2010

Selection of Web Services Based on Opinion Mining of Free-Text User Reviews

Nicholas A. Thurow
Cedarville University, nthurow@cedarville.edu

John D. Delano
Cedarville University, jdelano@cedarville.edu

Selection of Web Services Based on Opinion Mining of Free-Text User Reviews

Research-in-Progress

Nicholas A. Thurow
Cedarville University,
Department of Business Administration
251 N. Main St., Cedarville, OH 45314
nthurow@cedarville.edu

John D. Delano
Cedarville University,
Department of Business Administration
251 N. Main St., Cedarville, OH 45314
jdelano@cedarville.edu

Abstract

When multiple web services exist that perform identical tasks, non-functional attributes must be considered in order to choose the best service. Quality-of-service (QoS) attributes are often used to differentiate functionally redundant web services. However, ranking services according to QoS attributes is a complex problem. Additionally, the use of test data to establish those QoS ratings does not always yield accurate results. Therefore, this paper proposes a method that utilizes opinion mining techniques to extract information about the QoS attributes of a web service based on free-text user reviews. This method not only has the advantage of using real-world data rather than test data, but it also ensures that a variety of use cases are tested that would be common in the everyday usage of that service.

Keywords: Service-oriented architecture, web services, opinion mining, quality of service

Introduction

Service oriented architecture (SOA) has the potential to transform the way business transactions are carried out by streamlining business processes and integrating those processes with other businesses (Papazoglou and Heuvel 2007). SOA provides a platform-independent architecture for organizations to communicate, which allows an organization to enjoy the advantages of this powerful technology without forcing it to drastically change the way it conducts business or the platforms it uses to automate business processes (Erl 2005). The primary medium through which SOA is implemented is through XML-based web services (Chaari et al. 2008). XML provides a way for applications to communicate in a standardized way. In addition, XML provides a means of data validation, and enables standards, such as SOAP, WSDL, and UDDI, which further enable service-oriented principles such as reusability, composability, and loose coupling.

Existing research has already attempted to catalog and locate software components based on their functional characteristics (Vitharana et al. 2003). However, it is not uncommon for multiple web services to perform identical tasks. Thus, when services are redundant, non-functional aspects (such as QoS attributes) must be used to differentiate these services (Wang et al. 2006). While extant research has addressed the evaluation of web services based on quality of service (QoS) metrics, the current approaches introduce subjective biases that are difficult to control. This paper will attempt to address this weakness by proposing a new method for evaluating web services, based on mining opinions stored in free-text user reviews.

This paper is structured as follows: the next section will provide a review of QoS metrics and their relationship with the underlying XML technology. The following section will examine the current methods for evaluating QoS metrics to select web services. The next section will present the proposed method for selecting web services. We then describe our research method and discuss the impact of our findings. The paper concludes with a discussion of current limitations, future research plans, and a conclusion.

QoS Metrics and XML

Though XML-based web services provide an excellent vehicle for experiencing a SOA, they are optimized for SOA principles such as reusability rather than for QoS metrics like performance. Table 1 lists some of the QoS metrics that are usually identified for web services (Mani and Nagarajan 2002). Thus, though web services accomplish the primary goals of SOA, they do so in a fairly inefficient manner (Kohlhoff and Steele 2003). Some of the main performance concerns for XML-based web services are the use of SOAP to serialize and de-serialize data, the parsing of XML documents, and the verbosity of XML.

Table 1. Quality of Service Metrics

Quality-of-Service Metric	Description
Availability	The probability that a service is available.
Accessibility	The probability that an available web service is able to serve requests. Even when a web service is available, it may not be accessible, such as when the service is experiencing a high degree of usage.
Integrity	The ability of a web service to correctly carry out a transaction. Ideally, a web service applies the ACID principle to all transactions, meaning transactions are atomic (if any part of the transaction fails, any and all changes are rolled back), consistent (data conforms to the set of integrity constraints, once the transaction is completed), isolated (transactions are unaware of each other and the effects of transactions are not visible until the transaction is completed), and durable (once a transaction completes, and only when a transaction completes, changes made are guaranteed to persist) (Gray and Reuter 1993).
Performance	The amount of latency (time between a request and a response) and throughput (ability to serve multiple requests) of a web service.
Reliability	The number of failures in a given time period.
Regulatory	The level of compliance of web services with certain rules, laws, and standards, such as SOAP, UDDI, and WSDL. Compliance with regulations and standards is a fundamental reason for web services' usefulness as a foundation for realizing SOAs (Erl 2005).
Security	The presence of access controls, such as authentication and encryption are vital to preserving confidentiality and non-repudiation.

