# Service Quality Improvement in Web Service Based Machine Transaltion

**Sapa Chanyachatchawan** and **Virach Sornlertlamvanich** and **Thatsanee Charoenporn**
National Electronics and Computer Technology Center (NECTEC)
112 Phahon Yothin Rd., Klong 1, Klong Luang, Pathumthani 12120,Thailand
{sapa.cha,virach.sor,thatsanee.cha}@nectec.or.th

## Abstract

There are many approaches to increase web service based machine translation result. However, perfect result alone does not guarantee the quality of translation service or user satisfaction. This paper proposes framework to improve translation service by using non functional attributes information. In this paper, we present methodology to measure quality of composite translation service using existing services information and also the guideline for selecting the composition web service which has highest quality of service.

## 1 Introduction

The advantage of web service based machine translation is the ability to create new language pairs from existing language pairs. This process is based on web service composition in SOA (Service Oriented Architecture). Langrid Project (NICT, 2011) is an example of machine translate service based on web service composition technique. Langrid user can create multihop translation from existing language pairs. The automatic composition process increases accessibility for end users transparently. The most challenging task among the automatic composition processes is the discovery process. Based on W3C, web service description standard (WSDL1.1) defines only input and output name and basic data type for web service with a few descriptions. OWL-S is used to embed semantic into service input and output which enable software agent to discover service. However, translation accuracy does not relate to quality of composite service or user satisfaction. By embedding QoS (Quality of Service) attributes as nonfunctional attributes into web service description, we can improve quality of composite service result. This paper proposes machine translation service framework that can automatically create new language pair from existing resources. The objective of this paper is to provide framework with model to managed nonfunctional attributes and semantic of service. Framework is presented in section 2, where component and overall process are explained in brief. In section 3 discovery process is presented in general idea along with model for evaluate quality of web service and selection method.

## 2 Framework

In this section, we describe framework for managing web service composition process. The framework is illustrated in Figure 1. From user aspect, our framework is service provider. Users do not need to search and compose web service by themselves. In this paper, existing translation service and composite translation service are called service and composite service respectively.

Component of the framework consists of

- *Proxy Agent* is responsible to interact with users and manage user session.
- *Discovery Agent* is responsible to search and interact with external web service registies.
- *Service Agent* is responsible to invoke external web services.
- *Repository* is responsible for record composite web services and non functional attributes information that do not included in the original WSDL.
- *Compose Agent* is responsible for 1) interact with other component in framework 2) compose web service 3) evaluate web service quality.

The process flow of framework is describe as follow:



*Proceedings of Workshop on Language Resources, Technology and Services in the Sharing Paradigm*, pages 16–23,
Chiang Mai, Thailand, November 12, 2011.

