Integrating Semantic Concept Similarity in Model-Based Web Applications

Cristiano Rocha\textsuperscript{1,2}, Daniel Schwabe\textsuperscript{1}, and Marcus Poggi\textsuperscript{1}

\textsuperscript{1} Department of Informatics, Catholic University, Rio de Janeiro
{dschwabe, poggi}@inf.puc-rio.br
\textsuperscript{2} Milestone
crocha@milestone-ti.com.br

Abstract

Model-based design methods, and model-based architectures, have gained adoption in authoring applications for the WWW. This is further reinforced by the increasing visibility of the Semantic Web, where models are intrinsic, described as metadata associated with the data made available to users and applications. Several proposals attempt to leverage this additional information to improve search functionalities, by incorporating semantic similarity (or proximity) measures into the search mechanism. In this paper we show how the availability of a semantic similarity evaluation engine can be used to enhance several functionalities of Web-based applications. In particular we will show how such an infra-structure can be used to detect and suggest new relation instances, as well as propose an inferred ordering for the presentation of related information that reflects the semantic closeness of the corresponding information. The proposed engine is based on a hybrid spread activation algorithm applied to the concept instances graph.

1. Introduction

In this paper we propose a semantic similarity framework that makes it possible to provide a numeric strength or similarity between two nodes, corresponding to two concept instances, in an application. We envision this type of measure to be very useful in various applications.

In web applications where the input of information happens in a distributed way one of the biggest problems is to guarantee that this process happens in the most natural and user friendly way, and also to guarantee that all the information is consistent. Here we show how the use of a similarity measure can help this process by suggesting new links which the system detects as having a great likelihood of existing. The absence of this link might be due to an error or inconsistency in the insertion of the information. If this is the case, the relations suggestion functionality will help the user in identifying this inconsistency and correcting it. If it is not the case, it might help overcome a lack of knowledge about the problem domain on the part of the user who is responsible for entering new information into the system.

A second scenario occurs when the user entering new information only partially knows the problem domain, and is therefore unable to explicitly link the new information with other existing information in the database. The system is then able to suggest likely links, which are easier for the user to recognize as being valid. This scenario occurs frequently in large organizations, where people responsible for entering new information only know their own areas and has little or no knowledge about other areas in the organization.

Another problem we are going to address is the presentation order of links in an application. By presenting links to related information in an order reflecting the semantic closeness of the corresponding information, it is expected that it will be easier for the user to find relevant information.

Both functionalities that will be discussed in the paper could be implemented using any similarity measure which can somehow calculate the similarity or proximity of two concepts or pages in a web application. However, with the advent of the Semantic Web and model based design methods, we envision that similarity measures which explore the semantics of the underlying model and the instances of an application can provide powerful measures that will outperform existing techniques in a number of contexts. We propose two different algorithms for calculating similarities measures which rely on the processing of semantic information related to the instances of the application: weight mapping and spread activation algorithms.

The similarity processing framework proposed here can also be successfully used for searching in model-based applications. In \cite{7} we presented a novel approach for combining traditional Information Retrieval techniques with spread activation and weight mapping techniques, in order to provide proximity semantic searches that combine textual information with semantic information. Here, we use the same processing framework for providing the desired functionalities; more information and details on the algorithms can be found in that paper.
2. Similarity Processing Framework

2.1. Weight Mapping

This technique tries to explore the fact that ontologies and their instances carry much more information than what is explicitly stated, as there is much “hidden” information entailed by the relations (i.e., a semantically-based linking structure). In traditional ontologies, it is only possible to indicate the presence or absence of a relation between two concept instances. In many situations, however, it would be desirable to also express some strength associated with the relation. The classical way is to associate a numerical value to the corresponding link. One of the ideas in this work is to extract knowledge from the ontology and its instances in order to obtain a numerical weight for each existing relation instance in the model. A similar idea was presented in [8], to provide a novel approach for ranking the results of ontology-based searching in the Semantic Web, with good results. We call “Weight Mapping” the technique of calculating a numerical weight value for each relation instance, based on the analysis of the link structure of the knowledge base.

