Abstract: Semantic annotation of images is an important step to support semantic information extraction and retrieval. However, in a multi-annotator environment, various types of conflicts such as converting, merging, and inference conflicts could arise during the annotation. We devised conflict detection patterns based on different data, ontology at different inference levels and proposed the corresponding automatic conflict resolution strategies. We also constructed a simple annotator model to decide whether to trust a given piece of annotation from a given annotator. Finally, we conducted experiments to compare the performance of the automatic conflict resolution approaches during the annotation of images in the celebrity domain by 62 annotators. The experiments showed that the proposed method improved 3/4 annotation-accuracy with respect to a naïve annotation system.
The Conflict Detection and Resolution in Knowledge Merging for Image Annotation

Cheng-Yu Lee¹ and Von-Wun Soo¹²
¹Department of Computer Science, National Tsing Hua University
HsinChu Taiwan 300
²Department of Computer Science and information engineering, National University of Kaohsiung, Kaohsiung, Taiwan, 811
E-mail: {leoli;soo}@cs.nthu.edu.tw

Abstract
Semantic annotation of images is an important step to support semantic information extraction and retrieval. However, in a multi-annotator environment, various types of conflicts such as converting, merging, and inference conflicts could arise during the annotation. We devised conflict detection patterns based on different data, ontology at different inference levels and proposed the corresponding automatic conflict resolution strategies. We also constructed a simple annotator model to decide whether to trust a given piece of annotation from a given annotator. Finally, we conducted experiments to compare the performance of the automatic conflict resolution approaches during the annotation of images in the celebrity domain by 62 annotators. The experiments showed that the proposed method improved 3/4 annotation-accuracy with respect to a naïve annotation system.

Keywords: semantic web, ontology, image annotation, conflict detection & resolution.
The Conflict Detection and Resolution in Knowledge Merging for Image Annotation

Cheng-Yu Lee¹ and Von-Wun Soo¹²
¹Department of Computer Science, National Tsing Hua University
HsinChu Taiwan 300
²Department of Computer Science and information engineering, National University of Kaohsiung, Kaohsiung, Taiwan, 811
E-mail: {leoli;soo}@cs.nthu.edu.tw

Abstract
Semantic annotation of images is an important step to support semantic information extraction and retrieval. However, in a multi-annotator environment, various types of conflicts such as converting, merging, and inference conflicts could arise during the annotation. We devised conflict detection patterns based on different data, ontology at different inference levels and proposed the corresponding automatic conflict resolution strategies. We also constructed a simple annotator model to decide whether to trust a given piece of annotation from a given annotator. Finally, we conducted experiments to compare the performance of the automatic conflict resolution approaches during the annotation of images in the celebrity domain by 62 annotators. The experiments showed that the proposed method improved 3/4 annotation-accuracy with respect to a naïve annotation system.

Keywords: semantic web, ontology, image annotation, conflict detection & resolution.

1. Introduction

To retrieve a desired image is not an easy task (Deb, 2004). In order to making historical images accessible by historians and general image-seekers, we had developed a serial of researches on semantic and ontology-based information retrieval systems for image retrieval (Soo & Lee, 2002–2003, 2005; Lee et. al., 2004). In these systems, annotators are asked to describe the images in terms of natural language texts in Mandarin Chinese or in a special format based on their understanding. Then, the system translates each statement into the standard rdf:triples (e.g. <Subject, Predicate, Object>) based on it’s domain knowledge. Finally, the system saves those triples as the annotations, which can be used in subsequent information retrieval or extraction, to describe the content of images.
One of the major motivations of Semantic Web (W3C, McGuinness & Harmelen, 2003, Motik & Glavinic, 2000, Hendler, Berners, & Miller, 2002, and DARPA) is to enable web resources to support semantic information extraction and retrieval based on such formal description language standards as RDF, DAML, and OWL. The main idea is to have a set of well-defined machine readable semantic tags accessible by general web users and allow them to define sharable ontologies for a specific domain. The software agents can then utilize the sharable ontology to conduct inferences and analyze the domain knowledge on web. Each piece of knowledge or annotation in our systems is represented in terms of a standard rdf:triple as <Subject-slot, Predicate-slot, Object-slot>.

In our previous work (Soo & Lee, 2002-2003), we concluded that the image annotation is a crucial step in a semantic image retrieval system and the quality and quantity of annotations could affect the performance of retrieval significantly. Therefore, we developed an annotation guide agent (AGA) (Lee & Soo, 2004; Lee et al., 2004) that could aid human annotators to achieve a more complete and useful annotations by asking critical properties and suggesting possible domain common sense. However, in a web-based image annotation system, the annotators could be worldwide and their annotation behaviors can be unpredictable. As we know, “an image is worth more than a thousand words” and “Every one's faults are not written in their foreheads”, the different viewpoint and different purpose of annotators may lead the annotations to inconsistency.

Conducting consistent and complete/useful image annotation at the same time is not a trivial task and could be also a trade-off. When the quantity of annotations has been scaled up by an automatic process (e.g. AGA) or multiple annotating strategies are adopted, the contradictions and conflicts of annotations can usually arise. Therefore, to manage the conflicts arising in the annotation task is the main theme of this paper. This paper intends to resolve the annotation conflicts issues within the framework of the semantic web knowledge representation. Since conflict resolution is a complicated and difficult task, this study instead attempts to prevent the generation of conflicts as much as possible at the annotation phase. The organization of this paper is: in section 2, the fundamental concepts and issues are introduced. In section 3, related work in knowledge and data conflicts is discussed. In this study we classify the conflicts into two types, the conspicuous conflicts and inconspicuous conflicts. The details will be discussed in the section 4 and section 5. In section 6, the experiments that were used to evaluate the performance of image retrieval based on the annotations facilitated by automatic conflict resolution methods are described. In section 7, we discuss the experimental results and make conclusions.
2. Fundamental Concepts and Issues Related to Conflict Resolution in Annotation

Before we get into the core, some assumptions and fundamental concepts are introduced in this section.

2.1 The Situations that a Conflict Happens
A possible conflict in our image annotation and retrieval system happens in the following phases (Figure 1).

- Conversion phase: a conflict happens in the translation from user’s annotation descriptions to the internal representation of the system.
- Merging phase: a conflict happens in merging the annotations between two different annotators (or between an annotator and a software agent) for the same image.
- Consistency of annotation: a conflict happens in the annotations (after the merging process). The annotation could still have potential contradictions in which are produced by a single annotator or by a merging process (Inference conflicts). For example, some descriptions could be right when we examine them individually, however, they may lead to a wrong conclusion when they are putting together, or they may contradict to a fact that the system might not know yet.

Figure 1. The possible situation that a conflict occurs.
2.2 A Right Annotation from Different Views

The annotation with multiple sources could face a difficult problem: “who/what is right?” As we mentioned, “an image is worth more than a thousand words”, to detect a conflict in annotations that come from different annotators with different view, can be in fact an unfeasible task. In this study, we adopt several methods to resolve the “right annotation” problem:

1. We adopt a single unique ontology as the base for translating the annotations and queries into semantic instances; it could map the annotations into a specific predefined domain. For example, the property of “height” can only be annotated in the numbers going with a dimension in our system, then the adjective descriptions such as “very tall” and “short” will be pruned and ignored.

2. Different types of properties could result in conflicts in different ways. For example, some property such as “FirstName” is a unique property for a specific person and it can’t be expressed in different terms, but other properties such as “Alias” could have multiple values and can be annotated by more than one term.

3. For those Chinese terms sharing the same semantic code in our thesaurus are treated as the synonym.

4. The referential correct annotations in this paper are predefined. Briefly speaking, for the retrieval purpose, the most popular terms (used by most people) could be treated as the most appropriate annotation, even it could be wrong.

2.3 Assumptions

This study resolve the possible conflicts encountered in image annotation based on the semantic web representation. Its assumptions are:

1. The ontology is unified represented and error-proof, but can be incomplete.

2. An annotator can access the previous annotations that have been done by others.

3. The annotation will be translated into the semantic rdf:triple <Subject, Predicate, Object/Value> correctly.

4. The language in the system is in Mandarin Chinese; but the ontology is established in English (except some value in an object-slot that is expressed in Chinese).

2.4 Domain Knowledge

2.4.1 Domain Ontology

Domain ontology is an essential part in our system and many other semantic web researches (Cranefield, 2001, Bernd & Irini 1999, Smith & Welty, 2004, Patel & Hayes, 2004, Staab & Erdmann, 2000, Decker & Melnik, 2000, Doan & Madhavan, ...
Based on the previous works, we realize that the expressing power and completeness of the ontology tend to affect the performance of the system. Therefore, we adopt the semantic web standard OWL to represent ontology in terms of rdf:triples. 

In our current domain ontology, there are 14 classes and 129 properties in the ontology that was constructed by the tool Protégé 2000 (Protégé). The concept classes described in the ontology are described in Figure 2.

Figure 2. The inheritance relations of classes in our domain ontology.

Figure 2 represents some inheritance relations among 14 classes (the class “Thing” is a pseudo-class in Protégé 2000), in which the solid arrows denote the “SubClassOf” property and the dotted arrows denote the “DisjointWith” property. For example, the class “Animal” is a super class of “Person”, and is disjoint with “Item” which is another subclass of “Physical Thing”. We also have 129 additional properties in this domain for describing the relations among the 14 classes. Figure 3 shows the properties that are related to “Animal”. In this case, there are seven data-type properties to describe “Animal”. For example, “hasGender” is related to two attribute values (“Male” or “Female”), “hasAge” to an integer, and “has Alias” to a string. Thirteen object properties that establish the relations between classes; For example, “hasMother” is related to another animal and “Live” is related to a location. The properties with a star mark such as “hasCategoryName” indicate that they are inherited from a super class “Physical Thing”.

Someone may question about the disjoining relation between Animal and Item: “Some items such as a teddy bear might have an animal looks but not a real animal.
bear”. To deal with such a problem, we devise a property named “MimicTo” to specify an animal-like item.

![Diagram of Animal relations]

Figure 3. The conceptual relations of an “Animal”.

