# AN AGENT-BASED INFORMATION CUSTOMIZATION SYSTEM USING CBR AND ONTOLOGY

Hyun Jung Lee
*Korea University Business School, Anam-dong Seongbuk-gu, Seoul, 136-701, Korea*

Mye M. Sohn*
*Department of Systems Management Engineering, Sungkyunkwan University*
*300, Chunchun-dong, Jangan-gu, Suwon, Kyunggi-do, 440-746, Korea*

Keywords: Case-based Reasoning, Ontology, Agent-based System, Information Customization.

Abstract: An *Agent-based Information Customizing System* is composed of a *Case Generation Agent* to transform unstructured documents to a structured form such as cases, and a *Case Customization Agent* to select the most similar case from the case-base and adjust the selected case depending on the user's information requirements. The developed case contains features and their values, and each defined case that is based on the information can each has a different feature set. Thus, it is possible to represent the contained details in documents, and it is easier to find more appropriate information from the case pool. Two-step similarity calculation using features and values and domain ontology are applied to find an appropriate case. A case customization process is suggested to adjust the case. In our future work, the suggested system would be applied to traveller's systems to integrate information and will prove its effectiveness.

## 1 INTRODUCTION

The information users are focusing on the speedy collection, the appropriate processing and the utility of the information on the web. Information users have recently been able to obtain the needed information from a variety of resources. However, it is not easy to find customised information from these resources. Until now, information resources such as the Web, don't have the ability to completely filter out the customised information that may be obtained from inaccurate or excessive information pools according to the information users' requirements. For example, most travellers spend a lot of time and effort to find customised travelling information that suits each traveller's purpose via the Web or the Internet (Pan B., and D. R. Fesenmaier, 2006). It's true that it is too difficult to select, customize and integrate the information based on unstructured documents without the help of the information worker.

In this paper, we propose an agent-based information customization system that supports select and integrating the dispersed information on the Web depending on the information user's requirements, which is composed of the transformation phase and the customization phase.

The transformation phase transforms the information or the data, which is collected from a variety of information resources, into a structured data format for an intelligent agent's processing. As a structured format, cases may be appropriate to transform unstructured documents into structured documents. Even though we consider other structured formats such as rules using if-then format, frames or logic using taxonomical expression of knowledge, etc., when we transform unstructured data or a document into a rule or a frame format, we will most likely lose a lot of information.

There are two reasons why we chose the case as a structured data format. First, information users usually want to refer to qualified information that is based on other users' experiences and accumulated knowledge. Information users have recently preferred to use web-based information for reference. They want to perform web searches to build customised cases through retrieving, reusing, and

* Corresponding author.



Lee H. and M. Sohn M. (2010).
AN AGENT-BASED INFORMATION CUSTOMIZATION SYSTEM USING CBR AND ONTOLOGY.
In *Proceedings of the 2nd International Conference on Agents and Artificial Intelligence - Artificial Intelligence*, pages 285-290
DOI: 10.5220/0002723702850290

revising the accumulated cases on a case-by-case basis. Second, when we use cases, we have less information loss than when we use rules or frames. Cases can include context information that contains the tacit knowledge within documents, and it will be helpful to maximize the information users' satisfaction.

In the customization phase, we generate new information that satisfies the users' requirements by using case-based reasoning. The proposed case-based reasoning uses the two-step similarity calculation method. In step 1, the similarity between the information that the users' requires and the features of the cases is calculated by using the requirements-feature similarity method. In step 2, the requirement-value similarity is used to calculate the similarity between the details of values and the users' requirements.

The proposed case is structured by features and values. A value is defined by sentences that include features. Therefore, a feature can be matched into several sentences as values. To find a similar case, we filter the cases according to the similarity between case and the information of the user's requirements. Ontology is needed to determine the semantic relationship between the features or values. The selected case through the two-step similarity calculation can be adjusted according to the user's requirements, which is conducted by the Case Customization Agent.