Different ideas were tested in devising a calculation that can generate a strength formula for each existing relation instance in the knowledge base. In [7] we proposed three different measures - cluster, specificity and combined- which we found very useful in developing our system. We are aware that the choice of these measures is totally application and task dependent. Here we will just briefly present the three proposed measures. For deeper information on the motivations behind them and explanations of the formulas the user should refer to [7].

The first measure tries to establish the degree of similarity between two related concept instances in a relation. The similarity measure used is very similar to the cluster function used in [2], obtained by specializing that function for concepts that relate to each other. Formula 1 indicates the similarity between concept instances $C_i$ and $C_k$. The value $n_{ijk}$ represents that concept $C_j$ is related to concept $C_i$ (it is 1 if the concepts are related and 0 otherwise). The value $n_{ik}$ represents the fact that both concepts $C_i$ and $C_k$ are related to concept $C_j$ (1 if both concepts $C_j$ and $C_k$ are related to $C_i$ and 0 otherwise).

The second measure is similar to the $idf$ (inverse domain frequency) measure [9] widely used in Information Retrieval (although in I.R. the log function is normally used). It is useful when the user wants to give the semantic of specificity or differentiation to the relation. Formula (2) was used for the specificity measure. The value $n_i$ is equal to the number of instances of the given relation type that have $k$ as its destination node.

The third measure is the combined measure, obtained as the product of the two previous ones. Its calculation resembles the $tf-idf$ strategy [9] that is commonly used in classic Information Retrieval since it combines a similarity with a specificity measure. In the general case this combined measure proved to be the best one in our applications. Other similarity and specificity measures might be used in the future to achieve better results.

\[
W(C_j, C_k) = \frac{\sum_{i=1}^{n} n_{ijk}}{\sum_{i=1}^{n} n_{ij}} \quad (1)
\]

\[
W(C_j, C_k) = \frac{1}{\sqrt{n_i}} \quad (2)
\]

2.2 Hybrid Spread Activation

The other strategy we use to calculate this similarity measure employs spread activation techniques. Such techniques are among the most used processing frameworks for semantic networks, having been successfully applied in several fields, particularly in Information Retrieval applications [3,4]. Given an initial set of activated concepts and some restrictions, activation flows through the network reaching other concepts which are closely related to the initial concepts. It is very powerful to perform proximity searches, where given an initial set of concepts the algorithm returns other concepts which are strongly connected to them. Inferences occur naturally in this process, since the result set may contain nodes that are not directly linked to the initial set of nodes. An overview of spread activation techniques is presented in [4].

Usually spread activation techniques are used either on semantic networks, where each edge in the network has only a label associated to it, or in association networks, where each edge has only a numeric weight associated to it. In [7] we showed how to use the weight mapping techniques to construct a hybrid instances network, where each relation instance has both a semantic label and a numerical weight, and use spread activation on this network. The intuition behind this idea is that better results in the search process can be achieved using the semantic information together with sub-symbolic (numerically encoded) information extracted from the instances. Several works in the literature present spread activation algorithms either in semantic [3] or in associative nets [2]. However, there are few works that use both approaches together.

The algorithm has as a starting point an initial set of instances in the ontology, henceforth called nodes, which have an initial activation value; in the functionalities proposed in this paper this value will be 1.0. All nodes not in the initial set have their initial activations set to zero. The initial nodes are put in a priority queue, ordered non-increasingly with respect to the node’s activation value.
The node with the highest activation value is then taken out of the queue and processed. If it satisfies all the restrictions, it propagates its activation to its neighbors. The neighboring nodes which were activated, and are not currently in the priority queue, are added to it. The priority queue is then reordered. The node that was just processed is added to the results list, which contains all the nodes that have been processed and are the result of the spread activation process.

This process repeats itself until a specified state is achieved (a defined output size for example), or there are no further nodes to be processed in the priority queue. At the end of the propagation a final set of nodes and their respective activation values are obtained. The total complexity of the spread activation algorithm proposed is $O(|E| \times \log |V|)$, where $E$ is the number of existing relation instances and $V$ is the number of concept instances. For a more detailed explanation about the propagation the user should refer to [7].

### 3. Ordering Related Elements

In hypermedia applications, a node is typically related to various other nodes. Most of the times, these relations are symbolic - they hold or they don’t. There is no idea of strength or intensity of a relation.

