, KV has 1.6B triples, of which 324M have a confidence of 0.7 or higher, and 271M have a confidence of 0.9 or higher. This is about 38 times more than the largest previous comparable system (DeepDive [32]), which has 7M confident facts (Ce Zhang, personal communication). To create a knowledge base of such size, we extract facts from a large variety of sources of Web data, including free text, HTML DOM trees, HTML Web tables, and human annotations of Web pages. (Note that about 1/3 of the 271M confident triples were not previously in Freebase, so we are extracting new knowledge not contained in the prior.)

Third, we perform a detailed comparison of the quality and coverage of different extraction methods, as well as different prior methods. We also demonstrate the benefits of using multiple extraction sources and systems. Finally, we evaluate the validity of the closed world assumption, which is often used to automatically evaluate newly extracted facts given an existing knowledge base (see Section 6).

In the following sections, we describe the components of KV in more detail. We then study the performance of each part of the system in isolation and in combination, and show that fusion of multiple complementary systems and data sources considerably improves precision at a given recall level

2. OVERVIEW

KV contains three major components:

- **Extractors**: these systems extract triples from a huge number of Web sources. Each extractor assigns a confidence score to an extracted triple, representing uncertainty about the identity of the relation and its corresponding arguments.
- **Graph-based priors**: these systems learn the prior probability of each possible triple, based on triples stored in an existing KB.
- **Knowledge fusion**: this system computes the probability of a triple being true, based on agreement between different extractors and priors.

Abstractly, we can view the KV problem as follows: we are trying to construct a weighted labeled graph, which we can view as a very sparse $E \times P \times E$ 3d matrix G, where E is the number of entities, P is the number of predicates, and $G(s,p,o) = 1$ if there is a link of type p from s to o, and $G(s,p,o) = 0$ otherwise. We want to compute $\Pr(G(s,p,o) = 1|\cdot)$ for candidate (s,p,o) triples, where the probability is conditional on different sources of information. When using extractions, we condition on text features about the triple. When using graph-based priors, we condition on known edges in the Freebase graph (obviously we exclude the edge we are trying to predict!). Finally, in knowledge fusion, we condition on both text extractions and prior edges.

We describe each of three components in more detail in the following sections. Before that, we discuss our training and test procedure, which is common to all three approaches.

2.1 Evaluation protocol

Using the methods to be described in Section 3, we extract about 1.6B candidate triples, covering 4469 different types of relations and 1100 different types of entities. About 271M of these facts have an estimated probability of being true above 90%; we call these "confident facts". The resulting KB is much larger than other automatically constructed KBs, as summarized in Table 1.

To evaluate the quality of our methods, we randomly split this data into a training set (80% of the data) and a test set (20% of the data); we infer labels for these triples using the method described below. To ensure that certain common predicates (e.g., relating to geographical containment) did not dominate the performance measures, we took at most 10k instances of each predicate when creating the test set. We then pooled the samples from each predicate to get a more balanced test set.

If the test set contains the triple (s,p,o), then the training set is guaranteed not to contain the same triple. However, it may contain (s,p,o') or (s,p',o) or (s',p,o). For example, suppose s is Barack Obama, p is father-of, and o is Sasha Obama. Then the training set may contain the fact that Barack is the father of Malia Obama, or that Barack lives in the same place as Sasha Obama, etc. In graph terminology, we are leaving out edges at random from the training set, and asking how well we can predict their presence or absence.

Name	# Entity types	# Entity instances	# Relation types	# Confident facts (relation instances)
Knowledge Vault (KV)	1100	45M	4469	271M
DeepDive [32]	4	2.7M	34	7M[a]
NELL [8]	271	5.19M	306	0.435M[b]
PROSPERA [30]	11	N/A	14	0.1M
YAGO2 [19]	350,000	9.8M	100	4M[c]
Freebase [4]	1,500	40M	35,000	637M[d]
Knowledge Graph (KG)	1,500	570M	35,000	18,000M[e]

Table 1: Comparison of knowledge bases. KV, DeepDive, NELL, and PROSPERA rely solely on extraction, Freebase and KG rely on human curation and structured sources, and YAGO2 uses both strategies. Confident facts means with a probability of being true at or above 0.9.

[a]Ce Zhang (U Wisconsin), private communication
[b]Bryan Kiesel (CMU), private communication
[c]Core facts, `http://www.mpi-inf.mpg.de/yago-naga/yago/downloads.html`
[d]This is the number of non-redundant base triples, excluding reverse predicates and "lazy" triples derived from flattening CVTs (complex value types).
[e]`http://insidesearch.blogspot.com/2012/12/get-smarter-answers-from-knowledge_4.html`

An alternative approach to constructing the test set would have been to leave out all edges emanating from a particular node. However, in such a case, the graph-based models would have no signal to leverage. For example, suppose we omitted all facts about Barack Obama, and asked the system to predict where he lives, and who his children are. This would be possible given text extractions, but impossible given just a prior graph of facts. A compromise would be to omit all edges of a given type; for example, we could omit connections to all his children, but leave in other relations. However, we think the random sampling scenario more accurately reflects our actual use-case, which consists of growing an existing KB, where arbitrary facts may be missing.

2.2 Local closed world assumption (LCWA)

All the components of our system use supervised machine learning methods to fit probabilistic binary classifiers, which can compute the probability of a triple being true. We give the details on how these classifiers are constructed in the following sections. Here, we describe how we determine the labels (we use the same procedure for the training and test set).

