Artificial Intelligence 174 (2010) 726–748

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Analysis of a probabilistic model of redundancy in unsupervised information extraction

Doug Downey [a,*], Oren Etzioni [b], Stephen Soderland [b]

[a] Northwestern University, 2133 Sheridan Road, Evanston, IL 60208, United States
[b] University of Washington, Box 352350, Seattle, WA 98195, United States

ARTICLE INFO

Article history:
Received 7 July 2009
Received in revised form 14 April 2010
Accepted 26 April 2010
Available online 29 April 2010

Keywords:
Information extraction
Unsupervised
World Wide Web

ABSTRACT

Unsupervised Information Extraction (UIE) is the task of extracting knowledge from text without the use of hand-labeled training examples. Because UIE systems do not require human intervention, they can recursively discover new relations, attributes, and instances in a scalable manner. When applied to massive corpora such as the Web, UIE systems present an approach to a primary challenge in artificial intelligence: the automatic accumulation of massive bodies of knowledge.

A fundamental problem for a UIE system is assessing the probability that its extracted information is correct. In massive corpora such as the Web, the same extraction is found repeatedly in different documents. How does this redundancy impact the probability of correctness?

We present a combinatorial "balls-and-urns" model, called URNS, that computes the impact of sample size, redundancy, and corroboration from multiple distinct extraction rules on the probability that an extraction is correct. We describe methods for estimating URNS's parameters in practice and demonstrate experimentally that for UIE the model's log likelihoods are 15 times better, on average, than those obtained by methods used in previous work. We illustrate the generality of the redundancy model by detailing multiple applications beyond UIE in which URNS has been effective. We also provide a theoretical foundation for URNS's performance, including a theorem showing that PAC Learnability in URNS is guaranteed without hand-labeled data, under certain assumptions.

1. Introduction

Automatically extracting knowledge from text is the task of *Information Extraction* (IE). When applied to the Web, IE promises to radically improve Web search engines, allowing them to answer complicated questions by synthesizing information across multiple Web pages. Further, extraction from the Web presents a new approach to a fundamental challenge in artificial intelligence: the automatic accumulation of massive bodies of knowledge.

IE on the Web is particularly challenging due to the variety of different concepts expressed. The strategy employed for previous, small-corpus IE is to hand-label examples for each target concept, and uses the examples to train an extractor [19, 38,7,9,29,27]. On the Web, hand-labeling examples of each concept are intractable—the number of concepts of interest is simply far too large. IE *without* hand-labeled examples is referred to as *Unsupervised* Information Extraction (UIE). UIE

* Corresponding author.
E-mail addresses: ddowney@eecs.northwestern.edu (D. Downey), etzioni@cs.washington.edu (O. Etzioni), soderlan@cs.washington.edu (S. Soderland).
URLs: http://www.cs.northwestern.edu/~ddowney/ (D. Downey), http://www.cs.washington.edu/homes/etzioni/ (O. Etzioni), http://www.cs.washington.edu/homes/soderlan/ (S. Soderland).

0004-3702/$ – see front matter
doi:10.1016/j.artint.2010.04.024

systems such as KnowItAll [16–18] and TextRunner [3,4] have demonstrated that at Web scale, automatically-generated textual patterns can perform UIE for millions of diverse facts. As a simple example, an occurrence of the phrase "*C* such as *x*" suggests that the string *x* is a member of the class *C*, as in the phrase "films such as Star Wars" [22].[1]

However, all extraction techniques make errors, and a key problem for an IE system is determining the probability that extracted information is correct. Specifically, given a corpus, and a set of extractions X_C for a class C, we wish to estimate $P(x \in C|corpus)$ for each $x \in X_C$. In UIE, where hand-labeled examples are unavailable, the task is particularly challenging. How can we automatically assign probabilities of correctness to extractions for arbitrary target concepts, *without* hand-labeled examples?

This paper presents a solution to the above question that applies across a broad spectrum of UIE systems and techniques. It relies on the *KnowItAll hypothesis*, which states that extractions that occur more frequently in distinct sentences in a corpus are more likely to be correct.

KnowItAll hypothesis: Extractions drawn more frequently from distinct sentences in a corpus are more likely to be correct.

The KnowItAll hypothesis holds on the Web. Intuitively, we would expect the KnowItAll hypothesis to hold because although extraction errors occur (e.g., KnowItAll erroneously extracts `California` as a `City` name from the phrase "states containing large cities, such as California"), errors occurring in distinct sentences tend to be different.[2] Thus, typically a given erroneous extraction is repeated only a limited number of times. Further, while the Web does contain some misinformation (for example, the statement "Elvis killed JFK" appears almost 200 times on the Web according to a major search engine), this tends to be the exception (the correct statement "Oswald killed JFK" occurs over 3000 times).

At Web-scale, the KnowItAll hypothesis can identify many correct extractions due to *redundancy*: individual facts are often repeated many times, and in many different ways. For example, consider the TextRunner Web information extraction system, which extracts relational statements between pairs of entities (e.g., from the phrase "Edison invented the light bulb," TextRunner extracts the relational statement `Invented(Edison, light bulb)`). In an experiment with a set of about 500 million Web pages, ignoring the extractions occurring only once (which tend to be errors), TextRunner extracted 829 million total statements, of which only 218 million were unique (on average, 3.8 repetitions per statement). Well-known facts can be repeated many times. According to a major search engine, the Web contains over 10,000 statements that Thomas Edison invented the light bulb, and this fact is expressed in dozens of different ways ("Edison invented the light bulb," "The light bulb, invented by Thomas Edison," "Thomas Edison, after ten thousand trials, invented a workable light bulb," etc.).

Although the KnowItAll hypothesis is simply stated, leveraging it to assess extractions is non-trivial. For example, the 10,000th most frequently extracted `Film` is dramatically more likely to be correct than the 10,000th most frequently extracted `US President`, due to the relative sizes of the target sets. In UIE, this distinction must be identified without any hand-labeled data. This paper shows that a probabilistic model of the KnowItAll hypothesis, coupled with the redundancy of the Web, can power UIE for arbitrary target concepts. The primary contributions are discussed below.

1.1. The urns model of redundancy in text

The KnowItAll hypothesis states that the probability that an extraction is correct increases with its repetition. But by how much? How can we precisely quantify our confidence in an extraction given the available textual evidence?

We present an answer to these questions in the form of the Urns model—an instance of the classic "balls-and-urns" model from combinatorics. In Urns, extractions are represented as draws from an urn, where each ball in the urn is labeled with either a correct extraction, or an error—and different labels can be repeated on different numbers of balls. Given the frequency distribution in the urn for labels in the target set and error set, we can compute the probability that an observed label is a target element based on how many times it is drawn. A key insight of Urns is that when the frequency distributions have predictable structure (for example, in textual corpora the distributions tend to the Zipfian), they can be estimated without hand-labeled data.

We prove that when the frequency of each label in the urn is drawn from a mixture of two Zipfian distributions (one for the target class and another for errors), the parameters of Urns can be learned without hand-labeled data. When the data exhibits a certain separability criterion, PAC learnability is guaranteed. We also demonstrate that Urns is effective in practice. In experiments with UIE on the Web, the probabilities produced by the model are shown to be 15 times better, on average, when compared with techniques from previous work [14].

[1] Here, the term *class* may also refer to relations between multiple strings, e.g. the ordered pair `(Chicago, Illinois)` is a member of the `LocatedIn` class.

[2] Two sentences are *distinct* when they are not comprised of exactly the same word sequence. We stipulate that sentences be distinct to avoid placing undue credence in content that is simply duplicated across many different pages, a common occurrence on the Web.

1.2. Paper outline

The paper proceeds as follows. We describe the URNS model in Section 2, experimentally demonstrate its effectiveness in UIE, and detail applications beyond UIE in which the model has been employed. The theoretical results characterizing the URNS model are presented in Section 3. We discuss future work in Section 4, and conclude.

2. The URNS model

In this section, we describe the URNS model for assigning probabilities of correctness to extractions. We begin by formally introducing the model, then describe our implementation and a set of experiments establishing the model's effectiveness for UIE.

The URNS model takes the form of a classic "balls-and-urns" model from combinatorics. We first consider the single urn case, for simplicity, and then generalize to the full multiple *Urns Model* used in our experiments.

We think of IE abstractly as a generative process that maps text to extractions. Extractions repeat because distinct sentences may yield the same extraction. For example, the sentence containing "Scenic towns such as Yakima…" and the sentence containing "Washington towns such as Yakima…" both lead us to believe that Yakima is a correct extraction of the relation `City(x)`.

Each potential extraction is modeled as a labeled ball in an urn. A *label* represents either an instance of the target relation, or an error. The information extraction process is modeled as repeated draws from the urn, with replacement. Thus, in the above example, two balls are drawn from the urn, each with the label "Yakima". The labels are instances of the relation `City(x)`. Each label may appear on a different number of balls in the urn. Finally, there may be balls in the urn with *error labels* such as "California", representing cases where the extraction process generated a label that is *not* a member of the target relation.

Formally, the parameters that characterize an urn are:

- C – the set of unique target labels; $|C|$ is the number of unique target labels in the urn.
- E – the set of unique error labels; $|E|$ is the number of unique error labels in the urn.
- $num(b)$ – the function giving the number of balls labeled by b where $b \in C \cup E$. $num(B)$ is the multi-set giving the number of balls for each label $b \in B$.

Of course, extraction systems do not have access to these parameters directly. The goal of an extraction system is to discern which of the labels it extracts are in fact elements of C, based on the number of repetitions of each label. Thus, the central question we are investigating is: *given that a particular label x was extracted k times in a set of n draws from the urn, what is the probability that* $x \in C$*?*

In deriving this probability formally below, we assume the system has access to multi-sets $num(C)$ and $num(E)$ giving the number of times the labels in C and E appear on balls in the urn. In our experiments, we provide methods that estimate these multi-sets in the unsupervised and supervised settings.

We derive the probability that an element extracted k of n times is of the target class as follows. First, we have that:

$$P(x \text{ appears } k \text{ times in } n \text{ draws} | x \in C) = \sum_{r \in num(C)} \binom{n}{k} \left(\frac{r}{s}\right)^k \left(1 - \frac{r}{s}\right)^{n-k} P\big(num(x) = r | x \in C\big)$$

where s is the total number of balls in the urn, and the sum is taken over possible repetition rates r.

Then we can express the desired quantity using Bayes Rule:

$$P(x \in C | x \text{ appears } k \text{ times in } n \text{ draws}) = \frac{P(x \text{ appears } k \text{ times in } n \text{ draws} | x \in C) P(x \in C)}{P(x \text{ appears } k \text{ times in } n \text{ draws})}. \tag{1}$$

Note that these expressions include prior information about the label x—for example, $P(x \in C)$ is the prior probability that the string x is a target label, and $P(num(x) = r | x \in C)$ represents the probability that a target label x is repeated on r balls in the urn. In general, integrating this prior information could be valuable for extraction systems; however, in the analysis and experiments that follow, we make the simplifying assumption of uniform priors, yielding the following simplified form:

Proposition 1.

$$P(x \in C | x \text{ appears } k \text{ times in } n \text{ draws}) = \frac{\sum_{r \in num(C)} (\frac{r}{s})^k (1 - \frac{r}{s})^{n-k}}{\sum_{r' \in num(C \cup E)} (\frac{r'}{s})^k (1 - \frac{r'}{s})^{n-k}}.$$

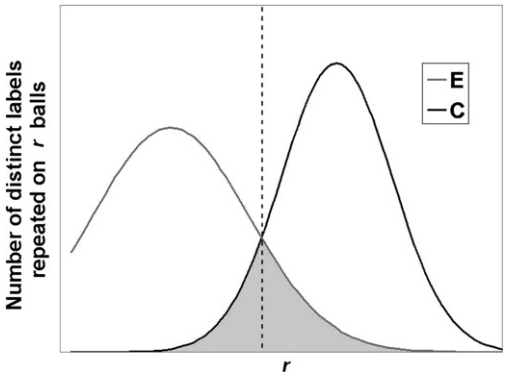

Fig. 1. Schematic illustration of the number of distinct labels in the C and E sets with repetition rate r. The "confusion region" is shaded.

2.0.1. The uniform special case

For illustration, consider the simple case in which all labels from C are repeated on the same number of balls. That is, $num(c_i) = R_C$ for all $c_i \in C$, and assume also that $num(e_i) = R_E$ for all $e_i \in E$. While these assumptions are unrealistic (in fact, we use a Zipf distribution for $num(b)$ in our experiments), they are a reasonable approximation for the majority of labels, which lie on the flat tail of the Zipf curve.