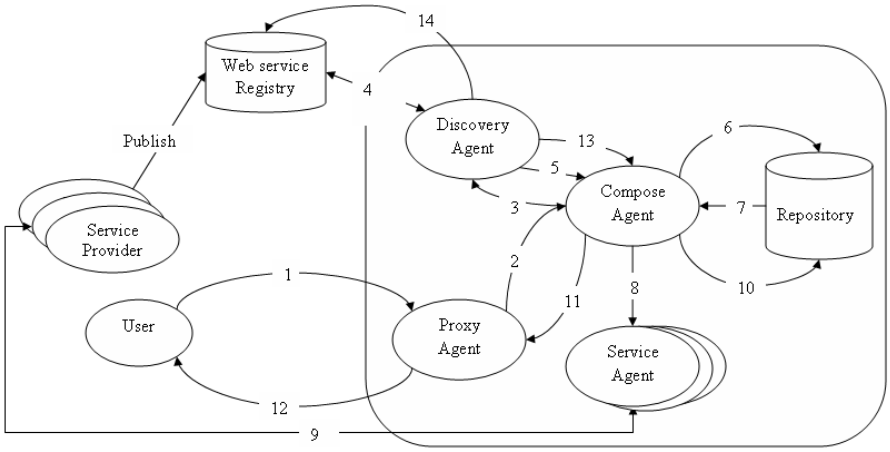


Figure 1: Framework

1. User send request to *Proxy Agent*. User request consists of input, output, and also semantic information.
2. User send request to *Proxy Agent*. User request consists of input, output, and semantic information.
3. *Proxy Agent* forwards request to *Compose Agent*.
4. *Compose Agent* forwards request parameters to *Discovery Agent*.
5. *Discovery Agent* searches services from web service registry.
6. *Discovery Agent* returns candidate results to *Compose Agent*
7. *Compose Agent* consults *Repository* if there is additional information about candidate results.
8. *Repository* returns related information to *Composer Agent*.
9. *Compose Agent* evaluates, selects web service, creates workflow, creates instances of *Service Agent*.
10. *Service Agents* binds web service.
11. After finish execution, *Service Agents* returns result to *Compose Agent*.
12. *Compose Agent* updates service information in *Repository*
13. *Compose Agent* forwards result to *Proxy Agent*.
14. *Proxy Agent* returns output to user
15. *Compose Agent* creates WDSL for composite service and forward to *Discovery Agent* and stores in *Repository*
16. *Discovery Agent* publishes composite service to web service registry

## 3 Evaluate quality of sevices

After discovery process, there are many composite services that satisfy user request. In order to select the highest quality service, quality of service is calculated using non functional attributes of service.

### 3.1 Non functional attributes of web services

Non function attributes can represent QoS(Quality of Service) which use to differentiate good service from others. QoS values in interval or ratio scale can be concerned as criterias for selecting optimal services. QoS values in nominal scale value use to reduce number of services. Some fundamental attributes will be used in this paper as an example

For any web service $S$, the $QoS$ attributes list below:

**Cost of service:** denotes as $QoS_{Cost}(S)$, this is the cost to pay for service provider in order to run service. The attributes consist of two parts; first part is taken directly from service provider called *direct cost*. Second part

is cost for set up and maintenance services called *indirect cost* which assume that the value is constant for every process throughout the whole system.

**Time of service:** denotes as $QoS_{Time}(S)$, this is the time measure from invoke service to receive respond from service in case of service process successfully. Time of service consists of *process time* and *delay time*. *process time* is time needed to run instance of service, *delay time* is overhead time. This value is kept in *Repository* and be updated every time that process finish as following equation

$$QoS_{Time}(S) =$$

$$\frac{((QoS_{Time}(S)_{N-1})*(N-1))+Time_N(S)}{N}$$

whereas $QoS_{Time}(S)_N$ is the average time of process after be invoked for $N$ time. This information is kept in *Repository*.

**Failure ratio:** denotes as $QoS_{Failure}(S)$, is ratio between number of failure and total number of execution. *Failure ratio* initial value is set to 0 and be updated by following formula

In case of service terminate normally;

$$QoS_{Failure}(S) = \frac{((QoS_{Failure}(S)_N)*(N-1))}{N}$$

In case of service terminate abnormally;

$$QoS_{Failure}(S) =$$

$$\frac{((QoS_{Failure}(S)_N)*(N-1))+1}{N}$$

whereas $QoS_{Failure}(S)_N$ is the failure ratio of process after be invoked for $N$ time.This information is kept in *Repository*.

**Unavailability:** denotes as $QoS_{Unavl}(S)$, is value to represent the unavailability of services. $QoS_{Unavl}(S)$ is obtained using this formula:

$$QoS_{Unavl}(S) = \frac{T(S)}{C}$$

Whereas, $T(S)$ is total amount of time (seconds) which service $S$ is not available in last $C$ seconds, $C$ is set by framework.

**User satisfaction:** denotes as $QoS_{Sat}$, is cardinal scale value represent satisfaction level of user, this value is variance depend on each users. This information is kept in *Repository*.

**Security level:** denotes as $QoS_{Sec}$, is cardinal scale value represent security level, this value is taken directly from service or trusted third party providers. This value is taken directly from service providers.

**Bandwidth:** denotes as $QoS_{Band}$, is bandwidth required for running process. This value is taken directly from service providers.

There are number of basic attributes used for measure $QoS$ in streaming application which allow some errors and lossy information.

**Error:** denotes as $QoS_{Error}$, is represent total number of noise and error (in bytes) occur while execute services.

**Delay:** denotes as $QoS_{Delay}$, is total delay and jitter time (in seconds) while execute services.

Some nominal scale non functional attributes that can not be convert to ratio scale, such as user context. These attributes are used to prune web services. Examples of these attributes are:

**Context:** denotes as $QoS_{Context}$, is set of context attributes represent context of users and their environment, such as location, demography information, or user browser environment.