For (s, p, o) triples that are in Freebase, we assume the label is true. For triples that do not occur in Freebase, we could assume the label is false (corresponding to a closed world assumption), but this would be rather dangerous, since we know that Freebase is very incomplete. So instead, we make use a somewhat more refined heuristic that we call the *local closed world assumption.*

To explain this heuristic, let us define $O(s, p)$ as the set of existing object values for a given s and p. This set will be a singleton for functional (single-valued) predicates such as place of birth, but can have multiple values for general relations, such as children; of course, the set can also be empty. Now, given a candidate triple (s, p, o), we assign its label as follows: if $(s, p, o) \in O(s, p)$, we say the triple is correct; if $(s, p, o) \notin O(s, p)$, but $|O(s, p)| > 0$, we say the triple is incorrect (because we assume the KB is *locally complete* for this subject-predicate pair); if $O(s, p)$ is empty, we do not label the triple, and we throw it out of our training / test set.

This heuristic is also used in previous works such as [15]. We empirically evaluate its adequacy in Section 6, by comparing to human-labeled data. There are more sophisticated methods for training models that don't make this assumption (such as [28, 36]), but we leave the integration of such methods into KV to future work.

3. FACT EXTRACTION FROM THE WEB

In this section, we summarize the extractors that we use to build KV, and then we evaluate their relative performance.

3.1 Extraction methods

3.1.1 Text documents (TXT)

We use relatively standard methods for relation extraction from text (see [16] for a recent overview), but we do so at a much larger scale than previous systems.

We first run a suite of standard NLP tools over each document. These perform named entity recognition, part of speech tagging, dependency parsing, co-reference resolution (within each document), and entity linkage (which maps mentions of proper nouns and their co-references to the corresponding entities in the KB). The in-house named entity linkage system we use is similar to the methods described in [18].

Next, we train relation extractors using distant supervision [29]. Specifically, for each predicate of interest, we extract a seed set of entity pairs that have this predicate, from an existing KB. For example, if the predicate is `married_to`, the pairs could be `(BarackObama, MichelleObama)` and `(BillClinton, HillaryClinton)`. We then find examples of sentences in which this pair is mentioned, and extract features/ patterns (either from the surface text or the dependency parse) from all of these sentences. The features that we use are similar to those described in [29].

In a bootstrapping phase, we look for more examples of sentences with these patterns occurring between pairs of entities of the correct type. We use the local closed world assumption to derive labels for the resulting set of extractions. Once we have a labeled training set, we fit a binary classifier (we use logistic regression) for each predicate independently in parallel, using a MapReduce framework. We have trained extractors for 4469 predicates, which is much more than previous machine reading systems.

3.1.2 HTML trees (DOM)

A somewhat different way to extract information from Web pages is to parse their DOM trees. These can either come from text pages, or from "deep web" sources, where data are stored in underlying databases and queried by filling HTML forms; these sources together generate more than 1B pages of data in DOM tree format [7]. To extract triples from DOM trees, we train classifiers as in the text case, except that we derive features connecting two entities from the DOM trees instead of from the text. Specifically, we use the lexicalized path (along the tree) between the two entities as a feature vector. The score of the extracted triples is the output of the classifier.

3.1.3 HTML tables (TBL)

There are over 570M tables on the Web that contain relational information (as opposed to just being used for visual formatting) [6]. Unfortunately, fact extraction techniques developed for text and trees do not work very well for tables, because the relation between two entities is usually contained in the column header, rather than being close by in the text/ tree. Instead, we use the following heuristic technique. First, we perform named entity linkage, as in the text case. Then we attempt to identify the relation that is expressed in each column of the table by looking at the entities in each column, and reasoning about which predicate each column could correspond to, by matching to Freebase, as in standard schema matching methods [42]. Ambiguous columns are discarded. The score of the extracted triple reflects the confidence returned by the named entity linkage system.

3.1.4 Human Annotated pages (ANO)

There are a large number of webpages where the webmaster has added manual annotations following ontologies from *schema.org*, *microformats.org*, *openGraphProtocol.org*, etc. In this paper, we use *schema.org* annotations. Many of these annotations are related to events or products, etc. Such information is not currently stored in the knowledge vault. So instead, in this paper we focus on a small subset of 14 different predicates, mostly related to people. We define a manual mapping from *schema.org* to the Freebase schema for these different predicates. The score of the extracted triple reflects the confidence returned by the named entity linkage system (the same one we use for TXT triples).

3.2 Fusing the extractors

We have described 4 different fact extraction methods. A simple way to combine these signals is to construct a feature vector $\vec{f}(t)$ for each extracted triple $t = (s, p, o)$, and then to apply a binary classifier to compute $\Pr(t = 1|\vec{f}(t))$. For simplicity and speed, we fit a separate classifier for each predicate.

The feature vector is composed of two numbers for each extractor: the square root[5] of the number of sources that the extractor extracted this triple from, and the mean score of the extractions from this extractor, averaging over sources (or 0 if the system did not produce this triple).

The classifier learns a different weight for each component of this feature vector, and hence can learn the relative reliabilities of each system. In addition, since we fit a separate classifier per predicate, we can model their different reliabilities, too.

The labels for training the fusion system come from applying the local closed world assumption to the training set. Since this is a very low-dimensional classification problem, we initially used a linear logistic regression model. However, we observed considerably better performance by using boosted decision stumps [35]. This kind of classifier can learn to quantize the features into bins, and thus learn a non-linear decision boundary.

[5]The motivation for using $\sqrt{n}$, where n is the number of sources, is to reduce the effect of very commonly expressed facts (such as the birth place of Barack Obama). Results are similar if we use $\log(1 + n)$. Note that we perform de-duplication of sources before running the extraction pipelines.

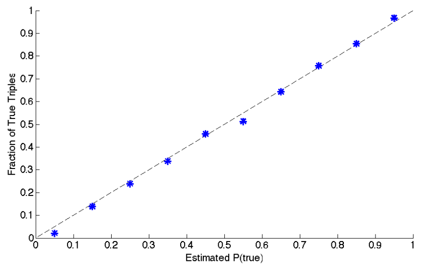

Figure 1: True probability vs estimated probability for each triple in KV.