2.4.2 The Mandarin Chinese Thesaurus

The thesaurus is another important knowledge source for a software agent. It could provide term association; in particular: synonym, generalized (broader), and specialized (narrow) terms. We have constructed a Mandarin Chinese thesaurus based on a Chinese thesaurus known as “Tong-Yi Tsu Lin”, the total size of lexicons is more than 70,000 words that is organized in a semantic hierarchy, unfortunately, the antonym terms were not specified expressively. The hierarchy is divided into 4 levels: first level is the most abstracted layer that consists of Person, Article, Action, etc. and the fourth level as the most specific layer consisting of the synonyms of specific concept words. There are 12 categories/concept words in the 1st, 94 in the 2nd, and 1428 in the 3rd level respectively.

2.4.3 The Common Sense Inference Rules

The common sense is a knowledge source for the system to reduce the efforts that might be needed for a human annotator in annotating an image. Common sense features refer to the simple, familiar and well-known facts that could be inferred in the
problem domain (Liu & Singh, 2003, Liberman, 2002-2004). The system could use statistical correlation learning method to correlate the common sense facts in any open information resources. Once the correlation among domain facts is established, it does not require human experts to intervene and can infer automatically the highly correlated facts implied in the information resources. In the annotation problem domain, we represent common sense as the association inference rules that can infer the concept association of one annotation based on others. One way to acquire the common sense association is to find triples (simple facts) that tend to have the same value in a class of objects based on their occurrence frequencies.

[Example 1] Supposed that an annotator was annotating an image of a “general” and the AGA had recorded many of its annotation instances as: “the person’s occupation is a General, and the person’s gender is Male”. According to the annotations, the AGA might be able to infer that the “gender” of a General could be a “Male” since most instances in the instance pool that were annotated as a “General” turned out to be a “Male” also.

Association rules (Han & Kamber, 2000) are used to implement the presupposition common sense (abbreviated as common sense below). An association rule can be expressed in terms of an associated relation that consists of two parts: the prerequisite (the left-hand-side of the rule) and the outcome (the right-hand-side of the rule) denoted as:

\[
R_i: <\text{Prerequisite}> \rightarrow <\text{Outcome}>
\]

It indicates that a prerequisite implies the outcome. And both prerequisite and outcome could be a triple (\(<\text{Subject, Predicate, Object}>\)). So a chain of relations may exist, e.g. (\(<\text{triple 1}>\rightarrow <\text{triple 2}>\rightarrow <\text{triple 3}>\)).

[Example 2] Supposed that all instances in Example 1 that are annotated as \(<\text{Person, hasOccupation, “general”}>\) has a high frequency to be annotated as \(<\text{Person, hasGender, “male”}>\) as well, this correlation fact can be written as an association rule:

\[
R_1: <\text{Person, hasOccupation, “General”}> \rightarrow <\text{Person, hasGender, “Male”}>
\]

If the confidence and support measures of \(R_1\) are greater than a predefined threshold, it will be treated as a common sense. Later, when an annotator is annotating a new image using the triple \(<\text{Person, hasOccupation, “General”}>\), the AGA will trigger this rule and suggests \(<\text{Person, hasGender, “Male”}>\) as a possible common sense. It
would be up to the human annotator to decide if the inferred common sense is correct. The way to determine an association rule as a common sense is using the confidence and support measures to decide if an association rule is well justified. If both the confidence and support measures of a rule exceed their thresholds, AGA will claim the corresponding triple as a common sense. The confidence and support measures are defined below.

**Definition**

Supposed there are totally $u$ subjects in all instances (one instance may be comprised of more than one subject) and there are $\alpha$ subjects that contain prerequisites (one or more triples), and $\beta$ subjects that contain both prerequisites and outcomes. The **confidence** and **support** of an association rule $R_i$ are defined as:

\[
\text{Confidence}(R_i) = P(<\text{Outcome}|<\text{Prerequisite}>) = (\beta/u)/(\alpha/u) = \beta/\alpha, \quad (1)
\]

\[
\text{Support}(R_i) = P(<\text{Prerequisite} \cap <\text{Outcome}>) = \beta/u. \quad (2)
\]

Confidence($R_i$) denotes the chance that the outcome of $R_i$ exists in a corpus given the prerequisite of $R_i$ and Support($R_i$) denotes the ratio of the number of cases that the requisite and the outcome of $R_i$ coexist over the number of total cases that refer to the same subject.

### 2.5 The Reliability of Knowledge and an Annotator

The reliability of an annotator denotes a simple user model for the annotator. Since annotators do not behave consistently and uniformly (Lee & Soo, 2004), to identify the type of an annotator is one of the difficult tasks for system to assist annotation. In order to rank the degree of trustworthy for an annotator, we define a measure called **annotator faith degree (AFD)** to indicate the reliability of an annotator.

**Definition**

**Annotator faith degree** (AFD) is a real number between $[0,1]$ that is a measure of how much an AGA could trust the annotator. It denotes as AFD($a$), where $a$ is an annotator.

We also define knowledge faith degree (KFD) as a measure of the reliability of a
piece of knowledge. In real world, no all knowledge is correct and accurate all the time. The uncertainty of knowledge usually leads to inconsistency and conflicts. Therefore recording a KFD for each piece of knowledge could help the system to deal with conflicts.

**Definition**

*Knowledge faith degree* (KFD) is a real number between [-1,1] that is a measure of how much the system or a human annotator could trust a piece of knowledge. Denote as $KFD_a(k)$, where $a$ is an actor (the AGA or a human annotator) and $k$ is the piece of knowledge.

A little different from AFD, the number of KFD lies in the interval [-1, 1]. “-1” means that the system believes a piece of knowledge is absolutely wrong. On the other hand, the value should be “1” if the piece of knowledge is absolutely right. And a “0” denotes unknown. If we trust some one $a$ with $AFD(a)=\delta$, and $a$ believe in a piece of knowledge $\lambda$ that $KFD_a(\lambda)=\zeta$, so we can conclude that we believe in $\lambda$ through $a$ is $KFD(\lambda)=AFD(a)*KFD_a(\lambda)=\delta*\zeta$. On the other hand, if $a$ believe that $\lambda$ is wrong with faith $\zeta$, then $KFD_a(\lambda)=-\zeta$, our belief of $\lambda$ through $a$ will be $AFD(a)*KFD_a(\lambda)=-\delta*\zeta$ as well.

As we mentioned above, the piece of knowledge in this paper is expressed in terms of a triple <Subject, Predicate, Object>, so a KFD value is attached to each triple. The KFD of the predefined knowledge in the thesauruses and domain ontology is set as either -1 or 1 (assumption 1 in section 2.1) and the KFD’s of commonsense inference rules are based on their support and confidence values as in formula (3).

$$KFD_d(R) = \text{Confidence}(R) \times \text{Enhanced\_Function}(\text{Support}(R))$$  \hspace{1cm} (3)

As usual, the support values of most association rules are low (e.g. the accepted rules are between 0.2 and 0.3 in our system), and they have a great gap with the confidence value (over 0.9), so we use an Enhanced function to increase the weighting of support when it is above the threshold (the rule had been accepted). The enhanced function of Support must map its value into the range of [0, 1]. The function $f(x) = x^\gamma$ [x to the power $\gamma$, where $\gamma$ is set as 0.1 in our system] is chosen as the enhanced function in our system, as show in Figure 4.
The KFD of the inference results is calculated as:

$$\text{KFD}_a(R(k)) = \text{KFD}_a(R) \times \Pi_k(\text{KFD}_a(k)),$$

where $R(k)$ denotes the inferred results of rule $R$ with an input (Prerequisite) $k$. The KFD of a piece of annotation $k$ basically depends on the reliability of its annotator.

$$\text{KFD}_a(K_b(k)) = \text{AFD}(b) \times \text{KFD}_b(k)$$

where $K_b(k)$ denotes that the knowledge $k$ is provided by annotator $b$.

Figure 4: The enhanced mapping function between the original support value $x$ and enhanced support value $f(x)$.

[Example 3] The KFD of an annotation $TI$ by system $s$ is a product of AFD of annotator $a$ and its KFD of annotation $TI$ (usually set as 1 or $-1$). And the KFD of an inferred result by rule $RI$ is the product of KFD of rule $RI$ and the KFD of its prerequisite $P$. Supposed that $\text{KFD}_a(TI) = 1$, $\text{AFD}(a) = 0.8$, $\text{Confidence}(RI) = 0.9$, $\text{Support}(RI) = 0.3$, $P$ of $RI$ is $TI$. Then

$\text{KFD}_a(K_a(TI)) = \text{AFD}(a) \times \text{KFD}_a(TI) = 0.8 \times 1 = 0.8$,

$\text{KFD}_s(RI) = \text{Confidence}(RI) \times \text{Enhanced\_Function}(\text{Support}(RI)) = 0.9 \times 0.886 = 0.798$,

$\text{KFD}_s(RI(TI)) = \text{KFD}_s(RI) \times \text{KFD}_a(P) = 0.798 \times \text{KFD}_a(K_a(TI)) = 0.798 \times 0.8 = 0.64$,

So the KFD of $TI$ provided by $a$ is 0.8, and the KDF of the inference result according
to $R_i$ and its prerequisite $T_i$ by the system $s$ is 0.64.

3. Related work

The research on conflicts can be divided into two major paradigms, the knowledge merging conflicts (Ram & Park, 2004, Chomicki & Lobo, 2000-2003, and Amgoud & Parsons, 2002) and the resource competition conflicts (Beroggi & Mirchandani, 2000, and Arai & Sycara, 2000). In semantic web community, the conflicts are usually referred to the knowledge merging conflicts.

Similar to the database community (Ram & Park, 2004, Chomicki & Lobo, 2000-2003, and Amgoud & Parsons, 2002), the semantic web community has to maintain the integrity of various pieces of knowledge. However, in comparison with database community, the semantic web community has to deal with more different types of knowledge sources and a more complicated ontology. Since the metadata descriptions cannot be guaranteed to be absolutely correct, even if the ontologies are unified represented and shared, the domain ontology usually can be still very complicated and the annotators may possess quite different concepts about the domain knowledge.

