Overcoming Semantic Drift in Information Extraction

Zhixu Li [§1][*], Hongsong Li [†2], Haixun Wang [†3], Yi Yang [‡4], Xiangliang Zhang [§5], Xiaofang Zhou [♭‡6]
[§]*King Abdullah University of Science and Technology, Saudi Arabia*
[†]*Microsoft Research Asia, Beijing, China* [‡]*The University of Queensland, Brisbane, Australia*
[♭]*School of Computer Science and Technology, Soochow University, China.*
[1,5]{zhixu.li, xiangliang.zhang}@kaust.edu.sa, [2]hongsli@microsoft.com,
[3]haixun@gmail.com, [4]yi.yang@uq.edu.au, [6]zxf@itee.uq.edu.au

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 demonstrate the effectiveness of our approach in cleaning information extraction errors caused by semantic drift. We perform experiments on a isA relation iterative extraction, where 90.5 million of isA pairs are automatically extracted from 1.6 billion web documents with a low precision. The experimental results show our DP cleaning method enables us to clean more than 90% incorrect instances with 95% precision, which outperforms the previous approaches we compare with. As a result, our method greatly improves the prevision of this large isA data set from less than 50% to over 90%.

Categories and Subject Descriptors

H.2.4 [**Database Management**]: Systems—*Textual Databases*

General Terms

Algorithms, Performance, Experimentation

1. INTRODUCTION

Iterative bootstrapping is used extensively in information extraction (IE) [22, 28, 25, 14]. 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 [14]; and it is domain and language independent [11]. Indeed, bootstrapping approaches have shown good performance in many web-scale information extraction tasks [1, 24].

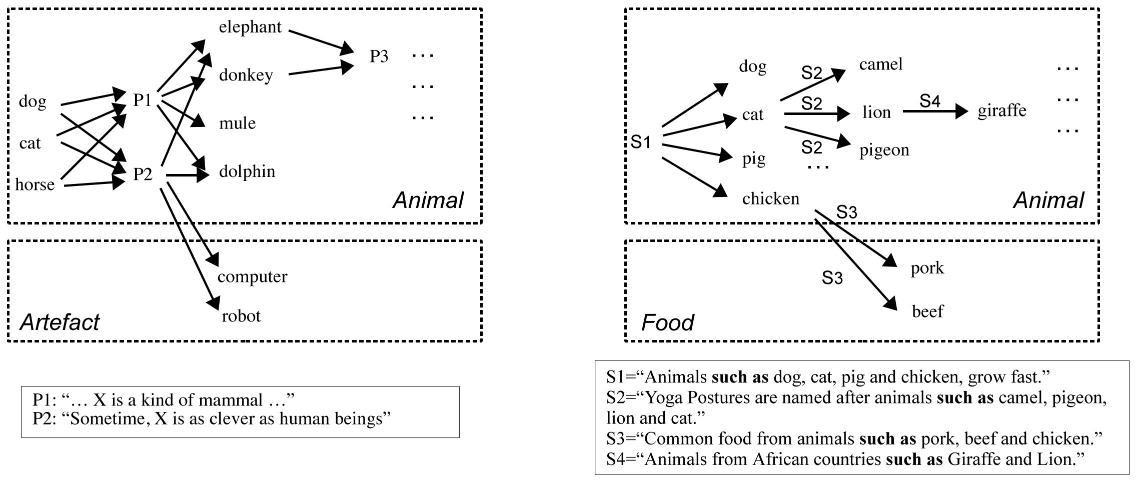
One of the biggest issues of iterative information extraction is *semantic drift* [17, 5]: As iterations proceed, the extractions may shift from the target class to some other classes [5]. As we know, state-of-the-art iterative IE methods can be divided into two categories, syntax-based and semantic-based, 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. 1(a), 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 [24] proposed a *semantic-based* iterative mechanism. For a given syntactic pattern, it performs multiple semantic iterations. Each iteration extracts new results to be added to 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 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 a sentence S_3= "Common food from animals **such as** *pork*, *beef*, and *chicken*" in Fig. 1(b). Assume

[*]This work was done during the first author's internship at Microsoft Research Asia.



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

Figure 1: Snapshots of Iterative Extraction for "Animal" with Two Different Bootstrapping Mechanisms

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 concepts such as "Food". □

There is already some work on reducing semantic drift. For example, Type Checking [14] 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 [17, 20, 4] 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) [17], or by combining evidences from multiple extractions [20, 4]. 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* 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. 1(a) and the instance "chicken" in Fig. 1(b). 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 [5, 14, 20, 4], which are hard to be captured by a single approach or model. In contrast, we identify two types of DPs that hold four strong properties (see Sec. 2.2), which enable us to identify DPs and eventually identify erroneous extractions more effectively.

Overcoming semantic drift through DPs raises some nontrivial challenges. Firstly, 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). Secondly, 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 then identify some important properties for differentiating the two types of DPs from non-DPs. For instance, a non-DP probably only introduces high-frequency instances into a class, since these instances are correct ones which are also introduced by many other instances under the target class. In contrast, a second-type DP only introduces low-frequency instances, since these instances does not belong to the target class and thus are seldom introduced by other instances under the target class. Whereas, a first-type DP probably introduces both high-frequency instances and low-frequency instances. Thus, 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. 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.

As an alternative, we resort to a supervised learning approach that takes advantages of all the identified properties of DPs in detecting DPs. However, one drawback of supervised learning is that it relies on large training data. In our case, we perform extraction on a large number of (e.g., millions of) target concepts, which means we need labeled data for each concept. This is infeasible.

To address this issue, we start with some seed instances labeled strictly based on the mutual exclusive heuristic [5]. We then introduce a semi-supervised, multi-task learning method that not only leverages unlabeled training data for a better understanding of the new data (semi-supervised), but also improves the classifier for one concept by exploiting its related concepts (multi-task).

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 adopt several DP-based cleaning strategies 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, and as a result, we clean a very large proportion of semantic drift errors.

We perform experiments on a semantic-based isA relation iterative extraction, where 90.5 millions of isA pairs in 13.5 million concepts are extracted from 1.6 billions of web pages with *such as* pattern. The results show that our DP cleaning method enables us to achieve 90% precision and 94% recall, which outperforms the previous approaches we compare with.

