Context Focused Intelligence Service within Big Data Environment: Knowledge Enabled Activeness, Efficiency and Precision

Topics
Topic 3: Data, Information and Knowledge
Topic 12: ISR for Decision Making

Authors
Xin Jin
Shiqiang Zong
Youjiang Li
Shanshan Wu
Wenke Yin
Weiyi Ge

Science and Technology on Information Systems Engineering Laboratory, Nanjing, Jiangsu, 210014, P.R. China

Point of Contact
Xin Jin

Name of Organization
Science and Technology on Information Systems Engineering Laboratory

Address
Nanjing, Jiangsu, 210014, P.R. China

Email
venus_163@163.com
Context Focused Intelligence Service within Big Data Environment: Knowledge Enabled Activeness, Efficiency and Precision

Xin Jin, Shiqiang Zong, Youjiang Li, Shanshan Wu, Wenke Yin, Weiyi Ge
(Science and Technology on Information System Engineering Laboratory, Nanjing 210007, China)

Abstract:
Big Data causes the problem of “Information Overwhelming”, which makes traditional intelligence service no longer effective. Required intelligence is submerged by huge amount of noise data, thus difficult to locate. The only way to get it is by keyword-based search, which however, brings down the efficiency and precision of intelligence service. To solve the problem, a domain knowledge based method is proposed. Firstly, context elements are defined to model commander decision task, and the mapping relations from commander decision task to commonly required intelligence are defined as domain knowledge. Then in running phases, commander decision task is monitored, and keyword expressions are automatically generated by applying the according pieces of knowledge. From user view, search results are provided actively along his decision making process. Moreover, it is proved by experiments that intelligence service efficiency and precision is improved by this method.

1 Introduction

In recent years, Big Data has become truth and caused broad discussion [1-3]. Increasing number of C2 systems are mining Big Data to improve their intelligence service. However, mining Big Data is not as easy as using their own well organized databases.

In traditional C2 systems, intelligence data is obtained from specific sources. The relations between intelligence supply and demand rarely change [4, 5]. However, it is different when transferred to Big Data environment. Useful intelligence is hidden within the sea of data. There are great many sources, providing intelligence in different types, redundant contents, and heterogeneous forms. Intelligence distribution is unclear to users. No one knows exactly which source contains his required intelligence. The only way to get it is by keyword-based search. Intelligence requirement is modeled into keywords. Which intelligence from which source user can get depends on dynamic match, while the result quality depends on the precision of user described requirement models. In Big Data environment however, to precisely model an info-req is difficult. User needs to choose keywords carefully to avoid ambiguity, and modify keywords repeatedly to get precise search results. These works detract commander’s focus from decision making, thus cause low efficiency.

A simple idea is to make the INT-REQ (intelligence requirement) generation process automatic. In our researches on C2 endeavors, by analysis of various commanders’ search operations, we found their intelligence requirement highly depends on the decision task, that on processing tasks of a same type, requirements have similar content and scope. There must be latent relations between task types
and INT-REQs, which may be learned as a kind of domain knowledge, and then applied to realize such automation.

By this idea, a knowledge driven method is proposed, to automatically generate INT-REQ against Big Data, by making use of domain knowledge. In this method, machine keeps aware of commander decision task, matches appropriate knowledge to generate INT-REQ contents and restrictions, and translated them into keyword expressions, which is finally submitted to search engines, databases and subscription systems. From user view, the whole process from INT-REQ generation to search result feedback is automatic, makes the intelligence service active. Moreover, it is proved by experiments that intelligence service efficiency and precision is improved by this method.

2 Related Researches

On helping user expressing information requirement, there are methods in IR (information retrieval) domain, e.g. keyword generation [6] and query recommendation [7]. Applied knowledge includes user searching behavior analysis [8], term relationship [6], and semantic relativity [9]. They boosted the prosperity of search engine advertisement, and got fast progress driven by great commercial interest. However, they all require user to input some initial keywords as a hint, and then give suggestions based on it. In our method, machine keeps aware of commander’s decision making task, and actively guesses what intelligence he require, without need for any keyword inputting. Mechanisms behind are basically different.

On knowledge based methods, Google and IBM now master the state-of-the-art. Google’s knowledge graph [10] is a formal knowledge definition, by which user input queries are mapped to a graph of concepts and instances with connections in between. IBM’s Robot Watson [11] can understand questions expressed in natural language, and quickly return precise answers by analyzing a huge knowledge base. Common knowledge learned from e.g. dictionaries, cyclopedias, newspapers, has been successfully applied to solve the open-domain IR and QA (Question Answering) problems in the 2 systems. However, we believe domain knowledge could improve domain system performance to a higher extent. For example, domain knowledge has been applied in decision support systems [12-13] to improve algorithm performance. On applying domain knowledge to solve problem of intelligence collection from Big Data, however, few researches have been found.