Summarization of non functional attributes is presented in Table 1

### 3.2 Normalize QoS value

In order to compare or measure different attributes, QoS need to be normalized. Each attributes is assigned preference of its value (minimum, maximum). Each attributes are normalized as following:

**Cost of service:** is normalized by using transform table because of cost of service should not be linear function. Table 2 shows the example of normalization of $QoS_{Cost}(S)$.

**Time of service:** is normalized using formula:

$$\frac{QoS_{Time}(S)}{C_{MaxTime}}$$

Table 1: Non functional attributes summarization

| Attribute | Scale | Source | Description |
|---|---|---|---|
| $QoS_{Cost}(S)$ | Ratio | Service Provider | Service cost |
| $QoS_{Time}(S)$ | Ratio | *Repository* | Average execute time |
| $QoS_{Failure}(S)$ | Ratio | *Repository* | Failure ratio |
| $QoS_{Unavl}(S)$ | Ratio | *Repository* | Unavailability ratio |
| $QoS_{Sat}(S)$ | Cardinal | *Repository* | User satisfaction |
| $QoS_{Sec}(S)$ | Cardinal | Service Provider | Security level |
| $QoS_{Band}(S)$ | Ratio | Service Provider | Reqiured Bandwidth |
| $QoS_{Error}(S)$ | Ratio | *Repository* | Number of error information |
| $QoS_{Delay}(S)$ | Ratio | *Repository* | Service delay time |
| $QoS_{Context}(S)$ | Nominal | User | User context |

Table 2: Cost of service transform table example

| Cost(Dollars) | Value |
|---|---|
| 0-0.5 | 0.0 |
| 0.5-1 | 0.3 |
| 1-5 | 0.5 |
| 5-10 | 0.8 |
| $\geq 5$ | 1.0 |

Whereas $C_{MaxTime}$ is the maximum execute time assigned by framework.

**Failure ratio:** does not need to be normalized, because $QoS_{Failure}(S) \in [0, 1]$

**Unavailability:** does not need to be normalized, because $QoS_{Unavl}(S) \in [0, 1]$

**User satisfaction:** is not normalized and will be used as constraint.

**Security level:** is not normalized and will be used as constraint.

**Bandwidth:** is normalized using formula:

$$\frac{QoS_{Band}(S)}{C_{MaxBand}}$$

Whereas $C_{MaxBand}$ is the maximum bandwidth of framework.

**Error:** is normalized using formula:

$$\frac{QoS_{Error} - C_{MinError}}{C_{MaxError} - C_{MinError}}$$

Whereas $C_{MinError}$ and $C_{MaxError}$ are minimum and maximum error that framework allow to happen.

**Delay:** is normalized using formula:

$$\frac{QoS_{Delay} - C_{MinDelay}}{C_{MaxDelay} - C_{MinDelay}}$$

Whereas $C_{MinDelay}$ and $C_{MaxDelay}$ are minimum and maximum error that framework allow to happen.

**Context:** can not be normalized and will be used as constraint.

Table 3 is the summarization of normalized attributes.

### 3.3 Quality of composite service

Once the service has been composed, the $QoS$ of composite service will be calculated. Workflow of composite service determines how $QoS$ be computed. Workflow of composite service is divided into four types 1)sequential, 2)parallel, 3)conditional, 4)loop, and 5)complex. Parallel, conditional, loop and complex workflow are reduced into one atomic task.As the result of reduction, new workflow will consist of sequential workflow only, and then sequential workflow is reduced to one atomic workflow. Only ratio scale and interval scale attributes will computed here. Hence, $QoS$ of composite service is calculated.

#### 3.3.1 Sequential workflow

Sequential workflow $CS$ (Figure 2)consists of $n$ sequential process denote as $S_i; 1 \leq i \leq n$. The work flow start at service $S_1$ and finish at service $S_n$. Process $S_i$ must finish before process $S_{i+1}$ can start.