3.3 Calibration of the probability estimates

The confidence scores from each extractor (and/or the fused system) are not necessarily on the same scale, and cannot necessarily be interpreted as probabilities. To alleviate this problem, we adopt the standard technique known as Platt Scaling (named after [33]), which consists of fitting a logistic regression model to the scores, using a separate validation set. Figure 1 shows that our (fused) probability estimates are well-calibrated, in the following sense: if we collect all the triples that have a predicted probability of 0.9, then we find that about 90% of them are indeed true. Each individual extractor is also calibrated (results not shown).

3.4 Comparison of the methods

Using the four extractors described earlier applied to a very large web corpus, we extract about 1.6B triples. Table 2 shows the number of triples from each system. We see that the DOM system extracts the largest number of triples overall (about 1.2B), of which about 94M (or 8%) are high confidence (with a probability of being true at or above 0.9; see the penultimate column of Table 2). The TBL system extracts the least number of triples overall (about 9.4M). One reason for this is that very few columns in webtables (only 18% according to [17]) map to a corresponding Freebase predicate. The ANO and TXT systems both produce hundreds of millions of triples.

In addition to measuring the number of triples at different confidence levels, it is interesting to consider the area under the ROC curve (AUC score). This score is equal to the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one. We computed the AUC scores for the different extraction meth-

System	#	# > 0.7	# > 0.9	Frac. >0.9	AUC
TBL	9.4M	3.8M	0.59M	0.06	0.856
ANO	140M	2.4M	0.25M	0.002	0.920
TXT	330M	20M	7.1M	0.02	0.867
DOM	1200M	150M	94M	0.08	0.928
FUSED-EX.	1600M	160M	100M	0.06	0.927

Table 2: Performance of different extraction systems.

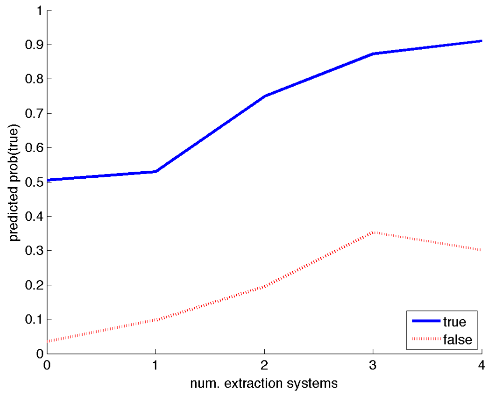

Figure 2: Predicted probability of each triple vs. the number of systems that predicted it. Solid blue line: correct (true) triples. Dotted red line: incorrect (false) triples.

ods on different test sets, namely, only using the extractions produced by each system (obviously, the test sets were distinct from the training sets). The test set for computing the AUC score for the fused extractors was the union of all the test sets of the individual systems.

We see that the DOM system has the highest AUC score, so although it produces a large number of low confidence triples, the system "knows" that these are likely to be false. The table also illustrates the benefits of fusing multiple extractors: we get about 7% more high confidence triples, while maintaining a high AUC score (see the last row of the table). Not surprisingly, however, the performance of the fusion system is dominated by that of the DOM system. In Section 5, we shall show much greater gains from fusion, when we combine graph priors with extractors.

3.5 The beneficial effects of adding more evidence

Figure 2 shows how the overall predicted probability of each triple changes as more systems extract it. When no systems extract a given triple, we rely on our prior model (described in Section 4); averaging over all the triples, we see that the prior probability for the true triples is about 0.5, whereas the prior probability for the false triples is close to 0. As we accumulate more evidence in favor of the triple, our belief in its correctness increases to near 1.0 for the true triples; for the false triples, our belief also increases, although it stays well below 0.5.

Figure 3 shows how the probability of a triple increases with the number of sources where the triple is seen. Again, our final belief in true triples is much higher than in false

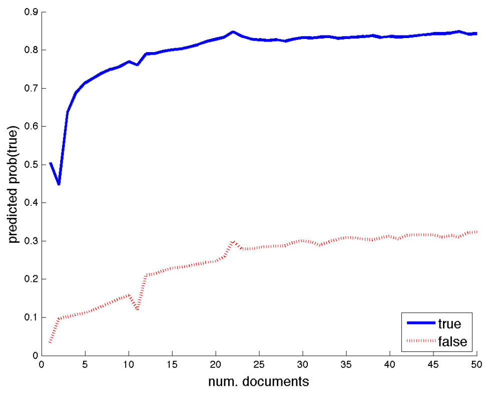

Figure 3: Predicted probability of each triple vs. the number of unique web sources that contain this triple (axis truncated at 50 for clarity).

triples. To prevent over-counting of evidence, we only count each triple once per domain, as opposed to once per URL; for example, if we extract a triple asserting that Barack Obama was born in Kenya from `myblogger.com/page1` and `myblogger.com/page2`, we only count this once.

4. GRAPH-BASED PRIORS

As mentioned in the introduction, facts extracted from the Web can be unreliable. A good way to combat this is to use prior knowledge, derived from other kinds of data. In this paper, we exploit existing triples in Freebase to fit prior models, which can assign a probability to any possible triple, even if there is no corresponding evidence for this fact on the Web (cf. [2]). This can be thought of as link prediction in a graph. That is, we observe a set of existing edges (representing predicates that connect different entities), and we want to predict which other edges are likely to exist. We have tried two different approaches to solving this problem, which we describe below.