SOAP has emerged as the standard messaging format for web services. However, this XML-based protocol suffers from basic inefficiency that other models such as EJB, CORBA, and DCOM avoid (Jun et al. 2006). The process of converting binary data into an XML, textual representation and then back to binary is a resource-intensive process. Furthermore, though converting binary data to plain text produces human-readable data, the result is far more verbose than the binary representation. For example, though it is clear what `<weight>255</weight>` means when seen in an XML document, the tradeoff is that 20 bytes of data are needed (using UTF-8 encoding) to transmit an 8-bit number like 255.

Not only does the verbosity of XML documents result in larger transmission times, but it also adds to the performance problems involved with parsing an XML document. Before data in an XML document can be de-serialized, it must be extracted from the document. This extra step further increases total latency of a web service, and verbose, human-friendly XML documents suffer the largest performance penalty because more data must be buffered, parsed, and handled. Due to the increase in latency and decrease in throughput resulting from large XML files, much research has been carried out in an attempt to find the best way to compress XML files (Ng et al. 2006). The first of these XML compressors, XMill, is able to outperform general-purpose compressors (such as gzip) in both time and compression ratio (Liefke and Suciu 2000). However, the resulting documents are no longer human-readable and must be decompressed in order to be parsed, queried, or updated, therefore allowing an increase in throughput at the cost of an increase in latency. XGRIND attempts to solve both the latency and throughput problems by leaving data in a human-readable form, using a string-compression algorithm known as the Huffman

algorithm (Tolani and Haritsa 2002). Other XML compression algorithms have also been proposed, however their usefulness in the realm of web services is unclear due to the fact that all clients must use the same compression and decompression algorithms. Web services rely on standards to provide interoperability, so unless a particular algorithm is adopted as a standard, it will not be able to enhance the QoS of web services.

Existing Web Service Selection Methods

Almost as important as choosing the service that most closely mirrors desired functionality is choosing a service that meets company QoS requirements. Low latency and high throughput may be more desirable for certain industries, such manufacturing, while integrity might be more critical to another industry, such as e-commerce businesses. For example, if a system failure occurs during the order placement process, a customer does not want to be left wondering whether or not his or her purchase went through. In still other industries, security may be a primary concern. Therefore, QoS requirements must be considered when choosing a web service.

To solve the problem of selecting the web service that meets QoS requirements, Chaari et al. (2008) propose an extension to the WS-Policy standard to include details about QoS requirements and QoS capabilities. The authors suggest the publication of details about QoS capabilities to a service's UDDI registration entry. The WS-Policy extension includes details such as the policy category (which maps to one of the QoS categories discussed above), the QoS expression (the QoS attribute, the desired value for that attribute, the units that the value is in, and an operator, such as "greater" or "equal"), as well as a parameter identifying whether or not the assertion is negotiable. Figure 1 shows a sample excerpt of a QoS requirement contained in a UDDI entry, as proposed by Chaari et al. (2008), that asserts a response time of less than ten seconds.