A major advantage of the hypermedia paradigm is precisely the ability to show related information to a given node, typically through a list of links. Sometimes this list is ordered according to the value of some attribute of the related nodes, such as alphabetically ordered on the destination node’s name. In many applications where a relevance ordering is used, it is determined manually, and explicitly specified to the system. This approach is very costly and almost intractable if done for all nodes in the database, and all their corresponding relations.

The idea of ordering the relations of a current node is a classic technique in Adaptive Hypermedia area, attempting to help users find their paths by adapting link presentation to the goals, knowledge and other characteristics of an individual user. One of the techniques for adaptive link presentation is the adaptive ordering which was successfully used in various applications [1]. Typically, this ordering is based on user profile, goals and previous navigation history.

Here, the proposed functionality calculates the strength of the relations based on the analysis of the graph containing all instances of nodes in the knowledge base. This makes it possible to automatically generate an ordering according to the strength of the relation between two nodes. The idea is to use the weights generated by the weight mapping techniques to propose an automatic ordering of the related elements.

### 3.2 Tests and Results

For example, consider the graph shown in Figure 1. Two possible orderings for the Area nodes which are related to the Professor node “Schwabe” are ascending alphabetical order (“Hypermedia”, “Software Engineering”, “Web Services”), and decreasing strength order, i.e., clustering measure, (“Web Services,” “Software Engineering”, “Hypermedia”). The second ordering gives more information to the user of the application, as it allows concluding that Professor “Schwabe” has a stronger relation to the “Web Services” Area compared to the other areas (in this example, he published more articles in the “Web Services” Area than in the other areas).

![Example of an instances graph in the research domain with all its nodes and relations](image)

The application used to test the results of both proposed functionalities was the PUC-Rio Informatics Department website (http://www.inf.puc-rio.br). In this web site it is possible to obtain information about the main research areas, professors, projects, students, labs and publications in the department. The knowledge base has around 2,630 node instances together with 6,554 relation instances. A small part of the research ontology used in the Web Site is shown in Figure 2.

There are several advantages in using this application as a test case, as it is representative of many similar cases found in practice. First, precise analysis of results is straightforward, since the domain experts are the professors and students, which we could consult as needed. Another interesting point is the fact that the website database contains various inconsistencies and errors due to the input process used. Since each professor is responsible for entering his information, some provide a lot of information, whereas others rarely input new information.

An additional problem is the fact that information regarding students and laboratories are typically incorrect or incomplete, since there is no person specifically responsible for entering this information, as students are not allowed to input information directly into the system.

Various tests were carried out to evaluate the ordering of related elements applied to some of the relation types in
each one of the knowledge bases. The goal of the each test was to evaluate if the ordering obtained reflected the strength of the relations, and therefore was valuable to the user of the application.

![Conceptual model for the PUC-Rio Informatics Department website](image)

The results obtained in all test cases were very good, as almost all the lists were positively evaluated by the experts, agreeing that the ordering provided by the system made good sense. The results obtained (Table 1) show that the weight mapping techniques can be very useful to propose an ordering of the related elements of a node instance. They can provide extra information for the user of an application, or to an algorithm that processes that knowledge base, such as the spread activation algorithm.

<table>
<thead>
<tr>
<th>Relation type</th>
<th># Instances Analyzed</th>
<th>% Positively evaluated</th>
</tr>
</thead>
<tbody>
<tr>
<td>Professor-Area</td>
<td>15</td>
<td>100.00%</td>
</tr>
<tr>
<td>Laboratory-Professor</td>
<td>7</td>
<td>85.00%</td>
</tr>
<tr>
<td>Area-Professor</td>
<td>10</td>
<td>100.00%</td>
</tr>
<tr>
<td>Professor-Student</td>
<td>4</td>
<td>100.00%</td>
</tr>
</tbody>
</table>

We also tested this functionality in the context of the Portinari Project website (http://www.portinari.org.br), an application documenting the artwork, life and times of a famous Brazilian painter. The main difference is that for this application we used the specificity measure to calculate the weights of the relations. For lack of space, we do not present a detailed analysis of the results for this application. The tests were conducted similarly as for the PUC-Rio Informatics Department website. Four types of relations were considered, with a total of 80 different orderings analyzed. We obtained a positive evaluation in 96.25% of the tests which show that the functionality was very successful in this application as well.