4.1 Path ranking algorithm (PRA)

One way to perform link prediction is to use the path ranking algorithm of [24]. Similar to distant supervision, we start with a set of pairs of entities that are connected by some predicate p. PRA then performs a random walk on the graph, starting at all the subject (source) nodes. Paths that reach the object (target) nodes are considered successful. For example, the algorithm learns that pairs (X, Y) which are connected by a marriedTo edge often also have a path of the form $X \xrightarrow{\text{parentOf}} Z \xleftarrow{\text{parentOf}} Y$, since if two people share a common child, they are likely to be married. The quality of these paths can be measured in terms of their support and precision, as in association rule mining (cf., [15]).

The paths that PRA learns can be interpreted as rules. For example, consider the task of predicting where someone went to college. The algorithm discovers several useful rules, shown in Table 3. In English, the first rule says: a person X is likely to have attended school S if X was drafted from sports team T, and T is from school S. The second rule says: a person is likely to attend the same school as their sibling.

F1	P	R	W	Path
0.03	1	0.01	2.62	/sports/drafted-athlete/drafted,/sports/sports-league-draft-pick/school
0.05	0.55	0.02	1.88	/people/person/sibling-s, /people/sibling-relationship/sibling, /people/person/education, /education/education/institution
0.06	0.41	0.02	1.87	/people/person/spouse-s, /people/marriage/spouse, /people/person/education, /education/education/institution
0.04	0.29	0.02	1.37	/people/person/parents, /people/person/education, /education/education/institution
0.05	0.21	0.02	1.85	/people/person/children, /people/person/education, /education/education/institution
0.13	0.1	0.38	6.4	/people/person/place-of-birth, /location/location/people-born-here, /people/person/education, /education/education/institution
0.05	0.04	0.34	1.74	/type/object/type, /type/type/instance, /people/person/education, /education/education/institution
0.04	0.03	0.33	2.19	/people/person/profession, /people/profession/people-with-this-profession, /people/person/education, /education/education/institution

Table 3: Some of the paths learned by PRA for predicting where someone went to college. Rules are sorted by decreasing precision. Column headers: F1 is the harmonic mean of precision and recall, P is the precision, R is the recall, W is the weight given to this feature by logistic regression.

Since multiple rules or paths might apply for any given pair of entities, we can combine them by fitting a binary classifier (we use logistic regression). In PRA, the features are the probabilities of reaching O from S following different types of paths, and the labels are derived using the local closed world assumption. We can fit a classifier for each predicate independently in parallel. We have trained prior predictors for 4469 predicates using Freebase as training data. At test time, given a new (s, p, o) triple, we look up all the paths for predicate p chosen by the learned model, and perform a walk (on the training graph) from s to o via each such path; this gives us a feature value that can be plugged in to the classifier.

The overall AUC is 0.884, which is less than that of the fused extractor system (0.927), but is still surprisingly high.

4.2 Neural network model (MLP)

An alternative approach to building the prior model is to view the link prediction problem as matrix (or rather, tensor) completion. In particular, the original KB can be viewed as a very sparse $E \times P \times E$ 3d matrix G, where E is the number of entities, P is the number of predicates, and $G(s, p, o) = 1$ if there is a link of type p from s to o, and $G(s, p, o) = 0$ otherwise. We can perform a low-rank decomposition of this tensor by associating a latent low dimensional vector to each entity and predicate, and then computing the elementwise inner product:

$$\Pr(G(s,p,o) = 1) = \sigma\left(\sum_{k=1}^{K} u_{sk} w_{pk} v_{ok}\right) \tag{1}$$

where $\sigma(x) = 1/(1+e^{-x})$ is the sigmoid or logistic function, and $K \sim 60$ is the number of hidden dimensions. Here $\vec{u}_s$, $\vec{w}_p$ and $\vec{v}_o$ are K-dimensional vectors, which embed the discrete tokens into a low dimensional "semantic" space. If we ignore the sigmoid transform (needed to produce binary responses), this is equivalent to the PARAFAC method of tensor decomposition [14, 5, 11].

A more powerful model was recently proposed in [37]; this associates a different *tensor* with each relation, and hence has the form

$$\Pr(G(s,p,o) = 1) = \sigma\left(\vec{\beta}_p^T f\left[\vec{u}_s^T \vec{W}_p^{1:M} \vec{v}_o\right]\right) \tag{2}$$

where $f()$ is a nonlinear function such as tanh, $\vec{\beta}_p$ is a $K \times 1$ vector, and $\vec{W}_p^m$ is a $K \times K$ matrix. Unfortunately, this model requires $O(KE + K^2MP)$ parameters, where M is the number of "layers" in the tensor W.

In this paper, we considered a simpler approach where we associate one *vector* per predicate, as in Equation 1, but then use a standard multi layer perceptron (MLP) to capture interaction terms. More precisely, our model has the form

$$\Pr(G(s,p,o) = 1) = \sigma\left(\vec{\beta}^T f\left[\vec{A}\left[\vec{u}_s, \vec{w}_p, \vec{v}_o\right]\right]\right) \tag{3}$$

where $\vec{A}$ is a $L \times (3K)$ matrix (where the $3K$ term arises from the K-dimensional $\vec{u}_s$ and $\vec{w}_p$ and $\vec{v}_o$) representing the first layer weights (after the embeddings), and $\vec{\beta}$ is a $L \times 1$ vector representing the second layer weights. (We set $L = K = 60$.) This has only $O(L + LK + KE + KP)$ parameters, but achieves essentially the same performance as the one in Equation 2 on their dataset.[6]

Having established that our MLP model is comparable to the state of the art, we applied it to the KV data set. Surprisingly, we find that the neural model has about the same performance as PRA when evaluated using ROC curves (the AUC for the MLP model is 0.882, and for PRA is 0.884).