The $QoS$ of $CS$ can be obtained as follows:

$$QoS_{Time}(CS) = \sum_{i=1}^{n} QoS_{Time}(S_i)$$

Table 3: Normalized functional attributes

| Attribute | Range | Preferred value | Description |
|---|---|---|---|
| $QoS_{Cost}(S)$ | $[0, 1]$ | Minimum | Service cost |
| $QoS_{Time}(S)$ | $[0, 1]$ | Minimum | Average execute time |
| $QoS_{Failure}(S)$ | $[0, 1]$ | Minimum | Failure ratio |
| $QoS_{Unavl}(S)$ | $[0, 1]$ | Minimum | Unavailability ratio |
| $QoS_{Sat}(S)$ | - | - | User satisfaction |
| $QoS_{Sec}(S)$ | - | - | Security level |
| $QoS_{Band}(S)$ | $[0, 1]$ | Minimum | Required Bandwidth |
| $QoS_{Error}(S)$ | $[0, 1]$ | Minimum | Number of error information |
| $QoS_{Delay}(S)$ | $[0, 1]$ | Minimum | Service delay time |
| $QoS_{Context}(S)$ | - | - | User context |

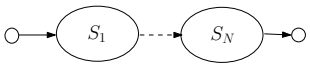


Figure 2: Linear workflow

$$QoS_{Cost}(CS) = \sum_{i=1}^{n} QoS_{Cost}(S_i)$$
$$QoS_{Failure}(CS) = 1 - \prod_{i=1}^{n}(1 - QoS_{Failure}(S_i))$$
$$QoS_{Unavl}(CS) = 1 - \prod_{i=1}^{n}(1 - QoS_{Unavl}(S_i))$$
$$QoS_{Band}(CS) = MAX_{1\leq i\leq n}(QoS_{Band}(S_i))$$
$$QoS_{Error}(CS) = \sum_{i=1}^{n} QoS_{Error}(S_i)$$
$$QoS_{Delay}(CS) = \sum_{i=1}^{n} QoS_{Delay}(S_i)$$

#### 3.3.2 Parallel workflow

Parallel workflow (Figure 3) $CS$ consists of $n$ parallel process denote as $S_i; 1 \leq i \leq n$, each process work independently and can start at same time.

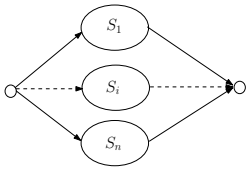


Figure 3: Parallel workflow

The $QoS$ of $CS$ can be obtained as follows:

$$QoS_{Time}(CS) = MAX_{1\leq i\leq n}(QoS_{Time}(S_i))$$
$$QoS_{Cost}(CS) = \sum_{i=1}^{n} QoS_{Cost}(S_i)$$
$$QoS_{Failure}(CS) = 1 - \prod_{i=1}^{n}(1 - QoS_{Failure}(S_i))$$
$$QoS_{Unavl}(CS) = 1 - \prod_{i=1}^{n}(1 - QoS_{Unavl}(S_i))$$
$$QoS_{Band}(CS) = \sum_{i=1}^{n} QoS_{Band}(S_i)$$
$$QoS_{Error}(CS) = \sum_{i=1}^{n} QoS_{Error}(S_i)$$
$$QoS_{Delay}(CS) = MAX_{1\leq i\leq n}(QoS_{Delay}(S_i))$$

#### 3.3.3 Conditional workflow

Conditional workflow $CS$ (Figure 4)consists of $n$ process denote as $S_i; 1 \leq i \leq n$, only one process will be execute. $p_i$ is the probability of process $S_i$ to be execute and $\sum_i^n p_i = 1$, these value store from $Repository$.

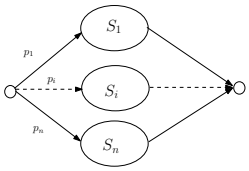


Figure 4: Conditional workflow

The $QoS$ of $CS$ can be obtained as follows:

$$QoS_{Time}(CS) = \sum_{i=1}^{n}(p_i * QoS_{Time}(S_i))$$
$$QoS_{Cost}(CS) = \sum_{i=1}^{n}(p_i * QoS_{Cost}(S_i))$$
$$QoS_{Failure}(CS) = \sum_{i=1}^{n}(p_i * QoS_{Failure}(S_i))$$
$$QoS_{Unavl}(CS) = \sum_{i=1}^{n}(p_i * QoS_{Unavl}(S_i))$$
$$QoS_{Band}(CS) = \sum_{i=1}^{n}(p_i * QoS_{Band}(S_i))$$
$$QoS_{Error}(CS) = \sum_{i=1}^{n}(p_i * QoS_{Error}(S_i))$$
$$QoS_{Delay}(CS) = \sum_{i=1}^{n}(p_i * QoS_{Delay}(S_i))$$

#### 3.3.4 Loop workflow

For loop workflow (Figure 5), there is condition of loop to simplify the calculation. Give $CS$ is composite service that created by repeat execution of service S with $p$ is the chance that service will be repeat and loop must be execute service S at least one time.