Not always the best ordering for a relation is obtained analyzing the instances graph. For example, in the case of publications of a professor it is not clear what is the semantics for evaluating that a publication is more important than another. Typically the importance of a publication is given by where it was published - conference proceedings, journal, magazine, etc. This information is usually stored as a property or attribute of the publication. Therefore, in this case, an ordering by the value of a specific attribute could be more interesting than the one provided by the weight mapping technique. Consequently, one should bear in mind that the best ordering for a relation will be determined by the type of tasks the user of the application has to accomplish. It should be a decision of the knowledge engineer of the given application domain which ordering to use, depending on the usage contexts.

4. Detecting and Suggesting Relations

Although one of the main advantages of hypermedia applications is the possibility of connecting different
concepts that have some relation among them, this is only true if the links provided are meaningful. The process of adding relationships among concepts is usually done by human beings in a totally manual way. This is a hard task that consumes a lot of time and requires great knowledge of the specific domain of the application.

For example, consider the PUC-Rio Department of Informatics website. There are approximately 1,600 publications stored in the website. For each stored publication, the website includes a list of relations to its authors, the areas in which the publication is relevant, the projects which are related to the publications, etc., requiring a tremendous amount of work. Many times, as in this case, this task is shared among various users, in order to make it a little easier. For example, each professor is responsible for entering the information regarding his own publications. In many cases, there is decentralized input of information, and various inconsistencies can arise from this process, mainly due to incomplete knowledge on the part of the user entering the information.

In various knowledge bases there exists redundancy. For example, in the research domain a professor who has several publications in a specific area has a great chance of being related to that area. If this information is not explicit in the knowledge base by a direct edge connecting the professor to the given area, this might be an inconsistency or error in the knowledge base.

The idea behind the proposed functionality of detecting and suggesting relations is to identify, for the user or the administrator of the system, possible relations among concepts, which are not explicit in the knowledge base but have a great possibility of existing. That is, the system detects possible new relations which were not previously in the knowledge base. Not every detected relation comes from an inconsistency in the knowledge base. Sometimes, a relation might not exist at an initial moment but, over time, that relation becomes latent due to the modifications that are happening in the knowledge base.

This kind of functionality can also benefit other processes. In particular, it can be very helpful in the process of updating the knowledge base. Most hypermedia applications have their knowledge bases updated constantly (that is, some concepts are inserted in the base, other concepts are deleted; some relations among the existing concepts are added, and others are deleted). This functionality can help the user in the task of inserting new concepts and relations among concepts by using the pre-existing knowledge in the base.

In this scenario, it would be very useful if, as the user starts to input and insert new relations, the system could suggest other probable relations. For instance, the user could begin by inserting the relation with the professor who wrote the publication. After that, the system would automatically suggest as related areas to the publication, from the main research areas of the given professor. After the confirmation of the areas related to the publication by the user, the system could suggest as probable co-authors of the publication, students and professors who typically write publications together in those areas with the given professor, and so on and so forth.

To suggest a new relation, the user provides a starting node, which the system uses as the input node for the spread activation algorithm. To prevent suggestion of links already present in the knowledge base, the spread activation must be configured by adding a restriction rejecting all nodes to which the given node already has a relation. The nodes obtained from the spread activation algorithm are then presented to the user as possible related nodes to the given node; the user has the option of immediately inserting any suggested relation in the knowledge base.

In addition to suggesting relations, it also associates a numeric weight that indicates the strength of that suggestion. The analysis of this weight is difficult, and varies from relation to relation. Naturally, an instance of a relation suggested which has a higher value than another suggested instance of the same relation, has a higher likelihood of being true.

Next, an example will be presented to clarify the use of this functionality. Considering the instances graph shown in Figure 1, it is possible to observe that professor “Schwabe” has a relation with three distinct areas (“Web Services”, “Hypermedia” and “Software Engineering”). If the system was asked to propose new relations of the type Professor-Area, it could suggest the relation with the area of “Information Retrieval”, since professor “Schwabe” has a publication in this area, and also advises a student in it.