Using this method, users are able to discover services, which meet specific performance criteria. The service composition engine constructs a matrix containing the list of desired QoS attributes along with the list of services in the registry. A "1" is placed wherever a particular service meets the requirements of a particular attribute, and a "0" is placed where the service does not satisfy the requirements. The services are then ranked according to the number of 1's contained in their row of the matrix, thus making the best service the one that satisfies the most requirements. In addition to discovering new services, though, this WS-Policy extension can also be used to find substitute services. In this case, a search is performed on the registry using the performance criteria of the old service as requirements for the new service.

```
<wspes:Assertion name="AssertionA">
      <wspes:expression>
            <wspes:paramater>qosid:ResponseTime</wspes:parameter>
            <wspes:Value>10</wspes:Value>
            <wspes:Unit>qosu:seconds</wspes:Unit>
            <wspes:Operator>less</wspes:Operator>
      </wspes:expression>
</wspes:Assertion>
```

Figure 1. WS-Policy Extension (Chaari et al. 2008)

In contrast to searching for a list of services that meet particular criteria, there are several approaches that use a linear or integer-programming approach to find the best service composition based on the QoS attributes of a given set of services (Kokash and D'Andrea 2007). Zeng et al. (2004) define five parameters to use in an objective function: execution price, execution duration, reputation, reliability, and availability. Integer programming is then used to determine the highest quality service. Similar to Zeng et al. (2004), Wang et al. (2006) use a normalization algorithm to discover the best service composition. However, rather than consider only numeric qualities, non-numeric qualities such as exception handling and accuracy are also considered.

Kokash and D'Andrea (2007) developed an approach for evaluating quality of redundant service compositions through analysis of risk. Rather than looking at performance metrics of various services, the authors take a considerably more pessimistic view of web services by evaluating the likelihood that each service will fail, and then attempt to minimize that risk. They express total risk r as the probability p of a threat multiplied by the magnitude q of its impact (such as monetary loss or breach of reputation). Thus, the total risk of service e can be represented by $r(e) = p(e)q(e)$. Using both qualitative and quantitative data, the authors then compute the total risk of each service and use the data to maximize a target function.

Proposed Web-Service Selection Method

Previous research in evaluating web services based on QoS has relied heavily on results received in test environments and on a model or function with various weights assigned to different metrics. As indicated earlier, these models suffer especially from the subjectivity of the coefficients given to each metric. Also, the results of tests performed in test environments may not be completely reliable. Zhu et al. (2006) point out that test results matter little if they do not apply in the real world. Given that there are potentially many different usage scenarios of web services, it is reasonable to conclude that some of those scenarios may not be included in the test environment, due to the fact that the scenario may not be predictable. Furthermore, performance of a web service can vary significantly based on the size of the XML payload. As a result, producing useful and accurate tests is not a simple task.

Furthermore, different performance aspects of a web service may matter more to some individuals than to others. Wu and Chang (2007) point out that even though a web service may be functionally appropriate for a particular situation, slow speed may make the service prohibitive. Likewise, a web service that may not be as highly ranked, as other services in some areas may be more suited to a given use case because the service may satisfy different QoS requirements. It is difficult for the selection algorithms discussed previously to account for such requirements.

Thus, this paper proposes a method for ranking web services according to QoS attributes that are mined from free-text user reviews. Using user reviews to evaluate the quality of a web service ensures that service rankings are based on real-world data rather than data produced in a test environment, thus answering some of Zhu et al.'s (2006) concerns. Additionally, this method relies on the opinions of many different users to determine a web service's rating for a particular QoS metric rather than the opinions of a few researchers, which statistically should result in a less biased rating. Finally, the use of user feedback also ensures that a variety of scenarios will be tested, as opposed to tests performed in a controlled environment, which may fail to cover a range of use cases.

One of the reasons that user-provided free-text reviews are attractive when evaluating web services is that a user review is more credible in the eyes of a consumer than a description written by the owner of the web service (Li et al. 2009). As with the creators of any product, the creators of a web service are biased towards their own creation and may downplay the negative aspects of their service while focusing on the positive aspects. Consumers of the service, however, are typically in a better position to evaluate the strengths and weaknesses of a service because they are experiencing that service in a real-world environment.

However, provider-created information is easier to evaluate than user-created information because it is generally presented in a standard form. Also, an emotional review by a user can make a product look better or worse than it actually is. User reviews may sometimes be incomplete, dishonest, and occasionally malicious. Fortunately, a field of research known as opinion mining has made great progress in extracting useful data from user-supplied reviews.