Define p to be the precision of the extraction process; that is, the probability that a given draw comes from the target class. In the uniform case, we have:

$$p = \frac{|C|R_C}{|E|R_E + |C|R_C}.$$

The probability that a *particular* element of C appears in a given draw is then $p_C = p/|C|$, and similarly $p_E = (1-p)/|E|$.

We use a Poisson model to approximate the binomial from Proposition 1. That is, we approximate $(\frac{r}{s})^k(1-\frac{r}{s})^{n-k}$ as $\lambda^k e^{-\lambda}/(\binom{n}{k}k!)$, where λ is $\frac{rn}{s}$. Using this approximation, with algebra we have:

$$P_{USC}(x \in C | x \text{ appears } k \text{ times in } n \text{ draws}) \approx \frac{1}{1 + \frac{|E|}{|C|}(\frac{p_E}{p_C})^k e^{n(p_C - p_E)}}. \tag{2}$$

In general, we expect the extraction process to be noisy but informative, such that $p_C > p_E$. Notice that when this is true, Eq. (2) shows that the odds that $x \in C$ increase exponentially with the number of times k that x is extracted, but also decrease exponentially with the sample size n.

A few numerical examples illustrate the behavior of this equation. The examples assume that the precision p is 0.9. Let $|C| = |E| = 2000$. This means that $R_C = 9 \times R_E$—target balls are nine times as common in the urn as error balls. Now, for $k = 3$ and $n = 10{,}000$ we have $P(x \in C) = 93.0\%$. Thus, we see that a small number of repetitions can yield high confidence in an extracted label. However, when the sample size increases so that $n = 20{,}000$, and the other parameters are unchanged, then $P(x \in C)$ drops to 19.6%. On the other hand, if C balls repeat much more frequently than E balls, say $R_C = 90 \times R_E$ (with $|E|$ set to 20,000, so that p remains unchanged), then $P(x \in C)$ rises to 99.9%.

The above examples enable us to illustrate the advantages of URNS over the noisy-or model used in previous IE work [25,1]. The noisy-or model for IE assumes that each extraction is an independent assertion, correct a fraction p of the time, that the extracted label is correct. The noisy-or model assigns the following probability to extracted labels:

$$P_{noisy\text{-}or}(x \in C | x \text{ appears } k \text{ times}) = 1 - (1-p)^k.$$

Therefore, the noisy-or model will assign the same probability—99.9%—in *all three* of the above examples. Yet, as explained above, 99.9% is only correct in the case for which $n = 10{,}000$ and $R_C = 90 \times R_E$. As the other two examples show, for different sample sizes or repetition rates, the noisy-or model can be highly inaccurate. This is not surprising given that the noisy-or model ignores the sample size and the repetition rates. Section 2.2 quantifies the improvements over the noisy-or obtained by URNS in practice.

2.0.2. Applicability of the URNS *model*

Under what conditions does our redundancy model provide accurate probability estimates? We address this question formally in Section 3, but informally two primary criteria must hold. First, labels from the target set C must be repeated on more balls in the urn than labels from the E set, as in Fig. 1. The shaded region in Fig. 1 represents the "confusion region"—if we classify labels based solely on extraction count, half of the labels in this region will be classified incorrectly, even with the ideal classifier and infinite data, because for these examples there simply isn't enough information to decide whether they belong to C or E. Thus, our model is effective when the confusion region is relatively small. Secondly, even for a small confusion region, the sample size n must be large enough to approximate the two distributions shown in Fig. 1; otherwise the probabilities output by the model will be inaccurate.

2.0.3. Multiple urns

We now generalize our model to encompass multiple urns. When we have multiple extraction mechanisms for the same target class, we could simply sum the extraction counts for each example and apply the single-urn model as described in the previous section. However, this approach forfeits *differences* between the extraction mechanisms that may be informative for classification. For example, an IE system might employ several patterns for extracting city names, e.g. "cities including x" and "x and other towns." It is often the case that different patterns have different modes of failure, so labels extracted by multiple patterns are generally more likely to be correct than those appearing for a single pattern. Previous work in *co-training* has shown that leveraging distinct uncorrelated "views" of the data is often valuable [5]. We model this situation by introducing multiple urns, where each urn represents a different pattern.[3]

Instead of n total extractions, in the multi-urn case we have a sample size n_m for each urn $m \in M$, with the label for example x appearing k_m times. Let $A(x, (k_1, \ldots, k_m), (n_1, \ldots, n_m))$ denote this event. Further, let $A_m(x, k, n)$ be the event that label x appears k times in n draws from urn m, and assuming that the draws from each urn are independent, we have:

Proposition 2.

$$P\big(x \in C \,\big|\, A\big(x, (k_1, \ldots, k_m), (n_1, \ldots, n_m)\big)\big) = \frac{\sum_{c_i \in C} \prod_{m \in M} P(A_m(c_i, k_m, n_m))}{\sum_{x \in C \cup E} \prod_{m \in M} P(A_m(x, k_m, n_m))}.$$

With multiple urns, the distributions of labels among balls in the urns are represented by multi-sets $num_m(C)$ and $num_m(E)$. Expressing the correlation between $num_m(x)$ and $num_{m'}(x)$ is an important modeling decision. Multiple urns are especially beneficial when the repetition rates for elements of C are more strongly correlated across different urns than they are for elements of E—that is, when $num_m(x)$ and $num_{m'}(x)$ are proportionally more similar for $x \in C$ than for $x \in E$. Fortunately, this turns out to be the case in practice in IE. We describe our method for modeling multi-urn correlation in Section 2.1.1.

2.1. Implementation of URNS

This section describes how we implement URNS for both UIE and supervised IE, and identifies the assumptions made in each case.

In order to compute probabilities for extracted labels, we need a method for estimating $num(C)$ and $num(E)$. For the purpose of estimating these sets from labeled or unlabeled data, we assume that $num(C)$ and $num(E)$ are Zipf distributed, meaning that if c_i is the ith most frequently repeated label in C, then $num(c_i)$ is proportional to i^{-z_C}. We can then characterize the $num(C)$ and $num(E)$ sets with five parameters: the set sizes $|C|$ and $|E|$, the shape parameters z_C and z_E, and the extraction precision p.

2.1.1. Multiple urns

To model multiple urns, we consider different precisions p_m for each urn, but make the simplifying assumption that the size and shape parameters are the same for all urns. As mentioned above, we expect repetition rate correlation across urns to be higher for elements of the C set than for the E set. We model this correlation as follows: first, elements of the C set are assumed to come from the same location on the Zipf curve for all urns, that is, their relative frequencies are perfectly correlated. Some elements of the E set are similar, and have the same relative frequency across urns—we refer to these as *global* errors. However, the rest of the E set is made up of *local* errors, meaning that they appear for only one kind of mechanism (for example, "Eastman Kodak" is extracted as an instance of `Film` only in phrases involving the word "film", and not in those involving the word "movie."). Formally, local errors are labels that are present in some urns and not in others. Each type of local error makes up some fraction of the E set, and these fractions are the parameters of our correlation model. Assuming this simple correlation model and identical size and shape parameters across urns is too restrictive in general—differences between mechanisms are often more complex. However, our assumptions allow us to compute probabilities efficiently (as described below), and don't appear to hurt performance significantly in practice (i.e. when compared with an "ideal" model as in Section 2.2.1).

With this correlation model, if a label x is an element of C or a global error, it will be present in all urns. In terms of Proposition 2, the probability that a label x appears k_m times in n_m draws from m is:

$$P\big(A_m(x, k_m, n_m)\big) = \binom{n_m}{k_m} \big(f_m(x)\big)^{k_m} \big(1 - f_m(x)\big)^{n_m - k_m} \tag{3}$$

where $f_m(x)$ is the frequency of label x. That is,

$$f_m(c_i) = p_m Q_C i^{-z_C} \quad \text{for } c_i \in C,$$
$$f_m(e_i) = (1 - p_m) Q_E i^{-z_E} \quad \text{for } e_i \in E.$$

[3] We may lump several patterns into a single urn if they tend to behave similarly.

In these expressions, i is the frequency rank of the label, assumed to be the same across all urns, and Q_C and Q_E are normalizing constants such that

$$\sum_{c_i \in C} Q_C i^{-z_C} = \sum_{e_i \in E} Q_E i^{-z_E} = 1.$$

For a local error x which is not present in urn m, $P(A_m(x, k_m, n_m))$ is 1 if $k_m = 0$ and 0 otherwise. Substituting these expressions for $P(A_m(x, k_m, n_m))$ into Proposition 2 gives the final form of our URNS model.

2.1.2. Efficient computation

A feature of our implementation is that it allows for efficient computation of probabilities. In general, computing the sum in Proposition 2 over the potentially large C and E sets would require significant computation for each label. However, given a fixed number of urns, with $num(C)$ and $num(E)$ Zipf distributed, an integral approximation to the sum in Proposition 2 (using a Poisson in place of the binomial in Eq. (3)) can be solved in closed form in terms of incomplete Gamma functions. The details of this approximation and its solution for the single-urn case are given in Section 3.[4] The closed form expression can be evaluated quickly, and thus probabilities for labels can be obtained efficiently. This solution leverages our assumptions that size and shape parameters are identical across urns, and that relative frequencies are perfectly correlated. Finding efficient techniques for computing probabilities under less stringent assumptions is an item of future work.

2.1.3. Supervised parameter estimation

In the event that a large sample of hand-labeled training examples is available for each target class of interest, we can directly estimate each of the parameters of URNS. In our experiments, we use Differential Evolution to identify parameter settings that approximately maximize the conditional log likelihood of the training data [40].[5] Differential Evolution is a population-based stochastic optimization technique, appropriate for optimizing the non-convex likelihood function for URNS. Once the parameters are set, the model yields a probability for each extracted label, given the number of times k_m it appears in each urn and the number of draws n_m from each urn.

2.1.4. Unsupervised parameter estimation

Estimating model parameters in an unsupervised setting requires making a number of assumptions tailored to the specific task. Below, we detail the assumptions employed in URNS for UIE. It is important to note that while these assumptions are specific to UIE, they are *not* specific to a particular target class. As argued in [17], UIE systems cannot rely on per-class information—in the form of either assumptions or hand-labeled training examples—if they are to scale to extracting information on arbitrary classes that are not specified in advance.

Implementing URNS for UIE requires a solution to the challenging problem of estimating $num(C)$ and $num(E)$ using only untagged data. Let U be the multi-set consisting of the number of times each unique label was extracted in a given corpus. $|U|$ is the number of unique labels encountered, and the sample size $n = \sum_{r \in U} r$.

In order to learn $num(C)$ and $num(E)$ without hand-labeled data, we make the following assumptions:

- Because the number of different possible errors is nearly unbounded, we assume that the error set is very large.[6]
- We assume that both $num(C)$ and $num(E)$ are Zipf distributed where the z_E parameter is set to 1.
- In our experience with KNOWITALL, we found that while different extraction rules have differing precision, each rule's precision is stable across different classes [17]. For example, the precision of the extractor "cities such as x" and "insects such as y" are similar. URNS takes this precision as an input. To demonstrate that URNS is not overly sensitive to this parameter, we chose a fixed value (0.9) and used it as the precision p_m for all urns in our experiments.[7] Section 2.2.5 provides evidence that the observed p value tends to be relatively stable across different target classes.

We then use Expectation Maximization (EM) over U in order to arrive at appropriate values for $|C|$ and z_C (these two quantities uniquely determine $num(C)$ given our assumptions). Our EM algorithm proceeds as follows:

1. Initialize $|C|$ and z_C to starting values.
2. Repeat until convergence:
   (a) **E-step** Assign probabilities to each element of U using Proposition (1).
   (b) **M-step** Set $|C|$ and z_C from U using the probabilities assigned in the E-step (details below).

[4] For the multi-urn solution, which is obtained through a symbolic integration package and therefore complicated, we refer the reader to the Java implementation of the solution which is available for download—see [12], Appendix A.

[5] Specifically, we use the Differential Evolution routine built into Mathematica 5.0.

[6] In our experiments, we set $|E| = 10^6$. A sensitivity analysis showed that changing $|E|$ by an order of magnitude, in either direction, resulted in only small changes to our results.

[7] A sensitivity analysis showed that choosing a substantially higher (0.95) or lower (0.80) value for p_m still resulted in URNS outperforming the noisy-or model by at least a factor of 8 and PMI by at least a factor of 10 in the experiments described in Section 2.2.1.