Many conflict problems in the semantic web community can’t be resolved using the database methods, because the ontology is different from the database schema (much larger and more complex). In addition, the knowledge sources in semantic web are quite open and the knowledge providers (annotators) are usually not under control and the annotations depend heavily upon their viewpoints. So the annotation conflict problems cannot be resolved in a simple manner. They should be resolved individually based on different causes of conflict situations.

A little different from other rule base conflict detecting approach (Grosof, 1997 and Dietrich & Kozlenkov, 2003), which usually detecting conflicts based on an extended rule base, the inconspicuous conflicts that we proposed tend to focus on detecting the conflicts that haven’t be considered (perfectly), such as a defect in an ontology, a contradictory conflict after inferences, and an erroneous outcome from a faulty annotation (knowledge).

The related research communities to the annotation one are text-based information retrieval and extraction communities known as TREC. However, most of the work in TREC (Aronson et al., 2004; Ferres et al., 2004; Harabagiu et al., 2003; Hou et al., 2003; Lee et al., 2004; Lin et al., 2002; Settles and Craven, 2004; Tong, 2004) did not address the potential conflicts in their benchmark corpus caused by annotation. The annotations could be conducted by a single annotator or a team of annotators; but only a final standard annotation is used as benchmark for performance evaluation. For
example: the question answering systems (Ferres et al., 2004; Harabagiu et al., 2003), usually attempt to find the highest ranked answers rather than to resolve the possible contradictions in the answers.

4. The conspicuous conflicts

As mentioned in end of section 1, the conflicts are classified into two types, a conspicuous conflict and an inconspicuous conflict. The conspicuous conflict is discussed in this section.

**Definition**

A **Conspicuous Conflict**: A conflict between two knowledge pieces (triples) that are related to each other with at least one attribute. The knowledge pieces could include the annotations from an annotator, the ontology (OWL) and background knowledge (the commonsense and a thesaurus).

4.1 The Completeness of Conspicuous conflicts

As the definition: two triples are related conspicuously only when at least one attribute are related to each other. A conspicuous conflict could happen at seven possible situations as shown in Table 1.

<table>
<thead>
<tr>
<th>situation</th>
<th>subject</th>
<th>predicate</th>
<th>object</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Equal</td>
<td>Not equal</td>
<td>Not equal</td>
</tr>
<tr>
<td>2</td>
<td>Not equal</td>
<td>Equal</td>
<td>Not equal</td>
</tr>
<tr>
<td>3</td>
<td>Not equal</td>
<td>Not equal</td>
<td>Equal</td>
</tr>
<tr>
<td>4</td>
<td>Equal</td>
<td>Equal</td>
<td>Not equal</td>
</tr>
<tr>
<td>5</td>
<td>Equal</td>
<td>Not equal</td>
<td>Equal</td>
</tr>
<tr>
<td>6</td>
<td>Not equal</td>
<td>Equal</td>
<td>Equal</td>
</tr>
<tr>
<td>7</td>
<td>Equal</td>
<td>Equal</td>
<td>Equal</td>
</tr>
</tbody>
</table>

Strictly speaking, “not equal subject” (situations 2, 3, and 6) can not be treated as conspicuous conflicts since they have different subjects. On the other hand, situations 1 and 5 would not become a conflict if they are sanctioned by the domain ontology. Therefore we classify the possible conspicuous conflicts into two types, data conflicts that deal with situations 4 and 7 and ontology conflicts that deal with others.

4.2 Data conflicts
In our system, the data conflicts are defined as follows:

[Definition]
Data conflicts: A single annotation (triple) of an annotator conflicts with the existing knowledge pieces in the value (Object) slot of the triples.

The possible data conflicts for a specific subject in our domain include the following cases:

1. Duplicated data of a limited-cardinality property: Annotating a previously annotated annotation with limited-cardinality (functional) property repeatedly.
2. Different values of a limited-cardinality property: Annotating with a different value to a previously annotated annotation with limited cardinality (functional) property. It includes (a) Synonym term, (b) Generalized term, (c) Specialized term, (d) Antonym term, and (e) Unrelated/different-view term.
3. Duplicated data of an unlimited-cardinality property: Annotating a previous annotated annotation with unlimited-cardinality property repeatedly.
4. Different values of an unlimited-cardinality property: Annotating a different value to a previous annotated annotation with unlimited-cardinality property. It can also have five possible situations as in case 2.
5. Annotation at different abstraction levels: Different annotators may use concepts at different abstraction levels to describe the same thing. It includes: (a) Different subject granularities, (b) Different property granularities, (c) Different class object granularities, and (d) Different data object granularities.

For example, the different of annotation between “Person” and “Animal”, “hasColor” and “hasSkinColor”, as well as “Taiwan” and “Taipei”.

4.3 Data conflict Detection and Resolution

In order to detect the data conflicts, several simple detecting rules are shown in Table 2. When the system receives a new annotation triple NA:<S1, P1, O1> from an annotator, the system will check with the rules in Table 2 against previous annotation PA:<S2, P2, O2> one by one if any potential conflict can be detected. In table 2, O1 ≡ O2 indicates that O1 and O2 are synonyms, O2 = ¬ O1 indicates O1 and O2 are antonyms.

And the possible resolution actions that the system may take to resolve the detected
data conflicts are shown in Table 3. Basically, the actions that the system can take are: abandon the new one, add the new one, replace the old one with the new one, replace the old one with the modified new one, and update KFD. The action “Abandon NA” means the system does not accept the new piece of annotation NA, “Add NA” means the system accepts the new piece of annotation, “Replace PA by NA” and “Replace NA(PA) by NA*(PA*)” mean the system detects the conflict between the new and old annotations and decide to either to remove the old one or to make some modifications on the new piece of annotation respectively. In this study, we only modify the predicate of NA (PA) with its descendants, the new triple named NA*(PA*), when the PA have a specialized predicate, the NA will be modified, vice versa. “Update KFD” means the system would update the KFD of a piece of annotation if more than one annotator adds the same piece of annotation.

Table 2. Rules for detecting data conflicts.

<table>
<thead>
<tr>
<th>Rule#</th>
<th>Rule Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>Duplicated value</td>
<td>Find a &lt;S_2, P_2, O_2&gt; that S_1=S_2, P_1=P_2, and O_1=O_2.</td>
</tr>
<tr>
<td>R2</td>
<td>Synonym value</td>
<td>Find a &lt;S_2, P_2, O_2&gt; that S_1=S_2, P_1=P_2, and O_1 ≡ O_2.</td>
</tr>
<tr>
<td>R3</td>
<td>Generalized value</td>
<td>Find a &lt;S_2, P_2, O_2&gt; that S_1=S_2, P_1=P_2, and O_1 generalizes O_2.</td>
</tr>
<tr>
<td>R4</td>
<td>Specialized value</td>
<td>Find a &lt;S_2, P_2, O_2&gt; that S_1=S_2, P_1=P_2, and O_2 generalizes O_1.</td>
</tr>
<tr>
<td>R5</td>
<td>Antonym value</td>
<td>Find a &lt;S_2, P_2, O_2&gt; that S_1=S_2, P_1=P_2, and O_2 = ¬ O_1.</td>
</tr>
<tr>
<td>R6</td>
<td>Different values</td>
<td>Find a &lt;S_2, P_2, O_2&gt; that S_1=S_2, P_1=P_2, and O_2 ≠ (¬ O_1 or related to O_1).</td>
</tr>
<tr>
<td>R7</td>
<td>Functional-property</td>
<td>Find &lt;P_1, cardinality, 1&gt; or &lt;P_1, type, functional-property&gt;</td>
</tr>
<tr>
<td>R8</td>
<td>Generalized predicate</td>
<td>Find a &lt;S_2, P_2, O_2&gt; that S_1=S_2, and P_1 generalizes P_2 or P_2 generalizes P_1.</td>
</tr>
</tbody>
</table>

Table 3. Possible resolution actions for the detected data conflicts.

<table>
<thead>
<tr>
<th>Action #</th>
<th>Action name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>Abandon NA</td>
<td>Abandon NA &lt;S_1, P_1, O_1&gt;</td>
</tr>
<tr>
<td>A2</td>
<td>Add NA</td>
<td>Add NA &lt;S_1, P_1, O_1&gt; as a new instance</td>
</tr>
<tr>
<td>A3</td>
<td>Replace PA with NA</td>
<td>Remove PA &lt;S_2, P_2, O_2&gt; and Add NA &lt;S_1, P_1, O_1&gt;</td>
</tr>
</tbody>
</table>
According to the above conflict detection rules and resolution actions, we can resolve the data conflicts shown in Appendix A. For example: the “Duplicated data in limited-cardinality property” will be detected, if a new triple matches R1 (duplicated) or R7 (is functional) and the actions the system can take are: abandon the new annotation (A1) and update the KFD (A5).

Summarizing the above analysis, we integrate the detection and resolution of data conflicts in Table 4. Since the action “Add NA (A2)” is a standard action in our annotation system (if a new annotation is not abandoned, it will be added as an instance), the system can ignore it during conflict resolution. Be noted that that δ may be modified during the annotation process.

Table 4. The examination steps for detecting and resolving the data conflicts.

<table>
<thead>
<tr>
<th>Examine order</th>
<th>Detecting rules</th>
<th>Resolving Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>R8</td>
<td>A4</td>
</tr>
<tr>
<td>2</td>
<td>R1</td>
<td>A1, A5</td>
</tr>
<tr>
<td>3</td>
<td>R2, R7</td>
<td>A1, A5</td>
</tr>
<tr>
<td>4</td>
<td>R3, R7</td>
<td>A1, A5</td>
</tr>
<tr>
<td>5</td>
<td>R4, R7</td>
<td>A3</td>
</tr>
<tr>
<td>6</td>
<td>R5</td>
<td>If (AFD(p)&gt;AFD(c)+δ) A1 else (A3)</td>
</tr>
<tr>
<td>7</td>
<td>R6, R7</td>
<td>If (AFD(p)&gt;AFD(c)+δ) A1 else (A3)</td>
</tr>
</tbody>
</table>

Detecting rules that list in table 2 can be classified into three types: R1 to R6 – the possible states of object-slot, R7 – the cardinality number of predicates, and R8 – a generalized relationship between 2 predicates. Where the R1 to R6 will be a conflict when they co-occur with R7 (A limited cardinality with a contracted value), and R8 is a detecting rule which concert about the relationship between 2 predicates, therefore, the R8 should be considered first (if the R8 matched, system don’t need to consider
the other rules), then will be the R1 to R6 with R7 (the states of object-slot under the cardinality number of predicates). In which, since the R1 (Duplicated value) and R5 (Antonym value) will be a conflicts with entire state of cardinality number of the predicates, (that means duplicated/antonym is a conflict in any state of predicate.) so the R1 and R5 can be examined individually.

