# Diagnosing and Minimizing Semantic Drift in Iterative Bootstrapping Extraction

Zhixu Li, Ying He, Binbin Gu, An Liu, Hongsong Li, Haixun Wang, *Fellow, IEEE*, and Xiaofang Zhou, *Fellow, IEEE*

**Abstract**—Semantic drift is a common problem in iterative information extraction. Previous approaches for minimizing semantic drift may incur substantial loss in recall. We observe that most semantic drifts are introduced by a small number of questionable extractions in the earlier rounds of iterations. These extractions subsequently introduce a large number of questionable results, which lead to the semantic drift phenomenon. We call these questionable extractions Drifting Points (DPs). If erroneous extractions are the "symptoms" of semantic drift, then DPs are the "causes" of semantic drift. In this paper, we propose a method to minimize semantic drift by identifying the DPs and removing the effect introduced by the DPs. We use isA (concept-instance) extraction as an example to describe our approach in cleaning information extraction errors caused by semantic drift, but we perform experiments on different relation extraction processes on three large real data extraction collections. The experimental results show that our DP cleaning method enables us to clean around 90 percent incorrect instances or patterns with about 90 percent precision, which outperforms the previous approaches we compare with.

**Index Terms**—Information extraction, semantic drift, drifting point

## 1 Introduction

Iterative bootstrapping is used extensively in Information Extraction (IE) [17], [25], [31], [34]. Starting with a small number of seed instances in a target semantic class, it iteratively adds new instances selected by a model. The method is attractive because it requires minimal supervision; it is efficient for even tera-scale extraction [17]; and it is domain and language independent [13]. Indeed, bootstrapping approaches have shown good performance in many web-scale IE tasks [1], [30].

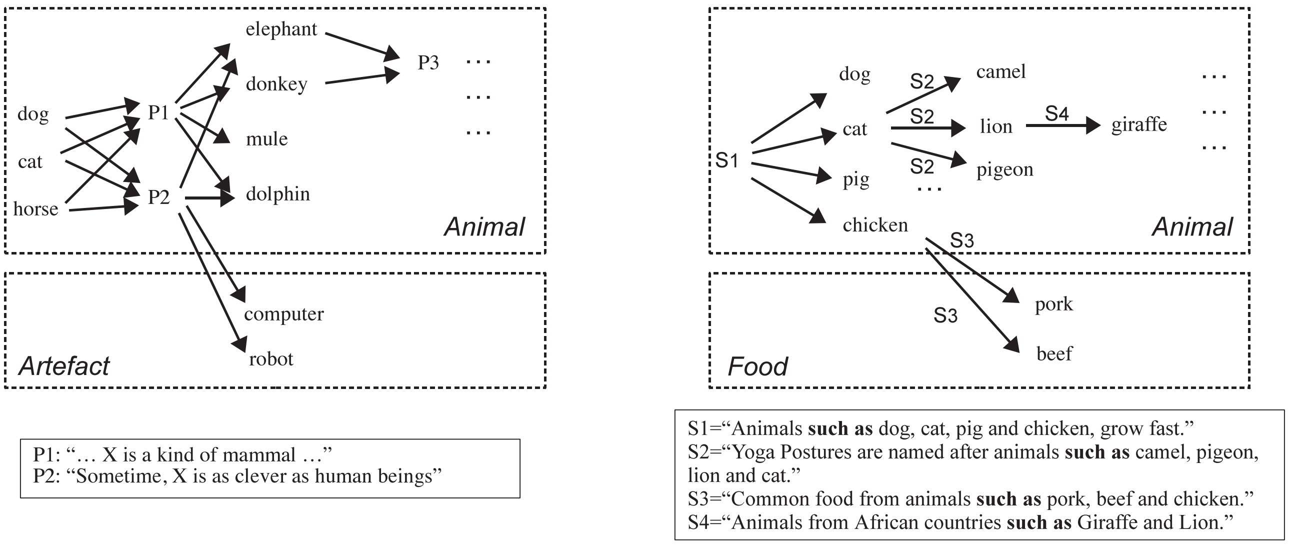
One of the biggest issues of iterative information extraction is *semantic drift* [5], [20]: As iterations proceed, the extractions may shift from the target class to some other classes [5]. State-of-the-art iterative IE methods can be divided into two categories, syntax-based and semantic-based, both of which have the semantic drift problem.

*Syntax-based Extraction:* Most iterative IE systems, including KnowItAll [7], Snowball [1], TextRunner [6], and NELL [3], are *syntax-based*, that is, each iteration finds additional syntactic patterns that can be used for information extraction. In other words, they rely on more syntactic patterns to produce more results. As depicted in Fig. 1a, given seeds such as *dog* and *cat* in the "Animal" class (or so called concept), we may discover a syntactic pattern $P_1 =$ "... X is a mammal ...", which enables us to find other animals such as *elephant* and *dolphin*. However, it may also produce syntactic patterns such as $P_2 =$ "Sometimes, X is as clever as human beings", which is error prone. It may produce extractions such as *computer* or *robot*, which in turn, will provide more irrelevant syntactic patterns. Eventually, the extraction for the "Animal" concept drifts to some other concepts.

*Semantic-based Extraction:* Some recent work [30] proposed a *semantic-based* iterative mechanism. For a given syntactic pattern, it performs multiple semantic iterations. Each iteration extracts new results to be added into a knowledge base, which enables it to understand more text and produce more extractions in the next iteration. Let us consider the syntactic pattern **such as** for extracting isA relationships. In the first iteration, the system has no knowledge, and only sentences that match the pattern without any ambiguity are used for extraction. For example, given a sentence $S_1 =$ "Animals **such as** *dog, cat, pig and chicken* ...", we extract *dog, cat, pig, chicken* as instances of "Animal". In the next iteration, (dog isA Animal) becomes part of our knowledge and may enable us to understand sentences that we were not able to understand in the previous iteration. For instance, given another sentence $S_4 =$ "Animals from African countries, **such as** *Giraffe and Lion*", where both "Animals" and "African countries" are candidate concepts, "Giraffe" and "Lion" are candidate instances. Since we know (Lion isA Animal), we may decide that in sentence $S_4$, **such as** modifies Animals rather than African countries. This knowledge

- Z. Li is with the School of Computer Science and Technology, Soochow University, Jiangsu Sheng 215000, China, and the Guangdong Key Laboratory of Big Data Analysis and Processing, Guangzhou 510006, P. R. China. E-mail: zhixuli@suda.edu.cn.
- Y. He, B. Gu, and A. Liu are with the School of Computer Science and Technology, Soochow University, Jiangsu Sheng 215000, China. E-mail: anliu@suda.edu.cn, {ying.hedy, gu.binbin}@hotmail.com.
- H. Li and H. Wang are with Microsoft Research Asia, Beijing 100080, China. E-mail: hongsli@microsoft.com, haixun@gmail.com.
- X. Zhou is with the School of Information Technology and Electrical Engineering, The University of Queensland, Brisbane QLD 4072, Australia, and the School of Computer Science and Technology, Soochow University, Jiangsu Sheng 215000, China. E-mail: zxf@itee.uq.edu.au.

*Manuscript received 8 June 2017; revised 3 Nov. 2017; accepted 3 Dec. 2017. Date of publication 13 Dec. 2017; date of current version 30 Mar. 2018. (Corresponding author: An Liu.)
Recommended for acceptance by L. Dong.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TKDE.2017.2782697*



P1: "… X is a kind of mammal …"
P2: "Sometime, X is as clever as human beings"

S1="Animals **such as** dog, cat, pig and chicken, grow fast."
S2="Yoga Postures are named after animals **such as** camel, pigeon, lion and cat."
S3="Common food from animals **such as** pork, beef and chicken."
S4="Animals from African countries **such as** Giraffe and Lion."

(a) "syntax-based" bootstrapping mechanism

(b) "semantic-based" bootstrapping mechanism

Fig. 1. Snapshots of iterative extraction for "Animal" with two different bootstrapping mechanisms.

enables us to obtain new knowledge (Giraffe isA Animal) instead of (Giraffe isA African country).

However, semantic-based extraction as described above is not immune to semantic drift. Consider the sentence $S_3$ = "Common food from animals **such as** *pork*, *beef*, and *chicken*" in Fig. 1b. Assume we already know (chicken isA animal), we may parse the sentence incorrectly to get wrong knowledge (pork isA animal) and (beef isA animal). Then the wrong knowledge probably will introduce more errors in the next iterations. Eventually, the extraction for the "Animal" concept drifts to other concept "Food".

There is already some work on reducing semantic drift. For example, Type Checking [17] checks whether the type of extracted instances matches the target class, and Mutual Exclusion [5] detects errors if extracted instances belong to mutually exclusive classes, such as "Animal" and "Artefact.". However, such constraints only tackle a small percentage of semantic drifts. Other methods [4], [20], [23] keep the most reliable instances in each iteration to maintain high precision, where an instance's "reliability" is determined either by some heuristic models (e.g., an instance is more reliable if it is extracted more frequently) [20], or by combining evidences from multiple extractions [4], [23]. Not surprisingly, these methods sacrifice recall for increased precision.

In this paper, we present a novel approach to overcome semantic drift. We consider semantic drift to be *triggered*[1] by certain patterns or instances that we call *Drifting Points (DPs)*. The DPs themselves are not necessarily erroneous extractions, rather, they trigger semantic drift from the target class to some other classes such as the pattern $P_2$ in Fig. 1a and the instance "chicken" in Fig. 1b. In that sense, erroneous extractions are the "symptoms" of semantic drift, while DPs are the "causes" of semantic drift. Identifying DPs enables us to cut off the propagation of semantic drift. Compared with detecting each erroneous extraction directly, focusing on DPs makes the problem much easier, as DPs are easier to model for two reasons: First, the number of DPs is much smaller than that of erroneous extractions, as one DP may introduce many errors. Second, there are various kinds of erroneous extractions [4], [5], [17], [23], which are hard to be captured by a single approach or model. In contrast, we identify two types of DPs that hold some strong features, which enable us to identify DPs and eventually identify erroneous extractions more effectively.

Overcoming semantic drift through DPs raises some nontrivial challenges. First, we must reach a high *precision* and *recall* in identifying DPs, as one incorrectly identified DP may let us take a number of correct extractions as erroneous extractions (false-negatives), while one missing DP may let us miss to identify a number of erroneous extractions (false-positives). Second, we need to clearly figure out the relationship between DPs and erroneous extractions, such that we can decide how to identify erroneous extractions with identified DPs. Note that not all the instances introduced by a DP are erroneous extractions. Take the DP instance "chicken" for example, though it introduces erroneous extractions like "pork" and "beef", it may also introduce correct extractions such as "duck" from the sentence "... animal **such as** *duck and chicken*". Hence, after DPs are detected, we should recognize which extractions introduced by a DP are erroneous ones.