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

2. OVERVIEW ON DRIFTING POINTS

In this paper, we are concerned with overcoming semantic drift in semantic-based isA relationships extraction. In the following, we first give definition of DPs in semantic-based isA relation extraction, and then define two types of DPs with their properties. Finally, we present the two challenging problems we need to solve in our method.

2.1 Drifting Points (DPs) in Semantic-based IsA Relation Extraction

In semantic-based isA relation extraction, we aim at extracting *Instances* such as *dog, cat, pig* for target *Class (or Concept)* such as "Animal" in an iterative bootstrapping manner with a Hearst (**such as**) pattern. In this context, we say an existing knowledge, i.e., an instance under a target class (or an isA pair), *triggers* the extraction of some other instances under the same class, if it enables us to understand a sentence, thereby generating these new isA pairs from the sentence. For example, *chicken* triggers the extraction of *pork, beef* under the "Animal" class in S_3 in Fig. 1(b). Also, we call these new generated instances (such as *pork, beef*) as *sub-instances* of the existing instance (such as *chicken*) that triggers them.

In semantic-based isA relation extraction, 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 as a* Drifting Error *w.r.t. a target class, if this instance 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% 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 is a Drifting Point w.r.t. a target class if it introduces drifting errors to this target class.*

2.2 Types of DPs

According to our observations, there are 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 not erroneous extractions by themselves but part of the instances triggered by this kind of DPs are drifting errors, such as *chicken* w.r.t. the "Animal" class as we show in Fig. 1(a). We formally define this kind of DPs as *Intentional DPs* below:

Definition 3. *An* **Intentional DP** *is a polysemous instance 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.*

In contrast, the second type of DP is an erroneous extraction by itself and all of the instances triggered by this kind of DP are drifting errors. We formally define them as *Accidental DPs* below:

Definition 4. *An* **Accidental DP** *is an extraction error by itself. It happens accidentally or randomly and then it triggers drifting errors in the subsequent iterations.*

An Accidental DP results from mistakes that happen accidentally. There are two situations in which an Accidental DP occurs. First, it might come from incorrectly parsed sentences. For example, the following sentence

"... animals other than *dogs* **such as** *cats* ..."

may be parsed incorrectly and produce (cat isA dog). Thus *cat* becomes an Accidental DP in the "dog" class as it will very likely introduce 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 countries **such as** *France, Portugal, New York, Mauritius, Norway* and *Japan* ...",

contains a wrong fact (New York isA country). Thus *New York* might become an Accidental DP in the "Country" class.

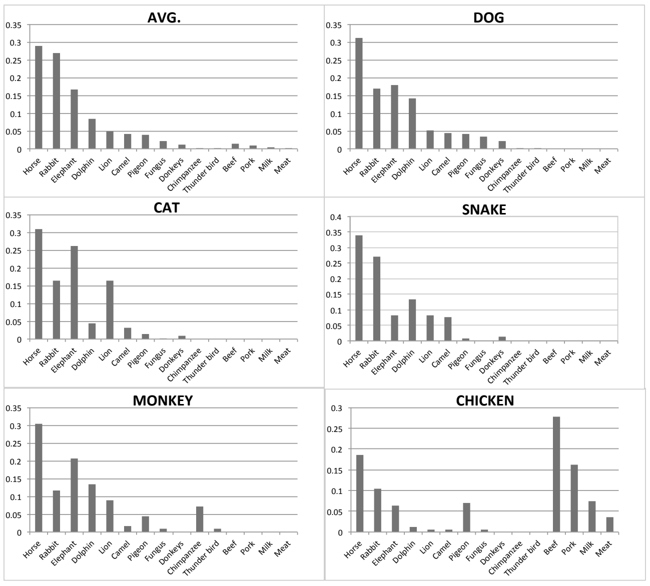


Figure 2: Distributions of instances triggered by DPs and non-DPs

2.3 Properties of DPs

We identify four important properties for the two types of DPs, which we leverage to detect DPs from non-DPs.

Property 1. *For a target class, the distribution of instances triggered by a DP is different from the distribution of instances that truly belong to the target class.*

Intuitively, a non-DP instance only triggers instances that belong to the target class, but a DP can potentially trigger instances of other classes. For example, Fig. 2 compares the distributions of instances triggered by some labeled DPs and non-DPs under "Animal" in our experimental data set (90.5 millions of isA pairs). Instances triggered by non-DP instances of "Animal" such as *dog*, *cat*, *snake*, and *monkey* are similar to the distribution of instances in "Animal" (denoted as the AVG distribution in Fig. 2). But a DP instance, such as *chicken*, triggers instances whose distribution is quite different from the AVG distribution. This is because it triggers not only "Animal" instances but also "Food" instances such as *beef*, *pork*, *milk*, and *Meat*.

Although Property 1 generally holds, there exist cases where the distribution of instances triggered by a non-DP is different from the AVG distribution. For example, a non-DP may trigger instances that are correct but are rarely or not as frequently triggered by others. For example, as shown in Fig. 2, *monkey* triggers the extraction of *chimpanzee* much more frequently than other non-DP instances.

Property 2. *If classes C_1 and C_2 are mutually exclusive, then an instance $e \in C_1 \cap C_2$ is very likely an Intentional DP.*

For example, *apple* is an instance of both "Fruit" and "Company", which are mutually exclusive, so *apple* is likely to be an Intentional DP. This property has been used to identify suspicious instances [5] based on pre-identified mutually exclusive classes.

Property 3. *An Accidental DP is usually supported by very weak evidence, that is, the instance is derived from very few (mostly only one) sentences.*

Intuitively, an Accidental DP is obtained by mistake. The probability that the same mistake occurrs repeatedly is low probability in different sentences. For example, from billions of pages, statements such as (cat isA Dog) and (New York isA Country) can only be found in one sentence. Therefore, an Accidental DP usually has a low frequency, i.e., supported by weak evidence.

However, we cannot soly rely on Property 3 to identify Accidental DPs, since as many as 50% of non-DPs in our extraction are only obtained once. Thus, if we regard low-frequency instances as Accidental DPs, 50% of non-DPs will be considered as Accidental DPs.