To illustrate that the neural network model learns a meaningful "semantic" representation of the entities and predicates, we can compute the nearest neighbors of various items in the a K-dimensional space. It is known from previous work (e.g., [27]) that related entities cluster together in the space, so here we focus on predicates. The results are shown in Table 4. We see that the model learns to put semantically related (but not necessarily similar) predicates near each other. For example, we see that the closest predicates (in the $\vec{w}$ embedding space) to the 'children' predicate are 'parents', 'spouse' and 'birth-place'.

4.3 Fusing the priors

We can combine the different priors together using the fusion method described in Section 3.2. The only difference is the features that we use, since we no longer have any extractions. Insead, the feature vector contains the vector of confidence values from each prior system, plus indicator values specifying if the prior was able to predict or not. (This lets us distinguish a missing prediction from a prediction score of 0.0.) We train a boosted classifier using these signals, and calibrate it with Platt Scaling, as before. Fusing the two prior methods helps performance, since they have complementary strengths and weaknesses (different inductive biases): the AUC of the fused system is 0.911.

[6]More precisely, [37] reported an 88.9% accuracy on the subset of Freebase data they have worked with (75,043 entities, 13 relations) when they replaced entities such as `BarackObama` by their constituting words `Barack` and `Obama`. Applying the same technique of replacing entities with consituting words, our simpler model got an accuracy of 89.1%.

Predicate	Neighbor 1	Neighbor 2	Neighbor 3
children	0.4 parents	0.5 spouse	0.8 birth-place
birth-date	1.24 children	1.25 gender	1.29 parents
edu-end	1.41 job-start	1.61 edu-end	1.74 job-end

Table 4: Nearest neighbors for some predicates in the 60d embedding space learned by the neural network. Numbers represent squared Euclidean distance. Edu-start and edu-end represent the start and end dates of someone attending a school or college. Similarly, job-start and job-end represent the start and end dates of someone holding a particular job.

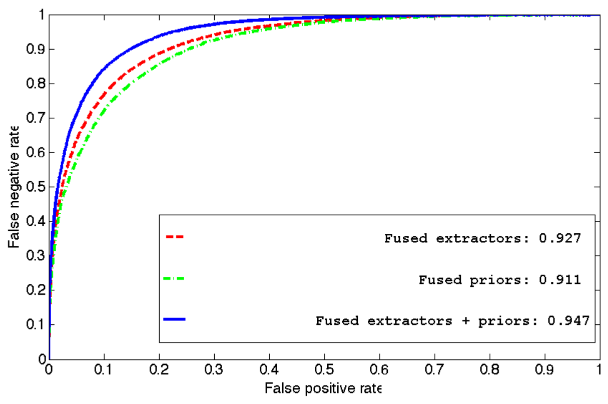

Figure 4: ROC curves for the fused extractor, fused prior, and fused prior + extractor. The numbers in the legend are the AUC scores.

5. FUSING EXTRACTORS AND PRIORS

We have described several different fact extraction methods, and several different priors. We can combine all these systems together using the fusion method described in Section 3.2. Figure 4 shows the benefits of fusion quantitatively. We see that combining prior and extractor together results in a significant boost in performance.

To more clearly illustrate the effect of adding the prior, Figure 5 plots the number of triples in each confidence bin for the (fused) extractor, the (fused) prior, and the overall system. We see that compared with considering only extractions, combining priors and extractors increases the number of high confidence facts (those with a probability greater than 0.9) from about 100M to about 271M. Of these, about 33% are new facts that were not yet in Freebase.

Figure 5 illustrates another interesting effect: when we combine prior and extractor, the number of triples about which we are uncertain (*i.e.*, the predicted probability falling in the range of [.3, .7]) has gone down; some of these triples we now believe to be true (as we discussed previously), but many we now believe to be false. This is a visual illustration that the prior can reduce the false positive rate.

We now give a qualitative example of the benefits of combining the prior with the extractor. The extraction pipeline extracted the following triple:[7]

[7]Here the predicate is a conjunction of two primitive predicates, `/people/person/education` and `/education/education/institution`, obtained by passing through a complex value type (CVT) node, aka an anonymous or "blank" node, representing the temporal event that Barry attended Madison.

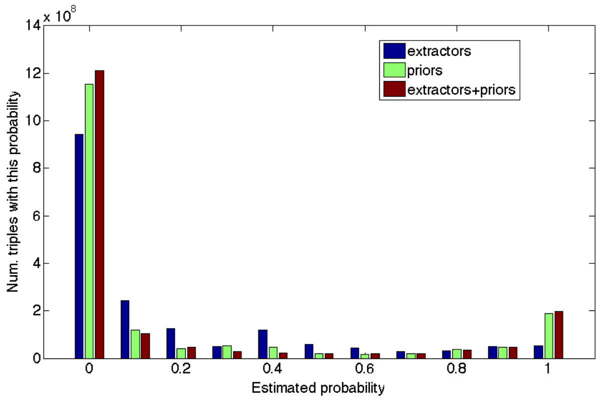

Figure 5: Number of triples in KV in each confidence bin.

```
<Barry Richter  (/m/02ql38b),
/people/person/edu./edu/edu/institution,
Universty of Wisconsin-Madison (/m/01yx1b)>
```

The (fused) extraction confidence for this triple was just 0.14, since it was based on the following two rather indirect statements:[8]

```
In the fall of 1989, Richter accepted a scholarship
to the University of Wisconsin, where he played for
four years and earned numerous individual accolades...
```

```
The Polar Caps' cause has been helped by the impact of
knowledgable coaches such as Andringa, Byce and former
UW teammates Chris Tancill and Barry Richter.
```

However, we know from Freebase that Barry Richter was born and raised in Madison, WI. This increases our prior belief that he went to school there, resulting in a final fused belief of 0.61.