In chapter 2, after review related studies, in chapter 3, the overall architecture of the proposed system is introduced. Next, we discuss the case and the sub-case generation, and we describe how to retrieve similar cases with using the two-step similarity calculation. In chapter 6, we introduce the case customization agent. Finally, we discuss future study and the conclusion of this study.

## 2 LITERATURE REVIEW

Information integration is the process of semantically and syntactically combining the data that comes from different resources with adhering to a unified format (Bernstein A. Philip and Laura M. Haas, 2008). To do this, it is important to classify the information resources such as the Web documents into structured or unstructured forms. The former are needed for transforming the data, which is obtained from databases with the heterogeneous schemas, into homogeneous data with using a semantically and syntactically. This can be accomplished by the ETL technology or the mediated schema (Halevy Y. Alon, *et al.*, 2005; Bernstein A. Philip and Laura M. Haas, 2008). The latter is able to extract information from unstructured document through keyword search, summary data, and pre-defined information structures (e.g., the customer's name and address, the product information and the brand name, and so on). An annotation such as the Resource Description Framework (RDF) can also be applied to the information selection and integration processes. The process of applying ontology to the semantic integration of information is getting more popular regardless of the structure of that information (Alexiev Vladimir, *et al.*, 2005).

However, It is too difficult to integrate information by using the selected documents based on a keyword search due to the increase of web documents from around the world by geographic progression. Even we use the RSS, we can still receive the selected information; so, the information users have to have a part in electing the appropriate information. In addition, it is necessary to check and resolve possible conflictions. Information users usually spend a lot of time and effort to get the customized information (Mani, I. and Bloedorn, E., 1998; Teufel, S. and Moens, K., 1997.). Therefore, it is critical to propose an adequate methodology of information integration. However, information is basically differentiated from physical products (Lee, Y., *et al.*, 2001; Karmarkar, U. S. and Apte, U. M., 2007). Therefore, to integrate information, research is needed to understand the contents in documents, to classify those contents and to integrate the contents. To accomplish this, we suggest an Agent-based Information Customization System with using the CBR and Ontology.

## 3 OVERAL ARCHITECTURE OF AGENT-BASED INFORMATION CUSTOMIZATION SYSTEM

In an *Agent-based Information Customization, Case Generation Agent* is for transforming the collected information into cases, and *Case Customization Agent* is for customizing the information depending on the information of the user's requirements. Their overall architecture is shown in Figure 1.

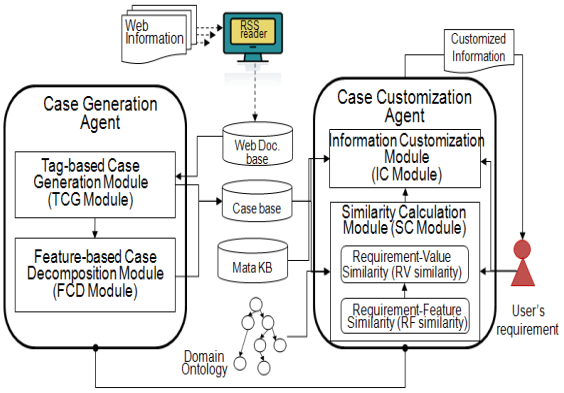


Figure 1: Architecture of agent-based information customization system.

- The *Case Generation Agent* is invoked whenever the RSS reader has requested to receive new documents. we assumed that the information is usually collected through a push-based RSS reader. The agent transforms the unstructured documents into structured ones, such as cases. The received unstructured web documents can be transformed to a case that is based on tags in the documents. The features of the case are replaced by tags in the documents, and the value of the feature is defined by sentences that include the feature. Each feature can be included in each sentence, and the sentence is defined by a sub-case.
- The *Case Customization Agent* is invoked to receive the information the user's requires and to select similar cases from the case base. A similarity calculation module calculates the similarities between a user's requirements and a feature set of a case, and between a user's requirements and the values. For the similarity calculation, we usually refer to the domain ontology. A selected similar case is customised according to a user's requirements with considering the constraints among the information.