Property 4. *An error extraction (e isA C) triggered by a DP is usually supported by weak evidence, since the extraction is usually not triggered by other instances of C.*

Intuitively, this property holds for error instances triggered by DPs. By definition, an error extraction triggered by a DP will not be triggered by non-DPs. Besides, it seldom happens that the same error instances will be triggered by different DPs according to Property 1. For example, *beef* is triggered by DP *chicken* for the "Animal" class in Fig. 1(b). It turns out when we extract isA relationships from billions of sentences, no instances other than *chicken* would trigger the extraction of *beef* as an instance of "Animal".

2.4 Problems Definition

Although the above four properties provide some clues to DPs and drifting errors caused by DPs, none of them is definite. In this paper, we introduce machine learning approaches that take advantage of the four properties to solve the following two problems: i) Identify the two types of DPs when we extract isA pairs for millions of concepts (Sec. 3); ii) Identify drifting errors introduced by the two types of DPs (Sec. 4).

3. DRIFTING POINTS DETECTION

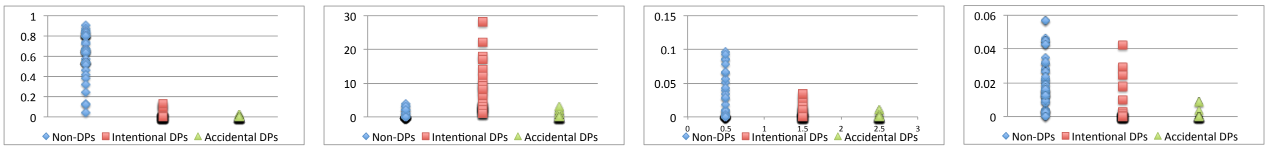
We resort to supervised learning methods for detecting DPs. For each concept, a *DP Detector* is trained to classify instances into three categories: (1) Intentional DPs, (2) Accidental DPs, or (3) non-DPs. One drawback of supervised learning is that it requires large amount of training data. To make things worse, we have millions of concepts.

In our approach, we use semi-supervised learning to 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. This is particularly necessary as lots of concepts do not have much training data among the millions of concepts. In the following, we start with feature selection, then we describe how to obtain a set of seed DPs and non-DPs, and finally, we present our method called Concept Adaptive Drift Detection for our purpose.

3.1 Designing Features

In the DP detector of the target concept C, each instance e is represented by a feature vector $x(e) = [f_1(e), f_2(e), ..., f_d(e)]^T \in \mathbb{R}^d$, where d is the number of features. Four features corresponding to the four properties of the DPs are explored here. 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. 3.

(1) Feature with Property 1: According to Property 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



(a) $f_1(e)$ (b) $f_2(e)$ (c) $f_3(e)$ (d) $f_4(e)$

Figure 3: Comparing the feature values of Intentional DPs, Accidental DPs and non-DPs under the "Animal" concept

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.

As we can see in Fig. 3(a), for $f_1(e)$, non-DPs mostly have much higher values than DPs, and the accidental DPs have very small values since they mostly only have very few sub-instances which are usually not correct instances of the concept. The values of Intentional DPs are usually a bit higher than Accidental DPs, but not as high as most of the non-DPs.

(2) Feature with Property 2: The second feature corresponding to Property 2 is the number of C's mutually exclusive concepts that also obtain e as their instance. That is,

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

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

As we can see in Fig. 3(b), all of the Intentional DPs have $f_2(e)$ values larger than 2, and most of the non-DPs and Accidental DPs have this feature value as 0. However, there are also a small number of non-DPs and Accidental DPs that have values larger than 0, because one such non-DP or Accidental DP might be accidentally (and incorrectly) learned as an instance of concept C' though $C' \perp C$.

(3) Feature with Property 3: The third feature corresponds to Property 3. Here we adopt a scoring function, denoted as $score(.)$, to estimate the probability of each instance being correct. The higher $score(e)$, the more likely that e is a good instance to the concept.

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

Most of the previous ranking methods [17, 15] tend to give higher scores to instances with higher frequencies. But frequency is not a good indicator, as it is common that the frequency of a drifting error is higher than that of a correct instance. In this paper, we employ a random walk based ranking model [23] to score instances. In particular, 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.

As we can see in Fig. 3(c), for $f_3(e)$, since both non-DPs and Intentional DPs are correct instances of the concept, they usually have relatively higher scores than Accidental DPs, which are usually drifting errors of the concept.

(4) Feature with Property 4: According to Property 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,

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

As we can see in Fig. 3(d), since sub-instances of non-DPs are usually correct, but have different scores due to different popularities, the $f_4(e)$ values of non-DPs under the "Animal" concepts distribute almost uniformly between 0 and 0.06. The sub-instances of Intentional DPs include both correct instances and drifting errors, thus the average scores of its sub-instances are mostly lower than 0.04. The sub-instances of Accidental DPs are drifting errors with low scores, thus the feature values of Accidental-DPs are very small (mostly less than 0.01).

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 [24]. 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.

3.2.1 *Using the Mutually Exclusive Assumption with Millions of Concepts*

Previous work [5] also used mutual exclusion for data cleansing. However, it is for certain pre-identified pairs of concepts (e.g., "cities" and "politicians"). When millions of concepts are involved in the extraction, it is impossible to pre-identify all pairs of concepts that are mutually exclusive or highly similar. As an alternative, we introduce a similarity measure to identify mutually exclusive concepts. We call the isA pairs extracted in the first iteration as *core pairs*. The core pairs have high quality, because we have not considered any ambiguous sentences yet. For a concept C, we call its instances in the core pairs as its *core instances*, and we denote them as $Core(C)$. The similarity between concepts C_1 and C_2 is defined as:

$$Sim(C_1, C_2) = Cosine(Core(C_1), Core(C_2)) \tag{5}$$

where $Cosine$ measures the cosine similarity between two sets. Fig. 4 shows the distribution of concept pairs for different similarity scores. From experimental studies on our labeled data, we found that 0.0001 is good threshold for mutually exclusiveness. That is, if $Sim(C_1, C_2) < 0.0001$, then C_1 and C_2 are mutually exclusive concepts.