As the number of e-commerce websites grows, so too does the number of customer reviews and the need to automatically extract useful information from those reviews (Hu and Liu 2004). Opinion mining attempts to satisfy this need by extracting opinion-bearing words (such as *great*, *amazing*, or *poor*) from a customer review and mapping those words to a reviewable aspect of the product. Each opinion-bearing word also has a semantic orientation, meaning a positive or negative effect. Thus, given a collection of documents containing opinions on an object, the three main steps of opinion mining as identified by Hu and Liu (2004, p. 169) are:

- Extracting object features that have been commented on in each document;
- Determining the semantic orientation (positive, negative, or neutral) of each opinion; and
- Grouping synonyms of features together (to account for the fact that users may use different words to refer to the same feature).

Jakob et al. (2009) have created a movie recommendation system, which they call HYRES that exploits free-text user reviews of movies and utilizes opinion mining techniques to recommend movies based on an individual's interest in past movies. The researchers used the online Internet Movie Database (IMDB), which requires users to post both star ratings and free-text reviews, as the basis for their study. Two pieces of information from the movie reviews were emphasized: the influence on the star rating that a given movie aspect (characters, storyline, etc.) has for a user, and how important a particular movie aspect is to a user (based on the number of opinions by a user regarding each aspect). Given these two pieces of information, a movie can be recommended to a user even if it has a mediocre overall rating as long as it scores well in the areas that most interest the user. Similarly, a movie need not be penalized if it rates poorly in those areas of low interest to the user (Jakob et al. 2009).

To begin the process of mining the reviews, clusters must first be created that represent common "reviewable" aspects of a movie (Jakob et al. 2009). Jakob et al. (2009) tested three different methods to create these clusters: manual clustering, semi-automatic clustering, and fully automatic clustering. Manual clustering, as the name implies, requires manual identification of both the clusters and the group of terms that will relate to each cluster. Similar to manual clustering, semi-automatic clustering requires the manual identification of the clusters themselves, however the terms relating to those clusters are extracted automatically from the review corpus (the entire body of user reviews that will be analyzed less any opinion-bearing words). The semantic relatedness of each word in the corpus is calculated for each cluster. For this method, the authors used Explicit Semantic Analysis to calculate relatedness, and the top twenty terms relating to each cluster were kept. Finally, fully automatic clustering automates the entire process, including the cluster selection. For this final method the researchers used Latent Dirichlet Allocation, a statistical model used to extract topics from text documents (Blei et al. 2003), to extract the clusters from the corpus. In addition, the researchers used the MALLET package—a "language processing, document classification, clustering, topic modeling, information extraction" tool (McCallum 2002)—to assign each word in the corpus to a cluster.

Once the clusters were created and the relevant terms in the corpus were each mapped to a cluster, the complex step of extracting opinions from the reviews began. This step involved linking the opinion-bearing words included in the corpus with the terms mapped to each cluster. By using the semantic orientation of the opinion-bearing word, a "vote" either for or against that cluster could be derived (Jakob et al. 2009). For example, if the phrase "the music was beautiful" appeared in a review, the opinion-bearing word ("beautiful") combined with the aspect lemma ("music") allowed them to determine that this user felt positively about the soundtrack of the film, because the semantic orientation of "beautiful" is positive and "music" is associated with the soundtrack cluster.

Linking opinion-bearing words to terms can be a difficult task, however, because of the complexity of the English language. The long, complex sentences containing many nested clauses that IMDB reviewers typically wrote made simple pattern-matching algorithms useless for linking opinions with terms (Jakob et al. 2009). Instead, the authors relied on the Stanford Parser (a free online language parser that breaks sentences down into grammatical parts) along with two generic dependency relation patterns (the fact that adjectives are the major way to express opinions and the fact that an intermediate word often separates an opinion-bearing word and the cluster term) to derive the opinions from each review.