We obtain $|C|$ and z_C in the M-step by first estimating the rank-frequency distribution for labels from C in the untagged data U. From U and the probabilities found in the E-step, we can obtain $E_C[k]$, the expected number of labels from C that were extracted k times for $k \geqslant 1$ (the $k = 0$ case is detailed below). We then round these fractional expected counts into a discrete rank-frequency distribution with a number of elements equal to the expected total number of labels from C in the untagged data, $\sum_k E_C[k]$. We obtain z_C by fitting a Zipf curve to this rank-frequency distribution by linear regression on a log–log scale.[8]

Lastly, we set $|C| = \sum_k E_C[k] + \mathit{unseen}$, where we estimate the number of unseen labels of the C set (i.e. those with $k = 0$) using Good–Turing estimation [20]. Good–Turing estimation provides an estimate of the *probability mass* of the unseen labels (specifically, the estimate is equal to the expected fraction of the draws from C that extracted labels seen only once). To convert this probability into a number of unseen labels, we simply assume that each unseen label has probability equal to that of the least frequent seen label. A potentially more accurate method would choose *unseen* such that the actual number of unique labels observed is equal to that expected by the model (where the latter is measured e.g. by sampling). Such methods are an item of future work.

This unsupervised learning strategy proved effective for target classes of different sizes; for example, URNS learned parameters such that the number of elements of the `Country` relation with non-negligible extraction probability was about two orders of magnitude smaller than that of the `Film` and `City` classes, which approximately agrees with the actual relative sizes of these sets.

2.2. URNS: Experimental results

How accurate is URNS at assigning probabilities of correctness to extracted labels? In this section, we answer this question by comparing the accuracy of URNS's probabilities against other methods from previous work.

This section begins by describing our experimental results for IE under two settings: unsupervised and supervised. We first describe two unsupervised methods from previous work: the noisy-or model and PMI. We then compare URNS with these methods experimentally, and lastly compare URNS with several baseline methods in a supervised setting.

We evaluated our algorithms on extraction sets for the classes `City(x)`, `Film(x)`, `Country(x)`, and `MayorOf(x,y)`, taken from experiments with the KNOWITALL system performed in [17]. The sample size n was 64,605 for `City`, 135,213 for `Film`, 51,390 for `Country` and 46,858 for `MayorOf`. The extraction patterns were partitioned into urns based on the name they employed for their target relation (e.g. "country" or "nation") and whether they were left-handed (e.g. "countries including *x*") or right-handed (e.g. "*x* and other countries"). We chose this partition because it results in extraction mechanisms that make relatively uncorrelated errors, as assumed in the multiple-urns model. For example, the phrase "Toronto, Canada and other cities" will mislead a right-handed pattern into extracting "Canada" as a `City` candidate, whereas a left-handed pattern is far less prone to this error. Each combination of relation name and handedness was treated as a separate urn, resulting in four urns for each of `City(x)`, `Film(x)`, and `Country(x)`, and two urns for `MayorOf(x, y)`.[9,10]

For each relation, we tagged a random sample of 1000 extracted labels, using external knowledge bases (the Tipster Gazetteer for cities and the Internet Movie Database for films) and manually tagging those instances not found in a knowledge base. For `Country` and `MayorOf`, we manually verified correctness for all extracted labels, using the Web. Countries were marked correct provided they were a correct name (including abbreviations) of a current country, and mayors were marked correct if the person was a mayor of the city at some point in time. In the UIE experiments, we evaluate our algorithms on all 1000 examples, and in the supervised IE experiments we perform 10-fold cross validation.

2.2.1. UIE experiments

We compare URNS against two other methods for unsupervised information extraction. First, in the *noisy-or* model used in previous work, an extracted label appearing k_m times in each urn is assigned probability $1 - \prod_{m \in M}(1 - p_m)^{k_m}$, where p_m is the extraction precision for urn m. We describe the second method below.

Our previous work on KNOWITALL used Pointwise Mutual Information (PMI) to obtain probability estimates for extracted labels [17]. Specifically, the PMI between an extracted label and a set of automatically generated *discriminator phrases* (e.g., "movies such as *x*") is computed from Web search engine hit counts. These PMI scores are used as features in a Naive Bayes Classifier (NBC) to produce a probability estimate for the label. The NBC is trained using a set of automatically bootstrapped seed instances. The positive seed instances are taken to be those having the highest PMI with the discriminator phrases

[8] To help ensure that our probability estimates are increasing with k, if z_C falls below 1, we adjust z_E to be less than z_C.

[9] Draws from URNS are intended to represent independent evidence. Because the same sentence can be duplicated across multiple different Web documents, in these experiments we consider only each *unique* sentence containing an extraction to be a draw from URNS. In experiments with other possibilities, including counting the number of unique documents producing each label, or simply counting every extraction of each label, we found that for UIE, performance differences between the various approaches were small compared to the differences between URNS and other methods.

[10] In the unsupervised setting, we assumed that the fraction of errors in the urns that are local is 0.1, and that errors appearing for only left- or only right-handed patterns were equally prevalent to those appearing for only one label. The only exception was the `City` class, where because the target class is the union of the two class names ("city" and "town") rather than the intersection (as with "film" and "movie"), we assumed that no local errors appeared for only one name. Altering these settings (or indeed, simply using a single urn—see Section 2.2.4) had negligible impact on the results in Fig. 2.

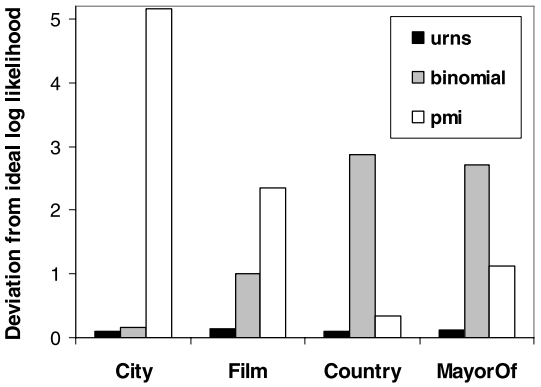

Fig. 2. Deviation of average log likelihood from the ideal for four relations (lower is better). On average, URNS outperforms noisy-or by a factor of 15, and PMI by a factor of 20.

Table 1
Improved efficiency due to URNS. The top row reports the number of search engine queries made by KNOWITALL using PMI divided by the number of queries for KNOWITALL using URNS. The bottom row shows that PMI's queries increase with k—the average number of distinct labels for each relation. Thus, speedup tends to vary inversely with the average number of times each label is drawn.

	City	Film	MayorOf	Country
Speedup	17.3×	9.5×	1.9×	3.1×
Average k	3.7	4.0	20.7	23.3

after the bootstrapping process; the negative seeds are taken from the positive seeds of other relations, as in other work (e.g., [25]).

Although PMI was shown in [17] to rank extracted labels fairly well, it has two significant shortcomings. First, obtaining the hit counts needed to compute the PMI scores is expensive, as it requires a large number of queries to a public Web search engine (or, alternatively, the expensive construction of a local Web-scale inverted index). Second, the seeds produced by the bootstrapping process are often noisy and not representative of the overall distribution of extractions [39]. This combined with the probability polarization introduced by the NBC tends to give inaccurate probability estimates.

2.2.2. Discussion of UIE results

The results of our unsupervised experiments are shown in Fig. 2. We plot deviation from the *ideal* log likelihood—defined as the maximum achievable log likelihood given our feature set. Specifically, for each class C define an ideal model $P_{ideal}(x)$ equal to the fraction of test set labels with the same extraction counts as x that are correct. We define the ideal log likelihood as:

$$\text{ideal log likelihood} = \sum_{x \in C} \log P_{ideal}(x) + \sum_{x \in E} \log\bigl(1 - P_{ideal}(x)\bigr). \tag{4}$$

Our experimental results demonstrate that URNS overcomes the weaknesses of PMI. First, URNS's probabilities are far more accurate than PMI's, achieving a log likelihood that is a factor of 20 closer to the ideal, on average (Fig. 2). Second, URNS is substantially more efficient as shown in Table 1.

This efficiency gain requires some explanation. These experiments were performed using the KNOWITALL system, which relies on queries to Web search engines to identify Web pages containing potential extractions. The number of queries KNOWITALL can issue daily is limited, and querying over the Web is, by far, KNOWITALL's most expensive operation. Thus, number of search engine queries is our efficiency metric. Let d be the number of discriminator phrases used by the PMI explained above. The PMI method requires $O(d)$ search engine queries to compute the PMI of each extracted label from search engine hit counts. In contrast, URNS computes probabilities *directly* from the set of extractions—requiring *no* additional queries, which cuts KNOWITALL's queries by factors ranging from 1.9 to 17.

As explained in Section 2.0.1, the noisy-or model ignores target set size and sample size, which leads it to assign probabilities that are far too high for the `Country` and `MayorOf` relations, where the average number of times each label is extracted is high (see bottom row of Table 1). This is further illustrated for the `Country` relation in Fig. 3. The noisy-or model assigns appropriate probabilities for low sample sizes, because in this case most extracted labels are in fact correct, as predicted by the noisy-or model. However, as sample size increases, the fraction of correct labels decreases—and the noisy-or estimate worsens. On the other hand, URNS avoids this problem by accounting for the interaction between target set size and sample size, adjusting its probability estimates as sample size increases. Given sufficient sample size, URNS performs close to the ideal log likelihood, improving slightly with more samples as the estimates obtained by the EM process

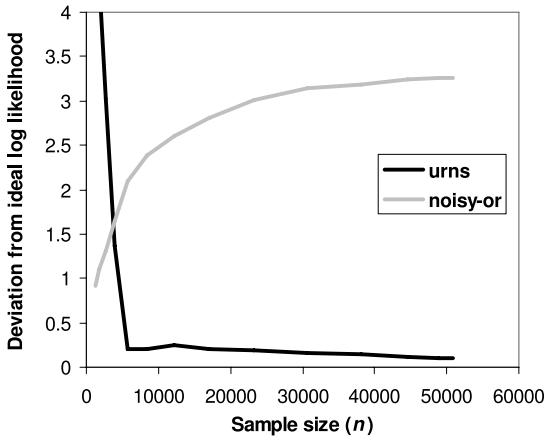

Fig. 3. Deviation of average log likelihood from the ideal as sample size varies for the `Country` relation (lower is better). URNS performs close to the ideal given sufficient sample size, whereas noisy-or becomes less accurate as sample size increases.

become more accurate. Overall, URNS assigns far more accurate probabilities than the noisy-or model, and its log likelihood is a factor of 15 closer to the ideal, on average. The very large differences between URNS and both the noisy-or model and PMI suggest that, even if the performance of URNS degrades in other domains, it is quite likely to still outperform both PMI and the noisy-or model.

Our computation of log-likelihood contains a numerical detail that could potentially influence our results. To avoid the possibility of a likelihood of zero, we restrict the probabilities generated by URNS and the other methods to lie within the range (0.00001, 0.99999). Widening this range tended to improve URNS's performance relative to the other methods, as this increases the penalty for erroneously assigning extreme probabilities—a problem more prevalent for PMI and noisy-or than for URNS. If we narrow the range by two digits of precision, to (0.001, 0.999), URNS still outperforms PMI by a factor of 15, and noisy-or by a factor of 13. Thus, we are comfortable that the differences observed are not an artifact of this design decision.

Lastly, although we focus our evaluation on the quality of each method's probability estimates in terms of likelihood, the advantage of URNS is also reflected in other metrics such as classification accuracy. When we convert each method's probability estimate into a classification (positive for a label *iff* the probability estimate is greater than 0.5), we find that URNS has an average accuracy of approximately 81%, compared with PMI at 63% and noisy-or at 47%. Thus, URNS decreases classification error over the previous methods by a factor of 1.9× to 2.8×. URNS *ranks* the majority of extracted labels in a manner similar to the noisy-or model (which ranks by overall frequency). Thus, URNS offers comparable performance to noisy-or in terms of e.g. area under the precision/recall curve [6]. However, the correlations captured by multiple urns can improve the ranking of sufficiently frequent labels, as detailed in Section 2.2.4.

2.2.3. Supervised IE experiments

We compare URNS with three supervised methods. All methods utilize the same feature set as URNS, namely the extraction counts k_m.

- **noisy-or** – Has one parameter per urn, making a set of M parameters $(h_1, \ldots, h_M)$, and assigns probability equal to
$$1 - \prod_{m \in M} (1 - h_m)^{k_m}.$$
- **logistic regression** – Has $M + 1$ parameters $(a, b_1, b_2, \ldots, b_M)$, and assigns probability equal to
$$\frac{1}{1 + e^{a + \sum_{m \in M} k_m b_m}}.$$
- **SVM** – Consists of an SVM classifier with a Gaussian kernel. To transform the output of the classifier into a probability, we use the probability estimation built-in to LIBSVM [8], which is based on logistic regression of the SVM decision values.