We also explore the idea of *highly similar* concepts. Two concepts (e.g., "nations" and "countries") are considered highly similar if instances belonging to one concept are likely to belong to the

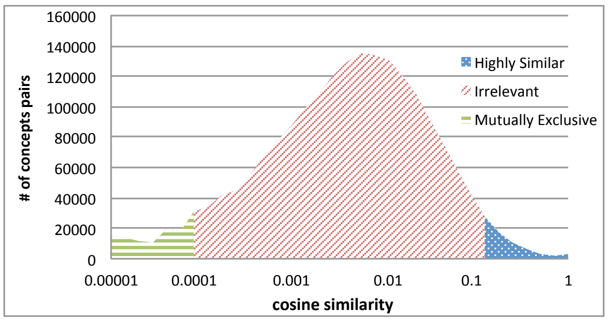


Figure 4: The distribution of cosine score between concepts

other. From observation in our experiments, we consider C_1 and C_2 highly similar if $Sim(C_1, C_2) > 0.1$. We could safely take the Mutually Exclusive concepts of a concept C as the Mutually Exclusive concepts of a concept C', if C and C' are highly similar concepts.

3.2.2 *Defining Evidenced Correct and Incorrect Instances*

In order for detecting obvious DPs and non-DPs, we also need to find some evidenced correct or incorrect instances based on mutually exclusive concepts. We say an instance is an *evidenced correct* instance of a concept if this isA relation is obtained from some verified sources or is among the highly frequent extracted pairs in the first iteration. In contrast, we say an instance is an *evidenced incorrect* instance of a concept if this instance is accidentally extracted for this concept for only once in some latter iteration other than the first one, but it is recognized as an evidence correct instance of some other concept that mutually exclusive to this concept. For example, assume "City" and "Country" are mutually exclusive, if we have evidence that *New York* is an instance of "City", but *New York* was accidentally extracted as an instance of "Country" only once in some latter iteration other than the first one, then we say *New York* is an evidenced incorrect instance of "Country".

3.2.3 *Heuristic Rules for Labeling Seed Instances*

To this point, we introduce 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% 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% of the one million concepts, and the left 33.6% 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 [19]. Hilbert space is an abstract vector space possessing the structure of an inner produce that allows length and angle to be measured [19]. The advantage of Hilbert space let us be able to perform full rank kernel Principal Component Analysis (PCA) [19] 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 very 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 (PCA) [19] 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 $\mathcal{X} = \{x_1, x_2, ..., x_n\}$ be the original feature representations, where $x_i \in \mathbb{R}^d$ denotes the i-th instance, and d is the dimension of features. Let $\phi : \mathbb{R}^d \to \mathcal{H}$ 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 $\alpha =$

$[\alpha_1, ..., \alpha_n]^T$ are coefficients such that $V = \sum_{i=1}^n \alpha_i$ [19], to obtain the new representations $\tilde{\mathcal{X}} = \{\tilde{x}_1, \tilde{x}_2, ..., \tilde{x}_n\}$. Let α_1, α_2, ..., α_r be the normalized eigenvectors corresponding to all the non-zero eigenvalues $0 < \lambda_1 \leq ... \leq \lambda_r$. 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 $\tilde{x}_j^p = \sum_{i=1}^r \alpha_i^p \left(\phi(x_i)\phi(x_j)^T\right)$, which composes $\tilde{x}_j = [\tilde{x}_j^1, \tilde{x}_j^p, ..., \tilde{x}_j^r]^T$.

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, $y_i^c = [0,0,1]$. 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 $F_c : \mathbb{R}^r \to \{0,1\}^3$, 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:

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

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 λ 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 labelled 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. Firstly, we learn the classifier starting with a small set of labeled instances and increasingly involving unlabeled instances. Secondly, we learn classifiers for the t concepts simultaneously such that we can share the knowledge among concepts to obtain better performance. This learning process brings challenges on defining the regularization function $\Omega(.)$. Next, we discuss how to define $\Omega(.)$ in two schemata.

1. Leveraging unlabeled data: We propose to exploit the manifold structure of both the labeled and unlabeled data via a statistical approach. Denote $\mathcal{N}_k(\tilde{x}_i)$ as the k-nearest neighbor set of $\tilde{x}_i$ including itself. Inspired by [26], for any instance $\tilde{x}_j \in \mathcal{N}_k$, we assume that its label can be predicted by its k-nearest neighbors via a local prediction function $f_i^c(\tilde{x}_j)$. For each $\tilde{x}_i$, we define $f_i^c(\tilde{x}_j) = (p_i^c)^T \tilde{x}_j + b_i^c$ where p_i^c is the local predictor, and b_i^c is the bias term. In all, we will have n local prediction functions for the n instances.

The inconsistency between the local prediction function f_i^c and W_c is $\|f_i^c(x_j) - W_c^T x_j\|_F^2$. To exploit the distribution of the unlabeled data, we propose to smooth the disagreement between the local prediction functions and the global prediction function by minimizing the following:

$$\min_{W_c, p_i^c} \sum_{i=1}^n \sum_{\tilde{x}_j \in \mathcal{N}_k(\tilde{x}_i)} \left(\|(p_i^c)^T \tilde{x}_j + b_i^c - W_c^T \tilde{x}_j\|_F^2 + \|p_i^c\|_F^2\right), \tag{9}$$

where the minimization of $\|p_i^c\|^2$ controls the complexity of the local predictor p_i^c. Denote $\tilde{X}_i = [\tilde{x}_i, \tilde{x}_{i_1}, ..., \tilde{x}_{i_k}]$, where each $\tilde{x}_{i_j}$ $(1 \leq j \leq k)$ is a k-nearest neighbor of $\tilde{x}_i$. We rewrite (9) as:

$$\min_{W_c, p_i^c} \sum_{i=1}^n \left(\|\tilde{X}_i^T p_i^c + b_i^c \mathbf{1}_{k+1} - \tilde{X}_i^T W_c\|_F^2 + \alpha\|p_i^c\|_F^2\right), \tag{10}$$

where α is the regularization parameter of the local function, $\mathbf{1}_{k+1}$ is vector of all ones in $(k+1)$-dimension. By setting the derivatives of (10) to zero w.r.t. b_i^c and p_i^c, we have:

$$b_i^c = \frac{1}{k+1}(\tilde{X}_i^T W \mathbf{1}_{k+1} - (p_i^c)^T \tilde{X}_i \mathbf{1}_{k+1}) \tag{11}$$

$$p_i^c = (\tilde{X}_i H \tilde{X}_i^T + \lambda I)^{-1} \tilde{X}_i H \tilde{X}_i^T W_c^{(i)} \tag{12}$$

where $H = I - \frac{1}{k+1}\mathbf{1}_{k+1}\mathbf{1}_{k+1}^T$ is the centering matrix and I is the identity matrix. Substituting (11) and (12) into (10) and denoting $\tilde{X} = [\tilde{x}_1, ..., \tilde{x}_n]$, we can rewrite Eq. (10) as follows:

$$\min_{W_c} Tr(W_c^T \tilde{X} (\sum_{i=1}^n S_i L_i S_i^T) \tilde{X}^T W_c), \tag{13}$$

where $Tr(\cdot)$ is the trace operator of a matrix, $S_i \in \{0,1\}^{n\times(k+1)}$ is a selection matrix in which $(S_i)_{u,v} = 1$ if $\tilde{x}_u$ is the v-th element in $\mathcal{N}_k(\tilde{x}_u)$, and $(S_i)_{u,v} = 0$ otherwise. And

$$L_i = H - H\tilde{X}_i^T (\tilde{X}_i H \tilde{X}_i^T + \lambda I)^{-1} \tilde{X}_i H. \tag{14}$$

Integrating $\|W_c\|_F^2$ and (13) into the regularization term of (8), we have the following object function for training a DP detector:

$$\begin{aligned} \min_{W_c} \sum_{i=1}^m &\|W_c^T \tilde{x}_i - y_i^c\|_F^2 + \\ &\lambda\left(Tr(W_c^T \tilde{X} (\sum_{i=1}^n S_i L_i S_i^T) \tilde{X}^T W_c) + \beta\|W_c\|_F^2\right). \end{aligned} \tag{15}$$

where β is the regularization parameter of W_c when we minimize $\|W_c\|_F^2$. Until now, the object function (15) is convex, which can be readily solved.

2. Multi-Concept Learning: In (15), the DP detectors of different concepts are trained separately. Next, we illustrate how we adapt knowledge among different concepts for a more discriminating Drift Detection. 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.

Suppose there are t concepts $\{C_1, C_2, ..., C_t\}$, with corresponding classifiers for their DP detectors as $\{W_1, W_2, ..., W_t\}$. We define a matrix $W = [W_1, W_2, ..., W_t]^T \in \mathbb{R}^{3t\times r}$ which is the horizontal concatenation of the DP detectors. Note that W encodes the information of all the drift detectors. With the new denotion W, we further extend (15) by training the DP detectors of different concepts in a joint framework. As shown in [8], sparse models can be used to exploit the shared information among multiple variables. In light of this, we propose to minimize the following object function, which trains the t DP detectors simultaneously.

$$\min_{W_c|_{c=1}^t} \sum_{c=1}^{t} \Bigg[\sum_{i=1}^{m} \|W_c^T \tilde{x}_i - y_i^c\|_F^2 + \lambda \Big(Tr(W_c^T \tilde{X} (\sum_{i=1}^{n} S_i L_i S_i^T) \tilde{X}^T W_c) + \beta \|W_c\|_F^2 \Big) + \Big(\sum_{i=1}^{r} \sum_{j=1}^{3t} |W_{ij}^{\frac{1}{2}}| \Big)^{\frac{1}{2}} \Bigg] \tag{16}$$

Let us define

$$A = \tilde{X} (\sum_{i=1}^{n} S_i L_i S_i^T) \tilde{X}^T. \tag{17}$$

Let $\tilde{X_c^l} = [\tilde{x}_1, ...\tilde{x}_m]$ be the data matrix comprising all the labeled training data for the c-th concept. Then (16) can be written as

$$\min_{W_c|_{c=1}^t} \sum_{c=1}^{t} \left\|(\tilde{X_c^l})^T W_c - Y_c\right\|_F^2 + \lambda \left(\sum_{c=1}^{t} Tr(W_c^T A W_c) + \beta \left\|W\right\|_{2,1} + \gamma \left\|W\right\|_F^2 \right) \tag{18}$$

where γ is the global regularization parameter.

Through optimizing (18), the t DP detectors of different concepts can be trained simultaneously, in which $\sum_{i=1}^{r} \sum_{j=1}^{3t} |W_{ij}^{\frac{1}{2}}|$ uncovers the shared structure of different detectors via $\ell_{2,1}$-norm minimization. In that way, the shared knowledge of different concepts can be transferred among the t drift detectors, thereby resulting in more reliable detectors.

Next, we illustrate how to optimize the object function (18). Let w^i be the i-th column of W, i.e., $W^T = [w^1, ..., w^r]^T$. We define a diagonal matrix D with its diagonal elements $D_{ii} = \frac{1}{2\|w^i\|}$. The problem in (18) is then equivalent to:

$$\min_{W_c|_{c=1}^t} \sum_{c=1}^{t} \left\|(\tilde{X_c^l})^T W_c - Y_c\right\|_F^2 + \lambda \left(\sum_{c=1}^{t} Tr(W_c^T A W_c) + \beta Tr(W^T D W) + \gamma \left\|W\right\|_F^2 \right) \tag{19}$$

By setting the derivative of (19) *w.r.t.* W_c to 0, we have:

$$\begin{aligned} &2\tilde{X}_c^l(\tilde{X}_c^l)^T W_c - 2\tilde{X}_c^l Y_c + 2\lambda A W_c + 2\lambda\beta D W_c + 2\lambda\gamma W_c = 0 \\ &\Rightarrow W_c = \left(\tilde{X}_c^l(\tilde{X}_c^l)^T + \lambda A + \lambda\beta D + \lambda\gamma I\right)^{-1} \tilde{X}_c^l Y_c \end{aligned} \tag{20}$$

Therefore, we can use Algorithm 1 to train the t detectors.