In order for detecting DPs precisely (Challenge 1), we define two types of DPs according to their different characteristics and impact on semantic drifts. In particular, the first type of DPs are usually polysemous instances such as "chicken", which are not erroneous extractions by themselves, but some of their introduced extractions might be erroneous ones. The second type of DPs are erroneous extractions by themselves and can only introduce erroneous extractions. Based on this categorization on DPs, we could identify some important features for differentiating the two types of DPs from non-DPs. For instance, the difference between the *frequency distributions* of instances introduced by a first-type DP, a second-type DP and a non-DP can be quite obvious. A non-DP probably usually introduces high-frequency instances only, given that these instances are correct ones which are also introduced by many other instances. In contrast, a

1. In this context, we say an existing knowledge, i.e., an instance or a pattern under a target class, *triggers* the extraction of some other instances under the same class, if it enables us to understand a sentence, thereby generating new instances from the sentence.

second-type DP only introduces low-frequency instances which do not belong to the class, while a first-type DP probably introduces both high-frequency instances and low-frequency instances. However, this is not a definite property, as non-DPs sometimes may also introduce low-frequency instances which are correct ones but rarely introduced by other instances. Therefore, it is inaccurate to detect DPs with any heuristic functions developed from this single property.

The main difficulty in Challenge 2 lies on how we detect the erroneous extractions from those introduced by the first-type of DPs. Note that only a part of instances introduced by a first-type DP are erroneous ones, but there are no reliable evidence can be leveraged to answer whether an instance introduced by a first-type of DP is a correct one or not. In that case, we propose to check the correctness of these instances at sentence extraction level. In particular, for each sentence that is introduced by a first-type DP, we check all possible extractions to this sentence, and score each of them using a probabilistic model based on the generated instances obtained from the extraction. We take the extraction with the highest score as the correct extraction to this sentence, while those instances obtained by incorrect extractions to this sentence will be rolled back.

In summary, our main contributions are the following:

- We propose a novel method to overcome semantic drift by identifying the cause of the semantic drift: the drifting points (DPs). Comparing previous approaches that focus on the phenomenon of semantic drifts, our approach achieves higher precision and recall as we are able to cut off the propagation of semantic drift.
- We use semi-supervised, multi-task learning based on a small number of automatically labeled training data. This method not only leverages unlabeled data for a better understanding of new data, but also exploits the knowledge in related concepts to improve the classifier learning for each concept. This enables us to detect DPs in millions of concepts.
- We design DP-based cleaning strategy to identify and roll back incorrect extractions introduced by different kinds of DPs. We thus effectively cut off the propagation of errors in the iterative extraction process to clean a very large proportion of semantic drift errors.

We perform experiments on three large real data extraction collections. The results show that our DP cleaning method enables us to clean around 90 percent incorrect instances or patterns with about 90 percent precision, which outperforms all the other approaches in comparison.

*Roadmap.* We define Drifting Points (DPs) in Section 2, and then present how we detect DPs in Section 3, We introduce how we roll back error extractions activated by DPs in Section 4. We report our experiment results in Section 5. After covering related work in Section 6, we conclude in Section 7.

## 2 Overview on Drifting Points (DPs)

In either semantic-based or syntax-based bootstrapped relation extraction process, we say *Semantic Drift* happens to a target class if the extractions shift from the target one to some other irrelevant classes. In particular, we define the erroneous extractions happen with semantic drift as *Drifting Errors* below:

**Definition 1 (Drifting Errors).** *We call an instance or a pattern as a* Drifting Error *w.r.t. a target class if it does not belong to the target class but some other classes irrelevant to the target one.*

Note that not all erroneous extractions are drifting errors. For example, erroneous extractions caused by typos like *Syngapore*, and *Micorsoft* are not drifting errors. In other words, drifting errors are those caused by semantic misunderstanding. However, according to our observations to the large data set we employ in our experiments, more than 90 percent erroneous extractions are drifting errors.

We define instances that trigger drifting errors to a target class as *Drifting Points (DPs)* below:

**Definition 2 (Dirfting Points (DPs)).** *We say an instance or a pattern is a Drifting Point w.r.t. a target class if it introduces drifting errors to this target class.*

According to our observations, there are generally two types of DPs which have different characteristics and different impact on semantic drifts, and thus are necessary to be treated differently.

The first type of DPs are polysemous instances/patterns. They are not drifting errors themselves but part of the instances triggered by them are drifting errors, such as *chicken* w.r.t. the "Animal" class as we show in Fig. 1a. We formally define this kind of DPs as *Intentional DPs* below:

**Definition 3.** *An* Intentional DP *is a polysemous instance/pattern that belongs to both the target class and another irrelevant class. Hence, it tends to introduce instances from the irrelevant class into the target class as drifting errors.*

The second type of DPs result from mistakes that happen accidentally. There are two situations in which this kind of DPs may occur. First, they might come from incorrectly parsed sentences. For example, the following sentence

"... <u>animals</u> other than *dogs* **such as** *cats* ..."

may be parsed incorrectly and produce (cat isA dog). Thus *cat* becomes a drifting error accidentally in the "dog" class as it will very likely introduce more drifting errors into the "dog" class. Second, although the sentence is parsed correctly, the knowledge in the sentence is incorrect. For example,

"He has toured in various <u>countries</u> **such as** *France, Portugal*, *New York*, *Mauritius*, *Norway* and *Japan*...",

contains a wrong fact (New York isA country). Thus *New York* might become a drifting error accidentally in the "Country" class. Thus, we formally define this kind of DPs as *Accidental DPs* below:

**Definition 4.** *An* Accidental DP *is a drifting error itself. It happens accidentally or randomly and brings in drifting errors in the subsequent extraction iterations.*

In the rest of this paper, we firs work on detecting the two types of DPs (Section 3), and then identifying drifting errors introduced by them (Section 4). For easier presentation, we use semantic-based relation extraction as an example scenario to present our approach in the following two sections, and we will explain the different settings in the syntax-based relation extraction scenario in the experiments section.

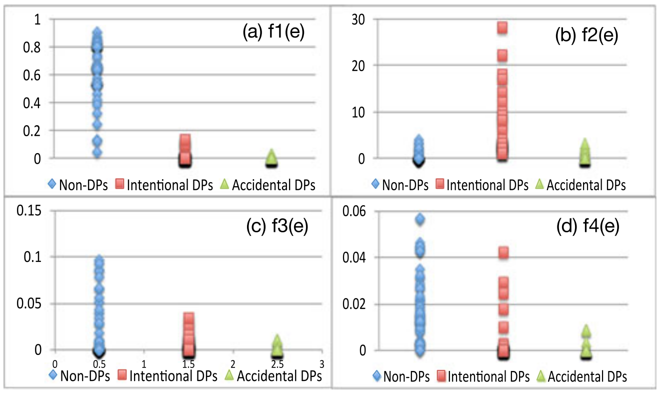


Fig. 2. Feature values of Intentional DPs, Accidental DPs, and non-DPs under the "Animal" concept.

## 3 Drifting Points (DPs) Detection

It is a nontrivial task to detect DPs from the extraction process. Although some properties of DPs could be identified (as listed in Appendix A, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109//TKDE.2017.2782697.) to provide some clues to detecting DPs, none of them is definite. Thus we would like to introduce machine learning approaches to solve the problem. Particularly, we resort to supervised learning methods for detecting DPs. For each concept, a *DP Detector* is trained to classify instances into three categories: (1) Intentional DPs, (2) Accidental DPs, or (3) non-DPs.

One drawback of supervised learning is that it requires large amount of training data. To make things worse, we have millions of concepts, among which many long-tail concepts do not have much training data. As a result, simple supervised machine learning tools which rely heavily on a large enough high-quality training data set for every concept do not apply in our case.

In this work, we propose a semi-supervised multi-task learning algorithm, which only needs a small set of labeled instances and can increasingly involves unlabeled instances to improve the model. Besides, we find ways to generate a small training data set automatically according to the definitions of DPs. Particularly, our semi-supervised learning algorithm could leverage unlabeled data for better understanding of the new data, and we use multi-task learning to improve our understanding of a certain concept by exploiting its related concepts.

In the following, we start with feature selection, and then describe how to obtain a set of seed DPs and non-DPs. Finally we present how we learn the DP detectors.

### 3.1 Designing Features

In the DP detector of the target concept $C$, each instance $e$ is represented by a feature vector $x(e) = [f_1(e), f_2(e), \dots, f_d(e)]^T \in \mathbb{R}^d$, where $d$ is the number of features. Several important features are considered as introduced below. To illustrate the effect of the four features, we depict the distribution of feature values of 1,097 manually labeled Intentional DPs, Accidental DPs and non-DPs under the "Animal" concept in our experimental data set in Fig. 2.

*(1) Feature 1.* The first feature explores the frequency distribution of instances triggered by an instance $e$. We take the similarity between the frequency distribution of instances triggered by $e$ and the frequency distribution of target concept $C'$s instances obtained in the first iteration as a feature, which is sufficient to distinguish many DPs and non-DPs. More specifically,

$$f_1(e) = Cosine(\vec{F}(sub(e)), \vec{F}(E(C,1))), \tag{1}$$

where $sub(e)$ denotes the set of instances triggered by $e$ (which will also be called as *Sub-instances* of $e$), and $E(C,i)$ is the set of instances that have already been learned under a given concept $C$ after $i$th iteration. Here $\vec{F}(X)$ is the frequency distribution of a set of instances $X$, and $Cosine(\vec{X},\vec{Y})$ measures the cosine similarity between the two vectors $\vec{X}$ and $\vec{Y}$ after they are mapped into the same space.

*(2) Feature 2.* The second feature is the number of $C'$s mutually exclusive concepts that also obtain $e$ as their instance,

$$f_2(e) = |\{C'|e \in E(C'), C' \perp C\}|, \tag{2}$$

where $E(C)$ is all of the learned instances under $C$, and $C' \perp C$ indicates concept $C'$ and $C$ are mutually exclusive.