In this paper, we did not record the historical annotations (Tansel et al., 1989; Pissinou et al., 1994; Su and Chen, 1993) as the evidences to judge the correctness of annotation, we believe that the historical records of annotation could be very important information in conflict handling and also the contraction condition might change overtime as the reviewer had pointed out. However, it requires a huge storage to keep the historical records. For example, supposed that a single triple be modified \( n \) time, and an image consist of \( m \) triples, if we have \( p \) images in our image base, then we would have to prepare \( n \times m \times p \) additional annotation space for the historical records. Even we have taken the historical information into consideration, we still cannot ensure to infer the correctness of an annotated triple. Therefore, we resolve the conflicts base on the reliability of annotators instead, that have indirectly taken into the historical annotation information into consideration. We have conducted other research on the annotator model (Lee and Soo, 2005) that elaborate the annotator could be modeled precisely.

4.4 Ontology (schema/structure) conflicts

Although the assumption is based on a unified and sharable ontology, however, the annotators still may very often violate or misunderstand the ontology. We define the ontology conflicts as follows.

[Definition]

**Ontology conflicts**: The annotation of an annotator conflicts with the existing ontology.

The possible ontology conflicts in our domain include: (The details of ontology conflicts are described in Appendix B)

1. **Subject conflict.** The conflict causes a wrong description about a subject that can be further divided into four cases.
2. **Property conflict.** The property conflict leads to a wrong property description that can be further divided into three cases.
3. **Object conflict.** The object conflict leads to a wrong description about an object that can be further divided into three cases.
4. Irrelevant triples. If any two of three slots are wrong, it usually leads to an irrelevant triple among the three tuples (subject, predicate, and object).

We use several simple checking rules in Table 5 to detect the ontology conflicts. When the system receives a new annotation triple \(<S, P, O>\) from an annotator, it will examine the rules against the ontology and take the actions in Table 6 to resolve the conflicts. In Tables 5 and 6, we do not include the generalized class of a subject since if a class \(C\) is a child of another class \(C'\), \(C\) will inherit all the attributes of \(C'\), except some peculiar ones that has nothing to do with \(C'\), so does the omission of the specialization of a predicate.

In Table 5, where the “Unrelated(x)” actions (actions 4 and 5) are adopted only when the number of candidates of \(x\) is less than a threshold constant (2 or 3 candidates), otherwise, it will be replaced by the “Abandon” action. Therefore, the analysis for resolving the ontology conflicts is shown in Appendix C. For example: the “Using a wrong but related subject \(S\)” will be detected, if the subject of the new triple is not related to the predicate (there is no relation between the subject and the predicate denote as \(\neg<S,P>\)), however, the predicate is related to object \(<P,O>\), and there is a descendant \(S^*\) of \(S\) that makes the \(S^*\) be related to \(P\) (Specialized\((S)\)), so the actions the system can take are: Suggest a new triple \(<S^*,P,O>\) such that \(S^*=\text{Specialized}(S)\).

Table 5. Detection rules for potential ontology conflicts.

<table>
<thead>
<tr>
<th>Rule#</th>
<th>Rule Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>S</td>
<td>Determine if (S) exists in ontology.</td>
</tr>
<tr>
<td>2</td>
<td>P</td>
<td>Determine if (P) exists in ontology.</td>
</tr>
<tr>
<td>3</td>
<td>O</td>
<td>Determine if (O) exists in ontology.</td>
</tr>
<tr>
<td>4</td>
<td>(&lt;S,P&gt;)</td>
<td>Determine if (&lt;S,P&gt;) is valid in ontology.</td>
</tr>
<tr>
<td>5</td>
<td>(&lt;S,O&gt;)</td>
<td>Determine if (&lt;S,O&gt;) is valid in ontology.</td>
</tr>
<tr>
<td>6</td>
<td>(&lt;P,O&gt;)</td>
<td>Determine if (&lt;P,O&gt;) is valid in ontology.</td>
</tr>
<tr>
<td>7</td>
<td>\text{Specialized}(S)</td>
<td>Determine if (&lt;\text{Specialized}(S),P&gt;) is valid in ontology.</td>
</tr>
<tr>
<td>8</td>
<td>\text{Similar}(S)</td>
<td>Determine if (&lt;\text{Similar}(S),P&gt;) is valid in ontology.</td>
</tr>
<tr>
<td>9</td>
<td>\text{Generalized}(P)</td>
<td>Determine if (&lt;S,\text{Generalized}(P),O&gt;) is valid in ontology.</td>
</tr>
<tr>
<td>(A^*)</td>
<td>\text{Cardinality}(P)</td>
<td>Cardinality ((P)=n), and # of (&lt;S,P,{O}\rangle &gt; n)</td>
</tr>
</tbody>
</table>

Table 6. Possible actions to resolve the ontology conflicts.

<table>
<thead>
<tr>
<th>Action #</th>
<th>Action name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>\text{Specialized}(S)</td>
<td>Suggest a new triple (&lt;S',P,O&gt;) such that (S'=\text{Specialized}(S)).</td>
</tr>
<tr>
<td>2</td>
<td>\text{Similar}(S)</td>
<td>Suggest a new triple (&lt;S',P,O&gt;) such that that (S'=\text{Similar}(S)).</td>
</tr>
</tbody>
</table>
3  Generalized(P)  Suggest a new triple <S,P',O> such that P'=Generalized(P).

4  Unrelated(P)  Suggest a new triple <S,P',O> where P' is a property of S, and O is a legal value of P'.

5  Unrelated(S)  Suggest a new triple <S',P,O> where S' is a subject that has property P.

6  Abandon  Abandon <S,P,O>

7* Remove to fit  Remove <S,P,{O}> while # of <S,P,{O}> = n according to KFD

5. The Inconspicuous Conflicts

In this section, we discuss the conflicts that are not conspicuous conflicts but they could cause inconsistency of knowledge and are known as inconspicuous conflicts. We propose ways to detect the inconspicuous conflicts and some rough methods for resolutions actions.

[Definition]

Inconspicuous conflicts: Different pieces of knowledge (either in annotations or in system knowledge) might not always show conspicuous conflicts. However, the combination of those pieces of knowledge may sometimes lead to inconsistency based on the background knowledge. Therefore we can also call it knowledge inconsistency conflicts.

5.1 The Completeness of Inconspicuous Conflicts

Since “an image is worth more than a thousand words” and the range of annotation cannot be limited, the completeness of the inconspicuous conflicts cannot be guaranteed. In this study, we only discuss the possible inconspicuous conflicts that can happen. The conflicts might be due to the lack of critical elements in the current version of semantic web ontology standard OWL (include 3 types), unreasonable annotations or might happen after an inference processing. The possible inconspicuous conflicts include five types.

5.2 The Types of Inconspicuous Conflicts

1. One-to-one property. The functional property P(x,y) may imply another functional property P'(y,x). However, the lack of one-to-one attribute mapping in the ontology
language may cause some one-to-one properties to become in conflict. For example, 
<Person1, hasBody, Body1> and <Person2, hasBody, Body1> becomes a conflict, 
because “hasBody” is a one-to-one property (namely, two different persons can’t own 
the same body in real world).

2. Not a Relation
The semantic web ontology languages (either DAML or OWL) lack of the capability 
to describe the “Negative attribute” of a property. It may lead to serious conflicts. For 
example, the property “hasAncestor” is a transitive property and the property 
“hasFriend” is a symmetric property. But we rarely declare that the “hasAncestor” 
absolutely is NOT a reflexive and symmetric property, and “hasFriend” is not a 
transitive relation, (since if Peter is a friend of mine, and John is Peter’s friend, these 
two facts can’t influence the friendship between me and John). So the declaration of 
relations should include is-relation (Positive), is-Not-relation (Negative), and 
No-relation (not need to declare). We propose four kinds of is-NOT-relation that 
include
(a) Not reflexive. The property P is-Not-reflexive implies APA is a illegal 
description. For example, “hasFather” is-Not-reflexive implies that <Person1, 
hasFather, Person1> is a conflict.
(b) Not symmetric. P is-Not-symmetric implies APB and BPA are conflicts. For 
example, “hasFather” is-Not-symmetric implies that <Person1, hasFather, 
Person2> and <Person2, hasFather, Person1> are conflicts.
(c) Not transitive. P is-Not-transitive implies APB, BPC, and APC are conflicts. 
For example, “hasFather” is-Not-transitive implies that <Person1, hasFather, 
Person2>, <Person2, hasFather, Person3>, and <Person1, hasFather, Person3> 
are conflicts.
(d) Not Cyclic. P is-Not-cyclic implies APB, BPC, and CPA are conflicts. For 
example, “hasFather” is-Not-cyclic implies that <Person1, hasFather, 
Person2>, <Person2, hasFather, Person3>, and <Person3, hasFather, Person1> 
are conflicts.

3. Mutual exclusion
Most property relations can coexist for the same subject-object pair. For example, the 
property relations “hasBoss” and “hasFather” can coexist for the same pair between 
your father and you because your father can be your boss as well. However, some 
property relations cannot coexist for the same subject-object pair. For example, the 
property relations of “hasFather”, “hasBrother” and “hasSister” are such kinds of 
properties. Therefore <Person1, hasFather, Person2>, <Person1, hasBrother, Person2>, 

19
and <Person1, hasSister, Person2> are in conflict because Person1 and Person2 can only have a unique relation among “hasFather”, “hasBrother”, and “hasSister” and cannot coexist for two same pair of persons.