The absence of this relation in the knowledge base could be due to an error. In this case, professor “Schwabe” is indeed related to the area of “Information Retrieval”, but this relation was not stored in the knowledge base due to errors in the information input process. It is also possible that, when the database started to be populated, this was really not one of his areas, but after he started publishing papers and advising students in that area, it became true, but this relation was never actually inserted in the knowledge base.

Another possibility is that professor “Schwabe” only has direct relations to his main research areas, and since “Information Retrieval” is not one of those, he has no direct relation to it in the knowledge base. In this case, the lack of this relation in the base is not an error. Even if this is the case, this inference is still very useful in various contexts. If a search for professors in this area is done in the system, it might be interesting to show professor Schwabe as one of the results, since he has at least some experience in the area, even though no explicit relation is actually stored. In any case, it is important to observe that the decision on whether or not to insert a suggested relation in the knowledge base is taken by the user(s).
There are several types of relations in this application, and evaluating the functionality for all of them would be too expensive. Some specific relations were evaluated. Relations involving Laboratories and Students were good candidates since they had much fewer instances than they should have in practice. A balanced analysis of this functionality should also test relations that were thoroughly filled - those relations where most of the actually occurring relations were already explicit in the knowledge base. The intuition we had and wanted to confirm was that for these relations the precision of the proposed functionality would be lower. We divided the relations in 3 distinct groups - strong, medium, and weak - based on the average number of instances the relation had, relative to the expected number of instances, using the semantics of each relation (e.g. a paper must have at least one author).

The algorithm suggests various new relation instances, ordered for each type of relation, and a real number - its weight - associated to each suggestion. To be of practical value, it is necessary to establish a threshold for the weights, to filter out meaningless suggestions. The difficulty is to establish this single threshold value, since it should be different for each type of relation, because the semantics of each relation type is completely different.

The approach used was to use existing relations in the database as a “training set”, to collect the weights assigned to them by the algorithm. The threshold value is obtained as a function of these collected weights, which in our case was \( \min \{ \min \text{weight}, (\text{avg weight} - \text{std. dev.}) \} \). Several other possibilities were considered, and for other domains this function may have to be adjusted.

To analyze the precision of the results obtained it was necessary to manually evaluate each suggestion proposed by the system, using the help of domain experts. They classified the suggestions as correct or incorrect. For instances where there was any doubt, we classified them as incorrect. The precision was calculated by identifying the percentage of correct relation suggestions. Recall could not be calculated since we did not have a list of all correct instances which were missing in the base (indeed, this is the reason why the functionality was very useful).

4.1. Tests and Results

Several tests were made to evaluate the suggestion of relations functionality in the PUC-Rio Department of Informatics website application. The basic methodology of the tests consisted of choosing a set of relations, and for each one of them asking the spread activation system to suggest new instances for them. After that, domain experts evaluated the proposed new relation instances, and decided whether or not they should exist in the knowledge base. Based on the hits and misses we calculated the precision rate of the functionality. The goal of the tests was to analyze if the proposed system suggests new relations with an acceptable precision, where the meaning of acceptable varies from application to application. In the positive case, it could be employed by the users of an application, either for error and inconsistency identification, or for aiding in the insertion of new instances in the knowledge base.

The graph presented in Figure 3 presents the change in the precision as function of the number of suggested relations. The horizontal axis represents the number of suggested relations for a given relation type, and the vertical axis represents the precision value at a given point. The graph was constructed as follows. For each relation type, we use the spread activation algorithm to obtain a list with all the relation suggestions for that type. This list was sorted from the best suggestion to the worst.

Table 2 presents the results obtained. As expected, the precision value diminishes as more relations instances are suggested. Also, the precision was much higher for the relations in the weak and medium groups. This is due to the fact that in these relations there are more missing relations instances, and therefore the level of correct suggestions tends to be higher. Generally speaking, 329 relations instances above the threshold were suggested with an average precision rate of 78.7%. Ignoring the threshold, the system proposed 834 new relation instances with an average precision of 75.9%. Both results are very encouraging, given the fact that the main goal of the functionality is not to automatically generate new relations instances but to identify them for the user, who has the option of accepting or not the suggestion. The correctly suggested relations instances were responsible for an increase of 10% in the number of relations in the base.