*(3) Feature 3.* The third feature investigates the probability of each instance being correct, which requires a proper ranking model. Among the existing ranking models, Random walk [26] has been shown to be a better model than others including Bayesian Sets [11] and PageRank [16] in ranking instances extracted in a syntax-based iterative extraction [28], thus we adopt random walk based ranking model here. Particularly, we build a random walk graph for each target class, where each instance under the class is taken as a node, and each sentence parsing be represented as edges pointing from an instance to its triggered sub-instances between nodes. The random walk score of an instance $e$ is the probability that we could randomly walk from the instances obtained in the first iterations to the node of the instance $e$. More specifically, we have

$$f_3(e) = score(e) = RW\ Score(e). \tag{3}$$

*(4) Feature 4.* The forth feature mainly concerns about the quality of sub-instances triggered by $e$, which could be reflected by the average score of the sub-instances triggered by $e$. That is,

$$f_4(e) = AVG(score(sub(e))). \tag{4}$$

*(5) Other Potential Features.* Besides the four features given above, we can also involve a number of potentially useful features into the feature vector. First, the number of sub-instances triggered by $e$. Second, the standard deviation of score of sub-instances of $e$ which estimates the dispersion exists from the average. Third, the ordinal number of the iteration that the instance was first discovered as the instance of $C$. Forth, the number of all instances under the target concept $C$. Fifth, the total number of iterations in extracting instances for $C$. More specifically,

$$\begin{aligned} f_5(e) &= |sub(e)| \\ f_6(e) &= StandDev(score(sub(e))) \\ f_7(e) &= level(e) \\ f_8(e) &= |E(C)| \\ f_9(e) &= MaxLevel(C), \end{aligned} \tag{5}$$

where $level(e)$ stands for the ordinal number of the iteration that the instance was first discovered, $|E(C)|$ is the number of $C$'s instances, and $MaxLevel(c)$ returns the number of iterations used to extract instances for $C$.

## 3.2 Preparing Training Set

We do not have labeled data for DPs or non-DPs, although evidenced correct isA pairs can be obtained from verified sources (such as Wikipedia), or from highly frequent extracted pairs in the first iteration [30]. In this section, we define some heuristic rules based on the definitions of DPs and the mutually exclusive assumption [5] to label a number of obvious Intentional DPs, Accidental DPs and non-DPs.

We discuss on how we define evidenced correct/incorrect instances by using the mutually exclusive assumption with millions of concepts in Appendix B, available in the online supplemental material, and then introduce the three heuristic rules for labeling obvious DPs and non-DPs below.

**Rule 1.** We label $e$ of concept $C$ as an Intentional DP if $e$ is an evidenced correct instance of $C$, but part of its sub-instances are evidenced correct instances of other concepts $C'$ where $C' \perp C$.

For example, we have *chicken* as an evidenced correct "Animal" instance, and also find that its sub-instances like *pork* and *beef* are evidenced instances of "Food". Since "Animal" and "Food" are mutually exclusive, we label *chicken* as an Intentional DP of "Animal".

**Rule 2.** We label $e$ of concept $C$ as an Accidental DP if $e$ is an evidenced incorrect instance of concept $C$.

For example, once *New York* is decided as an evidenced incorrect instance of "Country", it must be an evidenced correct instance of another concept, say "City", which should be mutually exclusive with "Country", then *New York* is very likely an Accidental DP of "Country".

**Rule 3.** We label $e$ of concept $C$ as a non-DP if $e$ and all its sub-instances are evidenced correct instances of $C$.

*Potential Problems.* A small number of instances (about 7 percent of all instances in our experiments) are labeled as Intentional DPs, Accidental DPs, or non-DPs. The strictness of the heuristic rules guarantees the correctness of the labeled data. However, the small training set only covers 66.4 percent of the one million concepts, and the left 33.6 percent have no training set, most of which are small concepts with no identified mutually exclusive concepts. On the other hand, the data labeled after the three rules probably has a biased distribution on feature 2 due to these rules labeling DPs and non-DPs relying on the mutually exclusive relation between concepts. Among all the four features, only feature 2 concerns about mutually exclusive relation. As a result, a DP detector trained on this kind of labeled data set may over-fit to a single dimension (feature 2).

## 3.3 Learning DP Detectors

In order to train DP detectors with the small set of biased training data, we have to address the problems with the data we described above.

First, to avoid over-fitting to a single dominant dimension, we follow a commonly used method which performs a non-linear mapping to transform the original data into the kernels of the data in a Hilbert space [22]. The advantage of Hilbert space let us be able to perform full rank kernel Principal Component Analysis (PCA) [22] to obtain a new representation of the original data, which won't be biased to a single dimension.

Second, for the problem that many concepts have no or only a very small training set, we propose to overcome this bottleneck through novel methods that leverage different kinds of knowledge instead of using more labels. Our premise is that it is cheaper to use a large amount of unlabeled data than to manually annotate a larger portion of instances. Also, humans often adapt knowledge obtained from previous experiences to improve the learning of new tasks. To address the problem of an insufficient number of labeled data, it is advantageous to adapt knowledge from other related concepts. In light of these, we propose a new algorithm, namely *Concept Adaptive Drift Detection* that not only leverages unlabeled data for a better understanding of new data, but also exploits the knowledge in other related concepts.

### 3.3.1 *Non-Linear Mapping*

We now introduce how to transform the original data into the kernels of the data in a Hilbert space $\mathcal{H}$ with a non-linear mapping, then we describe how to perform full rank kernel Principal Component Analysis [22] in the Hilbert space $\mathcal{H}$ to obtain a new representation of the original data.

Specifically, suppose there are $n$ instances under a concept and let $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$ be the original feature representations, where $x_i \in \mathbb{R}^d$ denotes the feature vector of the $i$th instance, and $d$ is the dimension of features. Let $\phi : \mathbb{R}^d \rightarrow \mathcal{H}$ be the non-linear mapping from $\mathbb{R}^d$ to the Hilbert space $\mathcal{H}$, such that $\phi(x_i)$ denotes the mapping of $x_i$ in $\mathcal{H}$. The covariance matrix in $\mathcal{H}$ is given by

$$C_{\mathcal{H}} = \frac{1}{n}\sum_{i=1}^{n} \phi(x_i)\phi(x_i)^T. \tag{6}$$

To perform rank kernel PCA in $\mathcal{H}$, we aim at finding the eigenvalues $\lambda \geq 0$ and the eigenvectors $V$ satisfying $\lambda V = C_{\mathcal{H}}V$. Although $\mathcal{H}$ could have an arbitrarily large, possibly infinite dimensionality, the inner product of any two data $\phi(x_i)$ and $\phi(x_j)$ can be explicitly expressed by a kernel matrix $K$, i.e., $K_{ij} = \phi(x_i)\phi(x_j)^T$. Thus we need to solve the eigenvalue problem $n\lambda\alpha = K\alpha$, where $\alpha = [\alpha_1, \ldots, \alpha_n]^T$ are coefficients such that $V = \sum_{i=1}^{n} \alpha_i$ [22], to obtain the new representations $\tilde{\mathcal{X}} = \{\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_n\}$. Let $\alpha_1, \alpha_2, \ldots, \alpha_r$ be the normalized eigenvectors corresponding to all the non-zero eigenvalues $0 < \lambda_1 \leq \ldots \leq \lambda_r$. Once we have obtained $\alpha_i|_{i=1}^{r}$, the mapping of the testing datum $x_j$ corresponding to the eigenvector $V^p$ $(1 \leq p \leq r)$ can be computed as $\tilde{x}_j^p = \sum_{i=1}^{r} \alpha_i^p(\phi(x_i)\phi(x_j)^T)$, which composes $\tilde{x}_j = [\tilde{x}_j^1, \tilde{x}_j^p, \ldots, \tilde{x}_j^r]^T$.

### 3.3.2 *Concept Adaptive Drift Detection*

Denote the new representation of an instance $x_i \in \mathbb{R}^d$ after the transformation as $\tilde{x}_i \in \mathbb{R}^r$. Suppose there are $t$ concepts. Let $y_i^c \in \{0,1\}^3$ be the label of an instance $\tilde{x}_i$ for the $c$th concept $(1 \leq c \leq t)$. If $x_i$ is an Intentional DP of the $c$th concept, $y_i^c = [1,0,0]$. If $x_i$ is an Accidental DP of the $c$th concept, $y_i^c = [0,1,0]$. If $x_i$ is a non-DP of the $c$th concept, $y_i^c = [0,0,1]$.

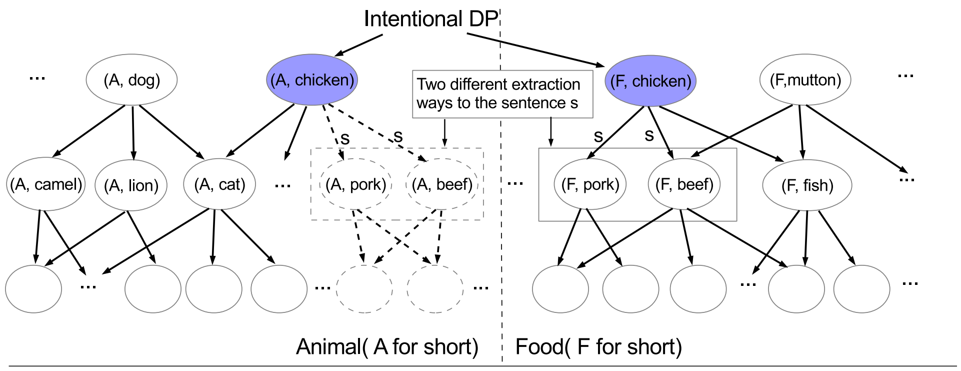


Fig. 3. An example triggering-extraction graph based on isA pairs.

This boolean labeling enables us to have equal distance between the three categories.

The DP detector of the $c$th concept can be represented as a function $F_c : \mathbb{R}^r \rightarrow \{0,1\}^3$, which maps the data from the representation $\tilde{x}_i$ to $y_i^c$. Suppose there are $n$ instances in all and the first $m(m < n)$ instances are labeled for the $c$th concept. A good DP detector $F_c$ can be trained within the regularized empirical error minimization framework as follows:

$$\min_{F_c} \sum_{i=1}^{m} loss\big(F_c(\tilde{x}_i), y_i^c\big) + \lambda\Omega(F_c), \tag{7}$$

where $loss(\cdot,\cdot)$ is a loss function to measure the distance between $F_c(\tilde{x}_i)$ and $y_i^c$, $\Omega(\cdot)$ is a regularization function on $F_c$ and $\lambda$ is the regularization parameter. Among various kinds of loss functions, we adopt the least square loss for its simplicity and effectiveness. Assuming that the training data are centered, we use the linear classifier $F_c(\tilde{x}_i) = W_c^T\tilde{x}_i$ as the DP detector, where $W_c \in \mathbb{R}^{r\times 3}$ is called the classifier for $F_c$. Then (7) can be rewritten as

$$\min_{W_c} \sum_{i=1}^{m} \|W_c^T\tilde{x}_i - y_i^c\|_F^2 + \lambda\Omega(W_c). \tag{8}$$

where $\|.\|_F$ denotes Frobenius norm. The classifier $W_c$ should be as small in magnitude as possible so that it would not over-fit the labeled data. $\|W_c\|_F^2$ is thus considered in the regularization term to control the complexity of $W_c$. However, the regularization function $\Omega(.)$ in our problem depends not only on $\|W_c\|_F^2$ but also other factors below.