6. EVALUATING LCWA

So far, we have been relying on the local closed world assumption (LCWA) to train and test our system. However, we know that this is just an approximation to the truth. For example, Freebase often lists the top 5 or so actors for any given movie, but it is unreasonable to assume that this list is complete (since most movies have a cast of 10–20 actors); this can result in false negatives (if our system predicts the name of an actor that is not on the list). Conversely (but less frequently), Freebase can contain errors, which can result in false positives.

To assess the severity of this problem, we manually labeled a subset of our balanced test set, using an in-house team of raters. This subset consisted of 1000 triples for 10

[8]Sources: `http://www.legendsofhockey.net/LegendsOfHockey/jsp/SearchPlayer.jsp?player=11377` and `http://host.madison.com/sports/high-school/hockey/numbers-dwindling-for-once-mighty-madison\-high-school\-hockey-programs/article_95843e00-ec34-11df-9da9-001cc4c002e0.html`

Labels	Prior	Extractor	Prior+ex
LCWA	0.943	0.872	0.959
Human	0.843	0.852	0.869

Table 5: AUC scores for the fused prior, extractor and prior+extractor using different labels on the 10k test set.

different predicates. We asked each rater to evaluate each such triple, and to determine (based on their own research, which can include web searches, looking at wikipedia, etc) whether each triple is true or false or unknown; we discarded the 305 triples with unknown labels.

We then computed the performance of our systems on this test set, using both LCWA labels and the human labels. In both cases, the system was trained on our full training set (i.e., 80% of 1.6B) using LCWA labels. The results are shown in Table 5. We see that the performance on the human labeled data is lower, although not by that much, indirectly justifying our use of the LCWA.

7. RELATED WORK

There is a growing body of work on automatic knowledge base construction [44, 1]. This literature can be clustered into 4 main groups: (1) approaches such as YAGO [39], YAGO2 [19], DBpedia [3], and Freebase [4], which are built on Wikipedia infoboxes and other structured data sources; (2) approaches such as Reverb [12], OLLIE [26], and PRISMATIC [13], which use open information (schema-less) extraction techniques applied to the entire web; (3) approaches such as NELL/ ReadTheWeb [8], PROSPERA [30], and DeepDive/ Elementary [32], which extract information from the entire web, but use a fixed ontology/ schema; and (4) approaches such as Probase [47], which construct taxonomies (is-a hierarchies), as opposed to general KBs with multiple types of predicates.

The knowledge vault is most similar to methods of the third kind, which extract facts, in the form of disambiguated triples, from the entire web. The main difference from this prior work is that we fuse together facts extracted from text with prior knowledge derived from the Freebase graph.

There is also a large body of work on link prediction in graphs. This can be thought as creating a joint probability model over a large set of binary random variables, where $G(s, p, o) = 1$ if and only if there is a link of type p from s to o. The literature can be clustered into three main kinds of methods: (1) methods that directly model the correlation between the variables, using discrete Markov random fields (e.g., [23]) or continuous relaxations thereof (e.g., [34]); (2) methods that use latent variables to model the correlations indirectly, using either discrete factors (e.g., [48]) or continuous factors (e.g., [31, 11, 20, 37]); and (3) methods that approximate the correlation using algorithmic approaches, such as random walks [24].

In the knowledge vault, we currently employ graph priors of the second and third kind. In particular, our neural tensor model is a continuous latent variable model, which is similar to, but slightly different from, [37] (see Section 4.2 for a discussion). Our PRA model is similar to the method described in [24], except it is trained on Freebase instead of on NELL. In addition, it uses a more scalable implementation.

Another related literature is on the topic of probabilistic databases (see e.g., [40, 43]). KV is a probabilistic database, and it can support simple queries, such as `BarackObama BornIn ?`, which returns a distribution over places where KV thinks Obama was born. However, we do not yet support sophisticated queries, such as JOIN or SELECT.

Finally, there is a small set of papers on representing uncertainty in information extraction systems (see e.g., [45, 25]). KV also represents uncertainty in the facts it has extracted. Indeed, we show that its uncertainty estimates are well-calibrated. We also show how they change as a function of the amount of evidence (see Figure 2).

8. DISCUSSION

Although Knowledge Vault is a large repository of useful knowledge, there are still many ways in which it can be improved. We discuss some of these issues below.

Modeling mutual exclusion between facts. Currently (for reasons of scalability) we treat each fact as an independent binary random variable, that is either true or false. However, in reality, many triples are correlated. For example, for a functional relation such as born-in, we know there can only be one true value, so the (s, p, o_i) triples representing different values o_i for the same subject s and predicate p become correlated due to the mutual exclusion constraint. A simple way to handle this is to collect together all candidate values, and to force the distribution over them to sum to 1 (possibly allowing for some "extra" probability mass to account for the fact that the true value might not be amongst the extracted set of candidates). This is similar to the notion of an X-tuple in probabilistic databases [40]. Preliminary experiments of this kind did not work very well, since the different o_i often represent the same entity at different levels of granularity. For example, we might have a fact that Obama was born in Honolulu, and another one stating he was born in Hawaii. These are not mutually exclusive, so the naive approach does not work. We are currently investigating more sophisticated methods.

Modeling soft correlation between facts. For some kinds of relations, there will be soft constraints on their values. For example, we know that people usually have between 0 and 5 children; there is of course a long tail to this distribution, but it would still be surprising (and indicative of a potential error) if we extracted 100 different children for one person. Similarly, we expect the date of birth of a person to be about 15 to 50 years earlier than the date of birth of their child. Preliminary experiments using joint Gaussian models to represent correlations amongst numerical values show some promise, but we still need to fully integrate this kind of joint prior into KV.

Values can be represented at multiple levels of abstraction. We can represent the world at different levels of granularity. For example, we can say that Obama is born in Honolulu, or in Hawaii, or in the USA. When matching extracted facts with those stored in Freebase, we use prior geographical knowledge to reason about compatibility. For example, if we extract that Obama was born in Hawaii, and we already know he was born in Honolulu, we consider this a correct extraction. In the future, we would like to generalize this approach to other kinds of values. For example, if we extract that Obama's profession is politician, and we already know his profession is president, we should regard the extracted fact as true, since it is implied by what we already know.