Parameters maximizing the conditional likelihood of the training data were found for the noisy-or and logistic regression models using Differential Evolution.[11] For those models and URNS, we performed 20 iterations of Differential Evolution

[11] For logistic regression, different convex optimization methods are applicable; however, in our experiments the Differential Evolution routine appeared to converge to an optimum, and we do not believe the choice of optimization method impacted the logistic regression results.

Table 2
Supervised IE experiments. Deviation from the ideal log likelihood for each method and each relation (lower is better). The overall performance differences are small, with URNS 19% closer to the ideal than noisy-or, on average, and 10% closer than logistic regression. The overall performance of SVM is close to that of URNS.

	City	Film	Mayor	Country	Average
noisy-or	0.0439	0.1256	0.0857	0.0795	0.0837
logistic regression	0.0466	0.0893	**0.0655**	0.1020	0.0759
SVM	0.0444	0.0865	0.0659	**0.0769**	0.0684
URNS	**0.0418**	**0.0764**	0.0721	0.0823	**0.0681**

Table 3
Label-precision of the K highest-ranked extracted labels for varying values of K between 10 and 200. Across the five K values shown, URNS reduces error over the single-urn model by an average of 29%.

Number of highest-ranked extracted labels	Single-urn	URNS
10	1	1
20	0.9875	1
50	0.925	0.955
100	0.8375	0.845
200	0.7075	0.71

using 400 distinct search points. In the SVM case, we performed grid search to find the kernel parameters giving the best likelihood performance for each training set—this grid search was required to get acceptable performance from the SVM on our task.

The results of our supervised learning experiments are shown in Table 2. URNS, because it is more expressive, is able to outperform the noisy-or and logistic regression models. In terms of deviation from the ideal log likelihood, we find that on average URNS outperforms the noisy-or model by 19%, logistic regression by 10%, but SVM by only 0.4%.

2.2.4. Benefit from multiple urns

The previous results use the full multi-urn model. How much of URNS's large performance advantage in UIE is due to multiple urns?

In terms of likelihood, as measured in Fig. 2, we found that the impact of multiple urns is negligible. This is primarily because the majority of extracted labels occur only a handful of times, and in these cases the multiple-urn model lacks enough data to estimate the correlation of counts across urns.

Multiple urns *can* offer some performance benefit, however, for more commonly extracted labels. We evaluated the effect of multiple urns for UIE across the four relations shown in Fig. 2, computing the average *label-precision at K*, equal to the fraction of the K highest-probability labels which are correct. The results under the single-urn and full URNS model are shown in Table 3 for varying K. The full URNS model always performs at least as well as the single-urn model, and sometimes provides much higher precision. In fact, using multiple urns reduces the error by 29% on average for the five K values shown in the table.

2.2.5. Is p a "universal constant"?

Our UIE experiments employed an extraction precision parameter p of 0.9. While URNS still massively outperforms previous methods even if this value is adjusted to 0.8 or 0.95, the accuracy of URNS's probabilities does degrade as p is altered away from 0.9.

In this section, we attempt to measure how consistent the observed p value is across varying classes. This experiment differs somewhat from those presented above. In order to test across a wide variety of classes, we moved beyond the KNOWITALL experiments from [17] and used the TEXTRUNNER system to provide instances of classes [3]. To choose classes to investigate, we randomly selected 12 nouns from WordNet for which there were at least 100 extractions (not necessarily unique) in TEXTRUNNER. We excluded nouns which were overly general such that nearly any extraction would be correct (e.g., the class `Example`) and nouns which are rarely or never used to name concrete instances (e.g., the class `Purchases`). The results in this section were compiled by querying TEXTRUNNER for 100 sentences containing extractions for each class.[12]

While TEXTRUNNER provides greater coverage than KNOWITALL, precision in general is lower. One of the inaccuracies of the TEXTRUNNER system is that it often fails to delimit the boundaries of extractions properly (e.g., it extracts the phrase "alkanes or cycloalkanes" as an instance of the `Solvents` class). We found that we could improve the precision of TEXTRUNNER by over 20% on average by post-processing all extractions, breaking on conjunctions or punctuation (i.e. the previous example becomes simply "alkanes"). Our results employ this heuristic.

The results of the experiment are shown in Table 4. For each class, "p Observed" gives the fraction of the 100 extractions tagged correct (by manual inspection). The average p value observed across classes of 0.84 is lower than the value of 0.9

[12] The list of excluded nouns and the labeled extractions for each selected class are available for download; see [12], Appendix A.

Table 4
Average p values for various classes, measured from 100 hand-tagged examples per class. Three of the 12 classes have p values in bold, indicating a statistically significantly difference from the mean of 0.84 (significance level of 0.01, Fisher Exact Test). However, if we adjust the estimate of p per class according to how frequently it occurs in the "such as" pattern (using the factor h_{class}; see text), none of the resulting $p + h_{class}$ values are significantly different from the mean.

Class	p Observed	$\frac{\text{Hits(class such as)}}{\text{Hits(class)}}$	p Observed + h_{class}
solvents	**0.98**	0.201	0.85
devices	0.93	0.022	0.87
thinkers	0.93	0.013	0.89
relaxants	0.92	0.010	0.89
mushrooms	0.86	0.001	0.90
mechanisms	0.85	0.017	0.80
resorts	0.85	0.002	0.88
flies	0.84	0.0004	0.93
tones	0.77	0.001	0.83
wounds	0.77	0.002	0.80
machines	**0.69**	0.002	0.71
cultures	**0.67**	0.002	0.70

we use in our previous experiments; this reflects the relatively lower precision of TextRunner as well as the increased difficulty of extracting common nouns (versus the proper noun extractions used previously). The results show that while there is substantial regularity in observed p values, the values are not perfectly consistent. In fact, three classes (with "p Observed" values in bold) differ significantly from the average observed p value (at significance level of 0.01, Fisher Exact Test).

Given that we observe variability in p values across classes, an important question is whether the correct p value for a given class, p_{class}, can be predicted. We observed empirically that the precision of extractions for a class increases with how relatively frequently the class name is used in extraction patterns. As an example, the phrase "cultures such as x" appears infrequently relative to the word "cultures," as shown in Table 4 in terms of Web hit counts obtained from a search engine. In turn, the class Cultures exhibits a relatively low p value. Intuitively, this result makes sense—class names which are more "natural" for naming instances should both appear more frequently in extraction patterns, and provide more precise extractions.

We can exploit the above intuition by adjusting the estimate of extraction precision for each class by a factor h_{class}. For illustration, based on the values in Table 4, we devised the following adjustment factor:

$$h_{class} = 0.08\left(-2.36 - \log_{10}\frac{\text{Hits(class such as)}}{\text{Hits(class)}}\right). \tag{5}$$

The adjustment factor can give us a more accurate estimate of the precision for a given class $p_{class} = p - h_{class}$.

Obviously, the expression h_{class} is heuristic and could be further refined using additional experiments. Nonetheless, adjusting by the factor does allow us to obtain better precision estimates across classes. The quantity "p Observed + h_{class}" has only 57% of the variance of the original "p Observed" (and the same mean, by construction). Further, none of the observed differences of "p Observed + h_{class}" are statistically significantly different from the original mean, using the same significance test employed previously.

Lastly, we should mention that even *without* any adjustment factor, the variance in p value across classes is not substantially greater than that employed in our sensitivity analysis in Section 2.2. Thus, we expect the performance advantages of Urns over the noisy-or and PMI models to extend to these other classes as well.

2.3. Urns: *Other applications*

Urns is a general model. For any classification task, if one of the features represents a count of observations following a mixture of Zipf distributions as assumed by Urns, the model can be employed. In this section, we highlight three examples of how the Urns model has been applied to tasks other than that of assigning probabilities of correctness to extractions.

2.3.1. *Estimating UIE precision and recall*

An attractive feature of Urns is that it enables us to estimate its expected recall and precision as a function of sample size. If the distributions in Fig. 1 cross at the dotted line shown then, given a sufficiently large sample size n, expected recall will be the fraction of the area under the C curve lying to the right of the dotted line.

For a given sample size n, define τ_n to be the least number of appearances k at which an extracted label is more likely to be from the C set than the E set (given the distributions in Fig. 1, τ_n can be computed using Proposition 1). Then we have:

$$\mathbf{E}[\mathit{TruePositives}] = |C| - \sum_{r \in num(C)} \sum_{k=0}^{\tau_n - 1} \binom{n}{k} \left(\frac{r}{s}\right)^k \left(1 - \frac{r}{s}\right)^{n-k}$$

Table 5
Estimating precision and recall in UIE. Listed is the URNS model estimate for precision and recall, along with the actual measured quantities, for four classes. The major differences between the classes—that the `MayorOf` and `Country` classes have roughly two orders of magnitude lower recall than the `City` and `Film` classes—is qualitatively reflected by the model.

	n	E[Recall]	Actual recall	E[Precision]	Actual precision
`City`	64605	12900	14300	0.78	0.84
`Country`	51390	37	176	0.63	0.77
`Film`	135213	25900	23400	0.79	0.68
`MayorOf`	46858	58	158	0.62	0.79

where we define "true positives" to be the number of extracted labels $c_i \in C$ for which the model assigns probability $P(c_i \in C) > 0.5$.

The expected number of false positives is similarly:

$$\mathbf{E}[\mathit{FalsePositives}] = |E| - \sum_{r \in num(E)} \sum_{k=0}^{\tau_n - 1} \binom{n}{k} \left(\frac{r}{s}\right)^k \left(1 - \frac{r}{s}\right)^{n-k}.$$

The expected precision of the system can then be approximated as:

$$\mathbf{E}[\mathit{Precision}] \approx \frac{\mathbf{E}[\mathit{TruePositives}]}{\mathbf{E}[\mathit{FalsePositives}] + \mathbf{E}[\mathit{TruePositives}]}.$$

To illustrate the potential benefit of the above calculations and evaluate their accuracy, we computed expected recall and precision for the particular $num(C)$ and $num(E)$ learned (in the unsupervised setting) in our experiments in Section 2.2. The results appear in Table 5. The recall estimates are within 11% of the actual recall (that is, the estimated number of correct examples in our set of extracted labels, based on the hand-tagged test set) for the `City` and `Film` classes. Further, the estimates reflect the important qualitative difference between the large `City` and `Film` classes as compared with the smaller `MayorOf` and `Country` classes.

Were we to increase the sample size n for the `Film` class and the `Country` class each to 1,000,000, the model predicts that we would increase our `Film` recall by 81%, versus only 4% for `Country`. Thus, the above equations allow an information extraction system to dynamically choose how to allocate resources to match given precision and recall goals, even in the absence of hand-labeled data.

2.3.2. Estimating the functionality of relations

Knowledge of which relations in a knowledge base are *functional* is valuable for a variety of different tasks. Previous work has shown that knowledge of functional relations can be used to automatically detect contradictions in text [11,34], and to automatically identify extractor errors in IE [1]. For example, if we know that the `Headquartered` relation is functional and we see one document asserting that *Intel* is headquartered in *Santa Clara*, and another asserting it is headquartered in *Phoenix*, we can determine that either the documents contradict each other, or we have made an error in extraction. In this section, we illustrate how URNS can be used to automatically compute the probability that a phrase denotes a functional relation.

The discussion in this section is based on a set of extracted *tuples*. An extracted tuple takes the form $R(x, y)$ where (roughly) x is the subject of a sentence, y is the object, and R is a phrase denoting the relationship between them. If the relation denoted by R is functional, then typically the object y is a function of the subject x. Thus, our discussion focuses on this possibility, though the analysis is easily extended to the symmetric case.

The main evidence that a relation $R(x, y)$ is functional comes from the distribution of y values for a given x value. If R denotes a function and x is unambiguous, then we expect the extractions to be predominantly a single y value, with a few outliers due to noise.

Example A in Fig. 4 has strong evidence for a functional relation. 66 out of 70 extractions for *was_born_in (Mozart, PLACE)* have the same y value. An ambiguous x argument, however, can make a functional relation appear non-functional. Example B refers to multiple real-world individuals named "John Adams" and has a distribution of y values that appears less functional than example C, which has a non-functional relation.

Logically, a relation R is functional in a variable x if it maps it to a unique variable y: $\forall x, y_1, y_2 R(x, y_1) \wedge R(x, y_2) \Rightarrow y_1 = y_2$. Thus, given a large random sample of ground instances of R, we could detect with high confidence whether R is functional. In text, the situation is far more complex due to ambiguity, polysemy, synonymy, and other linguistic phenomena.