With the small set of labeled data in hand, we propose a semi-supervised multi-task learning algorithm. First, we learn the classifier starting with a small set of labeled instances and increasingly involving unlabeled instances. In this step, we propose to exploit the manifold structure of both the labeled and unlabeled data via a statistical approach in training DP detectors of different concepts separately. Second, we learn classifiers for the $t$ concepts simultaneously such that we can share the knowledge among concepts to obtain better performance. The assumption is that the drift detectors of different concepts have some shared structural information in common. It is therefore reasonable to leverage the relevance between them to optimize the training of the drift detectors when we have few labeled data.

This learning process brings challenges on defining the regularization function $\Omega(.)$. Due to the limitation of space, we put details of this part in Appendix C, available in the online supplemental material.

## 4 Cleaning Drifting Errors

We now introduce how we clean drifting errors brought by detected DPs. For Accidental DPs, which are drifting errors themselves and trigger other drifting errors, we not only drop themselves, but also roll back all the extractions activated by them. Whereas, for Intentional DPs, we do not drop the DPs since they are correct instances themselves, but we check whether each extraction activated by an Intentional DP is a drifting error. In the following, we show how we perform such checking in Section 4.1, and then we describe how we rollback extractions triggered by detected DPs in Section 4.2.

### 4.1 Extractions Activated by Intentional DPs

Suppose the extraction we perform on a sentence is triggered by an Intentional DP. To judge whether the extraction is correct or not, we find all possible extraction ways to the sentence, and score each of them using a probabilistic graphic model base on the isA pairs obtained from the extraction. We take the extraction with the highest score as the expected correct extraction.

Specifically, assume the extraction to a sentence $s$ is triggered by an Intentional DP $(C, E)$. We denote the sentence as $s := \{\mathbf{C}_s, \mathbf{E}_s\}$ where $\mathbf{C}_s$ is the candidate set of concepts in the sentence, and $\mathbf{E}_s$ is the candidate set of instances in the sentence. Apparently, $C \in \mathbf{C}_s$ and $E \in \mathbf{E}_s$. For any instance $e \in \mathbf{E}_s$, it should only belong to one concept in $\mathbf{C}_s$ in the sentence $s$. Given the Intentional DP $(C, E)$, $e$ will be taken as an instance of $C$ and we will generate an isA pair $(C, e)$. But in the extraction to some other sentences, $e$ might be taken as an instance of different concept in $\mathbf{C}_s$. To find out which concept in $\mathbf{C}_s$ that each instance $e$ in $s$ most likely belongs to, we build a graph for the sentence $s$ to involve all relevant isA pairs into the graph, where each node denotes an isA pair, and the edge pointing from a node A to a node B means the isA pair in node A has triggered the extraction of the isA pair in node B. Then we will generate a graph as depicted in Fig. 3.

Based on this graph, we would like to use the Markov Blanket model [27] to help us estimate the correct probability of an isA pair $(C, e)$ triggered by an Intentional DP $(C, E)$. Since the deduction of the calculation is complicated, we use Lemma 1 and its proof to show how we calculate the correct probability of $(C, e)$, denoted as $Pr(C, e)$.

**Lemma 1.** *According to Markov Blanket model [27], the correct probability of an isA pair $(C, e)$ in an extraction-trigger graph can be calculated as*

$$Pr(C,e) \propto \frac{score(C,e)}{\sum_{C' \in \mathbf{C}_s} score(C',e)} \cdot \prod_{(C,e') \in N_c(C,e)} score(C,e'), \quad (9)$$

*where $N_c(C,e)$ is the set of isA pairs triggered by the node $(C,e)$ and $score(C,e)$ is the random walk score of $(C,e)$ calculated in the graph. Due to the space limitation, the proof to Lemma 1 is attached in Appendix D, available in the online supplemental material.*

Assume the extraction of a sentence $s := \{\mathbf{C}_s, \mathbf{E}_s\}$ is triggered by an Intentional DP $(C,E)$, we define the score that $C \in \mathbf{C}_s$ is the correct concept of the extraction as

$$Score(s,C) = \sum_{e' \in \mathbf{E}_s} \left( \frac{Pr(C,e')}{\sum_{C' \in \mathbf{C}_s} Pr(C',e')} \right). \quad (10)$$

If the concept $C$ does not hold the highest $Score(s,C)$ among all concept in $\mathbf{C}_s$, we say the current extraction to $s$ triggered by $(C,E)$ is incorrect, which will be rolled back.

**Example 1.** Given a sentence s:

$s =$ "food from animals **such as** *pork, beef and chicken*"

where $\mathbf{C}_s = \{$"food", "animal"$\}$, and $\mathbf{E}_s = \{$*pork, beef, chicken*$\}$. Assume (chicken isA animal) is an Intentional DP, and it triggers the extraction from $s$ wherein "animal" is the concept. We list the correct probability of every candidate concept and instance pair.

(pork, food, 0.235), (pork, animal, 0.005),
(beef, food, 0.214), (beef, animal, 0.008),
(chicken, food, 0.181), (chicken, animal, 0.165).

We have: $Score(s,\text{ "animal"}) = \frac{0.005}{0.005+0.235} + \frac{0.008}{0.008+0.214} + \frac{0.165}{0.165+0.181} = 0.021 + 0.036 + 0.476 = 0.533$. On the other hand, we have $Score(s,\text{"food"}) = \frac{0.235}{0.005+0.235} + \frac{0.214}{0.008+0.214} + \frac{0.181}{0.165+0.181} = 0.979 + 0.964 + 0.523 = 2.466$. As a result, the current extraction triggered by (chicken isA animal) is not the one with the highest score, thus will be rolled back.

## 4.2 DP-Based Cleaning Algorithm

After we identify all the DPs, we roll back extractions from sentences triggered by Accidental DPs and unqualified extractions from sentences triggered by Intentional DPs. We decrease the count of affected isA pairs in the knowledge base, and if the count of an isA pair becomes 0, the isA pair is removed from the knowledge base. This may trigger another wave of rolling back: Extractions from sentences triggered by these isA pairs will be rolled back as well. This roll-back process is performed iteratively until no more isA pairs can be removed from the knowledge base.

Besides, some DPs are only triggered by the DPs learned in earlier iterations. Removing DPs in earlier iterations consequentially clean DPs in the following iterations. Meanwhile, it eases the burden of the DP classification since many DPs' sub-instances have been removed. Therefore, our DP-based cleaning is conducted one iteration after one, until no DPs can be found and no cleaning is required.

At last, we also give the DP-based cleaning algorithm in Algorithm 1. Given the set of sentences $\mathbf{S} = \{S_1, S_2, \ldots, S_n\}$, where $S_i$ denotes the subset of sentences that are processed at the $i$th iteration. After all DPs are identified. From the subset of sentences $S_1$ that are processed in the beginning, we check whether the extractions from each sentence $s \in S_1$ are triggered by an Accidental DP or Intentional DP, where the extractions triggered by Accidental DPs and the unqualified extractions from sentences triggered by Intentional DPs will be withdrawn. After all sentences in $S_1$ are processed, we will update the frequencies of all learned isA pairs we learn. If an isA pair is already removed, i.e., the frequency of the isA pair becomes 0, then all the extractions to sentences that are triggered by this isA pair will be withdrawn directly. We repeat this process iteratively until no more isA pairs can be removed from the knowledge base. Until then, we move to check the sentences in the next iteration. The whole algorithm stops after we go through all sentences.

**Algorithm 1.** DP-Based Cleaning Algorithm

```
1: Input: The set of sentences $\mathbf{S} = \{S_1, S_2, \ldots, S_n\}$, where $S_i$
   denotes the subset of sentences that are processed at the $i$th
   iteration, the set of learned isA pairs and their frequencies
   $(Pairs, Freqs)$, all identified Accidental DPs and Intentional
   DPs.
2: for $i \leftarrow 1$ to $n$ do
     foreach $s \in S_i$ do
       if s is triggered by an Accidental DP then
         Withdraw all the isA pairs extracted from $s$;
       if $s := \{\mathbf{C}_s, \mathbf{E}_s\}$ is triggered by an Intentional DP
       then
         foreach $C \in \mathbf{C}_s$ do
           Calculate $Score(s,C)$ according to Eq. (10);
         if $C$ does not hold the highest $Score(s,C)$
         then
           Withdraw all the isA pairs extracted from
           $s$;
     repeat
       foreach $(pair, freq) \in (Pairs, Freqs)$ do
         if $freq = 0$ then
           Withdraw the extractions triggered by $pair$.
       Update $(Pairs, Freqs)$;
     until No more isA pairs can be removed;
```

## 5 Experiments

In our experiments, we first apply our approach on a semantic-based extraction process based on the Probase data,[2] and then on two syntax-based extraction processes based on the ClueWeb09 data[3] and CN-DBPedia data.[4]

### 5.1 Experiments on Semantic-Based Extraction

In this section, we first evaluate the effectiveness of our DP detection method, and then compare the performance of our approach with existing approaches.

#### 5.1.1 The Probase Data Set and Ground Truth

The Probase data is used for this group of experiments. After sentence de-duplication, there are 326,110,911 sentences extracted from 1,679,189,480 web pages. To the best of our knowledge, the scale of the data set is one order of

2. http://research.microsoft.com/en-us/projects/probase/
3. http://www.lemurproject.org/clueweb09/
4. http://kw.fudan.edu.cn

TABLE 1
Hearst Patterns (*NP* Stands for *Noun Phrase*)

| ID | Pattern |
|---|---|
| 1 | *NP* such as {*NP*,}*{(or — and)} *NP* |
| 2 | such *NP* as {*NP*,}*{(or — and)} *NP* |
| 3 | *NP* {,} including {*NP*,}*{(or — and)} *NP* |
| 4 | *NP* {, *NP*} * {,} and other NP |
| 5 | *NP* {, *NP*}*{,} or other *NP* |
| 6 | *NP* {,} especially {*NP*,}*{(or — and)} *NP* |

magnitude larger than the previously known largest corpus [21]. From these sentences, we used 7 hours and a cluster of 10 servers to extract isA pairs using the semantic-based iterative extraction method with the hearst patterns listed in Table 1. The method ran for about 100 iterations to extract about 143 millions isA pairs (90.5 millions distinct ones) under 13.5 million distinct concepts, where 99.999 percent were obtained in the first 10 iterations.