4. The data conflicts after logical inference

Logical inference may lead to many conflicts since the rules may not be suitable at all situations. To discuss the inferential conflicts easily, we define the following symbols:

- \( k_i \): a single piece of knowledge in term of a triple <S, P, O>
- \( r_i \): a single (common sense) inference rule represented as \(<\text{Prerequisite}> \rightarrow \langle\text{Outcome}\rangle\)

\[ K_x : \text{knowledge and beliefs of actor } x \text{ represented in terms of a set of triples } \{k_1, k_2, \ldots, k_n\} \]

\[ R_x : \text{inference rules of actor } x \text{ (inference rules) as a set of rules } \{r_1, r_2, \ldots, r_m\} \]

\[ IK_x(K_x, R_x) : \text{the inferred knowledge base of } K_x \text{ after applying inference rules } R_x. \]

The data conflicts caused by logical inference include the following four cases.

(a) Lack of knowledge may induce the inference conflicts. If two annotators have different set of knowledge (usually they do), the one with less knowledge may infer differently from the one with more knowledge and thus cause inferred data conflicts.

[Example 4] Supposed there are two annotators \( a \) and \( b \).

\( K_a = \{k_1, k_2\}, K_b = \{k_1\}, R_a = R_b = \{r_1, r_2\} \)

\( k_1 = \langle\text{Person1, hasFirstName, “Jacky”}\rangle. \)

\( k_2 = \langle\text{Person1, Live, Location1}, \text{<Location1, hasLocationName, “HongKong”}\rangle. \)

\( r_1 : \langle\text{Person, hasFirstName, “Jacky”}\rangle \rightarrow \langle\text{Person, hasGender, “female”}\rangle \)

(“Jacky usually named female at USA). \)

\( r_2 : \langle\text{Person, hasFirstName, “Jacky”}\rangle \text{ and } \langle\text{Location1, hasLocationName, “HongKong”}\rangle \rightarrow \langle\text{Person, hasGender, “male”}\rangle \)

(“Jacky usually is a name of a male at HongKong). \)

\( K_a(k_1,k_2,r_2) = \{\langle\text{Person1, hasGender, “male”}\rangle\}. \)

\( IK_a(k_1,r_1) = \{\langle\text{Person1, hasGender, “female”}\rangle\}. \)

Therefore annotators \( a \) and \( b \) have inferred different genders for Person1 and it causes a data conflict.

(b) A wrong piece of knowledge may cause inferred data conflicts. The mistakes of annotations may lead to inference failure and it happens quite often.
[Example 5] What if the \( k_2^* \) of \( b \) is wrong?

\( K_a = \{k_1\}, K_b = \{k_2^*\}, R_a = R_b = \{r_1\} \)
\( k_1 = <\text{Person1, hasGender, “male”}> \).
\( k_2^* = <\text{Person1, hasFirstName, “Michel”}>. \) (Typo error of “Michael”)

\( r_1: <\text{Person, hasFirstName, “Michel”} > \rightarrow <\text{Person, hasGender, “female”}> \)

(Michel usually named female).
\( K_a = \{<\text{Person1, hasGender, “male”}>\} \).
\( IK_b(k_1,r_1) = \{<\text{Person1, hasGender, “female”}>\} \).

Then annotators \( a \) and \( b \) apparently lead to inferred data conflict.

(c) The faulty inference rules cause inferred data conflicts. The faulty or improper inference rules may lead to inferred data conflicts especially when the inference rules are generated by an imperfect process (either the training cases are too small or too old or are barrowed from other domains).

[Example 6] \( K_a = \{k_1, k_2\}, K_b = \{k_1\}, R_a = R_b = \{r_1^*\} \)
\( k_1 = <\text{Person1, hasLastName, “Chen”}>. \)
\( k_2 = <\text{Person1, hasGender, “female”}>. \)

\( r_1^*: <\text{Person, hasLastName, “Chen”} > \rightarrow <\text{Person, hasGender, “male”}> \)

(Chen is usually a family name of a male if the erroneous inference rule is unfortunately induced from a military domain where everyone is a male).
\( K_a = \{<\text{Person1, hasLastName, “Chen”}>, <\text{Person1, hasGender, “female”}>\} \).
\( IK_b(k_1,r_1^*) = \{<\text{Person1, hasLastName, “Chen”}>, <\text{Person1, hasGender, “male”}>\} \).

Then annotators \( a \) and \( b \) apparently lead to inferred data conflict.

(d) Using different set of inference rules may lead to different influence results. The rules may be conflict with each other, especially when rules are generated from a statistical manner and the relaxations of chosen thresholds may lead to conflicts between different set of rules.

[Example 7] If the rules are established when most soldiers are males and some data shows that Mary is a name of a female, it will bring data conflicts.
\( K_a = \{k_1, k_2\}, K_b = \{k_1, k_2\}, R_a = \{r_1\}, R_b = \{r_2\} \)
\( k_1 = <\text{Person1, hasFirstName, “Mary”}>, k_2 = <\text{Person1, hasOccupation, “Soldier”}> \)

(Mary is a soldier).

\( r_1: <\text{Person, hasOccupation, “Soldier”} > \rightarrow <\text{Person, hasGender, “male”}> \)
(A soldier usually is a male)

\[ r_2: <\text{Person, hasFirstName, “Mary”}> \rightarrow <\text{Person, hasGender, “female”}> \]

(Mary is usually a name of a female)

\[ IK_a (k_1, k_2, r_1) = <\text{Mary, hasGender, “male”}> \]
\[ IK_b (k_1, k_2, r_2) = <\text{Mary, hasGender, “female”}> \]

Then annotators \( a \) and \( b \) apparently lead to inferred data conflict.

5. Unreasonable annotation

An unreasonable annotation refers to a triple that is in conflict with the real world knowledge that the system has not yet established. In some cases, the conflicts occur when some annotator makes an unreasonable annotation so that it is in conflict with the real world knowledge. The following case is an example that all annotations are legal except the last one that is unreasonable because the dogs in real world have no wings at all.

[Example 8]

\[ <\text{Animal1, hasCategoryName, “Dog”}>, \]
\[ <\text{Animal1, hasBody, Body1}>, \]
\[ <\text{Body1, hasBodyPart, BodyPart1}>, \]
\[ <\text{BodyPart1, hasBodyPartName, “Wing”}> \]

Since the piece of knowledge “Dogs has no Wing” is not in the system knowledge base, it is unlikely for the system to detect this unreasonable annotation. However, it is very difficult to formulate all unreasonable pieces of knowledge since they cannot be exhaustively enumerated. It is also related to the well-known frame problem in knowledge representation.

5.3 The Detection and Resolution of Inconspicuous Conflicts

Most of conspicuous conflicts may be detected and resolved by simple rules such as in PDL (Chomicki & Lobo, 2000). However, the conflicts due to “incomplete ontology”, “contradictory inference”, or “unreasonable facts” could not be detected by simple detecting rules. It is because their occurrences are based on an incomplete and inconsistent knowledge or the correctness of an annotation. And it is a matter of “who/what can be trusted?” instead of “who/what is correct?”. Therefore, the inconspicuous conflict detection and resolution could be a complex task.

The inconspicuous conflicts in the following cases are due to the incompleteness of domain ontology and they can’t be detected and resolved without related background
knowledge. Therefore the first strategy we adopt is to add some additional features into the domain ontology. However, we also need to determine if a property is attached with the features to prevent the conflicts. In the following cases, “P” denotes the prevention actions.

1. One-to-one property
   **P**: add an attribute to declare a property as a “OneToOne” Property.

2. Not a Relation
   **P**: add an attribute to declare a property as a “isNotRelation” Property.

3. Mutual exclusion
   **P**: add a relation to have the property <MutualExclusion>
   <MutualExclusion> Other properties </MutualExclusion>.

Given a input annotation <S,P,O>, the detection and resolution rules are:

a) One to one property (a functional property P(x,y) implies another functional property P'(y,x))
   **D**: Find <P, type, OneToOne> & <S', P, O>.
   **A**: Remove <S, P, O> or <S', P, O> according to KFD.

b) Not a Relation
   **D**: Find <P, type, isNotReflexive> and S=O.
   **A**: Abandon <S, P, O>.
   **D**: Find <P, type, isNotSymmetric> & <O,P,S>.
   **A**: Remove <S, P, O> or <O, P, S> according to KFD.
   **D**: <P, type, isNotTransitive > & {<S, P, O>,T1,T2} are Transitive.
   **A**: Remove <S, P, O> or T1 or T2 according to KFD.
   **D**: Find <P, type, isNotCyclic> & <O, P, O'> & <O', P, S>.
   **A**: Remove <S, P, O> or <O, P, O'> or <O', P, S> according to KFD.

c) Mutual exclusion
   **D**: Find <P, isMutualExclusion, P'> & <S, P', O>.
   **R**: Remove <S, P, O> or <S', P', O> according to KFD.

More complicated actions could be devised to resolve those conflicts. In this study, however, the only action we adopt to resolve the conflicts of incomplete ontology is simply to “Remove one of the triples”. The properties that could be the seed of potential inconspicuous conflicts of above description in the system are described in Appendix D.

4. Data conflicts after logical inference.

   In the previous section, we detect the data conflicts by simple detecting rules. However, we cannot use the same resolution actions when the inference rules are involved. The involvement of inference rules complicates the conflict resolution by
increases the potential solution space. The detecting and resolving methods are:

(a) **Inference conflicts due to the lack of a piece of knowledge**

The conflict arises when an annotator adds a new piece of knowledge and the new knowledge invokes a different rule inference $R_2(K^*)$.

$$K \rightarrow R_1(K), K^* \rightarrow R_2(K^*), \text{ and } R_1(K) \text{ conflicts with } R_2(K^*)$$

where $K$ is the original knowledge, $K^*$ is the extended knowledge after adding new piece of knowledge into $K$. The Prerequisite of $R_2$ includes the Prerequisite of $R_1$.