The algorithm used by Jakob et al. (2009) applies particularly well to the problem of selecting web services based on QoS metrics, because the clusters are already clearly defined in the form of the seven metrics presented earlier. Therefore, assuming a corpus exists that contains reviews of various web services, a semantic analysis could be performed, using semi-automatic clustering on the corpus to link terms to clusters. This analysis would then provide a means to select a web service based on its QoS metrics.

Using this opinion mining algorithm to rank web services would answer some of the concerns of Wu & Chang (2007) that were discussed earlier. Because different clusters exist that will essentially receive separate ratings, those clusters which are not as important in a particular scenario can easily have their impact on the final rating of the service reduced, while the results can also be modified to place a stronger emphasis on other more important clusters. As with unimportant aspects of movie recommendations, non-applicable QoS metrics could be ignored. Furthermore, the concerns raised by Zhu et al. (2006) are also addressed by this algorithm because web services will be ranked by users with real-world experience using the services and not based on test data or provider-created data.

Research Method and Results

To test the efficacy of the proposed web service selection method outlined above, we created and administered a web-based survey, where participants were asked to test several web services, and then provide a review of each web service. Each participant evaluated a total of four web services that each focused on common, low-level utility functions (dictionary definition lookup, interest calculator, sorting, and shopping cart/shipping calculator). Given the low-level characteristics of the web services that we created and the fact that users could only evaluate each web service within the context of the interface that we created, we eliminated the regulatory and security QoS metrics from consideration.

One challenge associated with user provided feedback is that users should have a vested interest in the object they are reviewing (Kirkland and Manoogian 2004). To increase each participant's vested interest, we required participants to test each service with different values at least five times. In addition, participants provided subjective, free-text reviews for two services, and they provided objective (Likert scale responses) evaluations for the remaining two web services. The survey randomized the order in which objective and subjective reviews were given. In addition to providing an objective or subjective evaluation, all participants were asked to provide an overall star rating of each web service. Finally, half of the participants were provided a table similar to Table 1 while completing their free-text reviews, and they were specifically asked to consider the attributes in this table when evaluating the service. The remaining participants were simply asked to provide a review of the service.

A total of 33 individuals (14 were upper-level computer science (CS) majors and 19 were full-time technical staff) from a small, mid-western university were invited to participate in the study, of which 25 provided responses (10 CS majors and 15 technical staff) for an overall response rate of 75.8% (71.4% of CS majors and 78.9% of technical staff). This resulted in a total of 50 objective evaluations and 47 free-text evaluations (not all participants fully completed the survey). We received at least 10 reviews for all services (dictionary [10], interest [11], sorting [12], and checkout [14]), which is consistent with the minimums suggested by Jakob et al. (2009).

To verify that the participants evaluated the web services themselves and not simply the provided interface, we manipulated three of the four web services to tease out specific QoS metrics, and we left the fourth web service as a control. We included code in the dictionary service to timeout on every third connection. Therefore we would expect users to rate the dictionary service relatively low in terms of availability. We modified the sorting service so that a few items were swapped out of place in the result. Therefore we would expect participants to rate the sorting service relatively low in terms of integrity. We also introduced a five to six second delay in the interest calculation service, so we would expect participants to rate the interest calculation service relatively low in terms of performance. To evaluate these expectations, we created dummy variables to represent each service. We then used logistic regression to determine the effect of the objective QoS attributes on each dummy variable. As expected, the integrity QoS metric was the only significant predictor of the sorting service (Wald = 5.938, $p = 0.15$), the performance QoS attribute was the only significant predictor of the interest calculation service (Wald = 7.717, $p = 0.005$), and the logistic regression model for the checkout service was not significant ($\chi^2 = 6.234$, $p = 0.284$). Somewhat unexpectedly, both the availability (Wald = 3.922, $p = 0.048$) and the integrity (Wald = 4.233, $p = 0.04$) QoS attribute were significant predictors of the dictionary service. Upon later review of the free-text reviews, we discovered that our dictionary service was not able to provide definitions for some terms, which may have caused the integrity metric to stand out.