As a preprocessing step, each web document is broken down into a set of sentences using standard sentence boundary disambiguation methods [30]. All sentences that match any Hearst pattern are collected. Then, we perform *de-duplication* to long sentences based on the assumption that exactly the same long duplicate sentences are usually copied from each other.

Our goal is to extract isA pairs from all sentences matching any Hearst pattern. Specifically, given a sentence $s$, from which we want to obtain

$$s = \{(c, e_1), (c, e_2), \dots, (c, e_m)\}, \tag{11}$$

where $c$ is the concept, $\{e_1, e_2, \dots, e_m\}$ are its entities, and each $(c, e_i)$ is an extracted isA pair from sentence $s$.

As presented in Fig. 4a, the number of learned distinct pairs increases dramatically from only about 16.8 millions in iteration 1 to more than 90.5 millions after all iterations. However, the precision of the learned isA pairs, as was observed from more than 10k sampled data, also drops dramatically from more than 90 percent in iteration 1 to lower than 50 percent after 5 iterations.

All error isA pairs can be classified into two categories: (1) syntactic errors, such as incomplete sentences, typos or error syntactic parsing; (2) semantic drift errors, when error concept is chose for instances within a sentence. Some of the semantic drift errors are caused by error sentence understanding or the knowledge in the sentence is incorrect, but mostly, they are caused by error sentence parsing supported by learned isA pairs in previous iterations. The percentage of semantic drift errors among all errors in each iteration jumps from no more than 30 percent in iteration 1, to more than 80 percent after several iterations. In this paper, we mainly focus on cleansing semantic drift errors from the 90.5 millions of harnessed concept-instance pairs. Our DP finding and cleaning algorithm is implemented on a cluster of 10 servers using the *Map-Reduce* model. The whole algorithm runs for about 20 hours.

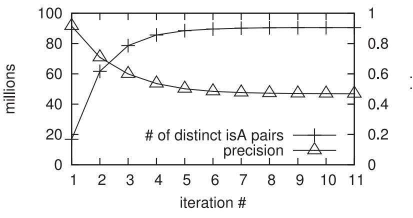
To set up the ground truth for evaluation, we manually labeled 1,115 Intentional DPs, 2,290 Accidental DPs, 4,408 non-DPs, 4,519 correct instances, and 5,979 drifting errors under 20 different concepts (listed in Table 2). Most of the 20 concepts are popular and large concepts such as “animal”, “company”, “woman”, given that semantic drift mostly occurs under popular concepts. To illustrate the effectiveness of our approach on tail concepts, we also involve one unpopular concept “key u.s. export” in our experiments. As we could observe in Table 2, among the 16 labeled instances for the “key u.s. export” concept, only 2 of them are labeled as errors, i.e., the error percentage is only 0.1250. But under some popular concepts such as “asian country”, “child”, “money” and “woman”, the percentage of error instances could be more than 50 percent.

### 5.1.2 Parameters Setting

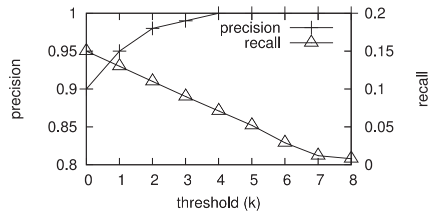
Before evaluating the performance of the DP detection method, and comparing the effectiveness of DP-based cleaning approach with other approaches, we need to decide two important components and parameters settings used in this paper. One is the scoring function $score(.)$ used in several features, the other is a threshold $k$ with Section 3.2, where only frequent pairs extracted from more than $k$ different sentences in the first iteration will be taken labeled as a correct instance.

*1. Scoring Function (Random Walk Model).* In defining the features for DP detection, we used a Random Walk based model to assign a score to each instance. To demonstrate the advantage of the Random Walk model, we compare it with two other models. One is a *Frequency* model, which gives each instance a score that is proportional to the frequency that the instance is learned under a concept. Another is a *PageRank* model, which do page rank [16] based on the same graph with the one used for random walk, except that the edges are undirected. Besides, we also use 0.15 as the teleporting probability, and we iterate the graph until the score vector converges. Table 3 lists the average precision of top 100, 1,000 or 2,000 instances under our labeled concepts by ranking them using one of the above models. As we can see, the Random Walk model reaches a higher precision than the Frequency model and the PageRank model.

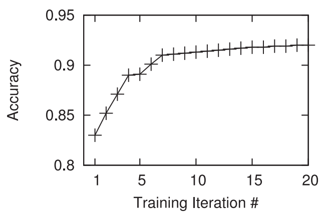
*2. Training Data Set Preparation.* A threshold $k$ is used in defining strong evidence for finding obvious DPs and non-DPs. As depicted in Fig. 4b, the average percentage of identified DPs and non-DPs decreases from 15 to 0.8 percent as $k$ increases from 0 to 8. On the other hand, the average precision of the labeled data increases from 0.902 to 0.993 as



(a) The number and precision of isA pairs



(b) The precision and recall of the labeled data



(c) The performance over iterations

Fig. 4. A number of figures for the experiments.

TABLE 2
The Statistics on Our Manually Labeled Instances under 20 Different Concepts

| concept | #Instances | The data we labeled | | | | | |
|---|---|---|---|---|---|---|---|
| | | #Correct instance | #Error instance | Error Percent | #Intent. DPs | #Accid. DPs | #Non-DPs |
| animal | 16,280 | 626 | 508 | 0.4479 | 32 | 256 | 809 |
| asian country | 784 | 74 | 165 | 0.6903 | 43 | 56 | 68 |
| child | 17,313 | 993 | 1,479 | 0.5983 | 527 | 642 | 181 |
| chinese city | 220 | 86 | 44 | 0.3384 | 5 | 8 | 50 |
| chinese food | 307 | 61 | 32 | 0.344 | 4 | 12 | 35 |
| chinese university | 46 | 23 | 9 | 0.2812 | 3 | 4 | 8 |
| computer | 7,264 | 235 | 785 | 0.7696 | 128 | 258 | 232 |
| computer software | 1,878 | 340 | 59 | 0.1478 | 12 | 23 | 208 |
| developing country | 282 | 42 | 64 | 0.6037 | 18 | 31 | 11 |
| disney classic | 142 | 43 | 26 | 0.3768 | 5 | 8 | 29 |
| key u.s. export | 26 | 24 | 2 | 0.125 | 1 | 5 | 6 |
| money | 3475 | 197 | 475 | 0.7068 | 17 | 86 | 79 |
| people | 76 | 47 | 8 | 0.1454 | 2 | 3 | 35 |
| phone | 3,193 | 276 | 122 | 0.3065 | 18 | 67 | 238 |
| president | 480 | 70 | 25 | 0.2631 | 2 | 4 | 67 |
| religion | 3,107 | 11 | 10 | 0.4761 | 49 | 143 | 79 |
| student | 19,930 | 100 | 424 | 0.8091 | 30 | 189 | 1,241 |
| u.s. state | 123 | 58 | 51 | 0.4678 | 5 | 13 | 53 |
| weather | 828 | 294 | 224 | 0.4324 | 11 | 29 | 72 |
| woman | 11,502 | 929 | 1,467 | 0.6122 | 203 | 453 | 907 |
| **Overall** | **87,246** | **45,19** | **5,979** | **0.5695** | **1,115** | **2,290** | **4,408** |

$k$ increases from 0 to 3. We could reach 100 percent precision when $k \geq 4$. To have the largest number of absolute correct labeled instances, we set $k = 4$ in our experiments. Finally, averagely we will have 7.1 percent of instances as labeled data, and 92.9 percent of instances as unlabeled data.

### 5.1.3 Our Approach versus Previous Approaches

We also compare our DP-based cleaning approach with several state-of-the-art approaches below: (1) *Mutual Exclusion (MEx):* This is the Mutual Exclusion Cleaning approach used in [5], which reports error instances belonging to mutually exclusive concepts. (2) *Type Checking (TCh):* This approach identifies drifting errors through type-checking [4], [17], where we use the well-developed entity recognition tool Stanford Named Entity Recognizer [10] to recognize entities with NLP tagging and grammar analysis. (3) *PRDual-Rank:* This approach [9] was used in a syntactic-based extraction process, which infers the quality of tuples and patterns from their neighbouring nodes, and only top-ranked ones in high quality will be kept. We adopt this technique on our data set by changing tuples and patterns into isA pairs and sentences respectively. (4) *Random Walk Rank (RW-Rank):* Similar to the PRDual-Rank method, here we use the random walk model to do the ranking, which has already demonstrated its advantages over other ranking models in Section 5.1.2.

We evaluate the cleaning results of these methods in the following four dimensions: (1) $p_{error}$ denotes the percentage of removed errors in all the removed instances; (2) $r_{error}$ denotes the percentage of removed errors in all the errors under each concept; (3) $p_{corr}$ denotes the percentage of remained correct instances in all the remained instances; (4) $r_{corr}$ denotes the percentage of remained correct instances in all the correct instances under each concept.