The $QoS$ of $CS$ can be obtained as follows:

$$QoS_{Cost}(CS) = \frac{QoS_{Cost}(S)}{(1-p)}$$
$$QoS_{Time}(CS) = \frac{QoS_{Time}(S)}{(1-p)}$$

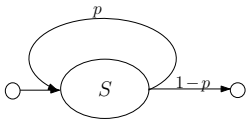


Figure 5: Loop workflow

$$QoS_{Failure}(CS) = 1 - \frac{(1-p)*(1-QoS_{Failure}(S))}{(1-p*(1-QoS_{Failure}))}$$
$$QoS_{Unavl}(CS) = 1 - \frac{(1-p)*(1-QoS_{Unavl}(S))}{(1-p*(1-QoS_{Unavl}))}$$
$$QoS_{Band}(CS) = QoS_{Band}(S_i)$$
$$QoS_{Error}(CS) = \frac{QoS_{Error}(S)}{(1-p)}$$
$$QoS_{Delay}(CS) = \frac{QoS_{Delay}(S)}{(1-p)}$$

#### 3.3.5 Complex workflow

Complex workflow $CS$ (Figure 6)consists of $N$ process denote as $S_i; 1 \leq i \leq n$, it is acyclic directed graph.

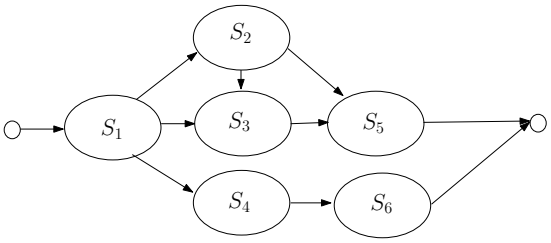


Figure 6: Complex workflow

The $QoS$ of $CS$ can be obtained as follows:

$$QoS_{Cost}(CS) = \sum_{i=1}^{n} QoS_{Cost}(S_i)$$
$$QoS_{Failure}(CS) = 1 - \prod_{i=1}^{n}(1 - QoS_{Failure}(S_i))$$
$$QoS_{Unavl}(CS) = 1 - \prod_{i=1}^{n}(1 - QoS_{Unavl}(S_i))$$
$$QoS_{Error}(CS) = \sum_{i=1}^{n} QoS_{Error}(S_i)$$

For the calculation of $QoS_{Time}$ and $QoS_{Delay}$, concept of finding critical path in work flow is applied, method such as *Finding the critical path in a time-constrained workflow*(Son and Kim, 2000) , *Finding Multiple Possible Critical Paths Using Fuzzy PERT* (Chen and Chang, 2001) or *Critical Path Method (CPM)*(Samuel, 2004) can be used to critical path. Given set $A$ that member of service $A$ are services in critical path. $QoS_{Time}$ and $QoS_{Delay}$ will be:

$$QoS_{Time}(CS) = \sum_{S_i \in A} QoS_{Time}(S_i)$$
$$QoS_{Delay}(CS) = \sum_{S_i \in A} QoS_{Delay}(S_i)$$

Due to complexity of workflow, the composite bandwidth will be use the maximum bandwidth required by services.

$$QoS_{Band}(CS) = MAX_{1 \leq i \leq n}(QoS_{Band}(S_i))$$

### 3.4 Objective function

Objective function is used to evaluate the fitness of composite service. As many attributes are considered, the single unique value is needed for comparison between each possible combination. In framework, $QoS$ of composite service is defined by formula:

$$QoS(X) = \sum_{i=1}^{N} (w_i * QoS_i(X))$$

whereas $N$=number of attributes; $w_i$=weight of attribute $i^{th}$

Our objective is to find composite service that have minimum $QoS$ value, thus objective function will be

$$\min QoS(X) = \min \left(\sum_{i=1}^{N} (w_i * QoS_i(X))\right)$$

some constraints are defined for composite service to represent real life constraints.

$$QoS_i(CS) \leq C_i \text{ for each } i \in 1, ..., N$$

whereas $N$=number of attributes; $C_i$=constraint of attribute $i^{th}$

### 3.5 Selecting web service

$QoS$ function from previous section consists of non linear parameter which make calculation complex. To simplify problem, some assumptions are given 1) suppose that only one possible workflow returned from discovery process 2) composite service is not streaming application. 3) user context is irrelevant. 4) workflow consist only sequential processes. Figure 7 show an example of such a workflow.