Given the relatively small sample size, we chose to manually code each free-text review into a tally of votes for or against each of the QoS attributes. Since human judgment is often cited as the gold standard, we believed that this would provide a best-case baseline by which to evaluate our proposed approach. The two authors independently coded all of the free-text reviews. Initial inter-rater reliability scores were quite low, but after discussion, we discovered that we handled differently the generic free-text comments, such as "the service worked well". After synchronizing our approach to these generic comments, inter-rater reliability scores were within acceptable limits (availability = 0.69, accessibility = 0.59, integrity = 0.63, performance = 0.77, reliability = 0.71).

We then repeated the logistic regression calculations described above using the average of the judge's scores for each QoS attribute as the independent variables. For the dictionary service, we expected the availability metric to be the only significant predictor, but there were no significant predictors at the 0.05 level; however, the accessibility metric was nearly significant at $p < 0.10$. For the interest calculation service, the performance attribute was the only significant predictor (Wald = 6.952, $p = 0.008$), and for the checkout service, no QoS attributes were significant, both as we expected. For the sort service, the logistic regression algorithm would not converge on a solution with all

independent variables in the model. Lack of convergence may have been a result of the limited sample size or complete separation (Allison 2008). In this case, we believe there was complete separation between the integrity attribute and the dependent variable, since a reduced model that excluded the integrity attribute reached convergence (Allison 2008). In the reduced model, both the performance (Wald = 3.882, $p = 0.05$) and reliability (Wald = 5.120, $p = 0.024$) attributes were significant; however, as indicated by Allison (2008), the estimates produced by the reduced model are biased. In the case of complete separation, Allison (2008) further recommends that the bivariate correlations between the excluded independent variable and the dependent variable be evaluated. The correlation coefficient between the integrity attribute and the sort service dummy variable was significant (-.634, $p < 0.001$), suggesting that a relationship between the integrity attribute and the sort service dummy variable does indeed exist, as we expected.

Discussion

The results presented above suggest that even though our analysis was based on the best-case scenario of human judgment, it is possible to extract objective QoS metric data from free-text user reviews of web services. This evidence also seems to suggest that certain QoS metrics are more easily captured from free-text data than others. For example, the results for the dictionary service seem to indicate some difficulty in distinguishing accessibility from availability, while the results of the interest and sort services suggest that performance and integrity were easier to identify. When regressing the QoS metrics on the star rating, the only significant predictor was the integrity metric ($t = 4.113$, $p < 0.001$). Integrity's significance suggests that it was the most prominent attribute on the minds of the participants when evaluating overall quality of service, which may have made it easier to identify during coding. The performance characteristic may have been easier to identify, simply because performance manifests itself in more easily identifiable terms. For example, two participants indicated that the interest calculation service was slower than the other services.

Anecdotally, when participants were not able to refer to the list of QoS metrics, they typically provided more comments related to the user interface of the web survey. For example, some participants complained about the usability of the checkout service, since it was not as full featured, as they would have liked. During the coding process, we simply ignored all user interface related comments, and so, not surprisingly, including the ShowQoS dummy variable in the logistic regression models, yielded no significance for that attribute in any of the models.

Limitations and Future Research Directions

As with all research studies, this study suffers from some limitations, but we believe that these limitations also provide insight for future research directions. The first limitation of our study was that participants were required to evaluate each web service in the context of our provided interface, instead of attempting to integrate each web service into their own solutions. As a result, our study sacrificed generalizability for efficiency. A more realistic, integration environment, where participants could download each service contract and provide their own test harness, would certainly help to focus each review on the web service itself. In addition, a realistic test environment would not only enable evaluation of the regulatory and security attributes that we were unable to test, but it would also likely increase each participant's vested interest, which would, in turn enable the participant to provide more specific reviews.

Second, our study only evaluated utility layer services (Erl 2005). Higher layer services may yield different priority structures for QoS metrics in the minds of the participants. For example, performance may become more prominent in orchestration layer services that coordinate many lower layer services. The main challenge with including higher layer services is limited scalability without a structured web service repository. Future research should investigate a proper theoretical design for a web service repository that could provide a means for storing, retrieving, testing, and reviewing web services, using a design science approach (Hevner et al. 2004). The services that are stored in this repository would also need to be functionally redundant to ensure that non-functional attributes of the services would be the focus of the user reviews. A reasonable number of individuals would then need to provide their reviews of those web services after consuming them in real-world environments.