As presented in Table 4, although Mutual Exclusion and Type Checking reach a high precision in removing drifting error isA pairs (91.19 and 94.23 percent), the recall of removing bad entities is pretty low (15.70 and 14.51 percent), which demonstrates that the two methods are very reliable, but have limitations in finding all drifting errors. With well-learned thresholds, both PRDual-Rank and RW-Rank reach much higher recall (65.45 and 58.31 percent) in identifying drifting errors, but on the other hand, relatively low precision (56.21 and 57.53 percent). This illustrate that, even if the association between sentences and isA pairs were taken into considered, ranking methods are still unproper to be used in overcoming semantic drift, due to relying on thresholds and ranking models. In comparison, our DP cleaning method reaches the highest precision (96.96 percent) and recall (91.45 percent). After the cleaning, the precision of the remained entities is 89.21 percent, which is much higher than that with other methods. Besides, the recall of the remaining entities is

TABLE 3
The Precision of Top 100, 1,000, or 2,000 Instances

| Ranking Model | p@100 | p@1000 | p@2000 |
|---|---|---|---|
| Frequency | 0.5903 | 0.4576 | 0.4421 |
| PageRank | 0.6544 | 0.5603 | 0.5068 |
| Random Walk | 0.7971 | 0.6111 | 0.5636 |

TABLE 4
Comparing Cleaning Performance with Other Methods

| Cleaning Method | $p_{error}$ | $r_{error}$ | $p_{correct}$ | $r_{correct}$ |
|---|---|---|---|---|
| Before Cleaning | - | - | 0.4305 | 1.0 |
| MEx | 0.9119 | 0.1570 | 0.4592 | 0.9832 |
| TCh | 0.9423 | 0.1451 | 0.4789 | 0.9724 |
| PRDual-Rank | 0.5621 | 0.6545 | 0.5812 | 0.6940 |
| RW-Rank | 0.5753 | 0.5831 | 0.5636 | 0.6509 |
| DP Cleaning | **0.9696** | **0.9145** | **0.8921** | **0.9393** |

TABLE 5
Example DPs under Some Selected Concepts

| Concepts | Animal | Computer | Chinese City | Phone | Woman |
|---|---|---|---|---|---|
| Intentional DPs | chicken | notebook | Beijing | Apple | Chanel |
| | mammal | mainframe | Shanghai | Samsung | - |
| | crow | dell | Hong Kong | Huawei | - |
| Accidental DPs | fur | monitor | Sichuan | IBM | LV |
| | blood | hard drive | Hebei | Google | Gucci |
| | foot | mouse | Liaoning | Baidu | Burberry |
| | tail | virus | Jiangsu | Sina | Coach |
| | eye | keyboard | Macau | Tencent | Versace |

TABLE 6
Comparing the Effectiveness of DP Detection Methods

| Detection Method | Precision | Recall | F1 |
|---|---|---|---|
| Ad-hoc 1 | 0.841 | 0.714 | 0.772 |
| Ad-hoc 2 | 0.836 | 0.702 | 0.763 |
| Ad-hoc 3 | 0.807 | 0.513 | 0.627 |
| Ad-hoc 4 | 0.787 | 0.561 | 0.655 |
| Supervised | 0.853 | 0.783 | 0.817 |
| Semi-Supervised | 0.906 | 0.910 | 0.908 |
| Semi-Supervised Multi-Task | **0.927** | **0.953** | **0.939** |

still high (93.93 percent), which means that only a small number of good entities are taken as drifting errors by mistakes.

### 5.1.4 Evaluation on DP Detection

In this experiment, we compare our method with several baseline methods below: (1) a conventional *Supervised Learning* method (using Random Forest, which is observed as a good classifier to our task), (2) a *Semi-Supervised Learning* method without performing multi-task learning together, (3) several ad-hoc methods, each of which is designed based on an individual feature in Section 3.1 with a well-learned threshold.

As listed in Table 6, the four ad-hoc methods can reach a good precision, but usually not good enough recall. A better precision (0.853) and recall (0.783) can be achieved by our supervised learning approach with automatically labeled training data. By using unlabeled data, the Semi-Supervised Learning could reach 5 percent higher precision and 13 percent higher recall. Finally, with the multi-task learning, the precision and recall can be further improved about 2 and 4 percent respectively. Thus, the Semi-Supervised Multi-Task Learning is more effective than the other methods. We also depict the improvement of the accuracy as we iteratively update the DP detectors with the Semi-Supervised Multi-Task Learning method in Fig. 4c. It takes 20 iterations to have the accuracy of the DP detectors become stable, and the accuracy improves from 0.835 in the first iteration to 0.921 in the 20th iteration. In Table 5, we also list some example DPs that we identified under several classes.

### 5.1.5 Evaluation on DP-Based Cleaning

We perform DP-based cleaning after we detect DPs. In the following, we first evaluate the precision and recall of checking extractions triggered by Intentional DPs, and evaluate the results of DP-based cleaning.

*1. Identifying Errors Triggered by Intentional DPs.* We labeled 7,800 sentence parsings triggered by labeled Intentional DPs under the 20 concepts, where 4,511 sentences are labeled as correct ones and the left are labeled as incorrect ones. Based on this ground truth, we evaluate the precision $p_{stc}$ and recall $r_{stc}$ of the bad parsing identification strategy to the labeled bad sentences under each concept. As listed in column 2 and column 3 in Table 7, under most of the 20 concepts, the strategy could successfully identify more than 95 percent of bad sentence extraction with about 90 percent precision. This contributes to the final DP cleaning results.

TABLE 7
The Evaluation Results on DP Cleaning

| concept | $p_{stc}$ | $r_{stc}$ | $p_{error}$ | $r_{error}$ | $p_{corr}$ | $r_{corr}$ |
|---|---|---|---|---|---|---|
| animal | 0.865 | 0.831 | 0.982 | 0.849 | 0.942 | 0.993 |
| asian country | 0.874 | 0.892 | 0.827 | 0.864 | 0.75 | 0.692 |
| child | 0.965 | 0.901 | 0.992 | 0.990 | 0.78 | 0.812 |
| chinese city | 0.861 | 0.814 | 0.75 | 0.692 | 0.92 | 0.938 |
| chinese food | 0.98 | 0.812 | 0.916 | 0.687 | 0.868 | 0.970 |
| chinese uni. | 1.0 | 0.891 | 1.0 | 0.714 | 0.8 | 1.0 |
| computer | 0.970 | 0.921 | 0.991 | 0.905 | 0.791 | 0.977 |
| computer s. | 0.801 | 0.789 | 0.638 | 0.605 | 0.927 | 0.936 |
| developing c. | 1.0 | 0.95 | 1.0 | 0.914 | 0.666 | 1.0 |
| disney classic | 0.893 | 0.756 | 0.733 | 0.846 | 0.92 | 0.851 |
| k. u.s. export | 1.0 | 0.967 | 1.0 | 0.95 | 0.857 | 1.0 |
| money | 0.976 | 0.854 | 0.927 | 0.719 | 0.690 | 0.917 |
| people | 1.0 | 0.944 | 1.0 | 0.92 | 0.87 | 1.0 |
| phone | 0.924 | 0.856 | 0.963 | 0.881 | 0.949 | 0.985 |
| president | 1.0 | 0.879 | 1.0 | 0.8 | 0.965 | 1.0 |
| religion | 1.0 | 0.931 | 1.0 | 0.8 | 0.833 | 1.0 |
| student | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| u.s. state | 1.0 | 1.0 | 0.863 | 1.0 | 1.0 | 0.943 |
| weather | 0.934 | 0.854 | 0.842 | 0.820 | 0.902 | 0.915 |
| woman | 1.0 | 0.974 | 0.965 | 0.972 | 0.930 | 0.914 |
| **Overall** | **0.953** | **0.891** | **0.969** | **0.914** | **0.892** | **0.939** |

*2. Rolling back DP-Triggered Extractions.* After identifying the DPs, we perform DP-based cleaning by rolling back extractions triggered by the DPs. Table 7 (from column 4 to column 7) shows the DP cleaning results under the 20 concepts. We can on average recognize 91.45 percent of error instances with 96.96 percent precision, which proves the effectiveness of our cleaning approach. As a result, the precision of the isA pairs we learned can be improved from about 43.05 percent to 89.21 percent, which is almost the same with the precision in iteration 1. Besides, only 6.07 percent of isA pairs are sacrificed, which is acceptable comparing to the great improvement of precision.

## 5.2 Experiments on Syntax-Based Extraction

To further prove the effectiveness of our approach, we also conduct experiments on syntax-based extraction process on the ClueWeb09 data and CN-DBPedia data respectively. Since the mechanism of syntax-based extraction is different from that of semantic-based extraction, the DPs we are looking for are synthetic patterns instead of instances of each concept, thus the features used in the DP detection model and the DP detection algorithm as well as the DP-based cleaning algorithm need to be adjusted accordingly.

*Features Adjustment.* Four important features and some other potential ones are introduced in Section 3.1 for detecting DPs in semantic-based extraction scenario. Similar features could be employed in the syntax-based extraction scenario. Let $p$ denote a pattern of a concept $C$, then we have its features briefly described below:

TABLE 8
The Statistics on Our Manually Labeled Patterns under Nine Different Relations on ClueWeb09

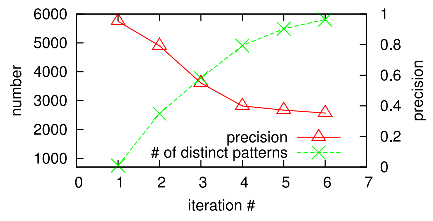
| Relation | #pattern | The data we labeled | | | | | |
|---|---|---|---|---|---|---|---|
| | | #Correct pattern | #Error pattern | Error Percent | #Intent.DPs | #Accid.DPs | #Non-DPs |
| TL | 202 | 141 | 61 | 0.302 | 0 | 56 | 146 |
| LO | 2,210 | 1,954 | 256 | 0.116 | 212 | 376 | 1,622 |
| AC | 412 | 393 | 19 | 0.046 | 74 | 24 | 314 |
| ME | 220 | 178 | 42 | 0.191 | 38 | 52 | 130 |
| MA | 209 | 163 | 46 | 0.22 | 23 | 49 | 137 |
| HO | 1,017 | 988 | 29 | 0.029 | 133 | 39 | 845 |
| JO | 767 | 704 | 63 | 0.082 | 79 | 50 | 638 |
| PR | 188 | 168 | 20 | 0.106 | 0 | 32 | 156 |
| HE | 587 | 416 | 171 | 0.291 | 75 | 157 | 355 |
| **Overall** | **5,812** | **5,105** | **707** | **0.154** | **554** | **735** | **4,231** |

- Feature 1 explores the frequency distribution of the instances triggered by $p$. We take the similarity between the frequency distribution of instances triggered by $p$ and the frequency distribution of target concept $C'$s instances given in the first iteration as the feature.
- Feature 2 is the number of $C'$s mutually exclusive concepts that also have $p$ as their pattern.
- For Feature 3, we still use the random walk score of $p$ as its correct probability, but the random walk graph now has two kinds of nodes, i.e., pattern nodes and instance nodes.
- Feature 4 of $p$ is the average random walk score of the instances that are brought into $C$ by $p$.
- The other potential features could also be adapted in similar ways.

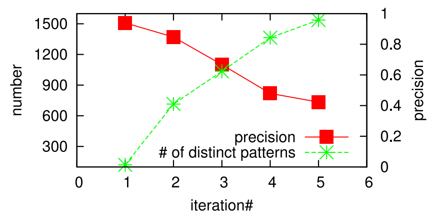
The adaptation to the DP detection algorithm and DP-based cleaning algorithm for syntax-based extraction is trivial. Due to the limitation of space, we do not give detai ls here.