**D: Data conflicts in both annotations are induced by two separate inference rules that have a subset relation between their prerequisites.**

**A: Choose $k$ with the most $KFD_a(R,(k))$ as the annotation.**

(b) **Wrong piece of knowledge induces inference conflicts.**

(c) **The faulty inference rules induce data conflicts.**

(d) **Different rule sets lead to inconsistent inference.**

The last three types of inference conflicts occur when a wrong piece of knowledge exists in the annotations or inference rules. The possible cases can be:

1. The new annotation invokes a rule that infers an annotation that is in conflict with a previous annotation.
2. The new annotation is in conflict with a previous annotation that is inferred by an inference rule.
3. The annotation invokes a rule that infers an annotation that is in conflict with a previous annotation that is inferred by some inference rule.

**D: Data conflicts that are induced by some inference rules over either one or both annotation(s) and the two inference rules do not have a subset relation.**

**A: Remove $R_i$ or $k_i$ with less $KFD_a(R_i)$ or $KFD_a(k_i)$ from**

(a) Previous human annotation,

(b) New human annotation +$\delta$,

(c) Rules that are activated by a previous annotation, and

(d) Rules that are activated by a new annotation.

**Until the conflict is resolved.**

5. Unreasonable annotation

An unreasonable annotation could affect the performance of image retrieval if it is not removed. However, as we discussed above an unreasonable annotation usually violates some facts that are hard to completely specify. Hence, we try to avoid it by two methods 1) using an annotator model (AFD) and 2) by common knowledge
inspection. The former is to update the AFD of an annotator according to his/her behaviors and annotations. For example, if an annotator whose annotations tend to be wrong in a conflict or whose annotations often lead to wrong retrieval results, the system will decreases her/his AFD with respect to the annotation. The latter is to use the information resources in the world wide web as the largest knowledge base and use such search engine as Google™ or Yahoo™ to search for related information in the web in order to check the validity of a piece of annotation (similar to a semantic indexing approach (Deerwester & Dumais, 1990, Dumais & Letsche, 1997, and Lee & Lee, 2005). The method can be briefly described as follows.

**D:** For a new annotation \(<S, P, O>\), if its predicate \(P\) is in the testable set \(TS\) and the annotator is not trustworthy, the system inspects \(n\) of triples whose predicates related to the annotation are in \(TS\).

**A:** If \(n^*\) of the \(n\) triples fail in the inspection, then abandon the annotation, else accept it.

The common knowledge inspection computes the coexisting ratio of two data values of a chosen predicate after querying the search engine by two separate terms or by two terms together respectively. The computation of coexisting ratio includes a) how many documents have been retrieved and b) how close of the two terms are in each document. For example, we search two terms, “Dog” (animal, hasCatalogName) and “Wings” (bodypart, hasbodypartName) using Google™ in Mandarin Chinese. The query using the combined term “DogWings” retrieved only 2 documents, and the query using separate terms “Dog” and “Wings” retrieved 14,600 documents. In contrast, if we search using two terms “Chicken” (animal, hasCatalogName) and “Wing” (bodypart, hasbodypartName) in Mandarin Chinese as queries, it retrieved 48,200 documents using the combined term “ChickenWings”, and it retrieved 48,300 documents using two separate terms “Chicken” and “Wings”. Moreover, we observed that the distances between “Chicken” and “Wings” were in general much closer than those between “Dog” and “Wing”, even the latter could still retrieved up to 14,600 documents. It implied “Chicken” and “Wings” could be more reasonable terms together than “Dog” and “Wings”. In other words, “Dog” has body part “Wings” can be unreasonable annotation.

### 5.4 Prevent the Interacting Conflicts among Conflict Resolutions

In the above conflict resolution process, a resolution of a conflict may sometimes
cause a new conflict with the resolution of another conflict. For example, a one-to-one property is the superset of a functional property and we cannot process them separately. Supposed that two annotations exist in the current knowledge base, T1: <Person1, hasBody, Body1> and T2: <Person2, hasBody, Body2> with KFD(T1) < KFD(T2). When a new annotation T’<Person1, hasBody, Body2> with KFD(T1) < KFD(T’) < KFD(T2), is added, T’ will replace T1 because “hasBody” is a functional property. However, T’ is also in conflict with T2, because the “hasBody” is also a one-to-one property, and T’ will be abandoned because its KFD is lower than T2. As a result in this case, abandon T1 could be a mistake, since the T’ could be a wrong annotation with respect to the T2 and should not be used to abandon T1.

In order to prevent the mistake during conflict resolutions, we adopted two separate queues as Removing Queue and Adding Queue to keep the annotations to be removed and added respectively. Those annotations that are waiting to be added to the knowledge base are kept in the Adding Queue while those annotations that are to be removed (no matter due to a conflict with the adding annotations or due to the logical inference from annotations that are in conflict with the adding annotations) from the knowledge base are kept in the Removing Queue. In the following example, T2 is kept in the Adding Queue, therefore, T1 is kept in the Removing Queue since T1 is in conflict with T2, and T3 and T4 are also kept in the Removing Queue since T3 and T4 are inferred from T1.

**Adding Queue = {<T2>}
Removing Queue = {(<T2>DC)<T1>, (<T1>→)T3, (<T1>→)T4}**

Where the description between brackets indicates the reason of removing, “DC” stands for “data conflicts” and “→” stands for “inference”.

Moreover, if T2 is not to be added for any reason (T2 was removed from the Adding queue before adding as an instance), then T1, T3 and T4 are taken away from the Removing Queue. Since the dependency and interactions among the annotations is taken care only at a single round of annotation in the system, it would not guarantee the consistency in the future (we cannot guarantee what will happen after T3 and T4 are removed from the instance, and there still could have conflicts between the Adding queue and the Removing queue. In this paper, the Adding queue has more priority than Removing queue, and it only overcomes part the problems. However, the conflict resolution strategies would gradually lead to more consistent and complete annotations.

6. Experiments & Discussion

6.1 The Initial Settings for Annotation Experiments
In order to illustrate the conflict detection and resolution problems in annotation, we adopt the domain on the images of celebrities, since there are more social relations among human subjects that could be tested for many kinds of annotation conflicts in comparison to the other domains. We asked annotators to annotate at such restricted aspects as human relations and some fundamental profile for a person (first name, last name, age, occupations, etc.) to increase the chance of conflicts. But we still left annotators some freedom to make their own annotations on a person. We chose ten images. Each of the images has one to three celebrities consisting of famous politicians, artists, and cartoon characters (totally 22 un-repeated persons). 62 students including some graduates and undergraduates were grouped randomly into 7 groups. Everyone in each group was asked to annotate the same three images that were randomly chosen from the ten images. In this way, each image was annotated by 16 to 20 annotators. In this paper, the number of annotation targets (22 celebrities in total in 10 images) is relatively large. Since the purpose of this experiment is to reveal the conflicts between image annotations by multiple annotators; if too many images are used as testing, the number of overlapping annotations for potential conflicts will be low. Therefore the number of images to be annotated is tentatively kept low enough to capture the potential conflicts among annotators. With sharing images, each image can be annotated by expert, common, and novice annotators, then we can examine and evaluate the conflict resolution method more easily.

After the annotation of the images, we estimated the AFD for each annotator. In Figure 5(a-d), the x-coordinate is the ID for each annotator (indexed from A00 to G08), and the y-coordinate indicates the AFD value for each annotator. Figure 5(d) shows that the average AFD for each of the 62 annotators that is averaged over the AFD's based on the annotation results from the first image as shown in Figure 5(a), the second image as shown in Figure 5(b) and the third image as shown in Figure 5(c) together. Each AFD value for an annotator over a specific image is calculated as following:
1. AFD is initially assigned as 0.5.
2. Each attribute of an image has x score for a correct annotation and -x points for the incorrect annotation, (x = 0.1 in our case)
3. If there are m annotators who made a correct annotation for the attribute, then the AFD for each correct annotator is incremented by an amount x/m.
4. If there are n annotators who made an incorrect annotation for the attribute, then the AFD for each incorrect annotator is decremented by an amount x/n.

The rules 3 and 4 reflect an unusual attribute that lead to an unusual annotation in either a good or a bad way.
Figure 5(a). The first AFD of the 62 annotators.

Figure 5(b). The second AFD of the 62 annotators.
Figure 5(c). The third AFD of the 62 annotators.

Figure 5(d). The average AFD of the 62 annotators.

Figure 5(d) shows that some annotators tend to have higher AFD’s than others, which means they are more familiar with the celebrity domain in comparison to others.

6.2 Performance Evaluation Experiments

After the initial setting of experiments, we conducted the comparison experiments to evaluate the performance of the proposed automatic conflict detection and resolution methods during the annotation of images. For each image, we chose a dozen of annotators randomly to conduct two separate groups of experiments. One was called the control group in which the annotators always replaced old annotations with new ones and abandoned the duplicated annotations. The other was called experiment group in which the system automatically detected and resolved the arising conflicts and assisted human annotation using the methods discussed in the above sections. In Figure 6, it illustrated the comparison of performances in the two groups of experiments in terms of the average number of correct and incorrect annotations. It showed that the average number of correct annotations in the experiment group is in general higher than that in the control group and the average number of incorrect annotations in the experiment group is lower than that in the control group.
Figure 6. The average number of correct and incorrect annotations on ten images for comparison of performances in the two groups of experiments.

Figure 7. The average accuracy of conflict resolution in the experiment group versus that in the control group.

Figure 7 shows the average accuracy of annotations for the ten images in each group, where the y-coordinate is the average accuracy of annotation that falls in the range $[1,-1]$ for each image and x-coordinate is the image number. The accuracy of annotation is calculated using following formula:

$$
\text{Accuracy} = \frac{\text{# of correct annotations} - \text{# of incorrect annotation}}{\text{total # of}}
$$
Since we need to increase the chance of conflicts in the experiments, the scope of annotation must be restricted. Therefore, the average accuracies of control and experiment groups are very low, they are 0.1854 and 0.3092 respectively. As shown in Figure 7, the performance of automatic data conflict resolution methods improves 12.38% from 18.54% of accuracy in comparison to a naïve annotation system that simply replaces the old annotation by new ones in dealing with conflicts. Furthermore, for each of the 10 images, we also tested 100 different annotator-sequences to examine the order sensitivity of the system. We calculated the variance for each of the two groups to examine the order sensitivity on each image, the variances are shown in Figure 8. It shows that the experiment group (solid line) tends to have lower variance than the control group (dotted line). It implies that the automatic conflict resolution methods are less sensitive to the order of annotators.