## 4 CASE GENERATION

The transformed cases from the gathered unstructured web documents have information structure. In the initial stage, a feature set of a case is defined by tags that are annotated on the document, and then the document is defined as a value of the feature set. If a document has the same feature set, then it can be added as another element to the value set. The transformed case could be broken up into several sub-cases. The sub-case can have a feature set and the values. The created cases and sub-cases can be applied as units to generate new information. Figure 2 illustrates this process:

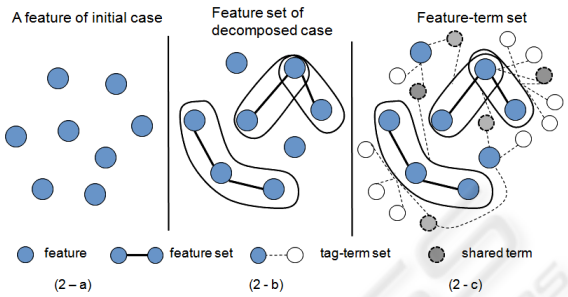


Figure 2: Feature and feature-term generation.

Figure (2-a) shows a mapping between tags (i.e.,$\{t_1, t_2, t_3, \ldots, t_n\}$), which are attached on the original web document. The tags are defined as a feature set $\{f_1 \; f_2 \; f_3 \; \ldots \; f_n\}$ in a case. The tag set doesn't contain any relationships between the tags, but the feature set represents the features and the relationships among the tags. Figure (2-b) illustrates the newly created feature sets. Each feature set has every relationship between features in a sub-case, such as $(\{f_1 \; f_2\} \; \{f_1 \; f_3\} \; \{f_1 \; f_2 \; f_3\} \; \ldots \; f_i \; f_j \; \ldots \; f_n\})$. Figure (2-c) shows the meaningful terms that are included in each sentence as values. The meaningful terms are defined by the classes and the instances of Ontology.

To transform the collected web document into a case, the following symbol is defined:

*$f_i$: a feature where it composes of a feature set of a case*
*$v_i$: a value where it composes of a value set of a case*
*$t_i$: a tag where it composes of a tag set of a document*
*$s_i$: a sentence where it composes of a document*
*$k_i$: a term keyword where it composes of a term set of a value*

$$feature\_set \quad \{f_i \mid i \geq 1\}$$
$$value\_set \quad \{v_i \mid i \geq 1\}, v_i \quad \{s_i \mid i \geq 1\}$$
$$term\_set \quad \{k_i \mid i \geq 1\}$$
$$negarion\_sign \quad \{NOT|null\}$$

In this research, the suggested case is composed of a 'Feature_set', a 'Value_set' and a 'Negation_sign', such as Case = {Feature_set, Value_set, Negation_sign}. The 'Feature_Set' is assumed that there are 'and' relationships between tags, such as Feature_set = $\{t_1 \; t_2 \; t_3 \ldots, t_n\}$. The 'Negation_sign' shows if the value contains a positive meaning or not. The case is illustrated as follows:

$$Case = \{feature\_set, value\_set, negation\_sign\};$$
$$= \{\{f_1 \wedge f_2 \wedge f_j \wedge \ldots \wedge f_m\}, \{v_1, v_2, \ldots, v_n\}, \{null\}\};$$

A case could be broken up into sub-cases that were the sentences of an original document. Each sentence can include each representative feature set that is selected according to ontology (thesaurus). The sub-case is illustrated as follows:

$$Sub\text{-}case = \{feature\_set, value\_set, negation\_sign\}$$
$$= \{\{f_1 \wedge f_2 \wedge f_j \wedge \ldots \wedge f_k\}, \{s_1, s_2, \ldots, s_l\}, \{null\}\}$$

In sub-case, a value set could be composed of several sentences and a value of a case could be decomposed by several sentences. Each sentence is defined by the Sub-case. Table 1 illustrates the algorithm for the generation of a case and a sub-case.