### 5.2.1 Data Sets and Ground Truth

The ClueWeb09 data could be split into 13,706,265 short sentences after sentence de-duplication, and we mainly focus on the relations between *Location* and *Organization* (or *Loc-Org* for short) in our experiments. Before doing syntax-based extraction, we have detected both Location and Organization entities in 8,095,360 sentences using Stanford Named Entity Recognizer.[5] To set up the ground truth for the evaluation, we manually labeled 552 Intentional DPs, 707 Accidental DPs, 4,553 non-DPs, 5,105 correct patterns, and 707 drifting errors under 9 different relationships (listed in Table 8), namely TL (an address that a terrorist attacks), LO (an address that an organization locates), AC (a venue of an organization), ME (a location that an organization holds meetings), MA (an organization that controls an area), HO (an area that an organization has office spots), JO (a member state of an organization), PR (a president of an organization), HE (the headquarter of an organization). As presented in Fig. 5a, the number of learned distinct patterns increases dramatically from about 700 in iteration 1 to 6,000 after 6 iterations. However, the precision of the learned Loc-Org Pairs also drops dramatically from more than 90 percent in iteration 1 to lower than 40 percent after 6 iterations.



(a) ClueWeb09



(b) CN-DBPedia

Fig. 5. The number and precision of patterns on the two syntax-based extraction data sets.

CN-DBPedia is a Chinese data set stored in the form of triples, which contains 66,966,039 subject-predicate-object expression. In our experiments, we mainly focus on five categories of relationships, namely person-company (PC), person-school (PS), person-association (PA), organization-location (OL) and organization-headquarter (OH). We manually labeled 239 intentional DPs, 269 Accidental DPs, 1,030 non-DPs (listed in Table 9) to set up the ground truth for evaluation. As presented in Fig. 5b, the number of learned distinct patterns increases from about 100 in iteration 1 to 1,500 after 5 iterations. But the precision of the learned patterns drops from 90+ to 40 percent after 5 iterators.

Due to the limitation of space, we do not present the experiments on the DP detection results here, but just list some example DPs that we identified under several relations on the two data sets in Tables 10 and 11 respectively.

### 5.2.2 Our Approach versus Previous Approaches

We now compare our DP-based cleaning approach with four state-of-the-art approaches on the same four dimensions on the two data set: (1) *Mutual Exclusion(MEx)*; (2) *Type Checking (TCh)*; (3) *PRDual-Rank*; and (4) *Random Walk Rank (RW-Rank)*. As listed in Table 12, although the MEx and TCh approaches achieve a high precision in removing drifting errors, they have a low recall in removing errors, which

5. https://nlp.stanford.edu/software/CRF-NER.shtml

TABLE 9
The Statistics on Our Manually Labeled Patterns under Five Different Relations on CN-DBPedia

| Relation | #pattern | The data we labeled | | | | | |
|---|---|---|---|---|---|---|---|
| | | #Correct pattern | #Error pattern | Error Percent | #Intent.DPs | #Accid.DPs | #Non-DPs |
| PC | 665 | 585 | 80 | 0.120 | 30 | 80 | 555 |
| PS | 410 | 300 | 110 | 0.268 | 90 | 110 | 210 |
| PA | 160 | 125 | 35 | 0.219 | 45 | 35 | 80 |
| OL | 120 | 95 | 25 | 0.208 | 25 | 25 | 70 |
| OH | 182 | 164 | 18 | 0.099 | 49 | 18 | 115 |
| **Overall** | **1,537** | **1,269** | **268** | **0.174** | **239** | **268** | **1,030** |

TABLE 10
Example DPs under Some Selected Relations on ClueWeb09

| Relation | LO | ME | AC | MA | HO | JO | HE |
|---|---|---|---|---|---|---|---|
| **Intentional DPs** | locate in<br>live in<br>have return to | be in<br>have return to<br>- | live in<br>be in<br>be now available for | be administer by<br>be active in<br>- | be on<br>locate in<br>be a division of | be a strong supporter of<br>be administer by<br>be in | found in<br>be home to<br>be the home of |
| **Accidental DPs** | reach<br>be a suburb in<br>be a town in | arrive from<br>leave for<br>be locate in | have win in<br>be out of<br>should leave | be a unit of<br>say in<br>be a part of | also work with<br>be a member of<br>be create by | be an agency of<br>be headquarter in<br>be allocate to | be form in<br>grow up in<br>meet in |

TABLE 11
Example DPs under Some Selected Relations on CN-DBPedia

| Relation | PC | PS | PA | OL | OH |
|---|---|---|---|---|---|
| **Intentional DPs** | legal representative<br>currently serves on | present post<br>dean | vice president<br>successive person in charge | headquarter<br>lie in | the company located in<br>locate in |
| **Accidental DPs** | dean<br>company organization | arrive from<br>leave for | director<br>general branch secretary | currently have jurisdiction over<br>affiliates | branch office<br>listing location |

demonstrates that in two data sets, the two approaches are reliable in finding errors, but have limitations in finding all drifting errors. With well-learned thresholds, both PRDual-Rank and RW-Rank reach much higher recall in identifying drifting errors, but on the other hand, relatively low precision. This illustrate that, also on the ClueWeb09 and CN-DBPedia Data Sets, even if the association between sentences and patterns were taken into consideration, ranking methods are still imperfect to be used in overcoming semantic drift, due to relying on thresholds and ranking models. In contrast, Our DP cleaning method reaches the highest precision and recall of the remained patterns. Besides, the precision and recall in removing error patterns are still high, given that only a spot of patterns are taken by mistakes.

## 6 Related Work

Semantic drift, also known as conceptual shift, has been known as a common problem in unsupervised iterative information extraction [17], [19], [25], [31], [34].

Existing IE systems tackled semantic drift through identifying and dropping drifting errors, but found limitations in reaching both a high precision and recall. In this paper, our method overcomes semantic drift by identifying the cause of semantic drift, i.e., DPs. After knowing the cause, we can cut off the propagation of drifting errors in iteration extractions.

Previous methods for identifying drifting errors can be roughly divided into two categories: (1) multi-class based, and (2) single-class based, according to the settings of IE systems that adopt them. In the following, we will cover the related work from the two aspects respectively.

Multi-class based methods were adopted by IE systems that performed iterative IE on multiple classes simultaneously, which identified drifting errors by comparing the extraction results between multiple classes [33].

TABLE 12
Comparing Cleaning Performance with Other Methods on ClueWeb09 and CN-DBPedia

| ClueWeb09 | $p_{error}$ | $r_{error}$ | $p_{correct}$ | $r_{correct}$ |
|---|---|---|---|---|
| Before Cleaning | - | - | 0.302 | 1.000 |
| MEx | 0.828 | 0.231 | 0.322 | 0.790 |
| TCh | 0.803 | 0.158 | 0.414 | 0.837 |
| PRDual-Rank | 0.688 | 0.657 | 0.539 | 0.678 |
| RW-Rank | 0.646 | 0.471 | 0.531 | 0.647 |
| DP-Cleaning | **0.936** | **0.897** | **0.887** | **0.853** |

| CN-DBPedia | $p_{error}$ | $r_{error}$ | $p_{correct}$ | $r_{correct}$ |
|---|---|---|---|---|
| Before Cleaning | - | - | 0.407 | 1.000 |
| MEx | 0.672 | 0.842 | 0.636 | 0.402 |
| TCh | 0.761 | 0.236 | 0.514 | 0.712 |
| PRDual-Rank | 0.714 | 0.842 | 0.856 | 0.365 |
| RW-Rank | 0.668 | 0.769 | 0.640 | 0.299 |
| DP-Cleaning | **0.924** | **0.865** | **0.814** | **0.857** |

Mutual Exclusion method is a representative one of this kind, which is based on the intuition that instances cannot belong to mutually exclusive semantic classes (unless the instances are ambiguous) [5]. Patterns and instances that violate this intuition will be taken as drifting errors. However, the mutual exclusion heuristic requires us to have prior knowledge on the exclusion between all classes, which can not be reached in reality when millions of classes are involved.

Single-class methods work on the extraction results of one class, however, most of which emphasize "probability assessment" (or called as "confidence"), which captures precision only in a heuristic manner [9].

For example, Riloff et al. [20] proposed to keep the most reliable instances in each iteration, and the "reliability" of an instance is decided by the number and quality of matched patterns to a class. Other methods relied on some heuristic models to assess the correct probability of extractions, such as according to the number and the reliability of patterns generating them [20], or combining evidences from multiple extractions [2], [18], [24]. Recently, Fang et al. [9] also modeled PRDual-Rank to capture the notion of precision and recall for both tuples and patterns in a principled way, and the confidence of an instance is decided by its relevant patterns. However, all these heuristic methods rely on arbitrary threshold to divide all extractions into two parts, which can hardly reach both high precision and satisfied recall. Differently, we model the DP detection problem into a learning problem, which uses some well-designed features based on the properties of DPs.

Other methods include the Espresso [12] method, which benefits from generic patterns by using a principled measure of instance and pattern reliability. The key idea of Espresso is recursive definition of pattern-instance scoring metrics [12]. Another method trains a relation classifier based on textual features that extracted from a large unlabled corpus with the help of Freebase, and shows that the syntactic parse features are particularly helpful for relations that are ambiguous or lexically distant in their expression [14].

Recently, Carlson et al. [4] also adopt a semi-supervised learning method by coupling the simultaneous training of many extractors [4]. They reported a high accuracy by enforcing constraints, including mutual exclusion, type checking [4], [17], given as domain knowledge, but on the other hand, also hurt the recall of the extraction. Similarly, our method for DP detection is also a semi-supervised learning method which starts with some seed instances that are labeled using strict rules based on the mutual exclusive heuristic [5]. However, we do not rely on strict constraints, but resort to a multi-task learning method that not only leverages unlabeled data for a better understanding of the new data (semi-supervised), but also improves the classifier for a concept by exploiting its related concepts (multi-task). As demonstrated in the experiments, our learning method reaches both high precision and satisfied recall.

To summarize, previous methods suffer either low precision or low recall because they focus on finding the semantic drift phenomenon. Our approach overcomes semantic drift by identifying the cause of the semantic drift: the drifting points. After knowing the cause, we can cut off the propagation of errors in iteration extractions.