We believe that, by learning a new kind of domain knowledge – mapping relations among decision task types and INT-REQs, it is possible to make intelligence search process automatic. This is the main idea of the research.

3 Modeling Language

As stated in above, to get required intelligence from Big Data, the requirement needs to be modeled precisely. The modeling language should be powerful, providing various descriptors to depict all kinds of expected features about the required intelligence, e.g. content, scope, timeliness, media type, probable sources, results order, and so on. On the other hand, a model of INT-REQ should be precise, with term semantics described, so as to avoid misunderstanding by different sources.
As a suggestion, TIREM (Task INT-REQ Express Model) is designed, as shown in Tab. 1, containing 7 main parts.

Tab. 1 TIREM Specification

<table>
<thead>
<tr>
<th>TIREM Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>&lt;REQ&gt;</strong></td>
</tr>
<tr>
<td><strong>&lt;LABEL&gt;</strong> Literal description of the INT-REQ &lt;/LABEL&gt;</td>
</tr>
<tr>
<td><strong>&lt;USER&gt;</strong> Intelligence user type &lt;/USER&gt;</td>
</tr>
<tr>
<td><strong>&lt;OPERATION&gt;</strong> User operation type &lt;/OPERATION&gt;</td>
</tr>
<tr>
<td><strong>&lt;MISSION&gt;</strong> User concerned mission type &lt;/MISSION&gt;</td>
</tr>
<tr>
<td><strong>&lt;SELECT&gt;</strong> Sub.prop1 &amp;/</td>
</tr>
<tr>
<td><strong>&lt;WHERE&gt;</strong> Sub.prop3 =/&lt;/&gt; Value a &amp;/</td>
</tr>
<tr>
<td><strong>&lt;FROM&gt;</strong> Source 1 &amp;/</td>
</tr>
<tr>
<td><strong>&lt;MEDIATYPE&gt;</strong> Type 1</td>
</tr>
<tr>
<td><strong>&lt;WHEN&gt;</strong></td>
</tr>
<tr>
<td><strong>&lt;START&gt;</strong> Start time of the INT-REQ &lt;/START&gt;</td>
</tr>
<tr>
<td><strong>&lt;END&gt;</strong> End time of the INT-REQ &lt;/END&gt;</td>
</tr>
<tr>
<td><strong>&lt;PERIOD&gt;</strong> Period for refresh &lt;/PERIOD&gt;</td>
</tr>
<tr>
<td><strong>&lt;REPEAT&gt;</strong> Period for repeated search &lt;/REPEAT&gt;</td>
</tr>
<tr>
<td><strong>&lt;EARLIEST&gt;</strong> Earliest publish time &lt;/EARLIEST&gt;</td>
</tr>
<tr>
<td><strong>&lt;LATEST&gt;</strong> Latest publish time &lt;/LATEST&gt;</td>
</tr>
<tr>
<td><strong>&lt;REALTIME&gt;</strong> Real-time or non &lt;/REALTIME&gt;</td>
</tr>
<tr>
<td><strong>&lt;/WHEN&gt;</strong></td>
</tr>
<tr>
<td><strong>&lt;ORDERBY&gt;</strong> 1st rule</td>
</tr>
</tbody>
</table>

**LABEL** is a literal description for human to understand the INT-REQ.

**USER**, **OPERATION** and **MISSION** define a decision task type from 3 dimensions. **USER** represents the role of the commander, e.g. commander, staff. Different users have different preference on supported intelligence. **OPERATION** represents the commander’s current decision making process, e.g. situation analyzing, mission planning. **MISSION** represents the mission instance the commander is currently assigned, e.g. attack an airport, organize a rescue.

**SELECT** and **WHERE** have the same meanings as in SQL. Difference lies in that each attribute following **SELECT** and **WHERE** is a “Subject-Property” pair. From logical view, a subject has one or more properties, following **SELECT** are value un-known ones, while following **WHERE** are value known ones used to filter subjects. In this way, a property has semantics, and all the properties following **SELECT** and **WHERE** belong to a same subject. With subject as a limit, a property will not be misunderstood as other subject’s properties, thus improves the precision.