Cardinal and nominal scale $QoS$ and non-linear composite $QoS$ ($QoS_{Failure}$, $QoS_{Unavl}$, $QoS_{Band}$, $QoS_{Sat}$ and $QoSSec$) are excluded from objective function and used to prune to discovered services. Hence, workflow objective function and constraints are solely linear function, and then 0-1 linear programming model is applied.

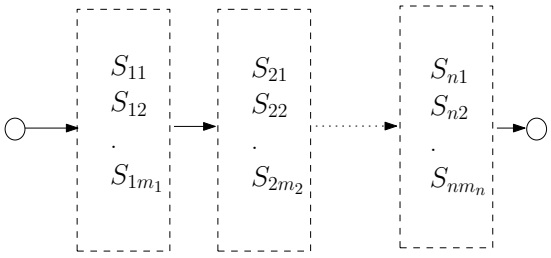


Figure 7: Output from discovery process

After discovery process, set of discovered service is pruned with set of constraints

$\{QoS_{Failure}$,$QoS_{Unavl}$,$QoS_{Band}$,$QoS_{Sat}$, $QoS_{Sec}\}$.

The workflow consists of $n$ process, each processes there is set or service with size $m_i; 1 \leq i \leq n$ that can fulfill process requirements.

Then we introduce set of variable $x_{ij}$ to represent decision variable.

$$\begin{pmatrix} x_{11} & & x_{n1} \\ \vdots & \ddots & \vdots \\ x_{1k} & & x_{nm_i} \end{pmatrix}$$

The variable $x_{ij}$ is correspond to $S_{ij}; 1 \leq i \leq n$ and $1 \leq j \leq m_i$. Whereas $n$ is number of process. $x_{ij} = 1$ iff service $x_{ij}$ has been selected, otherwise $x_{ij} = 0$

Hence, problem is transformed to linear programming problem:

minimize: $\sum_{i=1}^{n} \sum_{j=1}^{m_i} QoS(S_{ij}) * x_{ij}$

whereas $QoS(S_{ij}) = (w_{Time} * QoS_{Time}(S_{ij})) + (w_{Cost} * QoS_{Cost}(S_{ij}))$

subject to

$$\sum_{j=1}^{m_i} x_{ij} = 1$$
$$x_{ij} \in \{0, 1\}$$
$$QoS_{Time}(S_{ij}) \leq C_{Time}$$
$$QoS_{Cost}(S_{ij}) \leq C_{Cost}$$

## 4 Update repository information

After composite that have the best $QoS$ has been selected and executed, there are processes after finish. There are two cases 1) composite service terminate normally 2) composite service terminate abnormally. In later case, we update service information ($QoS_{Failure}$) in *Repository*. Process will not repeat because of services that makes composite service fail tends to have better $QoS$ value than others. As the result, other combination of this service must be excluded and rediscover web services again. In case of composite service terminate normally, service information ($QoS_{Failure}$, and $QoS_{Time}$) is updated in to *Repository*, and publish composite service to *Web service Registry* with $QoS$ information. $QoS$ information can be added to WSDL as extension (*Unreaveling the web services: an introduction to SOAP, WSDL, and UDDI*)(Curbera, 2002),using *Semantic Annotations for WSDL* ,or using OWL-S.