To decide whether R is functional in x for all x, we first consider how to detect whether R is *locally functional* for a particular value of x. We later combine the local functionality probabilities to estimate the global functionality of a relation.[13] Local functionality for a given x can be modeled in terms of the global functionality of R and the ambiguity of x.

[13] We compute global functionality as the average local scores, weighted by the probability that x is unambiguous.

A.	was_born_in(Mozart, PLACE): Salzburg(66), Germany(3), Vienna(1)
B.	was_born_in(John Adams, PLACE): Braintree(12), Quincy(10), Worcester(8)
C.	lived_in(Mozart, PLACE): Vienna(20), Prague(13), Salzburg(5)

Fig. 4. Functional relations such as example A have a different distribution of y values than non-functional relations such as C. Ambiguous x argument as in B, however, can make a functional relation *appear* non-functional.

We later outline an EM-style algorithm that alternately estimates the probability that R is functional and the probability that x is ambiguous.

Let θ_R^f be the probability that $R(x,\cdot)$ is locally functional for a random x, and let Θ^f be the vector of these parameters across all relations R. Likewise, θ_x^u represents the probability that x is locally unambiguous for random R, and Θ^u the vector for all x.

We wish to determine the *maximum a posteriori* (MAP) functionality and ambiguity parameters given the observed data D, that is $\arg\max_{\Theta^f,\Theta^u} P(\Theta^f,\Theta^u|D)$. By Bayes Rule:

$$P\big(\Theta^f,\Theta^u|D\big) \propto P\big(D|\Theta^f,\Theta^u\big)P\big(\Theta^f,\Theta^u\big). \tag{6}$$

We outline a generative model for the data, $P(D|\Theta^f,\Theta^u)$. Let R_x^* indicate the event that the relation R is locally functional for the argument x, and that x is locally unambiguous for R. Also, let D indicate the set of observed tuples, and define $D_{R(x,\cdot)}$ as the multi-set containing the frequencies for extractions of the form $R(x,\cdot)$.

Let us assume that the event R_x^* depends only on θ_R^f and θ_x^u, and further assume that given these two parameters, local ambiguity and local functionality are conditionally independent. We obtain the following expression for the probability of R_x^* given the parameters:

$$P\big(R_x^*|\Theta^f,\Theta^u\big) = \theta_R^f\theta_x^u.$$

We assume each set of data $D_{R(x,\cdot)}$ is generated independently of all other data and parameters, given R_x^*. From this and the above we have:

$$P\big(D|\Theta^f,\Theta^u\big) = \prod_{R,x}\big(P\big(D_{R(x,\cdot)}|R_x^*\big)\theta_R^f\theta_x^u + P\big(D_{R(x,\cdot)}|\neg R_x^*\big)\big(1-\theta_R^f\theta_x^u\big)\big). \tag{7}$$

These independence assumptions allow us to express $P(D|\Theta^f,\Theta^u)$ in terms of distributions over $D_{R(x,\cdot)}$ given whether or not R_x^* holds. We use a single-urn model to estimate these probabilities based on binomial distributions.

Let $k = \max D_{R(x,\cdot)}$, and let $n = \sum D_{R(x,\cdot)}$; we will approximate the distribution over $D_{R(x,\cdot)}$ in terms of k and n. In the single-urn model, if $R(x,\cdot)$ is locally functional and unambiguous, k has a binomial distribution with parameters n and p, where p is the precision of the extraction process. If $R(x,\cdot)$ is *not* locally functional and unambiguous, then we expect k to typically take on smaller values. Empirically, we find that the underlying frequency of the most frequent element in the $\neg R_x^*$ case tends to follow a Beta distribution.

Under the model, the probability of the evidence given R_x^* is:

$$P\big(D_{R(x,\cdot)}|R_x^*\big) \approx P\big(k,n|R_x^*\big) = \binom{n}{k}p^k(1-p)^{n-k}. \tag{8}$$

And the probability of the evidence given $\neg R_x^*$ is:

$$P\big(D_{R(x,\cdot)}|\neg R_x^*\big) \approx P\big(k,n|\neg R_x^*\big) = \binom{n}{k}\int_0^1 \frac{p'^{k+\alpha_f-1}(1-p')^{n+\beta_f-1-k}}{B(\alpha_f,\beta_f)}\,dp' = \frac{\binom{n}{k}\Gamma(n-k+\beta_f)\Gamma(\alpha_f+k)}{B(\alpha_f,\beta_f)\Gamma(\alpha_f+\beta_f+n)}, \tag{9}$$

where n is the sum over $D_{R(x,\cdot)}$, Γ is the Gamma function and B is the Beta function. α_f and β_f are the parameters of the Beta distribution for the $\neg R_x^*$ case (in practice, these are estimated empirically).

Substituting Eq. (9) into Eq. (7) and applying an appropriate prior gives the probability of parameters Θ^f and Θ^u given the observed data D. However, Eq. (7) contains a large product of sums—with two independent vectors of coefficients, Θ^f and Θ^u—making it difficult to optimize analytically.

If we knew which arguments were ambiguous, we would ignore them in computing the functionality of a relation. Likewise, if we knew which relations were non-functional, we would ignore them in computing the ambiguity of an argument. Instead, we initialize the Θ^f and Θ^u arrays randomly, and then execute an EM-style algorithm to arrive at a high-probability setting of the parameters.

Note that if Θ^u is fixed, we can compute the expected fraction of locally unambiguous arguments x for which R is locally functional, using $D_{R(x',\cdot)}$ and Eq. (9). Likewise, for fixed Θ^f, for any given x we can compute the expected fraction of locally functional relations R that are locally unambiguous for x.

Specifically, we repeat until convergence:

1. Set $\theta_R^f = \frac{1}{s_R} \sum_x P(R_x^*|D_{R(x,\cdot)})\theta_x^u$ for all R.
2. Set $\theta_x^u = \frac{1}{s_x} \sum_R P(R_x^*|D_{R(x,\cdot)})\theta_R^f$ for all x.

In both steps above, the sums are taken over only those x or R for which $D_{R(x,\cdot)}$ is non-empty. Also, the normalizer $s_R = \sum_x \theta_x^u$ and likewise $s_x = \sum_R \theta_R^f$.

By iteratively setting the parameters to the expectations in steps 1 and 2, we arrive at a good setting of the parameters.

The above algorithm is experimentally investigated in [34], showing that the technique effectively identifies functional relations, and can power effective contradiction detection.

2.3.3. Synonym resolution

The last application of URNS we will discuss is that of resolving which strings refer to the same objects or relations. In text, the same object is often referred to by multiple distinct names—"U.S." and "United States" each refer to the same country, for example. Likewise, relationships between objects are often expressed as multiple distinct paraphrases (e.g., "x is the capital of y" and "x, capital of y").

The RESOLVER system performs *Synonym Resolution*—taking as input a set of extracted tuples (as discussed above, e.g., `IsCapitalOf(D.C., United States)`) and returning a set of clusters, where each cluster contains coreferential object strings or relationship strings [42].

Here we provide a high-level description of how RESOLVER employs an URNS-like model, deferring to [42] for the details. Consider the task of determining whether two strings s_1 and s_2 refer to the same object, based on a set of tuples each including either s_1 or s_2 as an argument. RESOLVER specifies a urn-based generative process for the observed tuples; namely, the set of potential tuples for s_i are modeled as labels on balls in a urn, and the actual observed tuples involving s_i are modeled as draws from the urn. RESOLVER assumes that if s_1 and s_2 refer to the same object, then the urn contents for s_1 are maximally similar to those for s_2; otherwise, the two urns can differ to a greater or lesser degree. With this assumption, RESOLVER computes the probability that s_1 and s_2 co-refer based on how frequently they participate in similar tuples. This method is shown to be effective for resolving synonymous strings in practice.

2.4. Related work

In contrast to the bulk of previous IE work, our focus is on unsupervised IE (UIE) where URNS substantially outperforms previous methods (Fig. 2).

In addition to the noisy-or models we compare against in our experiments, the IE literature contains a variety of heuristics using repetition as an indication of the veracity of extracted information. For example, Riloff and Jones [33] rank extractions by the number of distinct patterns generating them, plus a factor for the reliability of the patterns. Our work is intended to formalize these heuristic techniques, and unlike the noisy-or models, we explicitly model the distribution of the target and error sets (our $num(C)$ and $num(E)$), which is shown to be important for good performance in Section 2.2.1. The accuracy of the probability estimates produced by the heuristic and noisy-or methods is rarely evaluated explicitly in the IE literature, although most systems make implicit use of such estimates. For example, bootstrap-learning systems start with a set of seed instances of a given relation, which are used to identify extraction patterns for the relation; these patterns are in turn used to extract further instances (e.g. [33,25,1,30]). As this process iterates, random extraction errors result in overly general extraction patterns, leading the system to extract further erroneous instances. The more accurate estimates of extraction probabilities produced by URNS would help prevent this "concept drift."

Skounakis and Craven [37] develop a probabilistic model for combining evidence from multiple extractions in a supervised setting. Their problem formulation differs from ours, as they classify each occurrence of an extraction, and then use a binomial model along with the false positive and true positive rates of the classifier to obtain the probability that at least one occurrence is a true positive. Similar to the above approaches, they do not explicitly account for sample size n, nor do they model the distribution of target and error extractions.

Culotta and McCallum [10] provide a model for assessing the confidence of extracted information using conditional random fields (CRFs). Their work focuses on assigning accurate confidence values to individual occurrences of an extracted field based on textual features. This is complementary to our focus on *combining* confidence estimates from multiple occurrences of the same extracted label. In fact, each possible feature vector processed by the CRF in [10] can be thought of as a virtual urn m in our URNS. The confidence output of Culotta and McCallum's model could then be used to provide the precision p_m for the urn.

Our UIE task is related to previous work in automatically devising logical statements from text [24,36] and unsupervised semantic role labeling [41,21,32]. UIE is distinct in that the target output is a knowledge base of factual relations, rather than an interpretation of text in terms of logic or labeled semantic roles. Because our UIE approach operates over a large corpus, we do not attempt to identify *all* semantic assertions in the text corpus. Instead, we focus on only factual assertions that can be identified automatically at relatively high precision (using e.g. extraction patterns), and present methods for combining this evidence at Web-scale.

Our work is similar in spirit to BLOG, a language for specifying probability distributions over sets with unknown objects [28]. As in our work, BLOG models can express observations as draws from an unknown set of balls in an urn. Whereas BLOG is intended to be a general modeling framework for probabilistic first-order logic with varying sets of objects, our work is directed at modeling redundancy in IE. We also provide supervised and unsupervised learning methods for our model that are effective for data sets containing many thousands of examples, along with experiments demonstrating their efficacy in practice.

One of the problems our EM-based algorithm for learning URNS parameters must solve is estimating the parameter $|C|$, the size of the target set. This problem has commonalities with the classic "capture–recapture" problem from ecology, in which the goal is to estimate the size of an animal population by capturing and marking a sample of the population, then re-sampling at a later time [31]. There are a number of significant differences between the capture–recapture problem and estimating URNS parameters, however. First, URNS attempts to learn the parameter $|C|$ from observations which are co-mingled with samples from a confounding error distribution. Second, URNS must also characterize how the frequencies of the target set vary (in terms of the Zipfian shape parameter z_C). In order to overcome these additional parameter estimation difficulties, URNS exploits problem structures often found in textual domains, such as the fact that extraction frequencies tend to be Zipf distributed.

3. URNS: Theoretical results

The URNS model was shown in the previous section to be effective in practice for UIE and other applications. In this section, we analyze the URNS model theoretically. To better understand the behavior of URNS, we would like the be able to characterize how class probability increases with extraction count. Further, we would like a guarantee on URNS's accuracy given sufficient unlabeled data. How does accuracy increase with sample size? Can the parameters of the model be learned from unlabeled data in general?

Specifically, we investigate the following questions in the context of a single-urn model:

1. In the model, at what *rate* does the probability that an extracted label is of the target class increase with the number of extractions k?
2. What are sufficient conditions for accurate classification, given the parameters of the model? What sample size n is sufficient to achieve a given level of classification accuracy?
3. Can the parameters of the model be learned from unlabeled data?
4. Can the URNS model provide accurate classifications for extractions, i.e. is PAC-learnability guaranteed?

We begin by considering the first two questions in the uniform special case previously introduced in Section 2.0.1. The uniform case, while not fully realistic, does provides qualitatively interpretable results useful for illustration. We then address all four questions in the more realistic Zipfian model used in our experiments.