In relational databases, one can simply consider a table as a class of subjects, a line as a single subject, a column as a property. In P/S (Publish/Subscribe) systems, the subscription request is constitute of value-known properties, used to filter message types, while value-unknown properties are contained in the message content. In un-structured text, subjects and properties can be implied from the literals. From “Beijing is the capital city of China” e.g., an RDF triple as <China, capital city, Beijing> can be extracted. In this way, **SELECT** and **WHERE** items can be translated to SQL queries, subscription requests and search keywords at different type of sources such as relational database, P/S.
system and search engine, respectively.

Sometimes for the required intelligence, there are preferred sources, as they are more trusted, more stable, easier to access or else. These sources can be listed after FROM, so as to improve the search precision and efficiency.

Each intelligence has a media type, e.g. text, image, video, audio, database, for-matted message and so on, while text can be further divided into structured and un-structured. MEDIATYPE limits the media type of the required intelligence.

WHEN describes required intelligence timeliness. Durative INT-REQ means that user need durative update within a period. 4 attributes are designed for durative INT-REQ. START and END means when the requirement is in effect and finished, while PERIOD and REPEAT are used to set real-time info update frequency and time interval to periodically recollect new published non-real-time-info. One-off INT-REQ means once a result set is received after a req-submission, an info-collection process is finished. 2 attributes are designed for one-off INT-REQ, means the publish time of info is no earlier than EARLIEST and no later than LATEST. REALTIME are used to limit require intelligence to real-time or non-real-time.

ORDERBY defines how search results are ranked and ordered in the returned list. As normally the returned list contains only the former N results, different settings of this item will get different results. Comprehensive ranking can be realize through setting more than one rules, with “|” as separator. For example, “correlation | trust level” means firstly order by result correlation with INT-REQ, and if there are 2 results with same rank, order by result sources’ trust level.

4 Dynamic Modeling Method

Unlike surfing on Internet, in C2 systems, INT-REQs are well-regulated. By analysis of normal commanders’ INT-REQs during decision making, through records of their database querying, web searching, and real-time intelligence subscription submitting operations, it is found that while a same type of commanders are processing a same type of decision making tasks, their required intelligence has similar scope, content and granularity, while differences mainly lie in the related entities and space-time contexts. For example, when a commander is planning an attack against some target, he needs to know the target’s defensive ability and current moving state, which varies according to different target types, and time-place where the attack happens.

As shown in Fig. 1, there are latent relations between user decision making tasks and their
INT-REQs, while these mapping relations have not been mined from the black box currently. Our idea is to mine the relations as knowledge, called “user-task-INT-REQ (UTIR) mapping knowledge”, and then apply it to infer user INT-REQ.

Based on this principle, our method is proposed: Firstly, classify user decision making tasks into different types, and design one template for each type according to UTIR mapping knowledge, describing what scope, content and granularity of intelligence is required on processing this type of tasks, with variables defined for varying factors. Then in application, when sensed which type of user task is under processing, choose the according template and assign values to its variables (like gap filling), thus an INT-REQ model is generated. The method contains 2 phases and 5 major steps as shown in Fig. 2, introduced in detail as follow.

4.1 Domain Knowledge Construction

Firstly, classify domain user decision making tasks. Divide them until the differences of INT-REQs between 2 types of tasks can be modeled as variables. For instance, at-tacking 2 targets of same type, their INT-REQs only differ in targets’ names, so the name is a variable. But target attack and defense belong to different task types, as their INT-REQs are totally different. Control of variable granularity is up to users.

Then, define task type IDs to differentiate one task type from another. Here, An ID containing 6 elements is suggested. 3 of them are UserType, OperationType, and ObjectType, describing user decision making operations. 3 others are ActorType, ActionType, and TargetType, describing the concerned missions. Any task can be described as some user doing some operation against some mission. So <U, O, O, A, A, T> is a globally unique ID for task types.

Then, for each task type, create a template and model the scope, content and granularity of the required intelligence, with varied factors defined as variables. There are 2 methods to do the modeling work. One is modeling by experience: Invite domain experts (experienced commanders etc.) to
describe the required intelligence according to their experiences, and let programmer to translate his description into template files. Another is modeling by learning: Record user INT-REQs during task processing, and analyze logs by aid of machine learning tools. There are 2 methods on recording: one is to record user operations like web searching, database querying or real-time intelligence subscribing; another is to allow user to view and modify the generated INT-REQ models, and record their modification operations. Here is no restriction or suggestion for the choice. The 2 methods can be used in together.