Dealing with correlated sources. In Figure 3, we showed how our belief in a triple increased as we saw it extracted from more sources. This is of course problematic if we have duplicated or correlated sources. Currently we have a very simple solution to this, based on counting each domain only once. In the future, we plan to deploy more sophisticated copy detection mechanisms, such as those in [10].

Some facts are only temporarily true. In some cases, the "truth" about a fact can change. For example, Google's current CEO is Larry Page, but from 2001 to 2011 it was Eric Schmidt. Both facts are correct, but only during the specified time interval. For this reason, Freebase allows some facts to be annotated with beginning and end dates, by use of the CVT (compound value type) construct, which represents n-ary relations via auxiliary nodes. (An alternative approach is to reify the pairwise relations, and add extra assertions to them, as in the YAGO2 system [19].) In the future, we plan to extend KV to model such temporal facts. However, this is non-trivial, since the duration of a fact is not necessarily related to the timestamp of the corresponding source (cf. [21]).

Adding new entities and relations. In addition to missing facts, there are many entities that are mentioned on the Web but are not in Freebase, and hence not in KV either. In order to represent such information, we need to automatically create new entities (cf. [46]); this is work in progress. Furthermore, there are many relations that are mentioned on the Web but cannot be represented in the Freebase schema. To capture such facts, we need to extend the schema, but we need to do so in a controlled way, to avoid the problems faced by open IE systems, which have many redundant and synonymous relations. See [17] for one possible approach to this problem.

Knowledge representation issues. The RDF triple format seems adequate for representing factual assertions (assuming a suitably rich schema), but it might be less appropriate for other kinds of knowledge (e.g., representing the difference between running and jogging, or between jazz music and blues). There will always be a long tail of concepts that are difficult to capture in any fixed ontology. Our neural network is one possible way to provide semantically plausible generalizations, but extending it to represent richer forms of knowledge is left to future work.

Inherent upper bounds on the potential amount of knowledge that we can extract. The goal of KV is to become a large-scale repository of all of human knowledge. However, even if we had a perfect machine reading system, not all of human knowledge is available on the Web. In particular, common sense knowledge may be hard to acquire from text sources. However, we may be able to acquire such knowledge using crowdsourcing techniques (c.f., [38]).

9. CONCLUSIONS

In this paper we described how we built a Web-scale probabilistic knowledge base, which we call Knowledge Vault. In contrast to previous work, we fuse together multiple extraction sources with prior knowledge derived from an existing KB. The resulting knowledge base is about 38 times bigger than existing automatically constructed KBs. The facts in KV have associated probabilities, which we show are well-calibrated, so that we can distinguish what we know with high confidence from what we are uncertain about. In the future, we hope to continue to scale KV, to store more knowledge about the world, and to use this resource to help various downstream applications, such as question answering, entity-based search, etc.

Acknowledgments

We thank John Giannandrea, Fernando Pereira, and Amar Subramanya for constructive suggestions that helped us improve this paper, and Kevin Lerman, Abhijit Mahabal, and Oksana Yakhnenko for developing most of the extraction pipeline. We also thank Eric Altendorf and Alex Smola, who were instrumental in the early phases of the project.

10. REFERENCES

[1] AKBC-WEKEX. The Knowledge Extraction Workshop at NAACL-HLT, 2012.

[2] G. Angeli and C. Manning. Philosophers are mortal: Inferring the truth of unseen facts. In *CoNLL*, 2013.

[3] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. DBpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735, 2007.

[4] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*, pages 1247–1250. ACM, 2008.

[5] A. Bordes, X. Glorot, J. Weston, and Y. Bengio. Joint learning of words and meaning representations for open-text semantic parsing. In *AI/Statistics*, 2012.

[6] M. Cafarella, A. Halevy, Z. D. Wang, E. Wu, and Y. Zhang. WebTables: Exploring the Power of Tables on the Web. *VLDB*, 1(1):538–549, 2008.

[7] M. J. Cafarella, A. Y. Halevy, and J. Madhavan. Structured data on the web. *Commun. ACM*, 54(2):72–79, 2011.

[8] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. H. Jr., and T. Mitchell. Toward an architecture for never-ending language learning. In *AAAI*, 2010.

[9] O. Deshpande, D. Lambda, M. Tourn, S. Das, S. Subramaniam, A. Rajaraman, V. Harinarayan, and A. Doan. Building, maintaing and using knowledge bases: A report from the trenches. In *SIGMOD*, 2013.

[10] X. L. Dong, L. Berti-Equille, and D. Srivastatva. Integrating conflicting data: the role of source dependence. In *VLDB*, 2009.

[11] L. Drumond, S. Rendle, and L. Schmidt-Thieme. Predicting RDF Triples in Incomplete Knowledge Bases with Tensor Factorization. In *10th ACM Intl. Symp. on Applied Computing*, 2012.

[12] A. Fader, S. Soderland, and O. Etzioni. Identifying relations for open information extraction. In *EMNLP*, 2011.

[13] J. Fan, D. Ferrucci, D. Gondek, and A. Kalyanpur. Prismatic: Inducing knowledge from a large scale lexicalized relation resource. In *First Intl. Workshop on Formalisms and Methodology for Learning by Reading*, pages 122–127. Association for Computational Linguistics, 2010.

[14] T. Franz, A. Schultz, S. Sizov, and S. Staab. TripleRank: Ranking Semantic Web Data by Tensor

Decomposition. In *ISWC*, 2009.