In the below, for notational convenience we will utilize in place of the multi-set $num(C)$ a multi-set F_C containing, for each element of C, the relative *fraction* of balls labeled with that element. We define F_E similarly, such that $\sum_{f \in F_C \cup F_E} f = 1$. Then the following expression (adapted from Eq. (1)) specifies the probability that x is an element of C given the observed values of k and n:

$$P(x \in C|k,n) = \frac{\sum_{f \in F_C} f^k (1-f)^{n-k}}{\sum_{f \in F_C \cup F_E} f^k (1-f)^{n-k}}.$$

We will also refer to the *classifier* output by URNS, which is a function from extracted labels to a binary value, indicating that URNS's probability is greater than 0.5 (positive) or less than 0.5 (negative).

3.1. Theoretical results: Known parameters

This section presents our theoretical results when the parameters of URNS are known. In the following, we examine URNS under two sets of assumptions, the Uniform Special Case (USC) and the Zipfian Case (ZC), defined below.

Theorems 3 and 5 address question (1) above in each model, describing how class probability increases with the number of times k a label is extracted. Specifically, we provide expressions for the increase in the odds ratio $odds(k,n) = P(x \in C|k,n)/(1 - P(x \in C|k,n))$ in terms of k. Theorems 4 and 7 address question (2). Let c_{known} indicate the classifier output by URNS when the parameters are known; we provide upper bounds on the expected error $E[error(c_{known})]$ in terms of the sample size n and the model parameters.

3.2. Analyzing the uniform special case

The *Uniform Special Case* (USC) of the URNS model, first introduced in Section 2.0.1, is characterized by the following assumptions:

USC1 Each target label has the same probability p_C of being selected in a single draw, and each error label has a corresponding probability p_E.
USC2 Each label from C is repeated on more balls in the urn than is each label from E (that is, $p_C > p_E$).
USC3 Frequency observations k are Poisson distributed (as in Eq. (2)).

3.2.1. Theoretical results in the USC

The following theorem states how the odds ratio $odds(k, n)$ increases with k in the USC.

Theorem 3. *In the USC*

$$\frac{odds(k_1, n)}{odds(k_2, n)} = \left(\frac{p_C}{p_E}\right)^{k_1 - k_2}. \tag{10}$$

Proof. Follows from the posterior probability in the USC (from Eq. (2)):

$$P(x \in C|k, n) = \frac{1}{1 + \frac{|E|}{|C|}(\frac{p_E}{p_C})^k e^{n(p_C - p_E)}}. \qquad \square \tag{11}$$

Along with assumption USC2, Theorem 3 illustrates that in the USC the odds that an element is a member of the target class increase exponentially with repetition. The increase is hastened when the target and error classes are less confusable (i.e. as p_C increases relative to p_E).

How accurately we can classify extracted labels, given the parameters of the model and the sample size? Let c_{known} indicate the URNS classifier when the parameters are known. The following theorem provides an upper bound on the error of c_{known} in the USC in terms of the sample size n, and the separability $p_C - p_E$ between the C and E sets.

Theorem 4. *In the USC, the expected error $E[error(c_{\text{known}})] < \epsilon$ when the sample size n satisfies:*

$$n \geqslant \frac{12 p_C \ln 1/\epsilon}{(p_C - p_E)^2}. \tag{12}$$

Proof. Define a model m with a threshold $\tau = \frac{p_C + p_E}{2}$ such that $P_m(x \in C|k, n) \geqslant 0.5$ whenever $k \geqslant n\tau$, and $P_m(x \in C|k, n) < 0.5$ otherwise. Since we can calculate the optimal threshold when the parameters are known, $E[error(c_{\text{known}})]$ is no worse than the expected error made by model m (which utilizes a potentially sub-optimal threshold). We express the expected error of model m over the full set $C \cup E$ by summing the expected contribution of each label (equal to the probability that the label appears a number of times resulting in misclassification).

$$E\big[error(c_{\text{known}})\big] = \frac{\sum_{x \in E} \sum_{k \geqslant n\tau} P(k|x \in E, n) + \sum_{x \in C} \sum_{k < n\tau} P(k|x \in C, n)}{|C \cup E|}. \tag{13}$$

Employing Chernoff bounds, we can bound the probability that a given label deviates from its expected frequency enough to be misclassified. The Chernoff bounds we employ state that for a random variable $X = \sum_i X_i$ equal to the sum of independent Bernoulli random variables X_i, the probability that X exceeds its expectation μ by more than a factor $(1 + \delta)$, for any $\delta > 0$, is bounded as:

$$P\big(X > (1 + \delta)\mu\big) < e^{-\mu\delta^2/3}. \tag{14}$$

Likewise, the probability that X is sufficiently less than its expectation is bounded as:

$$P\big(X < (1 - \delta)\mu\big) < e^{-\mu\delta^2/2} \tag{15}$$

for any $\delta > 0$.

Let $d = p_C - p_E$. Then we have:

$$\begin{aligned}
\sum_{k \geqslant n\tau} P(k|x \in E, n) &= \sum_{k \geqslant n(p_E + d/2)} P(k|x \in E, n) \\
&= P\big(k \geqslant n(p_E + d/2)|x \in E\big) \\
&\leqslant P\big(k > n(p_C + d/2)|x \in E\big) \\
&< e^{-nd^2/(12p_C)}
\end{aligned}$$

where the last inequality uses the Chernoff bound in Eq. (14) with $\mu = np_C$ and $\delta = d/(2p_C)$. Similarly, using the bound in Eq. (15), we have:

$$\begin{aligned}\sum_{k<n\tau} P(k|x\in C,n) &= \sum_{k<n(p_C-d/2)} P(k|x\in C,n)\\ &= P\big(k<n(p_C-d/2)|x\in C\big)\\ &< e^{-nd^2/(8p_C)}\\ &< e^{-nd^2/(12p_C)}.\end{aligned}$$

Algebra gives the final result. □

Theorem 4 yields the following corollary, which states that under the assumptions of the USC, even a weakly indicative extractor (one for which $p_C - p_E$ is just slightly greater than zero) can provide an arbitrarily accurate classifier, given sufficiently large n. This statement is akin to similar results in boosting algorithms in machine learning [35].

Corollary 1. *In the USC, for any $\epsilon > 0$, any extractor for which $p_C - p_E > 0$ can be used to achieve accuracy of $1-\epsilon$ given sufficient sample size n.*

3.3. Analyzing the Zipfian single-urn case

The USC is a reasonable approximation for labels on the flat tail of the Zipf curve, but it is clearly an oversimplification for all labels. The following theorems are analogous to those presented for the USC above, but employ the more realistic Zipfian single-urn assumptions. In particular, we assume that the target and error sets are governed by known Zipf distributions, described below, with sizes $|C|$ and $|E|$ and shape parameters z_C and z_E. Further, we assume draws are generated from a mixture of these Zipf distributions, governed by a known mixing parameter p giving the probability that a single draw comes from C:

$$p = \sum_{f\in F_C} f. \tag{16}$$

As in our experiments, we will find it more mathematically convenient to work with a continuous representation of the commonly discrete Zipfian distribution. Integrating over the continuous representation will allow us to arrive at closed-form expressions for class probability in terms of gamma functions (Theorem 5). In the discrete Zipfian case, it is assumed that the ith most frequent element of C has frequency α_C/i^{z_C}, for α_C a normalization constant. In our continuous representation, the frequency of each element of C is itself a random variable drawn by choosing a uniform x from the range $[1,|C|+1]$ and then mapping x to the curve $f_C(x) = \alpha_C/x^{z_C}$ to obtain a frequency. The normalization constant α_C is:

$$\alpha_C = \frac{p}{\int_1^{|C|+1} \frac{1}{x^{z_C}}\,dx}. \tag{17}$$

The normalization constant is chosen such that if we draw $|C|$ frequencies for the labels of the C set, the expected sum of the frequencies is p, as desired. The frequency of each element of E is defined analogously. We will refer to the functions f_C and f_E as *frequency curves*.

As in the USC, for a label in the ZC with underlying frequency f we assume the observed count k is Poisson distributed with expected value nf. Thus, the likelihood of observing an example of the set S (used to denote either of the C or E sets) a total of k times in n draws is:

$$P_{ZC}(k|x\in S,n) = \frac{1}{|S|}\int_1^{|S|+1} \frac{(n\alpha_S x^{-z_S})^k}{e^{n\alpha_S x^{-z_S}}k!}\,dx. \tag{18}$$

The solution of this equation in terms of incomplete gamma functions is given below in Theorem 5, Eq. (19).

We state the assumptions in the ZC as follows:

- ZC1 The distributions of labels from C and E are each Zipfian as defined above, with mixing parameter p. That is, the likelihood of the data is governed by Eq. (18).
- ZC2 Confidence increases with repetition; that is, $P(x\in C|k)$ increases monotonically with k.
- ZC3 The error label frequency curve has positive probability mass below the minimum target label frequency; that is $\alpha_E/i^{z_E} < \alpha_C/(|C|+1)^{z_E}$ for some known $i < |E|+1$.
- ZC4 Analogously, the target label frequency curve has positive probability mass above the maximum error label frequency; that is $\alpha_C/i^{z_C} > \alpha_E$ for some known $i > 1$.
- ZC5 Both the target and error set have non-zero probability mass in the urn; that is, $p, 1-p > M$ for some known lower bound $M > 0$.

Assumptions ZC3 and ZC4 encode an assumption that given a sufficient number of distinct labels in the urn, with high probability the most frequent labels will be target labels and the least frequent will be error labels. These assumptions will allow us to establish PAC learnability from unlabeled data alone.

To lend justification to the above assumptions, we note that we would expect them to hold at least approximately in Unsupervised Information Extraction applications. The Zipfian nature of extractions and monotonicity (ZC1 and ZC2) are well known to hold approximately in practice. Further, assumption ZC3 is certainly empirically true when one considers that, as a simple example, for any target set element there exist multiple less-frequent misspellings in the error set. Assumption ZC4 tends to be at least approximately true in practice: the most frequently extracted labels tend to be instances of the target class. Assumption ZC5 is nearly trivially true in practice, we would always expect the target and error sets to have probability mass above some non-zero minimum value.

3.3.1. Theoretical results in the ZC

We start by explicitly expressing how the odds that an element is a member of the target class increases with the number of repetitions:

Theorem 5. *In the ZC, the odds ratio*

$$\frac{odds(k+1,n)}{odds(k,n)} = \frac{(k-1/z_C)g(k,z_C,np,|C|+1,\alpha_C) + h(k-1/z_C, \frac{np}{(|C|+1)^{z_C}\alpha_C})}{(k-1/z_E)g(k,z_E,n(1-p),|E|+1,\alpha_E) + h(k-1/z_E, \frac{n(1-p)}{(|E|+1)^{z_E}\alpha_E})}$$

where

$$h(k',n') = n'^{k'}e^{n'}$$

and

$$P(k|x \in C,n) = \frac{np}{\alpha_C}^{1/z_C} g(k,z_C,np,|C|+1,\alpha_C) \tag{19}$$

with

$$g(k',z',n',s',\alpha') = \Gamma\left(k'-1/z', \frac{n'}{s'^{z'}\alpha'}\right) - \Gamma\left(k'-1/z', \frac{n'}{\alpha'}\right)$$

assuming that neither z_C nor z_E are exactly equal to 1.

Proof. Given that $|C|, |E|, k \geqslant 1$, and $z_C, z_E \neq 1$ the above result is obtained by symbolic integration in Mathematica and algebra.[14] □

Theorem 5 does not utilize any assumptions other than the Zipfian mixture (ZC1). Eq. (19) is the closed-form likelihood expression used to perform efficient inference in our experiments. Of course, the odds ratio given above is complex. An illustration of how class probability varies with k is shown in Fig. 5. In order to provide qualitative insights, the odds ratio should be simplified into a more interpretable bound; this is an item of future work.

We also wish to bound the classification error of URNS for the ZC. The following theorem provides a bound relative to the error of the *optimal classifier*, which utilizes both the URNS parameters *and* the precise frequencies of each label (rather than simply the observed counts). As such, the optimal classifier exhibits the best classification performance that can be achieved using the extraction count alone.

Definition 6. The **optimal classifier** is one which classifies each label optimally given knowledge of both the urn parameters, as well as the precise frequency in the urn of each label.

Define τ such that the classification threshold of the optimal classifier for a given n is equal to $n\tau$. From assumption ZC2, we know that a single such τ exists. Then the following theorem illustrates that as the sample size increases, the expected error falls off nearly linearly toward that of the optimal classifier.

Theorem 7. *In the ZC, given any $\delta > 0$, the expected error of urns is bounded as*:

$$E\left[error(c_{\text{known}})\right] \leqslant \beta + \frac{K_C(\delta) + K_E(\delta)}{(|C|+|E|)n^{1-\delta}}$$

where $K_C(\delta)$ and $K_E(\delta)$ are constants (with respect to n) defined below, and β is the expected error of the optimal classifier.