Third, our study relied on human coding of the free-text reviews to establish a best-case scenario. Future research should attempt to implement the Explicit Semantic Analysis (ESA) approach used by (Gabrilovich and Markovitch 2007) to classify free-text reviews in terms of the QoS metrics. ESA could also be used in the web service repository to locate compatible web services, based on textual search criteria (i.e. requirements specification documents).

One concern related to our proposed method of classifying free-text reviews relates to the integrity of the user reviews. For example, it might be possible for the owners of a particular service to provide fake or incorrect reviews that would inappropriately increase the ranking of the service. Indeed, we experienced this during our study, when we had to filter out some comments in user reviews that reflected criticisms of our test environment, rather than the web service itself. Additionally, owners of competing services may provide negative reviews in order to decrease the service's rating. Also, users often quote reviews posted by other users and comment on the accuracy of the quoted review rather than posting an original review of the product. Such instances would have to be taken into consideration with the proposed method in this paper. Finally, the opinion mining method used to evaluate web services would also have to consider the age of the review. The creator of a web service may fix certain issues with the service after users have already posted negative comments about that issue. To prevent those negative reviews from forever reducing the service's score, older reviews would have to be weighted less than more recent reviews.

A repository of web services likely would not be subject to the same degree of "spam" reviews that many websites experience. Since technical professionals (the direct consumers of web services) would be providing most, if not all, of the service reviews, more than likely those reviews would be of higher quality than user reviews found on e-commerce sites where a much larger audience posts their opinions. Li et al. (2009) have proposed a method for measuring the quality of a review by evaluating the entire body of a user's review in terms of metrics such as enrichment, temporality, and credibility (Li et al. 2009). The proposed opinion mining algorithm could incorporate these scores to determine a quality measurement of each review that could then be used to assign a higher or lower weight to the review.

Conclusion

This paper has presented a unique method to accomplish the challenging goal of selecting an appropriate web service for a given situation. The paper discussed the seven primary QoS metrics that are used to evaluate the quality of web services and discussed some of the causes for web service inefficiencies. In addition, the paper discussed some of the current methods that have been proposed to rank web services, including linear programming approaches and a risk avoidance approach, as well as some of the flaws of those approaches.

The paper also discussed the topic of opinion mining and suggested that a method that was originally designed to recommend movies, be applied to the selection of web services. Using this method, web services can be selected based on QoS metrics, and metrics that may not be important to a particular scenario can easily be ignored. This opinion mining method minimizes many of the problems associated with other selection methods, including basing selections on real-world experiences with web services rather than on test data. We conducted an initial study to determine the efficacy of our opinion mining approach. While we received mixed results, there is still evidence that this approach is worth exploring in the future.

References

Allison, P.D. "Convergence Failures in Logistic Regression," in: *SAS Global Forum 2008*, 2008, pp. 1-11.

Blei, D.M., Ng, A.Y., and Jordan, M.I. "Latent Dirichlet Allocation," *Journal of Machine Learning Research* (3:1) 2003, pp 993-1022.

Chaari, S., Badr, Y., and Biennier, F. "Enhancing Web Service Selection by QoS-Based Ontology and WS-Policy," in: *Symposium on Applied Computing*, ACM, Fortaleza, Ceara, Brazil, 2008, pp. 2426-2431.

Erl, T. *Service-Oriented Architecture: Concepts, Technology, and Design* Prentice Hall, Upper Saddle River, NJ, 2005.

Gabrilovich, E., and Markovitch, S. "Computing Semantic Relatedness using Wikipedia-based Explicit Semantic Analysis," Proceedings of IJCAI, Hyderabad, India, 2007, pp. 1606-1611.

Gray, J., and Reuter, A. *Transaction Processing: Concepts and Techniques* Morgan Kaufmann Publishers, Inc., San Francisco, 1993.