We also tested this functionality in the Portinari Project website. This application is interesting because its database is highly consistent, and the domain model is less redundant, in the sense that there are fewer semantically meaningful transitive paths to be explored by the spread activation algorithm. Given these characteristics, as expected, the suggestion of relations was not very effective. We can conclude that the utility of this functionality is proportional to the level of inconsistency of the knowledge base, and to the redundancy of the semantic domain of the application.

We also developed an interface where the inferred links are presented together with the existing links. We used different colors so the user could differentiate them and offered an easy one-click solution for the user to insert the inferred link, turning it into an explicit link in the database, if he wishes, and has the appropriate permissions. Users greatly appreciated this functionality.

The same ideas proposed here can also be used to suggest links for relations that do not exist in the conceptual model of the application, as opposed to
relational instances for existing ones. For example, it is possible to suggest links between professors, even though such relations are not present in the conceptual model.

In the case of the Portinari Project application the benefits of such a strategy become clearer. For example, in this domain, an exhibition is related to the paintings exhibited in it. A painting is related to the techniques used to paint it, and to its themes. An interesting suggestion of links would be to propose, for a given exhibition node, the techniques and themes most closely related to it, even though such relations do not exist in the conceptual model. This functionality works as an inference machine trying to do proximity search for node instances that are close to a particular node instance.

Some qualitative tests were done for this kind of relation suggestion in both applications, and the results obtained seemed to be very good. We intend to further explore this particular use in future works.

### 5. Related Work

An interesting system which uses spread activation techniques is the WebSCSA (Web Search by Constrained Spreading Activation) system proposed by Crestani [5]. This system searches for relevant Web pages by autonomously navigating through the Web using associations between pages. The navigation is processed and controlled by means of a Constrained Spreading Activation model. The first big difference to our work is...
that the spread activation is carried on the Web (not in a particular application) and therefore no domain information is available. Also, in our spread activation the similarity of pages is calculated using semantic information from the domain model while in WebSCSA the textual contents of the web page are considered.

ONTOCOPI [6] presents an approach similar to ours for processing ontology-based information through spread activation techniques for suggesting relations. It is applied for identifying communities of practices (COPs) in an organization. The system tries to suggest persons which are closely related and therefore have common interests. ONTOCOPI attempts to uncover informal COP relations (those which are often indeterminate and expensive to establish and monitor) by spotting patterns in the formal relations represented in ontologies, traversing the ontology from instance to instance via selected relations. The activation in their system is propagated through a semantic network only, and there exists no idea of extracting semantics from the link structure such as the weight mapping techniques proposed in our work. Their work uses the spread activation system and the suggestion of relations in a much narrower scope than the system proposed in this paper. We believe that our system could be successfully used for the same task as ONTOCOPI.

As previously mentioned, link ordering has been used for Adaptive Hypermedia applications; the main difference with respect to the one presented in this paper is that the type of information and the algorithms used in Adaptive systems is based on a model of the individual user, and its context of use. In our case, we use semantic information from the node instances and its relations, which is the same for all users. We envision the use of both technologies together as being an even more powerful method for ordering the presentation of links in hypermedia applications.

6. Conclusions

In this paper, we showed how a similarity processing engine can be used to provide some new functionality in model-based applications. The proposed engine uses semantic information from the model and its instances to explore the instances graph using a hybrid spread activation algorithm. The proposed engine proved to perform well in presenting links to related information in an order that reflects the semantic closeness of the corresponding information. It was also successfully used to suggest new relation instances to the user of an application, helping users in inserting new information in the database and also in identifying possible inconsistencies and errors in it.

As previously mentioned, we plan on integrating the proposed engine with adaptive hypermedia applications so we can use both semantic information from the domain and the user profile to perform adaptations. We also intend to investigate in more detail the applicability and utility of suggesting links for relations that do not exist in the conceptual model of the application and the results provided by it. In addition, we are also working on additional refinements in the proposed engine, experimenting with alternative functions, and other forms of exploiting semantic information.

Acknowledgement. The research presented in this paper was partly funded by scholarships from PUC-Rio and CNPq, and research grants from CNPq and FAPERJ. We also want to thank LES/LAC, TecWeb and Milestone laboratories for providing the necessary infra-structure for developing this work.

7. References