TIREM is suggested as the modeling language. A template is constituted of template ID and one or more models, each described as a “<REQ>…</REQ>” strings. An example is shown in Fig. 3. The template ID is same with <U, O, O, A, A, T> introduce in above. In the template, each variable is marked within an “&()” operator, and has been defined in the according task awareness function programs.

```
<TemplateID UserType = "Commander", OperationType = "Path plan", ObjectType= "Battle Plane", ActorType = "Battle Plane", ActionType = "Interception", TargetType = "Enemy Planes"><REQ> <!--Req 1: Information about enemy planes' flying ability-->
......
<SELECT>BattlePlane.flyingAbility</SELECT>
<WHERE>BattlePlane.Type = &(EnemyPlaneType)</WHERE>
</REQ>
<REQ> <!--Req 2: Real-time track of enemy planes within 10 minutes-->
......
<SELECT>EnemyPlane.RealTimeTrack</SELECT>
<WHERE>EnemyPlane.FlyingArea = &(TargetArea)</WHERE>
<WHEN>
  <START>&(CurrentTime)</START>
  <END>&(CurrentTime) + 600s</END>
  <PERIOD>5s</PERIOD>
</WHEN>
......
</REQ>
```

Fig. 3. Template example

Finally, construct domain ontology. Ontology formally represents knowledge as a set of concepts within a domain, using a shared vocabulary to denote the types, properties and relations among concepts. It can be used here to solve the problem of cross-domain heterogeneity on describing INT-REQ semantics. So, it is suggested to construct C2 domain ontology, and design templates using terms defined in the ontology. On source side, it is also suggested to construct ontology, and describe its intelligence metadata using terms defined in the ontology. However, different users and sources are in different domains, so their ontologies are heterogeneous to each other. To solve this problem, OM (ontology mapping) tools [15] can be used, which is able to find the concepts in different ontologies with similar meanings, properties and relations, and build alignment automatically. In this way, the INT-REQ model semantics will not be mistranslated by different sources.

Till now, constructed domain knowledge includes UTIR mapping template, do-main ontology and ontology mapping result built by OM tools. How to use the knowledge is introduced as follow.

**4.2 User Task Awareness**

In the user software (commander’s operation interface etc.), functions are required to keep aware
of current user tasks:
1. User awareness: identify user by monitoring login operation, and assign value to variable UserType;
2. User operation awareness: monitor user operations, including UI switching, tool operation, and so on. Define and code all supported operation types previously in preparing phase, and then in running phase, for each detected user operation, assign its type code to variable OperationType, and the operated object type code to variable ObjectType;
3. Concerned missions awareness: concerned missions are commonly described in some formatted files and imported into user software, containing descriptions about the executor, actions and targets, which can be resolved and assigned to variable ActorType, ActionType, TargetType;

Entity and context awareness: during the decision process, collect intelligence about various kinds of entity IDs, area IDs, and system time, and assign them to according variables in templates.

### 4.3 INT-REQ Content Generation

INT-REQ generation process is divided into 2 steps: choose one template and then fill all the gaps, as shown in Fig. 4.

![Fig. 4. Template selection and gap filling](image)

Match each online detected user task type with the templates in the base by ID. As template ID is globally unique, exactly one template will pass match each time.

The template passed match will be selected, with all its variables’ value assigned by the detected entities and contexts. After all variables have their values assigned, the INT-REQ content is generated, and the template is instantiated as an actual INT-REQ.

### 4.4 INT-REQ Semantics Description

Before the generated content becomes an INT-REQ model for publish, its semantics should be precisely described according to domain ontology, in order to get precise result feedback.

In TIREM syntax, semantic description pattern like “subject. property” is support-ed. With such description pattern, an INT-REQ is not only a combination of keywords, but explicitly indicates which properties of which subject is to be queried, and which other properties of the subject has what values.

Each subject and property is predefined in domain ontology, and identified by namespace. For example, “weapon:Lincoin” represents “Lincoin” aircraft carrier defined in a weapon ontology, and will not be mistranslated to some other concepts defined in other ontologies like “people:Lincoin” or
“car:Lincoln”.

In practices, we found problems about synonym. For example, “wp:F-18” and “wp:Hornet” are 2 IDs about a same type of plane. To avoid low recall of search result, both IDs should be included in template. Similarly, “wp:FlyingRange” and “wp:Voyage” both mean the longest distance a plane could fly through, and should both appear in template, as well. However, when designing templates, not all synonyms will be taken into consideration. There are 2 ways to solve the problem:

1. Build a table that keeps learning about new synonyms, and for each term in the generated INT-REQ content, expand it with its learned synonyms in its model. For example, “wp:F-18.wp:Voyage” can be expanded as “(wp:F-18 or wp:Hornet).(wp:Voyage or wp:FlyingRange)”;

2. Keep the terms in the INT-REQ content unchanged in the model, while during matching, use semantic match algorism to automatically map each term to its synonyms, based on domain ontology.