The only exception in Figure 8 is the image 3 in which the experiment group (Incorrect) has a slightly larger variance than the control group. The reason for this exception is the AFD model is not powerful enough to classify among annotators. The AFD’s of annotators in image 3 of the experiment group are very similar (located between 0.5 and 0.7).

**7. Conclusions and future works**

Conflicts in image annotation processes may affect the performance of image retrieval
In designing an automated semantic annotation system for image retrieval, conflict detection and resolution components are necessary for an AGA to mediate in the loop to aid human annotators to annotate images correctly. In order to evaluate the performance of conflict detection and resolution for image annotation, we first established a simple annotator model to aid the classification of "good/expert" and "bad/novice" annotators. It is worthwhile to mention that a good annotator model would help AGA to decide whether to trust the annotation by a given annotator. The annotator model could be a very important factor of this approach. However, as we observe in Figure 8, the user model we adopt was not good enough. In future work, a more sophisticated annotator model should not only improve the conflict resolution problem, but also should be used in AGA to deal with the annotation guiding problem.

The experimental results justified our conjectures that the conflict detection and resolution approaches based on appropriate annotator model could help annotators to annotate the image effectively. The performance comparison experiment showed the average accuracy in experiment group was improved about 12.38%. The automatic data conflict resolution methods had improved the accuracy. However, even assisted by the automatic conflict resolution methods, the average annotation accuracy was still very low (namely 30.93%). It is because there are still many conflicts with which we do not have a proper approach to deal with. The problems of multiple relations, the negative knowledge, and the unreasonable annotations are just some examples: Only a few types of multiple relations have been proposed in this paper. And it is hard to implement a faulty knowledge model to capture all negative knowledge in the open world environment. The same reason applies to dealing with the problems of unreasonable annotations. For example, we encountered five annotations that were annotated with "hasAlias" in which only two annotations were recognized correctly. Even we can filter some unreasonable annotation to decrease the chance of committing to incorrect annotations, we might also prune away some good annotations as well. The issues discussed above will leave as future work.

The issue: "How to determine a property is a One-To-One/is-Not-Relation/Mutual-exclusion property automatically?" will be considered in the future. If we can deal with this issue well, then the conflicts will be resolved more easily.

In the paper, we only used a single AFD value instead of a more complicated AFD set to model an annotator. The rationale behind it is that the application domain in this study is very narrow and a simple model can be much easier to implement and evaluated. Of course, a more elaborated annotator model will be more reliable for real applications and we have discussed this issue in other paper (Lee and Soo, 2005).
We also observe that several problems could affect the correctness using a general search engine approach for unreasonable annotation:

1. The polysemous terms will lead to a wrong conclusion. For example, if “Dogs” and “Wings” are the names of two rock bands that could relate them to each other, but it can be unreasonable in other annotation.
2. Some properties such as the name of a person, the quantity of some measure and a proper noun cannot be applicable.
3. The relationship of two terms cannot be defined precisely. When two terms are related in a unique way, we might not have proper expression to indicate the relations.

Although the search engine based approach has many drawbacks, but it provides at least some useful information to deal with unreasonable annotation and it can be improved to reach a better result that can be also in our future work.

In a summary, we have classified different types of conflicts and have devised their corresponding conflict detection methods and have implemented the automatic conflict resolution strategies to assist human annotators. The methodologies proposed have shed some light in dealing with the automatic conflict resolution problems in knowledge merging for image annotations.

Acknowledgment

This research is supported in part by Taiwan Ministry of Education Program for Promoting Academic Excellence of Universities under grant number 89-E-FA04-1-4 and by National Science Council under grant number 93-3112-B-007-009.

The ontology we constructed is using the Protégé-2000 that was developed by Stanford Medical Informatics at the Stanford University School of Medicine (Protégé).

References


Protégé 2.0 with OWL Plug-in, http://protege.stanford.edu/


Appendix

Appendix A. The detecting rules and resolution actions for data conflicts.

According to the detection rules and resolution actions in section 5, we describe the resolution of data conflicts in the following methods where the bold-type symbol “D” indicates the detecting rules, “A” indicates the resolution actions, and δ indicates a bias of a current annotator [based on the assumption 2 in section 2.1, in particular, δ > 0 when the annotation is directly annotated by a human annotator].

1. Duplicated data in limited-cardinality property.

   **D:** R1, R7. **A:** A1, A5.

2. Different values in limited-cardinality property.

   (a) Synonym term.

   **D:** R2, R7. **A:** A1, A5.

   (b) Generalized term.

   **D:** R3, R7. **A:** A1, A5.
(c) Specialized term.
(d) Antonym term.
   D: R5, R7.
   A: if (AFD(p)>> AFD(c)) A1 else A3.
(e) Irrelevant/different-view term
   D: R6, R7.
   A: if (AFD(p)> AFD(c)+δ) A1 else A3.

3. Duplicated data in unlimited-cardinality property.
   D: R1, ¬¬¬¬¬¬¬¬ R7.  A: A1, A5.

4. Different values in unlimited-cardinality property.
   (a) Synonym term.
   (b) Generalized term.
   (c) Specialized term.
   (d) Antonym term.
      D: R5, ¬¬¬¬¬¬¬¬ R7.
      A: if (AFD(p) > AFD(c)+δ) A1 else A3.
   (e) Unrelated term.

5. Annotation precision (at different abstraction levels)
   (a) Different subject granularities.
      Examine it using the rules in above 4 categories since the sub-class inherits
      the properties of its super-class.
   (b) Different property granularities.
      2. Examine it using the rules in the above 4 categories.
   (c) Different Class Object granularities.
Examine it using the rules in the above 4 categories.

(d) Different Data Object granularities

Examine it using the rules in the above 4 categories.

Appendix B. The Possible Ontology Conflicts.

1. Subject conflict. The conflict causes a wrong description about a subject that could be further divided into the following four cases.

   (a) Using a wrong attribute to describe a related subject. For example, in the triple <PhysicalThing, hasGender, “Male”>, the property “hasGender” is related to the concept of an “Animal” that is a subclass/descendant of the concept “PhysicalThing”, however, “hasGender” is not an attribute of “PhysicalThing”.

   (b) Using a wrong attribute-value pair to describe an unrelated subject. For example, in the triple <Item, hasGender, “Male”>, the property “hasGender” is not an attribute of the concept “Item” despite that the “Male” is a valid value.

   (c) Using a wrong identity name to refer to a subject. For example, in the triple <Person2, hasGender, “Male”> but “Person2” does not exist (Perhaps the annotator meant “Person1”).

   (d) Using an unknown subject name to refer to a subject. For example, in the triple <LEE, hasGender, “Male”> but LEE is not a legal class in the ontology.

2. Property conflict. The property conflict leads to a wrong property description that can be further divided into the following three cases.

   (e) Using a wrong attribute concept to describe a related property. For example, in the triple <Item, hasSkinColor, “Black”>, the concept “Item” does not relate to the property “hasSkinColor”, however “hasSkinColor” is sub-property/descendant of “hasColor” that is an attribute of concept “Item”, and “Black” is one of the valid colors.

   (f) Using a wrong value to describe an unrelated property. For example, in the triple <Item, hasGender, “Black”>, the “Item” doesn’t have property “hasGender” and none of Item’s properties are related to “hasGender”, and the value “Black” is not a valid value of “hasGender”.

   (g) Using unknown property. For example, in the triple <Item, hasMat, “Steel”>, “hasMat” is not a legal property in the ontology (it could be a typo error of “hasMaterial”).
3. Object conflict. The object conflict leads to a wrong description about an object.
   (h) Using a wrong object type. For example, in the triple <Person, hasMother, Item>, the object should be belong to a subclass/descendant of concept “Person” but not non-related <Item>.
   (i) Using a wrong data type value: For example, in the triple <Person, hasWeight, “very heavy”>, but the ontology requires a positive number type as valid value for the weight not a string description.
   (j) Using an unknown object class. For example, in the triple <Person, hasFather, M1>, “M1” is not a legal class in the ontology.
   (k) Quantity inconsistency (violation of cardinality constraint). The number of annotation does not fit the ontology. For example, in the triple <Person1, hasParent, Person2>, <Person1, hasParent, Person3>, and <Person1, hasParent, Person4> cause a conflict because the maximum cardinality of “hasParent” is 2 according to the ontology.

4. Irrelevant triples. If any two of three slots are wrong, it usually leads to an irrelevant triple in the three tuples (subject, predicate, and object). For example, <Item, hasGender, Action>, all the three tuples, “Item”, “hasGender”, and “Action” have nothing to do with each other

### Appendix C. The Detecting Rules and Resolving Actions for Ontology Conflicts.

In section 5, we describe the resolution of data conflicts using the following methods in which the bold-type symbol “D” indicates the detecting rules, “A” indicates the resolution actions.

1. Subject conflict:
   (a) Using a wrong but related subject S.
      D: ¬<S, P>, <P, O>, Specialized(S).
      A: Specialized(S).
   (b) Using a wrong and unrelated subject S.
      D: ¬<S, P>, <P, O>, ¬Specialized(S).
      A: Unrelated(S)/Abandon.
   (c) Using a wrong subject identity name for S.
      D: ¬S, <P, O>, Similar(S).
      A: Similar(S).
   (d) Using an unknown subject S.
      D: ¬S, <P, O>, ¬Similar(S).
      A: Unrelated(S)/Abandon.