Algorithm 1: Multi-Concept DP Detectors Training

```
1: Initialize $W_c$ for $c = 1, ..., t$ randomly;
2: Compute $L_i|_{i=1}^n$ according to (14);
3: Compute $A$ according to (17);
4: repeat
       Update each diagonal element of $D$ by $D_{ii} = \frac{1}{2\|w^i\|}$
       for $c \leftarrow 1$ to $t$ do
           Calculate $W_c|_{c=1}^t$ according to (20)
       end
   until The value of (18) Convergence;
5: Return $W_c|_{c=1}^t$.
```

Theorem 1. *The objective function value of Eq. (18) monotonically decreases in each iteration until convergence.*

Theorem 1 guarantees the convergence of the optimizing process in Algorithm 1. Due to the limitation of space, we put the proof of Theorem 1 in the appendix.

4. DRIFTING ERRORS CLEANING WITH DETECTED DPS

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, but we check whether each extraction triggered 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 Triggered 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 ways of extraction, and score each of them using a probabilistic model based 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 of a sentence s is triggered by an Intentional DP (C_e, 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 sentences. Apparently, $C_e \in \mathbf{C}_s$ and $e \in \mathbf{E}_s$. We define the score that $C \in \mathbf{C}_s$ is the correct concept of the extraction as:

$$Score(s, C) = \sum_{e' \in \mathbf{E}_s} \left(\frac{score(C, e')}{\sum_{C' \in \mathbf{C}_s} score(C', e')} \right) \tag{21}$$

where $score(C, e')$ is the random walk score of an isA pair (C, e'). If $Score(s, C_e)$ is not the highest score among all $Score(s, C)$, where $C \in \mathbf{C}_s$, we say the extraction of s triggered by (C_e, e) is a drifting error, and it will be rolled back.

EXAMPLE 1. *Given a sentence:*

$s =$ *"<u>food</u> from <u>animals</u>* **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 random walk scores of every candidate concept and instance pair.*

(pork, food, 0.15),	*(pork, animal, 0.001),*
(beef, food, 0.10),	*(beef, animal, 0.002),*
(chicken, food, 0.35),	*(chicken, animal, 0.250).*

We have: $Score(s, \text{"animal"}) = \frac{0.001}{0.001+0.15} + \frac{0.002}{0.002+0.10} + \frac{0.250}{0.250+0.35} = 0.006 + 0.019 + 0.416 = 0.441$. *On the other hand, we have* $Score(s, \text{"food"}) = 2.556$. *As a result, the current extraction triggered by (chicken isA animal) is not the one with the highest score, so we will roll back the extraction.*

4.2 Rolling Back DP Triggered Extractions

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 roll 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 can only be 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.

5. EXPERIMENTS

In our experiments, we evaluate the effectiveness of our DP detection method, and compare the performance of our approach with existing approaches.

5.1 Data Set and Ground Truth

We use the Probase data[1] released by the Probase people for 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 magnitude larger than the previously known largest corpus [18]. From these sentences, we used 7 hours and a cluster of 10 servers to extract isA pairs using the semantic-based iterative extraction method. 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% were obtained in the first 10 iterations.

As presented in Figure 5(a), 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% in iteration 1 to lower than 50% after 5 iterations.

To set up the ground truth for evaluation, we manually labeled 1115 Intentional DPs, 2290 Accidental DPs, 4408 non-DPs, 4519 correct instances, and 5979 drifting errors under 20 different concepts (listed in Table 1). 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 1, 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%.

5.2 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 in 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 [13] 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 2 lists the average precision of top 100, 1000 or 2000 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.

[1]http://research.microsoft.com/en-us/projects/probase/

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 2: The precision of top 100, 1000 or 2000 instances

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 3: Comparing cleaning performance with other methods

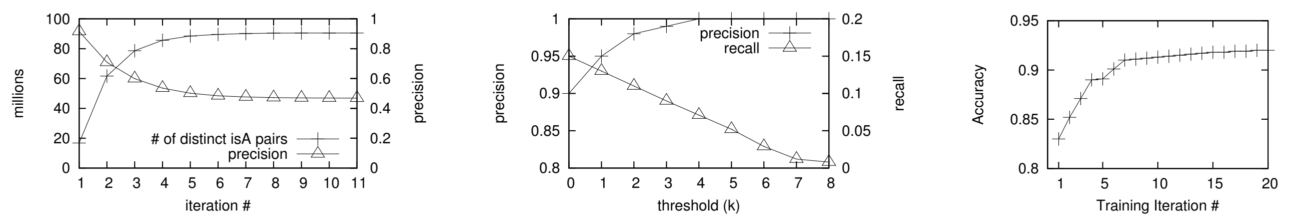
2. Training Data Set Preparation: A threshold k is used in defining strong evidence for finding obvious DPs and non-DPs. As depicted in Figure 5(b), the average percentage of identified DPs and non-DPs decreases from 15% to 0.8% as k increases from 0 to 8. On the other hand, the average precision of the labeled data increases from 0.902 to 0.993 as k increases from 0 to 3. We could reach 100% 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% of instances as labeled data, and 92.9% of instances as unlabeled data.

5.3 Our Approach v.s. 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 [14, 4], 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 5.2.

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 3, although Mutual Exclusion and Type Checking reach a high precision in removing drifting error isA



(a) The number and precision of isA pairs (b) The precision and recall of the labeled data (c) The performance over iterations

Figure 5: A number of figures for the experiments

concept	#Instances	#Correct instance	#Error instance	Error %	#Intent. DPs	#Accid. DPs	#Non-DPs
animal	16280	626	508	0.4479	32	256	809
asian country	784	74	165	0.6903	43	56	68
child	17313	993	1479	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	7264	235	785	0.7696	128	258	232
computer software	1878	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	3193	276	122	0.3065	18	67	238
president	480	70	25	0.2631	2	4	67
religion	3107	11	10	0.4761	49	143	79
student	19930	100	424	0.8091	30	189	1241
u.s. state	123	58	51	0.4678	5	13	53
weather	828	294	224	0.4324	11	29	72
woman	11502	929	1467	0.6122	203	453	907
Overall	**87246**	**4519**	**5979**	**0.5695**	**1115**	**2290**	**4408**

Table 1: The statistics on our manually labeled instances under 20 different concepts

pairs (91.19% and 94.23%), the recall of removing bad entities is pretty low (15.70% and 14.51%), 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%) in identifying drifting errors, but on the other hand, relatively low precision (56.21% and 57.53%). 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%) and recall (91.45%). After the cleaning, the precision of the remained entities is 89.21%, which is much higher than that with other methods. Besides, the recall of the remaining entities is still high (93.93%), which means that only a small number of good entities are taken as drifting errors by mistakes.