[14] For reference, the specific Mathematica commands involved in the proof are available online, see http://www.cs.northwestern.edu/~ddowney/data/urnsIntegration.html.

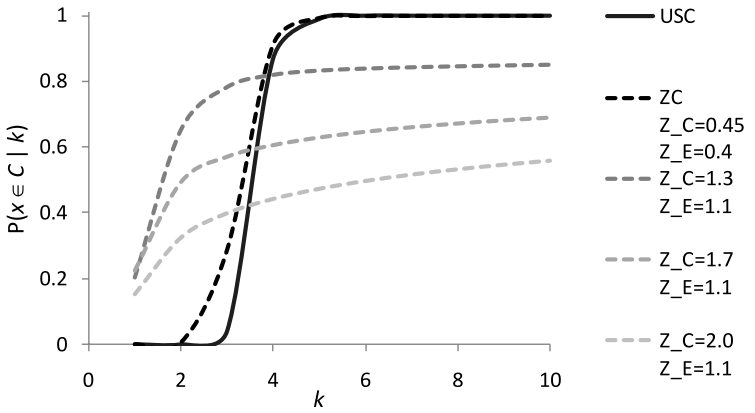

Fig. 5. Probabilities assigned in the Urns model, in the Uniform Special Case (USC), and Zipfian Case (ZC) as the Zipfian shape parameters vary. For very flat Zipf curves (z_C = 0.45, z_E = 0.4), ZC is similar to USC, but ZC differs from USC as the shape parameters increase and diverge from each other. A z_E value of 1.1 implies some errors have high extraction frequency, meaning that as k increases, class probability in the ZC converges to one more slowly than in the USC. In the above, $|E| = 20{,}000$, $|C| = 500$, $p = 0.9$, and $n = 10{,}000$.

The constants K_C and K_E are defined as follows (for S denoting C or E):

$$K_S(\delta) = \max\left(3/\delta, 1 + \int_1^{x_S^\tau - 1} \frac{1}{(\alpha_S x^{-z_S} - x_S^\tau)^2}\, dx\right) \tag{20}$$

where x_S^τ is defined to be the unique value such that $f_S(x_S^\tau) = \tau$, and α_S is the normalization constant (see Eq. (17)).

Proof. Following the proof of Theorem 4, we aggregate the probabilities that the elements are misclassified.

We present the analysis for expected error on elements of the C set; the E set is analogous. When the parameters are known, Urns makes errors the optimal classifier does not if and only if the true frequency of a target label x is greater than the threshold τ, but the observed count is less than $n\tau$. We bound the probability that an element with true frequency of $\alpha_C x^{-z_C} > \tau$ appears fewer than $n\tau$ times in n draws using Chebyshev's inequality. Chebyshev's inequality bounds the probability that a random variable Y with expectation μ and variance σ^2 appears sufficiently far from its expectation:

$$P\big(|Y - \mu| > r\sigma\big) \leqslant \frac{1}{r^2}.$$

For a Poisson random variable with expected value $p'n$ for $0 < p' < 1$, σ is bounded above by $\sqrt{n}$, so the above expression also bounds the probability that the deviation exceeds $r\sqrt{n}$. Setting $r\sqrt{n}$ equal to the smallest deviation resulting in misclassification ($n\alpha_C x^{-z_C} - nx_C^\tau$), and integrating over the frequency curve f_C, we have the following bound for the expected error on the C set:

$$E\big[error_C(c_{\text{known}})\big] \leqslant \beta_C + \int_1^{x_C^\tau} \min\left(1, \frac{1}{n(\alpha_C x^{-z_C} - x_C^\tau)^2}\right) dx \tag{21}$$

where β_C is the fraction of the expected error of the optimal classifier due to elements of C (namely, the probability mass of elements of C with frequency less than τ).

Define:

$$\gamma_n = \frac{1}{n} + \int_1^{x_C^\tau - 1/n} \frac{1}{n(\alpha_C x^{-z_C} - x_C^\tau)^2}\, dx \geqslant \int_1^{x_C^\tau} \min\left(1, \frac{1}{n(\alpha_C x^{-z_C} - x_C^\tau)^2}\right) dx.$$

We claim $\gamma_n \leqslant K_C(\delta)/n^{1-\delta}$, given which the theorem follows. The proof of the claim proceeds by induction. First, note that the $n = 1$ case, that $\gamma_1 \leqslant K_C(\delta)$, holds by construction of $K_C(\delta)$—the second term in the max function in Eq. (20) is equal to γ_1. Then assuming $\gamma_n \leqslant K_C(\delta)/n^{1-\delta}$, consider the $n+1$ case:

$$\gamma_{n+1} = \frac{n\gamma_n}{n+1} + \int_{x_C^\tau - 1/n}^{x_C^\tau - 1/(n+1)} \frac{1}{n(\alpha_C x^{-z_C} - x_C^\tau)^2}\, dx \leqslant \frac{K_C(\delta)n^\delta}{n+1} + \frac{1}{n^2} = \frac{K_C(\delta)}{(n+1)^{1-\delta}}\left(\frac{n^\delta}{(n+1)^\delta} + \frac{(n+1)^{1-\delta}}{K_C(\delta)n^2}\right).$$

It remains to show that:

$$\left(\frac{n^\delta}{(n+1)^\delta} + \frac{(n+1)^{1-\delta}}{K_C(\delta)n^2}\right) \leqslant 1. \tag{22}$$

With algebra, this is equivalent to the statement that $K_C(\delta)n^2((n+1)^\delta - n^\delta) \geqslant n+1$. From the generalized binomial theorem, $(n+1)^\delta$ is at least as large as $n^\delta + \delta n^{\delta-1} - \delta(1-\delta)n^{\delta-2}/2$. With algebra, we have:

$$K_C(\delta)n^2\big((n+1)^\delta - n^\delta\big) \geqslant \frac{K_C(\delta)\delta n^{1+\delta}}{2} \geqslant \frac{3n^{1+\delta}}{2} \geqslant n+1$$

as desired, using the fact that $K_C(\delta) \geqslant 3/\delta$. □

3.4. Theoretical results with unknown parameters

In unsupervised classification, in general we are not given the URNS parameters in advance, and must learn these from unlabeled data. In this section, we provide theorems bounding the error in unsupervised classification even when the parameter values are unknown. The following theorem shows that with high probability the parameter values of URNS can be estimated accurately from unlabeled data alone, as the total number of distinct labels in the urn $u = |C| + |E|$ increases, with n fixed.

Theorem 8. *In the ZC, for any $\delta, \epsilon > 0$, given sufficiently large $u = |C| + |E|$ for fixed n, we can obtain an estimate of the parameters of f_C and f_E such that with probability $1-\delta$ each estimate lies within ϵ of the true parameter value.*

Proof. The frequency curves f_C and f_E can be converted into functions $g_C(\lambda)$ and $g_E(\lambda)$ giving the probability density of a particular frequency λ for labels in the C (resp. E) set. These functions are themselves power law distributions. For example, in the error set case:

$$g_E(\lambda) = \begin{cases} \frac{L_E}{\lambda^{(1+z_E)/z_E}} & \text{for } a_E \leqslant x \leqslant b_E, \\ 0 & \text{for } x < a_E \text{ or } x > b_E, \end{cases} \tag{23}$$

for a suitable constant L_E where z_E indicates the exponent from the original frequency curve. The distribution of error labels in the model is completely characterized by four parameters: L_E and z_E, the minimal frequency a_E, and the maximal frequency b_E.

The probability that a particular label appears k times in n extractions can then be written as follows:

$$P(k|n) = \int_0^n \big(g_C(\lambda) + g_E(\lambda)\big)\frac{e^{-\lambda}\lambda^k}{k!}\,d\lambda. \tag{24}$$

Let $g(x) = g_C(x) + g_E(x)$. When written in the form of Eq. (24), the distribution over k becomes an instance of a *compound Poisson process*, for which the existence of effective estimators of $g(x)$ is well-known. In particular, Theorem 1 from [26] states that for any $x < n$ we can obtain a sequence of estimates $\hat{g}_u(x)$ of $g(x)$ such that $E[\hat{g}_u(x) - g(x)]^2 = o(1)$ as $u \to \infty$. Thus, for any given $\delta', \epsilon' > 0$, we have with probability $1-\delta'$ that $|\hat{g}_u(x) - g_(x)| < \epsilon'$ for u sufficiently large. It remains to convert this estimator of $g(x)$ into estimators of each of the URNS parameters. In the re-written model (Eq. (23)) we will employ, there are eight total parameters characterizing the mixture components g_C and g_E. We present the construction for the four parameters of g_E, the g_C case is analogous.

Consider two estimates $\hat{g}_u(x_0)$ and $\hat{g}_u(rx_0)$ where $x_0, rx_0 < \alpha_C/(|C|+1)$. That is, x_0 and rx_0 are sufficiently small that $g_C(x_0)$ and $g_C(rx_0)$ are zero by assumption ZC3. By algebra, in this region $(1+z_E)/z_E = (\ln g(x_0) - \ln g(rx_0))/(\ln r)$, so z_E is a continuous and bounded function of $g(x_0)$ and $g(rx_0)$ on the domain of interest. This implies we can estimate z_E within ϵ with probability $1-\delta$ given our estimator $\hat{g}_u$, for u suitably large. Likewise, L_E is a continuous and bounded function of $g(x)$ and z_E, so we can estimate L_E effectively.

It remains to obtain an estimator for the limits of support a_E and b_E. We begin with the minimal limit a_E. We construct from 0 to n a uniform lattice of estimates $\{\hat{g}_u(x_i)\}$ each ϵ apart. By assumption ZC5, $g_E(x) > M'$ for $x \in [a_E, a_E+\epsilon)$ for a known constant M' given that ϵ is sufficiently small. By taking u suitably large, we can ensure with probability $1-\delta$ that $\forall x_i < a_E$, $|\hat{g}_u(x_i) - g(x_i)| = |\hat{g}_u(x_i)| < M'/2$ and that the $x_j \geqslant a_E$ falling in the interval $[a_E, a_E+\epsilon)$ has estimate $\hat{g}_u(x_j) > M'/2$. Thus, the minimal x_i such that $\hat{g}_u(x) > M'/2$ is with probability $1-\delta$ an estimate within ϵ of a_E. Estimating the maximal limit of support b_E is similar. The same procedure is employed, except that because $g_C(b_E)$ is non-zero, we instead identify successive estimates $\hat{g}_u(x_k)$ and $\hat{g}_u(x_{k+1})$ that differ by a sufficiently large margin, where x_k is greater than our estimate for a_E. By taking u sufficiently large and ϵ sufficiently small, with probability $1-\delta$ the value x_k is within ϵ of b_E. □

3.4.1. PAC learnability under URNS

In this section, we show that a sufficiently informative extractor that follows the URNS model can be used to PAC learn from only unlabeled data. Here, we assume we have additional features for each label beyond just the extraction counts (for example, other features could include the co-occurrence counts of each label with textual contexts other than the extractors, as in [15]).

Our result is expressed in terms of a given, fixed concept class of binary classifiers mapping the input features to $\{0, 1\}$, denoted as $\mathcal{C}$—as is typical in the PAC-learning setting, we assume our target function (having zero error) is in $\mathcal{C}$.

Our result requires that a "separability" criterion holds on the concept class $\mathcal{C}$. This criterion states that no two distinct concepts in $\mathcal{C}$ agree on too large a fraction of the instance space:

Definition 9. A concept class $\mathcal{C}'$ is ϵ**-separable** if for any distinct concepts $c, c' \in \mathcal{C}'$, the fraction of examples $\mathbf{x} \in \mathcal{X}$ such that $c(\mathbf{x}) = c'(\mathbf{x})$ is less than $1 - \epsilon$.

We also require an extractor that is sufficiently informative. We state this criteria in terms of the minimal expected classification error that can be achieved using the extraction counts, in the limit of u and n large. This is equivalent to the area of the "confusion region" in Fig. 1, which we define formally as:

Definition 10. The **area of the confusion region** of an extractor is:

$$\min_{\tau}\left[\int_0^{\tau} g_C(\lambda)\,d\lambda + \int_{\tau}^{\infty} g_E(\lambda)\,d\lambda\right]. \tag{25}$$

Given this definition, we can state the following result, which shows that URNS is able to PAC learn from unlabeled data alone.