Table 1: Algorithm for Cases and Sub-case generation.

```
Tag_Set ← {t1, t2, ..., tn}
Document ← Web document
Feature_Set ← { }
Value_Set ← { }
Negation_Sign ← Null
Feture = Null
Value = Null

If Non_Exist (Tag set of document = Feature set of existing case)
Then
    Create Case (Feature_Set, Value_Set, Negation_Sign)
    Feature_Set of Case ← Tag_Set of a Document
    Value_Set of case ← the Document
    Negation_Sign ←{NOT|null}
Else
    Find the Case (Feature_Set, Value_Set, Negation_Sign)
    Value_Set of Case ← the Document
End if

Select sentences from the Document
Count the number of sentences in the document
Find features from the i'th sentence using ontology (or thesaurus)

N ← the number of sentences
i=0
While N<>0
    i =i+1
    If Non_Exist (Tag set of sentence = Feature set of existing case)
    Then
        Create Sub-case (Feature_Set, Value_Set, Negation_sign)
        Value of Sub-case ← the i'th sentence of the document
        Feature_set of Sub-case ← features of the i'th sentence
        Value_set of Sub-case ← the i'th sentence
        Negation_Sign ←{NOT|null}
    Else
        Find the Sub-case (Feature_Set, Value_Set, Negation_Sign)
        Value_Set of Sub-case ← the i'th sentence
    N ← N-1
While-end
```

# 5 CASE RETRIEVING USING TWO-STEP SIMILARITY

In this study, we assumed that the information user's requirements are sent to the agent-based information customization system. The similarity calculation module of the case customisation agent is invoked whenever the agent receives new documents.

## 5.1 Requirement-feature Similarity

The requirement-feature similarity (RF-similarity) is applied for selecting the most similar case, depending on the features, between the user's information requirements (with vector form) and the feature set. The details of the requirement-feature similarity are as follows.

First, the feature sets of cases are categorised into two groups by comparing information user's requirements with the feature sets of the cases. The one consists of the features that have exactly matched elements according to the user's requirements; the other consists of mismatch elements with user's requirements. If the feature set semantically subsumes an element of the requirements, then the features are grouped into the first group. For instance, if an element of the requirements is 'Seoul' and a feature set of a case is (Seoul, Korea), then the element is identical with the feature. The conceptual similarity for an exactly matched group is denoted in the first term of equation (1). The second term of equation (1) refers to the conceptual similarity between the features and the elements that have different values. We obtain the elements of the conceptual similarity between a user's requirements vector and the cases by using the weighted sum of the similarities between elements of the user's requirements and the features of the cases.

$$cs(r, c_i) = \sum_{\substack{r_k = f_{ij}, r_k \in f_{ij}, \\ k=1,2,\ldots,m_t}} cs(r_k, f_{ij}) + \sum_{\substack{r_k \neq f_{ij} \\ k=1,2,\ldots,mm_t}} MAX_{\forall r_k, f_{ij}} \left(cs\left(r_k, f_{ij}\right)\right) \quad (1)$$

The $cs(r, c_i)$ represents the similarity between a user's requirements and the $i^{th}$ case. $w_{m_t}$ and $w_{mm_t}$ represents the weights of the exactly matched terms and the mismatched terms. The $cs(r_k, f_{ij})$ represents the conceptual similarity between the $k^{th}$ element in the requirements and the $j^{th}$ feature of the $i^{th}$ case. The conceptual similarity calculation methods are feature-based, information contents-based and hybrids (Slimani T., B. Ben Yaghlane, and K. Mellouli, 2006, Song Ling et al., 2007). In this study, the edge-based conceptual similarity method is applied to the proposed system because it is intuitional and simple to use. After calculating the conceptual similarity, if the similarity is over the threshold value δ, then the value similarity between the user's requirements and the selected cases is calculated