## 7 Conclusion

In this paper, we propose a novel method to minimize semantic drift by identifying Drifting Points (DPs), which are the culprits of introducing semantic drifts. Compared to previous approaches which usually incur substantial loss in recall, DP-based cleaning method can effectively clean a large proportion of semantic drift errors while keeping a high recall. According to the experiments on a large data set, the DP cleaning method can effectively clean more than 90 percent of errors with more than 95 percent precision. After cleaning, the precision of extracted isA pairs is improved from about 50 to 90 percent. As a future work, we will adopt our method to overcome semantic drift happening to other types of relations in IE.

## Acknowledgments

This research is partially supported by the National Natural Science Foundation of China (Grant No. 61402313, 61572336, 61632016, 61472263, 61232006), the Postdoctoral Scientific Research Funding of Jiangsu Province (No. 1501090B), the National Postdoctoral Funding (No. 2015M581859, 2016T90493) and the Natural Science Research Project of Jiangsu Higher Education Institution (No. 17KJA520003). Z. Li and Y. He have contributed equally to this work.

## References

[1] E. Agichtein and L. Gravano, "Snowball: Extracting relations from large plain-text collections," in *Proc. 2nd ACM/IEEE-CS Joint Conf. Digital Libraries*, 2000, pp. 85–94.

[2] M. Cafarella, J. Madhavan, and A. Halevy, "Web-scale extraction of structured data," *ACM SIGMOD Rec.*, vol. 37 no. 4, pp. 55–61, 2009.

[3] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. Hruschka Jr., and T. Mitchell, "Toward an architecture for never-ending language learning," in *Proc. 26th AAAI Conf. Artif. Intell.*, 2010, pp. 1306–1313.

[4] A. Carlson, J. Betteridge, R. Wang, E. Hruschka Jr., and T. Mitchell, "Coupled semi-supervised learning for information extraction," in *Proc. Int. Conf. Web Search Data Mining*, 2010, pp. 101–110.

[5] J. Curran, T. Murphy, and B. Scholz, "Minimising semantic drift with mutual exclusion bootstrapping," in *Proc. 10th Conf. Pacific Assoc. Comput. Linguistics*, 2007, pp. 172–180.

[6] O. Etzioni, M. Banko, S. Soderland, and D. Weld, "Open information extraction from the web," *Commun. ACM*, vol. 51, no. 12, pp. 68–74, 2008.

[7] O. Etzioni, et al., "Web-scale information extraction in knowitall," in *Proc. 23rd Int. Conf. World Wide Web*, 2004, pp. 100–110.

[8] A. Evgeniou and M. Pontil, "Multi-task feature learning," in *Proc. 28th Int. Conf. Neural Inf. Process. Syst.*, 2007, vol. 19, Art. no. 41.

[9] Y. Fang and K. C.-C. Chang, "Searching patterns for relation extraction over the web: Rediscovering the pattern-relation duality," in *Proc. Int. Conf. Web Search Data Mining*, 2011, pp. 825–834.

[10] J. Finkel, T. Grenager, and C. Manning, "Incorporating non-local information into information extraction systems by gibbs sampling," in *Proc. Annu. Meet. Assoc. Comput. Linguistics*, 2005, pp. 363–370.

[11] Z. Ghahramani and K. Heller, "Bayesian sets," in *Proc. 28th Int. Conf. Neural Inf. Process. Syst.*, 2005, pp. 435–442.

[12] M. Komachi, T. Kudo, M. Shimbo, and Y. Matsumoto, "Graph-based analysis of semantic drift in espresso-like bootstrapping algorithms," in *Proc. Conf. Empirical Methods Natural Language Process.*, 2008, pp. 1011–1020.

[13] T. McIntosh and J. Curran, "Reducing semantic drift with bagging and distributional similarity," in *Proc. Annu. Meet. Assoc. Comput. Linguistics*, 2009, pp. 396–404.

[14] M. Mintz, S. Bills, R. Snow, and D. Jurafsky, "Distant supervision for relation extraction without labeled data," in *Proc. Annu. Meet. Assoc. Comput. Linguistics*, 2009, pp. 1003–1011.

[15] F. Nie, H. Huang, X. Cai, and C. Ding, "Efficient and robust feature selection via joint l21-norms minimization," in *Proc. 28th Int. Conf. Neural Inf. Process. Syst.*, 2010, pp. 1813–1821.

[16] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web," *Stanford InfoLab*, pp. 1–14, 1998.

[17] M. Pasca, D. Lin, J. Bigham, A. Lifchits, and A. Jain, "Names and similarities on the web: Fact extraction in the fast lane," in *Proc. Annu. Meet. Assoc. Comput. Linguistics*, 2006, pp. 809–816.

[18] M. Pennacchiotti and P. Pantel, "Entity extraction via ensemble semantics," in *Proc. Conf. Empirical Methods Natural Language Process.*, 2009, pp. 238–247.

[19] E. Riloff, "Automatically generating extraction patterns from untagged text," in *Proc. 26th AAAI Conf. Artif. Intell.*, 1996, pp. 1044–1049.

[20] E. Riloff and R. Jones, "Learning dictionaries for information extraction by multi-level bootstrapping," in *Proc. 26th AAAI Conf. Artif. Intell.*, 1999, pp. 474–479.

[21] A. Ritter, S. Soderland, and O. Etzioni, "What is this, anyway: Automatic hypernym discovery," in *Proc. 26th AAAI Conf. Artif. Intell.*, 2009, pp. 88–93.

[22] B. Schölkopf, A. Smola, and K. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Comput.*, vol. 10 no. 5, pp. 1299–1319, 1998.

[23] M. Skounakis and M. Craven, "Evidence combination in biomedical natural-language processing," in *Proc. Int. Workshop Data Mining Bioinform.*, 2003, pp. 25–32.

[24] P. Talukdar, J. Reisinger, M. Paşca, D. Ravichandran, R. Bhagat, and F. Pereira, "Weakly-supervised acquisition of labeled class instances using graph random walks," in *Proc. Conf. Empirical Methods Natural Language Process.*, 2008, pp. 582–590.

[25] M. Thelen and E. Riloff, "A bootstrapping method for learning semantic lexicons using extraction pattern contexts," in *Proc. Conf. Empirical Methods Natural Language Process.*, 2002, pp. 214–221.

[26] H. Tong, C. Faloutsos, and J. Pan, "Fast random walk with restart and its applications," in *Proc. IEEE Int. Conf. Data Mining*, 2006, pp. 613–622.

[27] I. Tsamardinos, C. F. Aliferis, A. R. Statnikov, and E. Statnikov, "Algorithms for large scale markov blanket discovery," in *Proc. 16th Int. FLAIRS Conf.*, 2003, vol. 2, pp. 376–380.

[28] R. Wang and W. Cohen, "Iterative set expansion of named entities using the web," in *Proc. IEEE Int. Conf. Data Mining*, 2008, pp. 1091–1096.

[29] L. Wasserman, "All of Statistics: A concise Course in Statistical Inference," Berlin, Germany: Springer Science & Business Media, 2013.

[30] W. Wu, H. Li, H. Wang, and K. Zhu, "Probase: A probabilistic taxonomy for text understanding," in *Proc. ACM SIGMOD Int. Conf. Manag. Data*, 2012, pp. 481–492.

[31] X. Yang and J. Su, "Coreference resolution using semantic relatedness information from automatically discovered patterns," in *Proc. Annu. Meet. Assoc. Comput. Linguistics*, 2007, pp. 528–535.

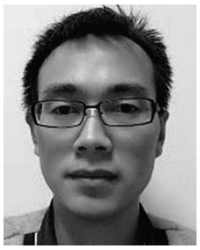
[32] Y. Yang, F. Nie, D. Xu, J. Luo, Y. Zhuang, and Y. Pan, "A multimedia retrieval framework based on semi-supervised ranking and relevance feedback," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 4, pp. 723–742, Apr. 2012.

[33] R. Yangarber, "Counter-training in discovery of semantic patterns," in *Proc. Annu. Meet. Assoc. Comput. Linguistics*, 2003, pp. 343–350.

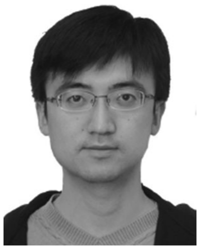
[34] H. Yu and E. Agichtein, "Extracting synonymous gene and protein terms from biological literature," *Bioinf.*, vol. 19, 2003, Art. no. i340.

**Zhixu Li** received the BS and MS degrees from the Renmin University of China, and the PhD degree from the University of Queensland, in 2006, 2009, and 2013, respectively. He is an associate professor in the Department of Computer Science & Technology, Soochow University, China. He previously worked as a research fellow with the King Abdullah University of Science and Technology. His research interests include the data cleaning, big data applications, information extraction, and retrieval.

**Ying He** is working toward the master's degree in the Research Center on Advanced Data Analytics (ADA), Soochow University, China. Her research interests include knowledge base, information extraction, and NLP.



**Binbin Gu** is working toward the master's degree in the Research Center on Advanced Data Analytics (ADA), Soochow University, China. His research interests include knowledge fusion, information extraction, and NLP. He has published several papers at DASFAA. He is the external reviewer of several international conferences such as ADC and WAIM.



**An Liu** received the PhD degree from both the City University of Hong Kong and University of Science and Technology of China, in 2009. He is an associate professor with the Soochow University. Prior to that in 2014, he was a senior research associate in the Joint Research Center of the City University of Hong Kong (CityU) and University of Science & Technology of China (USTC). His research interests include security, privacy, and trust in emerging applications; cloud computing; and services computing.

**Hongsong Li** received the PhD degree in computer science from Beijing Jiaotong University. He joined Microsoft Research Asia as an associate researcher in 2006. He has been with the Data intelligence and Tool Group, the Web Search and Mining Group, and the Data Management, Analytics and Services Group. His research interests include knowledge base, web data mining, machine learning, and NLP.

**Haixun Wang** received the PhD degree from the University of California, in 2000. He joined Google Research (Mountain View) in 2013. Prior to that, from 2009-2013, he was with Microsoft Research (Asia) where he led the database team. His research interests include text analytics, NLP, knowledge base, semantic network, artificial intelligence, database language and system, and graph data management. He is a fellow of the IEEE.

**Xiaofang Zhou** is a professor of computer science with The University of Queensland. He is the head of the Data and Knowledge Engineering Research Division. He is a specially appointed adjunct professor under the Chinese National Qianren Scheme hosted by the Renmin University of China (2010-2013), and by Soochow University where he has led the Research Center on Advanced Data Analytics (ADA) since July 2013. He has been working in the area of spatial and multimedia databases, data quality, high performance query processing, Web information systems, and bioinformatics, and has co-authored more than 250 research papers with many published in top journals and conferences. He is a fellow of the IEEE.