Hevner, A.R., March, S.T., Park, J., and Ram, S. "Design Science in Information Systems Research," *MIS Quarterly* (28:1) 2004, pp 75-105.

Hu, M., and Liu, B. "Mining and Summarizing Customer Reviews," in: *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, ACM, Seattle, WA, 2004, pp. 168-177.

Jakob, N., Weber, S.H., Muller, M.C., and Gurevych, I. "Beyond the Stars: Exploiting Free-text User Reviews to Improve the Accuracy of Movie Recommendations," in: *Proceedings of the Conference on Information and Knowledge Management*, Hong Kong, China, 2009, pp. 57-64.

Jun, W., Lei, H., and Chunlei, N. "Speed-up SOAP Processing by Data Mapping Template," in: *Proceedings of the International Conference on Software Engineering*, Shanghai, China, 2006, pp. 40-46.

Kirkland, K., and Manoogian, S. *Ongoing Feedback: How to Get It, How to Use It* Center for Creative Leadership, Greensboro, NC, 2004.

Kohlhoff, C., and Steele, R. "Evaluating SOAP for High Performance Business Applications: Real-Time Trading Systems," in: *Proceedings of the Twelfth International World Wide Web Conference*, Budapest, Hungary, 2003, pp. 1-9.

Kokash, N., and D'Andrea, V. "Evaluating Quality of Web Services: A Risk-Driven Approach," in: *Business Information Systems*, Springer Berlin, 2007, pp. 180-194.

Li, H.-H., Du, X.-Y., and Tian, X. "A Review-Based Reputation Evaluation Approach for Web Services," *Journal of Computer Science and Technology* (24:5) 2009, pp 893-900.

Liefke, H., and Suciu, D. "XMill: An Efficient Compressor for XML Data," *ACM SIGMOD Record* (29:2) 2000, pp 153-164.

Mani, A., and Nagarajan, A. "Understanding Quality of Service for Web Services: Improving the Performance of your Web Services," IBM DeveloperWorks, 2002.

McCallum, A.K. "MALLET: A Machine Learning for Language Toolkit," http://mallet.cs.umass.edu, 2002.

Ng, W., Lam, W.-Y., and Cheng, J. "Comparative Analysis of XML Compression Technologies," *World Wide Web* (9:1) 2006, pp 5-33.

Papazoglou, M.P., and Heuvel, W.-J. "Service Oriented Architectures: Approaches, Technologies and Research Issues," *International Journal on Very Large Data Bases* (16:3) 2007, pp 389-415.

Tolani, P.M., and Haritsa, J.R. "XGRIND: A Query-Friendly XML Compressor," in: *Proceedings of the International Conference on Data Engineering*, IEEE, San Jose, CA, 2002, pp. 225-234.

Vitharana, P., Zahedi, F.M., and Jain, H. "Knowledge-Based Repository Scheme for Storing and Retrieving Business Components: A Theoretical Design and an Empirical Analysis," *IEEE Transactions on Software Engineering* (29:7) 2003, pp 649-667.

Wang, X., Vitvar, T., Kerrigan, M., and Toma, I. "A QoS-Aware Selection Model for Semantic Web Services," in: *Proceedings of the 4th International Conference on Service-Oriented Computing*, Springer Berlin, Chicago, IL, 2006, pp. 390-401.

Wu, C., and Chang, E. "Intelligent Web Services Selection Based on AHP and Wiki," in: *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence*, IEEE Computer Society, Silicon Valley, CA, 2007, pp. 767-770.

Zeng, L., Benatallah, B., Ngu, A.H.H., Dumas, M., Kalagnanam, J., and Chang, H. "QoS-Aware Middleware for Web Services Composition," *IEEE Transactions on Software Engineering* (30:5) 2004, pp 311-327.

Zhu, L., Gorton, I., Liu, Y., and Bui, N.B. "Model Driven Benchmark Generation for Web Services," in: *Proceedings of the 2006 International Workshop on Service-Oriented Software Engineering*, ACM, Shanghai, China, 2006, pp. 33-39.