[15] L. A. Galárraga, C. Teflioudi, K. Hose, and F. Suchanek. Amie: association rule mining under incomplete evidence in ontological knowledge bases. In *WWW*, pages 413–422, 2013.

[16] R. Grishman. Information extraction: Capabilities and challenges. Technical report, NYU Dept. CS, 2012.

[17] R. Gupta, A. Halevy, X. Wang, S. Whang, and F. Wu. Biperpedia: An Ontology for Search Applications. In *VLDB*, 2014.

[18] B. Hachey, W. Radford, J. Nothman, M. Honnibal, and J. Curran. Evaluating entity linking with wikipedia. *Artificial Intelligence*, 194:130–150, 2013.

[19] J. Hoffart, F. M. Suchanek, K. Berberich, and G. Weikum. YAGO2: A Spatially and Temporally Enhanced Knowledge Base from Wikipedia. *Artificial Intelligence Journal*, 2012.

[20] R. Jenatton, N. L. Roux, A. Bordes, and G. Obozinski. A latent factor model for highly multi-relational data. In *NIPS*, 2012.

[21] H. Ji, T. Cassidy, Q. Li, and S. Tamang. Tackling Representation, Annotation and Classification Challenges for Temporal Knowledge Base Population. *Knowledge and Information Systems*, pages 1–36, August 2013.

[22] H. Ji and R. Grishman. Knowledge base population: successful approaches and challenges. In *Proc. ACL*, 2011.

[23] S. Jiang, D. Lowd, and D. Dou. Learning to refine an automatically extracted knowledge base using markov logic. In *Intl. Conf. on Data Mining*, 2012.

[24] N. Lao, T. Mitchell, and W. Cohen. Random walk inference and learning in a large scale knowledge base. In *EMNLP*, 2011.

[25] X. Li and R. Grishman. Confidence estimation for knowledge base population. In *Recent Advances in NLP*, 2013.

[26] Mausam, M. Schmitz, R. Bart, S. Soderland, and O. Etzioni. Open language learning for information extraction. In *EMNLP*, 2012.

[27] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. In *ICLR*, 2013.

[28] B. Min, R. Grishman, L. Wan, C. Wang, and D. Gondek. Distant supervision for relation extraction with an incomplete knowledge base. In *NAACL*, 2013.

[29] M. Mintz, S. Bills, R. Snow, and D. Jurafksy. Distant supervision for relation extraction without labeled data. In *Prof. Conf. Recent Advances in NLP*, 2009.

[30] N. Nakashole, M. Theobald, and G. Weikum. Scalable knowledge harvesting with high precision and high recall. In *WSDM*, pages 227–236, 2011.

[31] M. Nickel, V. Tresp, and H.-P. Kriegel. Factorizing YAGO: scalable machine learning for linked data. In *WWW*, 2012.

[32] F. Niu, C. Zhang, and C. Re. Elementary: Large-scale Knowledge-base Construction via Machine Learning and Statistical Inference. *Intl. J. On Semantic Web and Information Systems*, 2012.

[33] J. Platt. Probabilities for SV machines. In A. Smola, P. Bartlett, B. Schoelkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*. MIT Press, 2000.

[34] J. Pujara, H. Miao, L. Getoor, and W. Cohen. Knowledge graph identification. In *International Semantic Web Conference (ISWC)*, 2013.

[35] L. Reyzin and R. Schapire. How boosting the margin can also boost classifier complexity. In *Intl. Conf. on Machine Learning*, 2006.

[36] A. Ritter, L. Zettlemoyer, Mausam, and O. Etzioni. Modeling missing data in distant supervision for information extraction. *Trans. Assoc. Comp. Linguistics*, 1, 2013.

[37] R. Socher, D. Chen, C. Manning, and A. Ng. Reasoning with Neural Tensor Networks for Knowledge Base Completion. In *NIPS*, 2013.

[38] R. Speer and C. Havasi. Representing general relational knowledge in conceptnet 5. In *Proc. of LREC Conference*, 2012.

[39] F. Suchanek, G. Kasneci, and G. Weikum. YAGO - A Core of Semantic Knowledge. In *WWW*, 2007.

[40] D. Suciu, D. Olteanu, C. Re, and C. Koch. *Probabilistic Databases*. Morgan & Claypool, 2011.

[41] B. Suh, G. Convertino, E. H. Chi, and P. Pirolli. The singularity is not near: slowing growth of wikipedia. In *Proceedings of the 5th International Symposium on Wikis and Open Collaboration*, WikiSym '09, pages 8:1–8:10, 2009.

[42] P. Venetis, A. Halevy, J. Madhavan, M. Pasca, W. Shen, F. Wu, G. Miao, and C. Wi. Recovering semantics of tables on the web. In *Proc. of the VLDB Endowment*, 2012.

[43] D. Z. Wang, E. Michelakis, M. Garofalakis, and J. Hellerstein. BayesStore: Managing Large, Uncertain Data Repositories with Probabilistic Graphical Models. In *VLDB*, 2008.

[44] G. Weikum and M. Theobald. From information to knowledge: harvesting entities and relationships from web sources. In *Proceedings of the twenty-ninth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 65–76. ACM, 2010.

[45] M. Wick, S. Singh, A. Kobren, and A. McCallum. Assessing confidence of knowledge base content with an experimental study in entity resolution. In *AKBC workshop*, 2013.

[46] M. Wick, S. Singh, H. Pandya, and A. McCallum. A Joint Model for Discovering and Linking Entities. In *AKBC Workshop*, 2013.

[47] W. Wu, H. Li, H. Wang, and K. Q. Zhu. Probase: A probabilistic taxonomy for text understanding. In *SIGMOD*, pages 481–492. ACM, 2012.

[48] Z. Xu, V. Tresp, K. Yu, and H.-P. Kriegel. Infinite hidden relational models. In *UAI*, 2006.