2. Property conflict:
(a) Using a wrong but related property \(P\).
\[D: \neg <S,P>, <P,O>, \text{Generalized}(P).\]
\[A: \text{Generalized}(P).\]

(b) Using a wrong and unrelated property \(P\).
\[D: \neg <S,P>, <S,O>.\]
\[A: \text{Unrelated}(P) / \text{Abandon}.\]

(c) Using an unknown property \(P\).
\[D: \neg P, <S,O>.\]
\[A: \text{Unrelated}(P) / \text{Abandon}.\]

3. Object conflicts:

(a) Using a wrong and unrelated object \(O\).
\[D: <S,P>, \neg <P,O>.\]
\[A: \text{Abandon}.\]

(b) Using a wrong data type.
\[D: <S,P>, \neg <P,O>.\]
\[A: \text{Abandon}.\]

(c) Using an unknown object \(O\).
\[D: <S,P>, \neg O.\]
\[A: \text{Abandon}.\]

(d) Quantity inconsistency (cardinality limitation).
\[D: \text{Cardinality}(P).\]
\[A: \text{Remove to fit}.\]

4. More than two slots are wrong:

If two of three slots are wrong, it usually will lead to a triple with all tuples (subject, predicate, and object) expressed in different and unrelated manners such as \(<\text{Item}, \text{hasGender}, \text{Action}>\). This situation can be detected by
\[D: \neg <S,P>, \neg <P,O> \text{ and } \neg <S,O>.\]
\[A: \text{Abandon}.\]

As the annotations are in general unreliable, the conflict resolution action we adopted is to ignore the annotation and also to decrease the reliability of the annotator.

**Appendix D. The Properties Need to be Prevented from Inconspicuous Conflicts.**

In our working domain ontology, the related properties that need to be prevented for each types are:

<table>
<thead>
<tr>
<th>Type</th>
<th>Example Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>One-to-one</td>
<td>HasBody,</td>
</tr>
<tr>
<td>Property</td>
<td>Relations</td>
</tr>
<tr>
<td>-----------------------</td>
<td>--------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Not a reflexive</strong></td>
<td>hasBodyPart, composeOf, carry, hasMate, hasMother, hasFather, hasOlderBrother, hasYoungerBrother, hasOlderSister, hasYoungerSister</td>
</tr>
<tr>
<td><strong>Not a symmetric</strong></td>
<td>hasBodyPart, composeOf, carry, hasMother, hasFather, hasOlderBrother, hasYoungerBrother, hasOlderSister, hasYoungerSister.</td>
</tr>
<tr>
<td><strong>Not a transitive</strong></td>
<td>hasMother, hasFather.</td>
</tr>
<tr>
<td><strong>Not a cyclic</strong></td>
<td>hasBodyPart, composeOf, carry, hasMother, hasFather, hasOlderBrother, hasYoungerBrother, hasOlderSister, hasYoungerSister.</td>
</tr>
<tr>
<td><strong>Mutual Exclusion</strong></td>
<td>{hasMate, hasMother, hasFather}, {hasYoungerBrother, hasOlderBrother, hasOlderSister, hasYoungerSister, hasMother, hasFather}, and {periodBegin, periodEnd}</td>
</tr>
</tbody>
</table>
Dear Dr. Tefko and reviewers,

We are so happy to have a chance to publish our paper in “Information Processing & Management”. Thanks for the hard work of associate editor Dr. Tefko and reviewers, we have read every comment carefully, and revised our paper as possible as we can with those comments as attached below. If there are important points that we have overlooked and are not satisfactory, please point it out again.

Sincerely yours,

Chen-Yu Lee
Department of computer science,
National Tsing Hua University, Hsin Chu, Taiwan.

We have separated the comments into several isolated questions to make sure that we had answered all the comments with \( C_{ij} \) representing \( j^{th} \) comment of reviewer \( i \) and \( R_{ij} \) the reply of \( j^{th} \) comment of reviewer \( i \).

Reviewer #1:
\( C_{11} \): There are two areas where image annotation is important at present: (a) increased emphasis on manual annotation of images and video for the purpose of creating training data on which machine learning algorithms are run (b) the present large effort in manual annotation of images and video for retrieval, as done by broadcasters and image houses. Your work does not seem to address either though is possibly closest to (a) but I'm not sure. So who is it targeted at?

\( R_{11} \): This paper focuses on resolving conflicts for image annotation. The major purpose of our system tries to save the manual efforts to construct a large correct annotation that can be used in retrieval. But instead of using machine learning approach we construct a software agent to guide the annotation. The methods developed in the work can be used by broadcasters and image houses. We had enhanced our purpose in session 1 paragraph 1 to 3.

\( C_{12} \): page 2 - diagram - the section on consistency of annotation - won't this consistency element change during the annotation process by a given user, as he/she learns more about the image and the annotation - why is this a fixed value?

\( R_{12} \): Yes, the consistency of annotation will be changed dynamically during the annotation process, so it needs to be reconsidered when a new piece of knowledge is
acquired. We did not base on the assumption that the consistency is a fixed value. In fact, the conflict resolution has to be maintained dynamically to ensure the consistency all the time.

**C13**: page 6 - diagram - this is a messy diagram, not clear at all. TOO MANY CROSSING LINES.

**R13**: Figure 3 in page 6 had been revised into a simpler one. The crossing lines are avoided.

**C14**: page 8 - what is a "subject" in the definition?

**R14**: We have modified the term “subject” in the paper according to the two aspects: we use the term “subject” to refer a subject of annotation such as a person and use the term “subject-slot” to refer to a subject in a RDF:triple.

**C15**: page 6 introduces a Mandarin thesaurus - I am confused now, is this work on annotation in Chinese or English, since the other examples are in English.

**R15**: Yes, the annotation is in mandarin Chinese because of the practical domain applications. But for the purpose of presentation and publication we have translated the annotation in English. In page 1 in section 1 and assumption 4 in section 2.3, we have described this point clearly and also add the Chinese to all the examples.

**C16**: Creating single-valued numbers to represent an annotator’s faith degree and knowledge faith degree as absolute values is a bit inflexible ... I may know a lot about a certain domain and not much about another, yet I have a single value assigned to me as an annotator! That makes for mathematical convenience but is unreal.

**R16**: The reason that we only adopt a single AFD (annotator’s faith degree) is because the domain we used is quite narrow. We have shown by experiment that the use of knowledge faith degree can to some extent improve the effectiveness of the annotation. Definitely, a more complicated way to represent an annotator it could be more reliable. Please refer to our work in other paper (Lee and Soo, 2005)). We have added some statements in first paragraph of page 32 in the section of conclusion and future work about this point.

**C17**: Section on related work is very thin - there is much other work on conflict resolution in annotation, or in relevance judgments for example (see TREC) that could be covered here.

**R17**: The related work had been improved. Most of the work in TREC did not address the potential conflicts of annotation. The annotations could be conducted by a single
annotator or a team of annotators; but only a final standard annotation is used as benchmark for evaluation.

**C18:** The ontology you use, introduced on page 5 - is not designed for any particular application or domain

**R18:** The ontology in Figure 2 used lots of common concepts, however, the relations (properties) between those concepts is designed for the domain we used. Probably, it can be applied in other domain as well; however, we did not expect it as a general ontology.

**C19:** It seems that conflicts between annotators are resolved by correcting annotated but contradictory triples - shouldn't the over-riding of an annotation be recorded as circumstances in the annotation database may change and a previous contradiction no longer be a contradiction? By over-writing annotations you lose this historical aspect.

**R19:** We agree that the historical records of annotation could be very important information in conflict handling and also the contraction condition might change overtime as the reviewer had pointed out. However, it requires a huge storage to keep the historical records. For example, supposed that a single triple be modified n time, and an image consist of m triples, if we have p images in our image base, then we would have to prepare n*m*p additional annotation space for the historical records. Even we have taken the historical information into consideration, we still cannot ensure to infer the correctness of an annotated triple. Therefore, we resolve the conflicts base on the reliability of annotators instead, that have indirectly taken into the historical annotation information into consideration. We have added some explanation regarding this point in page 15 section 4.3.

**C1a:** There is some related work in the area of historical databases where the state of a database (cf set of annotations) is kept not just for the present time but for all previous times. That work might be relevant here.

**R1a:** Some papers are mentioned and discussed in page 15 section 4.3.

**C1b:** page 24, and elsewhere, you seem to ignore the polysemous nature of words - chick can be a bird we eat, or a term used to describe cowardice; wings are a body part of birds, and the name of a rock band of Paul McCartney, and an award given to pilots who qualify;

**R1b:** Actually, the polysemous terms is one of the reasons that makes the common knowledge inspection fails, but not the only one. Therefore, we add a paragraph in
Testing on 10 images, even with 60+ annotators, is too susceptible to the particular set of images having unusual characteristics. For example, two of them could have been Brad Pitt, who could have been all over the news in the previous week because he got divorced or something...which changes completely how people would annotate those two images, and that is 20% of your data. Your dataset is too small for reliable, repeatable, independent experimentation.

In this paper, each image is annotated by more than 16 annotators, but the number of annotation targets (there are 22 celebrities in total in 10 images) is relatively large. Since the purpose of this experiment is to reveal the conflicts between image annotations by multiple annotators; if too many images are used as testing, the number of overlapping annotations for potential conflicts will be low. Therefore the number of images to be annotated is tentatively kept low enough to capture the potential conflicts among annotators. This point of view is also explained in page 26 section 6.1.

Figure 5 - sort the x-axis values by their y-axis values to present this as a curve or something, not just random noise.

The x-axis is the annotator ID, so it makes no sense to sort it. But for the clarity of showing the AFD value in the figure, we have divided the Figure 5 into 4 figures (Figure 5(a)-5(d)) in page 27-28 section 6.1.

GENERAL: this is an interesting approach but the paper is weak in the sense that the experiments are too small to be reliable, the eventual application is undefined.

As the same point we mentioned in comment C1e the low number of test images is tentative and it does not affect the evaluation of our conflict resolution methods. On the contrary, using large number of images for experiment will make the evaluation harder. With sharing images, each image can be annotated by expert, common, and novice annotators, then we can examine and evaluate the conflict resolution method more easily.

Finally, the paper could benefit from proof reading by a native speaker of English and perhaps there is somebody in your organisation that could help you with that?

We have improved the English in this revision version.

Reviewer #2:

The quality of english should be significantly improved.
$R_{31}$: The same as $R_{1f}$. 