The second method is comparatively more expansible as the number of synonyms increases, while it requires sources to have semantic match ability and according ontologies. So the selection is up to users, here is no restriction.

4.5 INT-REQ Model Publish Management

After above steps, an INT-REQ model is formed. In this step, it will be published to various sources on Internet. The publish process needs management, because precise control of publish time is as important as precise semantic description. Intelligence presented to user at wrong time is a burden rather than support.

According to value of REALTIME, the INT-REQ will be published to real-time or non-real-time info-sources. According to START and END, the INT-REQ will be published and canceled. If REPEAT has value, the INT-REQ will be published repeatedly after the indicated time interval.

On source side, each received INT-REQ model will be translated into their supported formats. For search engines, it will be translated into keyword expressions. For data-bases, it will be translated into SQL queries. For P/S systems, it will be translated into subscription requests.

5 Implementation

Above method has been implemented in a test C2 system. Firstly, commander decision making tasks was divided into 55 combinations of 5 types of user operations (mission acceptance, situation analysis, operation planning, execution monitoring, effect assessment) and 11 types of concerned missions (fixed target striking, moving target interception, air attack defensing, anti-terrorism operation etc.), which forms a user task matrix, as shown in Fig. 5. Each cell of it represents a specific task type, with a template designed for it.
Template contents were designed according to commanders’ experiences. For above 55 task types, required intelligence includes real-time battlement dynamics, non-real-time battlement environment, structured data about weapons, unstructured data from intelligence, and so on, all modeled in their templates.

4 domain ontologies were constructed, including 84 concepts, 28 relations, and about 2k instances, with mapping relations built in between using ontology mapping tool Falcon-AO [15].

User task awareness module has been developed and embedded into the operation software, including time detection, user login process monitoring, formatted mission file analysis, work process monitoring, plan execution monitoring, and user operation listening.

Functions of INT-REQ content generation, semantics description and model publish management, along with model translation on source side were implemented following above methods. For synonym problem, we adopted the second method, as it’s more dispensable. The whole system architecture is shown as Fig. 6.
6 Experiments

To verify the efficiency and precision of the method, several groups of experiments have been taken. For each cell in the user task matrix, a commander is asked to input his INT-REQ manually, which is then compared with the automatically generated one (using above method) on efficiency (time cost). Then, use the 2 INT-REQs to do search twice, compare the result precision and recall, so as to measure precision.

6.1 Efficiency

Time costs of 55 tests are shown in Fig. 7. Mean time cost for manual mode is about half a minute, while for automatic mode, about half a second. It’s obvious automatic mode can effectively save user time on INT-REQ modeling.

![Fig. 7. Time cost comparison](image)

An example from experiments is given to show the process of INT-REQ generation, as shown in Fig. 5. In the example, battle plane’s “Flying Ability” is expanded as “Voyage”, “Extreme Speed” and
“Extreme Height”, “F-18” is expanded as “F-18” and “Hornet”, according to domain ontology named with “wp”. Even such a simple INT-REQ costs user much time and effort to model. But with domain knowledge built previously, machine’s high performance can accomplish this work in very short time.

6.2 Precision

Precision of 55 tests are shown in Fig. 8. Mean Precision for manual mode is about 38.7%, while for automatic mode, about 60.5%. It’s obvious automatic mode can effectively improve search result precision.

![Fig. 8. Precision comparison](image)

This can be explained by comparing INT-REQ model content. It was found that normally the generated INT-REQ models are more detailed, because lots of terms for restriction and expanding were added according to template and domain ontology. But for human, he usually needs several repeats of result viewing and INT-REQ remodeling works before able to get a model with the similar complexity as the generated one.

7 Conclusions

Big Data challenges the intelligence service mode in traditional C2 systems. To get precise intelligence, users spend much time and effort on search keywords preparing, which detract commander’s concentration on decision making. To solve the problem, a method is proposed, which keeps aware of commander’s decision making tasks, generates INT-REQ content and restrictions, describes its semantics according to domain knowledge, and publishes INT-REQ models to sources at appropriate time. It is proved by tests that by the method that: 1) Efficiency of search process is improved as user time cost on INT-REQ modeling is saved; 2) Precision of search results is improved, as the generated INT-REQ model is usually more detailed than the manually inputted ones; 3) Activeness is improved, as search results are returned automatically during commander’s decision making process.

Further researches are mainly focused on domain knowledge construction. UTIR mapping template is a basic form of domain knowledge, and the next step is to mine UTIR mapping rules form
the gradually accumulated UTIR mapping templates, so as to improve the method’s correctness performance.

References