5.4 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 property in Section 2.2 with a well-learned threshold.

As listed in Table 4, 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 labelled training data. By using unlabeled data, the Semi-Supervised Learning could reach 5% higher precision and 13% higher recall. Finally, with the multi-task learning, the precision and recall can be further improved about 2% and 4% 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 Figure 5(c). 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.

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**

Table 4: Comparing the effectiveness of DP detection methods

5.5 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.

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**

Table 5: The evaluation results on DP 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 5, under most of the 20 concepts, the strategy could successfully identify more than 95% of bad sentence extraction with about 90% precision. This contributes to the final DP cleaning results.

2. Rolling back DP-Triggered Extractions: After identifying the DPs, we perform DP-based cleaning by rolling back extractions triggered by the DPs. Table 5 (from column 4 to column 7) shows the DP cleaning results under the 20 concepts. We can on average recognize 91.45% of error instances with 96.96% 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% to 89.21%, which is almost the same with the precision in iteration 1. Besides, only 6.07% of isA pairs are sacrificed, which is acceptable comparing to the great improvement of precision.

6. RELATED WORK

Semantic drift has been known as a common problem in iterative information extraction (IE). 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. 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 [27]. 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. [17] 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 [17], or combining evidences from multiple extractions [21, 2, 16]. 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.

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 [14, 4], 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.

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% of errors with more than 95% precision. After cleaning, the precision of extracted isA pairs is improved from about 50% to 90%. As a future work, we will adopt our method to overcome semantic drift happening to other types of relations in IE.

8. REFERENCES

[1] E. Agichtein and L. Gravano. Snowball: Extracting relations from large plain-text collections. In *JCDL*, pages 85–94, 2000.

[2] M. Cafarella, J. Madhavan, and A. Halevy. Web-scale extraction of structured data. *ACM SIGMOD Record*, 37(4):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 *AAAI*, pages 1306–1313, 2010.

[4] A. Carlson, J. Betteridge, R. Wang, E. Hruschka Jr, and T. Mitchell. Coupled semi-supervised learning for information extraction. In *WSDM*, pages 101–110, 2010.

[5] J. Curran, T. Murphy, and B. Scholz. Minimising semantic drift with mutual exclusion bootstrapping. In *PACLING*, pages 172–180, 2007.

[6] O. Etzioni, M. Banko, S. Soderland, and D. Weld. Open information extraction from the web. *Communications of the ACM*, 51(12), 2008.

[7] O. Etzioni, M. Cafarella, D. Downey, S. Kok, A. Popescu, T. Shaked, S. Soderland, D. Weld, and A. Yates. Web-scale information extraction in knowitall. In *WWW*, pages 100–110, 2004.

[8] A. Evgeniou and M. Pontil. Multi-task feature learning. In *NIPS*, volume 19, page 41, 2007.

[9] Y. Fang and K. C.-C. Chang. Searching patterns for relation extraction over the web: rediscovering the pattern-relation duality. In *WSDM*, 2011.

[10] J. Finkel and et. al. Incorporating non-local information into information extraction systems by gibbs sampling. In *ACL*, 2005.

[11] T. McIntosh and J. Curran. Reducing semantic drift with bagging and distributional similarity. In *ACL/AFNLP*, pages 396–404, 2009.

[12] F. Nie, H. Huang, X. Cai, and C. Ding. Efficient and robust feature selection via joint l21-norms minimization. In *NIPS*, 2010.

[13] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Technical report, Stanford Digital Library Tech. Project, 1998.

[14] M. Pasca, D. Lin, J. Bigham, A. Lifchits, and A. Jain. Names and similarities on the web: Fact extraction in the fast lane. In *COLING/ACL*, pages 809–816, 2006.

[15] J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 1988.

[16] M. Pennacchiotti and P. Pantel. Entity extraction via ensemble semantics. In *EMNLP*, pages 238–247, 2009.

[17] E. Riloff and R. Jones. Learning dictionaries for information extraction by multi-level bootstrapping. In *AAAI*, pages 474–479, 1999.

[18] A. Ritter, S. Soderland, and O. Etzioni. What is this, anyway: Automatic hypernym discovery. In *AAAI*, pages 88–93, 2009.

[19] B. Schölkopf, A. Smola, and K. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319, 1998.

[20] M. Skounakis and M. Craven. Evidence combination in biomedical natural-language processing. In *BIOKDD*, pages 25–32, 2003.

[21] 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 *EMNLP*, pages 582–590, 2008.

[22] M. Thelen and E. Riloff. A bootstrapping method for learning semantic lexicons using extraction pattern contexts. In *EMNLP*, 2002.

[23] H. Tong, C. Faloutsos, and J. Pan. Fast random walk with restart and its applications. In *ICDM*, pages 613–622, 2006.

[24] W. Wu, H. Li, H. Wang, and K. Zhu. Probase: a probabilistic taxonomy for text understanding. In *SIGMOD*, 2012.

[25] X. Yang and J. Su. Coreference resolution using semantic relatedness information from automatically discovered patterns. In *ACL*, 2007.

[26] 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. *TPAMI*, 34:723–742, 2012.

[27] R. Yangarber. Counter-training in discovery of semantic patterns. In *ACL*, pages 343–350, 2003.

[28] H. Yu and E. Agichtein. Extracting synonymous gene and protein terms from biological literature. *Bioinformatics*, 19(suppl 1):i340, 2003.

APPENDIX

Lemma 1. *Given a positive semi-definite matrix Q, PQP^T is a positive semi-definite matrix for any matrix P.*

PROOF. As Q is semi-definite, it can be written as $Q = B^TB$. Thus we have $PQP^T = PB^TBP^T$. Denote $C = PB^T$. We have $PQP^T = CC^T$. Therefore, PQP^T is a positive semidefinite matrix. □

The following is the proof of Theorem 1.