Proposition 11. *If $\mathcal{C}$ is ϵ-separable, given an extractor that follows the ZC with confusion region of area less than $1 - \epsilon/2$, $\mathcal{C}$ is PAC-learnable from unlabeled data alone.*

Proof. By Theorem 8, with high probability we can obtain the parameters of URNS within an error of ϵ', for any $\epsilon' > 0$. Because the optimal classification threshold τ is a continuous and bounded function of the URNS parameters (see Eq. (25)), URNS can achieve accuracy arbitrarily close to the confusion region size. Thus, the error of URNS is less than $1 - \epsilon/2$, given n and u sufficiently large, meaning it assigns classifications different from those of the target classifier on fewer than $1 - \epsilon/2$ of the examples. By the separability criterion, the target concept is the only hypothesis that differs from the output of URNS on so few examples. Thus, an algorithm that returns the concept $c \in \mathcal{C}$ most similar to the output of URNS will always return the target concept. □

3.5. Related work

Joachims provides theoretical results in supervised textual classification that use the Zipfian structure of text to arrive at error bounds for Support Vector Machine classifiers on textual data [23]. The strong performance of SVMs in our supervised experiments corroborate Joachims's claim that these classifiers are effective on textual data. However, in contrast to Joachims's work, our theoretical results (and experiments) are focused on the unsupervised case. We show that when the Zipfian structure holds, unsupervised learning is possible under certain assumptions.

Our result showing that PAC Learnability is guaranteed in the URNS model (Proposition 11) extends a previous result showing that a single "monotonic feature" is sufficient to PAC-learn under certain assumptions (a monotonic feature is one, like the extraction counts we consider, whose value increases monotonically with class probability) [13]. The primary advantage of our result is that it does not require that the extraction counts be conditionally independent of the other features given the class, a strong assumption which is shown to be problematic in practice in [12]. Our result avoids this assumption by exploiting problem structure inherent in extraction, as expressed by the URNS model.

4. Future work

The techniques described in this paper leave open many potential areas of future work. One important direction is developing a probabilistic model for multiple extractors that is more flexible than multiple urns. The correlation model used for multiple urns is limited and can only handle a small, pre-defined set of distinct mechanisms. Language modeling techniques for UIE from recent work leverage all contextual information when assessing extractions, rather than relying on a select set of extraction patterns [15,2]. However, currently these techniques only rank extracted labels, and do not output probabilities or classifications. A model that produces probabilities of correctness without labeled data, like URNS, yet also leverages all available contextual information is an important target of future work.

When utilizing URNS for UIE in practice, the EM-based algorithm we employ to learn URNS parameters from unlabeled data could be improved in a number of ways. The algorithm often requires a sample size of hundreds or thousands of unlabeled observations of each class in order to be effective (as illustrated in Fig. 3). For classes where data is less plentiful, such as many of the relations extracted by the TEXTRUNNER system, the parameter learning algorithm is less effective. We expect that URNS could be modified to learn accurate parameters for much smaller data sets, through the use of priors or more robust likelihood-maximization techniques.

URNS also requires that a reasonable estimate of the precision of the extraction process be known. We demonstrated that this requirement is not prohibitive when extracting instances of classes drawn from WordNet, using generic extraction patterns; the extraction frequency can be assumed or adjusted from unlabeled text in such a way that the probabilities produced by URNS still offer large improvements over previous techniques. However, for "Open IE" systems such as TEXTRUNNER which discover target relations from text, the situation is more complex [3]. In TEXTRUNNER, extraction precision can vary greatly across the discovered relations; thus, the probabilities output by URNS in this case are less accurate. Automatically estimating extraction precision across relations in Open IE systems is an area of future work.

5. Conclusions

This paper described methods for identifying correct extractions in UIE, *without* the use of hand-labeled training data. The URNS model estimates the probability that an extraction is correct, based on sample size, redundancy, and corroboration from multiple distinct extraction rules. We described supervised and unsupervised methods for estimating the parameters of the model from data, and reported on experiments showing that URNS massively outperforms previous methods in the unsupervised case, and is slightly better than baseline methods in the supervised case. We also detailed several other applications in which the general URNS model of redundancy has been effective. Our theoretical results show how the accuracy of URNS improves with sample size, that the parameters of URNS can be estimated without hand-labeled data, and that URNS guarantees PAC-learnability from unlabeled data alone, given certain conditions.

Acknowledgements

This research was supported in part by NSF grants IIS-0535284 and IIS-0312988, DARPA contract NBCHD030010, ONR grants N00014-02-1-0324 and N00014-08-1-0431, and gifts from Google, and carried out at the University of Washington's Turing Center. The first author was supported by a Microsoft Research Graduate Fellowship sponsored by Microsoft Live Labs. Google generously allowed us to issue a large number of queries to their XML API to facilitate our experiments. We thank Pedro Domingos, Anna Karlin, Marina Meila, and Dan Weld for helpful discussions, and Jeff Bigham for comments on previous drafts. Also, thanks to Alex Yates for suggesting we consider this problem.

References

[1] E. Agichtein, L. Gravano, Snowball: Extracting relations from large plain-text collections, in: Proc. of the Fifth ACM International Conference on Digital Libraries, 2000.
[2] A. Ahuja, D. Downey, Improved extraction assessment through better language models, in: Human Language Technologies: Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL HLT), 2010.
[3] M. Banko, M. Cafarella, S. Soderland, M. Broadhead, O. Etzioni, Open information extraction from the Web, in: Proc. of IJCAI, 2007.
[4] M. Banko, O. Etzioni, The tradeoffs between traditional and open relation extraction, in: Proceedings of ACL, 2008.
[5] A. Blum, T. Mitchell, Combining labeled and unlabeled data with co-training, in: COLT: Proceedings of the Workshop on Computational Learning Theory, Morgan Kaufmann Publishers, 1998, pp. 92–100.
[6] M. Cafarella, D. Downey, S. Soderland, O. Etzioni, Knowitnow: Fast, scalable information extraction from the Web, in: Proc. of EMNLP, 2005.
[7] M. Califf, R. Mooney, Relational learning of pattern-match rules for information extraction, in: Working Notes of AAAI Spring Symposium on Applying Machine Learning to Discourse Processing, AAAI Press, Menlo Park, CA, 1998, pp. 6–11.
[8] C. Chang, C. Lin, LIBSVM: a library for support vector machines, 2001.
[9] F. Ciravegna, Adaptive information extraction from text by rule induction and generalisation, in: Proc. of the 17th International Joint Conference on Artificial Intelligence (IJCAI 2001), Seattle, Washington, 2001, pp. 1251–1256.
[10] A. Culotta, A. McCallum, Confidence estimation for information extraction, in: HLT-NAACL, 2004.
[11] M.-C. de Marneffe, A. Rafferty, C.D. Manning, Finding contradictions in text, in: ACL 2008, 2008.
[12] D. Downey, Redundancy in Web-scale information extraction: probabilistic model and experimental results, PhD thesis, University of Washington, 2008.
[13] D. Downey, O. Etzioni, Look ma, no hands: Analyzing the monotonic feature abstraction for text classification, in: Advances in Neural Information Processing Systems (NIPS) 21, 2008, January 2009.
[14] D. Downey, O. Etzioni, S. Soderland, A probabilistic model of redundancy in information extraction, in: Proc. of IJCAI, 2005.
[15] D. Downey, S. Schoenmackers, O. Etzioni, Sparse information extraction: Unsupervised language models to the rescue, in: Proc. of ACL, 2007.
[16] O. Etzioni, M. Cafarella, D. Downey, S. Kok, A. Popescu, T. Shaked, S. Soderland, D. Weld, A. Yates, Web-scale information extraction in KnowItAll, in: WWW, New York City, New York, 2004, pp. 100–110.
[17] O. Etzioni, M. Cafarella, D. Downey, S. Kok, A. Popescu, T. Shaked, S. Soderland, D. Weld, A. Yates, Unsupervised named-entity extraction from the Web: An experimental study, Artificial Intelligence 165 (1) (2005) 91–134.
[18] O. Etzioni, M. Cafarella, D. Downey, A. Popescu, T. Shaked, S. Soderland, D. Weld, A. Yates, Methods for domain-independent information extraction from the Web: An experimental comparison, in: Proc. of the 19th National Conference on Artificial Intelligence (AAAI-04), San Jose, California, 2004, pp. 391–398.
[19] D. Freitag, A. McCallum, Information extraction with HMMs and shrinkage, in: Proceedings of the AAAI-99 Workshop on Machine Learning for Information Extraction, Orlando, Florida, 1999.
[20] W.A. Gale, G. Sampson, Good–Turing frequency estimation without tears, J. Quantitative Linguistics 2 (3) (1995) 217–237.

[21] T. Grenager, C.D. Manning, Unsupervised discovery of a statistical verb lexicon, in: Conference on Empirical Methods in Natural Language Processing.
[22] M. Hearst, Automatic acquisition of hyponyms from large text corpora, in: Proc. of the 14th International Conference on Computational Linguistics, Nantes, France, 1992, pp. 539–545.
[23] T. Joachims, Learning to Classify Text Using Support Vector Machines: Methods, Theory and Algorithms, Kluwer Academic Publishers, Norwell, MA, USA, 2002.
[24] M. Liakata, S. Pulman, From trees to predicate-argument structures, in: Proceedings of the 19th International Conference on Computational Linguistics, Association for Computational Linguistics, Morristown, NJ, USA, 2002, pp. 1–7.
[25] W. Lin, R. Yangarber, R. Grishman, Bootstrapped learning of semantic classes from positive and negative examples, in: Proc. of ICML-2003 Workshop on The Continuum from Labeled to Unlabeled Data, Washington, DC, 2003, pp. 103–111.
[26] W.-L. Loh, Estimating the mixing density of a mixture of power series distributions, in: S.S. Gupta, J.O. Berger (Eds.), Statist. Decision Theory and Related Topics V, Springer, New York, 1993, pp. 87–98.
[27] A. McCallum, Efficiently inducing features of conditional random fields, in: Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence, Acapulco, Mexico, 2003, pp. 403–410.
[28] B. Milch, B. Marthi, S. Russell, D. Sontag, D.L. Ong, A. Kolobov, Blog: Probabilistic models with unknown objects, in: L.D. Raedt, T. Dietterich, L. Getoor, S.H. Muggleton (Eds.), Probabilistic, Logical and Relational Learning – Towards a Synthesis, in: Dagstuhl Seminar Proceedings, vol. 05051, Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany, 2006, http://drops.dagstuhl.de/opus/volltexte/2006/416 [date of citation: 2006-01-01].
[29] K. Nigam, J. Lafferty, A. McCallum, Using maximum entropy for text classification, in: Proc. of IJCAI-99 Workshop on Machine Learning for Information Filtering, Stockholm, Sweden, 1999, pp. 61–67.
[30] M. Pasca, D. Lin, J. Bigham, A. Lifchits, A. Jain, Organizing and searching the world wide web of facts – step one: The one-million fact extraction challenge, in: AAAI 2006, AAAI Press, 2006.
[31] K.H. Pollock, J.D. Nichols, C. Brownie, J.E. Hines, Statistical Inference for Capture–Recapture Experiments, Wildlife Society Monogr., vol. 107, 1990.
[32] S.D. Richardson, W.B. Dolan, L. Vanderwende, Mindnet: acquiring and structuring semantic information from text, in: Proceedings of the 17th International Conference on Computational Linguistics, Association for Computational Linguistics, Morristown, NJ, USA, 1998, pp. 1098–1102.
[33] E. Riloff, R. Jones, Learning dictionaries for information extraction by multi-level bootstrapping, in: AAAI/IAAI, 1999.
[34] A. Ritter, S. Soderland, D. Downey, O. Etzioni, It's a contradiction – no, it's not: A case study using functional relations, in: EMNLP, 2008.
[35] R.E. Schapire, The strength of weak learnability, Mach. Learn. 5 (2) (1990) 197–227.
[36] L. Schubert, Can we derive general world knowledge from texts?, in: Proceedings of the Second International Conference on Human Language Technology Research, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2002, pp. 94–97.
[37] M. Skounakis, M. Craven, Evidence combination in biomedical natural-language processing, in: BIOKDD, 2003.
[38] S. Soderland, Learning information extraction rules for semi-structured and free text, Mach. Learn. 34 (1–3) (1999) 233–272.
[39] S. Soderland, O. Etzioni, T. Shaked, D. Weld, The use of Web-based statistics to validate information extraction, in: AAAI-04 Workshop on Adaptive Text Extraction and Mining, 2004, pp. 21–26.
[40] R. Storn, K. Price, Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (4) (1997) 341–359.
[41] R.S. Swier, S. Stevenson, Unsupervised semantic role labelling, in: Proceedings on EMNLP, 2004, pp. 95–102.
[42] A. Yates, O. Etzioni, Unsupervised resolution of objects and relations on the Web, in: Proc. of HLT, 2007.