## 5 Related works

There are many related studies about quality of machine translation notably ones include *Automatic Evaluation of Machine Translation Quality Using Longest Common Subsequence and Skip-Bigram Statistics* (Lin and Och, 2004) and *METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments* (Banerjee and Lavie, 2005). There are researches about quality of service in service composition process such as *QoS-Aware Middleware for Web Services Composition - A Qualitative Approach* (Yeom, Yun and Min, 2006). There are two important processes that we do not focus in our framework. The first process is discovery process which are managed by *Discovery Agent* in our framework. Algorithms for discovery services are not included in this paper. There are many related studies in searching non perfect match web service such as *Automate Composition and Reliable Execution of Ad-hoc Processes* (Binder, Constatinescu, Faltings, Haller, and Turker, 2004) and *A software Framework For Matchmaking Based on Semantic Web Technology* (Li and Horrocks, 2003). Other research work as in *Ontology assisted Web services discover* (Zhang and Li, 2005) and *Web Service Discovery via Semantic Association Ranking and Hyperclique Pattern Discovery* (Paliwal, Adam, Xiong and Bornhovd, 2006) use semantic information to discover web services.

The second process is how to search for the optimal quality composite service from all possible combination of services. We can apply linear programming technique as described cutting method in *A lift-and-project cutting plane algorithm for mixed 01 programs* (Balas, Ceria and Cornuejols, 1993) to perform this search task.

## 6 Conclusion and future work

The main contribution of this paper is to propose a web service based machine translation framework that enhances quality of translation. We present the concept of embedding quality of service information, method to measurement QoS, and calculation of composite translation QoS.

For future work, we plan to work on simplifing the search space, discovery techniques using semantic, mathemetic model for solving integer programming problem, fault tolerance, and implementation of ontology to describe QoS attributes in services.

## References


Language Grid. 2011. *Language Grid* http://http://langrid.nict.go.jp/en/index.html

Samuel L. Baker. 2004 *Critical Path Method (CPM)* http://hadm.sph.sc.edu/COURSES/J716/CPM/CPM.html

Jin Hyun Son and Myoung Ho Kim. 2000. *Finding the critical path in a time-constrained workflow* Seventh International Conference on Real-Time Computing Systems and Applications (RTCSA'00) 2000 p. 102

Shyi-Ming Chen and Tao-Hsing Chang. 2001. *Finding Multiple Possible Critical Paths Using Fuzzy PERT* IEEE Transactions on Systems, Man and Cybernetics, Part B,2001

Walter Binder, Ion Constatinescu, Boi Faltings, Klaus Haller, and Can Turker. Barcelona, 2004 2004. *Automate Composition and Reliable Execution of Ad-hoc Processes* Second European Workshop on Multi-Agent Systems (EUMAS)

I. Li and Horrocks. 2003 *A software Framework For Matchmaking Based on Semantic Web Technology* In Proc, 12th Int Conf on the World Wide Web, 2003

Curbera F. 2002. *Unreaveling the web services: an introduction to SOAP, WSDL, and UDDI* IEEE Internet Computing, Vol. 6. No.2,pp. 86-93,2002.

Egon Balas ,Sebastian Ceria and Gerard Cornuejols. 1993 *A lift-and-project cutting plane algorithm for mixed 01 programs* Mathematical Programming, Volume 58, Numbers 1-3 / January, 1993,pp. 295-324

Po Zhang, Juanzi Li. *Ontology assisted Web services discover*. Service-Oriented System Engineering, 2005. IEEE International Workshop, 20-21 October 2005 Page(s):45 - 50

Paliwal, A.V.; Adam, N.R.; Hui Xiong; Bornhovd, C. 2006 *Web Service Discovery via Semantic Association Ranking and Hyperclique Pattern Discovery* Web Intelligence, 2006. WI 2006. IEEE/WIC/ACM International Conference on 18-22 Dec. 2006 Page(s):649 - 652

Issa, H.; Assi, C.; Debbabi, M.; 2006. *QoS-Aware Middleware for Web Services Composition - A Qualitative Approach* Computers and Communications, 2006. ISCC '06. Proceedings. 11th IEEE Symposium on 26-29 June 2006 Page(s):359 - 364

Chin-Yew Lin and Franz Josef Och 2004. *Automatic Evaluation of Machine Translation Quality Using Longest Common Subsequence and Skip-Bigram Statistics.* 42nd Annual Meeting of the Association for Computational Linguistics, Barcelona, Spain, July 21- 26, 2004.

Banerjee, S. and Lavie, A. 2005. *METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments* Workshop on Intrinsic and Extrinsic Evaluation Measures for MT and/or Summarization at the 43rd Annual Meeting of the Association of Computational Linguistics (ACL-2005), Ann Arbor, Michigan, June 2005