PROOF. As $(\tilde{X}_i^T\tilde{X}_i + \lambda I)^{-1}$ is a positive semidefinite matrix, according to Lemma 1, we can see that $H(\tilde{X}_i^T\tilde{X}_i + \lambda I)^{-1}H$ is a positive semidefinite matrix. Note that $H - H\tilde{X}_i^T(\tilde{X}_iH\tilde{X}_i^T + \lambda I)^{-1}\tilde{X}_iH = H(\tilde{X}_i^T\tilde{X}_i + \lambda I)^{-1}H$. Therefore, L_i is a positive semidefinite matrix. Then we have $q_i^TL_iq_i \geq 0\ \forall q_i \in \mathbb{R}^{k+1}$. Thus $\sum_{i=1}^n q_iL_iq_i^T \geq 0$, which indicates that $\sum_{i=1}^n S_iL_iS_i^T$ is a positive semidefinite matrix. According to Lemma 1, we can see that A is a positive semidefinite matrix. Denote $\tilde{W}_c = (\tilde{X}_c^l(\tilde{X}_c^l)^T + \lambda A + \lambda\beta D + \lambda\gamma I)^{-1}\tilde{X}_c^lY_c$. Then it can be inferred that:

$$\begin{aligned}&\sum_{c=1}^t \left\|(\tilde{X}_c^l)^T\tilde{W}_c - Y_c\right\|_F^2 + \lambda(\sum_{c=1}^t Tr(\tilde{W}_c^TA\tilde{W}_c)\\ &+\beta Tr(\tilde{W}^TD\tilde{W}) + \gamma\left\|\tilde{W}\right\|_F^2) \leq \sum_{c=1}^t \left\|(\tilde{X}_c^l)^TW_c - Y_c\right\|_F^2\\ &+\lambda(\sum_{c=1}^t Tr(W_c^TAW_c) + \beta Tr(W^TDW) + \gamma\|W\|_F^2)\end{aligned} \tag{22}$$

Therefore, we have

$$\begin{aligned}&\sum_{c=1}^t \left\|(\tilde{X}_c^l)^T\tilde{W}_c - Y_c\right\|_F^2 + \lambda(\sum_{c=1}^t Tr(\tilde{W}_c^TA\tilde{W}_c) + \beta\sum_{i=1}^r \frac{\left\|\tilde{w}^i\right\|_2^2}{2\left\|\tilde{w}^i\right\|_2}\\ &+\gamma\left\|\tilde{W}\right\|_F^2) \leq \sum_{c=1}^t \left\|(\tilde{X}_c^l)^TW_c - Y_c\right\|_F^2 + \lambda(\sum_{c=1}^t Tr(W_c^TAW_c)\\ &+\beta\sum_{i=1}^p \frac{\left\|w^i\right\|_2^2}{2\left\|w^i\right\|_2} + \gamma\|W\|_F^2)\\ &\Rightarrow \sum_{c=1}^t \left\|(\tilde{X}_c^l)^T\tilde{W}_c - Y_c\right\|_F^2 + \lambda\left(\sum_{c=1}^t Tr(\tilde{W}_c^TA\tilde{W}_c) + \beta\sum_{i=1}^p \left\|\tilde{w}^i\right\|_2\right.\\ &\left.-\beta(\sum_{i=1}^p \left\|\tilde{w}^i\right\|_2 - \sum_{i=1}^p \frac{\left\|\tilde{w}^i\right\|_2^2}{2\left\|\tilde{w}^i\right\|_2}) + \gamma\left\|\tilde{W}\right\|_F^2\right)\\ &\leq \sum_{c=1}^t \left\|(\tilde{X}_c^l)^TW_c - Y_c\right\|_F^2 + \lambda\left(\sum_{c=1}^t Tr(W_c^TAW_c) + \beta\sum_{i=1}^p \left\|w^i\right\|_2\right.\\ &\left.-\beta(\sum_{i=1}^p \left\|w^i\right\|_2 - \sum_{i=1}^p \frac{\left\|w^i\right\|_2^2}{2\left\|w^i\right\|_2}) + \gamma\|W\|_F^2\right)\end{aligned} \tag{23}$$

It has been shown in [12] that for any non-zero vectors $v_t^i|_{i=1}^r$:

$$\sum_i \left\|v_{t+1}^i\right\|_2 - \sum_i \frac{\left\|v_{t+1}^i\right\|_2^2}{2\left\|v_t^i\right\|_2} \leq \sum_i \left\|v_t^i\right\|_2 - \sum_i \frac{\left\|v_t^i\right\|_2^2}{2\left\|v_t^i\right\|_2} \tag{24}$$

where r is an arbitrary number. Thus, we can easily get the following inequality:

$$\begin{aligned}&\sum_{c=1}^t \left\|(\tilde{X}_c^l)^T\tilde{W}_c - Y_c\right\|_F^2 + \lambda(\sum_{c=1}^t Tr(\tilde{W}_c^TA\tilde{W}_c) + \beta\sum_{i=1}^p \left\|\tilde{w}^i\right\|_2\\ &+\gamma\left\|\tilde{W}\right\|_F^2) \leq \sum_{c=1}^t \left\|(\tilde{X}_c^l)^TW_c - Y_c\right\|_F^2 + \lambda(\sum_{c=1}^t Tr(W_c^TAW_c)\\ &+\beta\sum_{i=1}^p \left\|w^i\right\|_2 + \gamma\|W\|_F^2)\end{aligned} \tag{25}$$

$$\begin{aligned}&\Rightarrow \sum_{c=1}^t \left\|(\tilde{X}_c^l)^T\tilde{W}_c - Y_c\right\|_F^2 + \lambda(\sum_{c=1}^t Tr(\tilde{W}_c^TA\tilde{W}_c) + \beta\left\|\tilde{W}\right\|_{2,1}\\ &+\gamma\left\|\tilde{W}\right\|_F^2) \leq \sum_{c=1}^t \left\|(\tilde{X}_c^l)^TW_c - Y_c\right\|_F^2 + \lambda(\sum_{c=1}^t Tr(W_c^TAW_c)\\ &+\beta\|W\|_{2,1} + \gamma\|W\|_F^2)\end{aligned} \tag{26}$$

which indicates that the objective function value of (18) monotonically decreases until convergence. □