### 5.2 Requirement-value Similarity

The calculation of the requirement-value similarity (RV-similarity) uses the cases that are selected from the requirement-feature similarity calculation to find the most similar value. If the selected case from the requirement-feature similarity doesn't satisfy the user's requirements, then we have to find the most appropriate value from the sub-cases. To find the most appropriate value from the selected values, the domain ontology is applied. First of all, we parse the selected values into term sets which are composed of classes or instances in domain ontology. To determine value of requirement, we calculate semantic distance between requirements and term sets. If the value $v_{ij}$ *of* sub-case $i$ is composed of k number of terms $t_{ijk}$, then the semantic distance between a feature and a value $v_{ij}$ is calculated by formula (2). The number of edges between a feature node and the term nodes in the domain ontology is expressed by $|n_{ijk} - n_{feature}|$.

$$sd(feature, v_{ij}) = Min_{v_j}\left(\sum_k |n_{t_{ijk}} - n_{feature}|\right) \quad (2)$$

## 6 CASE CUSTOMIZATION

In the customization process, we adjust the selected case to get better customized information. Table 2 shows the algorithm.

The details of each step in this algorithm are as follows:

*Step1: Case Adjustment*

- If the selected case includes extra information that is not included in the user's requirements, then delete this from the selected case. To do this, find the feature of extra information and delete the sentences that include the features.
- If the selected case doesn't satisfy the user's requirements, then define the required features. Select the sub-cases that contain the features. The RV-similarity is processed to find the most similar value from values of the sub-cases.

*Step2: Confliction Check*

After adding or deleting a feature and a value in the earlier case, depending on the user's information requirements, it is necessary to check whether or not any conflict takes place among the contexts. As we have already mentioned, the relationship between the features in a feature set of a case is assumed to be an 'and' relationship such as $\{f_1 \wedge f_2 \wedge f_3, \ldots . f_n\}$.

Table 2: Algorithm for a Case customization.

```
//Step1. Case Adjustment
If the case has non-necessary information //
Then
    Find the non-necessary features
    Delete the values form the sentences including non-necessary features
Else if the case has insufficient information
    Define the features of additional information
    Select Sub-case including the features from the Sub-case-base
    Calculate the requirement-value similarity between requirements and values
    If (similarity threshold point of the Value > δ)
    Then
        Value_set of the Adjusted_Case ← the value
    Else
        Discard the value
    End if
End if

//Step2. Case Confliction Check using Values (Find the confliction between values in the
adjusted case)
Value_set = {v1, v2, ..., vn}
Term_set = {k1, k2, ..., km}
N ← Count the number of values from Value_Set
While N <> 0
  If the value includes Negation_Sign
  Then
      Check Term_Set between the value and the others
      Calculate the requirement-value similarity between them
      If (Similarity threshold point between them > δ)
      Then
          Add 'or' to between vi and vj
      Else
          Add 'and' to between vi and vj
      End if
  Else
  Endif
  N←N-1
End while

//Step3. Conflict Resolution
If Exist (Confliction between fi and fj)
Then
    Discard a feature and the value including lower similarity point between requirements and
    Term_set from the Feature_set and Value_set
Else if Exist (Confliction between fi and two more features)
Then
    Discard fi and the value from the Feature_set and Value_set
End if
End if

// Step4. Update Case-base
        Add the case to Case-base
```

However, according to deletion or updating, it is possible to build new relationships between features.

A confliction check is as follows. Feature $f_1$ is an element of a feature set = $\{f_1 \wedge f_2 \wedge f_3 \wedge f_4, \ldots, f_n\}$, and value $v_1$ is matched by feature $f_1$ and has a term set=$\{k_1, k_2, k_3, k_4, \ldots, k_p\}$. Feature $f_2$ is an element of a feature set = $\{f_1 \wedge f_2 \wedge f_3 \wedge f_4, \ldots, f_n\}$, and value $v_2$ is matched by feature $f_2$ and it has a term set=$\{k_1, k_2, k_3, k_4, \ldots, k_p\}$. The term set of value $v_1$ is compared with the term set of value $v_2$. The negation sign is used to check for conflict that happens among values even though these values contain the same keyword term set. To check the negation among values, the keywords of the term sets are compared and the similarity between them is measured. If the similarity is over the threshold δ, then the confliction is defined between them. The confliction between the features is illustrated as $\{f_1 \wedge f_2 \vee f_3 \wedge f_4, \ldots, f_n\}$, $\{(f_1 \wedge f_2) \vee f_3, \ldots, f_n\}$, and so on.

*Step3: Conflict Resolution*

If a confliction happens between feature $f_i$ and $f_j$, then it is necessary to determine which feature can be deleted from the feature set. Three possible resolutions are as follows.

- Delete the conflicting features and the values from the feature set and value set of the adjusted case.
- If a confliction happens between $f_i$ and $f_j$, then compare the term set of the value of the conflicted features with the user's requirements. Delete the less similar feature and value from the feature set and value set according to the requirement-value similarity check.
- If a feature has a confliction with one more value of features such as $\{(f_i \wedge f_j) \vee f_k\}$, then it means that there are conflictions between $f_i$ and $f_k$ such as $(f_i \vee f_k)$ and $f_j$ and $f_k$ such as $(f_j \vee f_k)$. Delete the feature $f_k$ from the feature set. So, if a feature is conflicting with one more feature, then delete the feature from the feature set.

*Step 4: Update the Newly Generated-case to the Case-base*

Update the case-base with the newly adjusted case, which include the new feature set and the value set.

# 7 CONCLUSIONS AND FURTHER RESEARCH

The agent-based information customization system contains two types of agents for information custom-tailoring. The case generation agent supports the transformation of documents as information resources into cases as structured information. The case customization agent uses the two-step similarity calculation and the case customization process.

In this research, the cases and sub-cases are composed of features and values. The similarity check to select the most similar case is based on RF-similarity and RV-similarity. In addition, ontology is applied to the similarity check to differentiate the similarities between the differently expressed features. To adjust the selected case, the customization process is suggested

The proposed system will be useful to information users who usually need to customise various types of information from a variety of resources. For instance, this system can be applied to travellers who usually customize travelling information systems to obtain qualified information. For future work, we are planning to apply the agent-based information customization system to a travelling information system to prove the effectiveness of our proposed system.

# ACKNOWLEDGEMENTS

This work was supported by Defense Acquisition Program Administration and Agency for Defense Development under the contract UD080042AD, Korea.

# REFERENCES

Alexiev Vladimir, *et al.,* 2005. Information Integration with Ontologies: Experiences from an Industrial Showcase. Wiley.

Bernstein A. Philip and Laura M. Haas, 2008. Information Integration in the Enterprise. Communications of the ACM, Vol. 51, No. 9.

Halevy Y. Alon, et al., 2005. Enterprise information integration: successes, challenges and controversies. Proceedings of the 2005 ACM SIGMOD international conference on Management of data.

Karmarkar, U. S., Apte, U. M., 2007. Operations management in the information economy: Information products, processes, and chains. Journal of Operations Management. Vol. 25, Issue 2, pp. 438-453.

Lee, Y., Allen, T., Wang, R., 2001. Information Products for Remanufacturing: Tracing the Repair of an Aircraft Fuel-Pump. *Sixth International Conference on Information Quality.*

Mani, I., Bloedorn, E., 1998. Machine learning of Generic and User-Focused Summarization. *In proceedings of the National Conference on Artificial Intelligence (AAAI).*

Pan, B., and Fesenmaier, D. R., 2006. online information search Vacation Planning Process. Annals of Tourism Research. Vol. 33, No. 3, pp. 809–832.

Teufel, S., Moens, K., 1997. Sentence extraction as a classification task. *Workshop 'Intelligent and scalable text summarization'*, ACL/EACL 1997.