Ontology search with user preferences

René Pázman

Softec Ltd., Bratislava, Slovakia,
rene.pazman@softec.sk,
WWW home page: http://rene.pazman.googlepages.com/

Abstract. Information seekers, especially in the area of job hunting, often know their preferences, at least approximately. Most existing search tools running within job portals allow users to define their preferences by means of explicitly stated search criteria. They often suffer the same problem: If there exists an interesting job for the user which do not fulfill the criteria exactly, they will not find it. The job applicant is then forced to reformulate the search question and start the searching over.
We propose a searching mechanism which uses the search criteria as a description of an ideal offer and matches it with the offers in an offer database. Each found offer in the more or less near neighborhood of the ideal offer is rated with a value based on the distance between the offer and the ideal offer. The searching mechanism is augmented by other user dependendent characteristics of search criteria, namely importance, precision and obligation in order to increase its efficiency.

1 Introduction

In the last decade or two, the spread of information portals offering some sort of offers over the internet has been evident. These portals supply information about offers from various domains and sources. Some of them are strictly specialized for some narrow domain (such as portals for IT jobs offering); some are not (e.g. internet shops). The content of these portals either originates from their providers' own sources, or it is acquired from other sources mainly from other portals and sites on the internet.

These portals allow users to find appropriate offers in various ways. One of the most common techniques is to let user describe his preferences and find the offers fulfilling them. The preferences are mostly seen as strict (crisp) search criteria and the user gets only those offers, which fulfill the criteria exactly. The offers' consumer is forced to repeatedly reformulate the search question and start the searching over. Nonetheless, some of the portals use non-strict (soft) view to the preferences — they use them as a specification of an ideal offer for the user and try to find offers which are closest to it.

The latter way of understanding of the user's preferences can be often seen in many universal search engines, such as Google. It is less often used in narrow domains, where the strict view on criteria prevails.

There are multiple ways of storing the offers on such portals. The most common way is to use a relational database system, such as MySql or Oracle. On the

other hand, a newer way of data representation together with the representation of its semantics has been developed within the Semantic web initiative — ontology. Ontology is usually expressed in the form of document or file that formally defines the relations among terms ([5]). Ontology thus allows expressing domain notions and relations between them explicitly. It supports effective and standard deductive reasoning about the information contained. It is an appropriate way of information representation when such reasoning is needed.

Our work aims at searching offers collected in a narrow domain ontology database in the non-strict (soft) manner. We propose a searching mechanism which uses the search criteria as a description of an ideal offer and matches them with the offers in the offer ontology database.

Together with the searching mechanism, we propose a way of formulating search criteria (using some additional search criteria characteristics) to guide the searching mechanism in order to maximize its efficiency. We keep the amount of information which is desired from the user on a rather low level: we require some supplementary information about the user's preferences that could be potentially learned in an intelligent adaptive system.

2 Search algorithm

Within an ontology expressed in an ontology language such as OWL (see [7]), classes are defined together with their properties. Our search algorithm presupposes such definition for offers too, where offers are the entities (or class instances) which are being searched for. Offers can have miscellaneous properties of various types. They can be literals, such as texts (strings), numbers (integers, float numbers) etc. or they can be instances of the same or other classes.

Such properties can be exploited during the specification of user preferences and afterwards during the offer search. It is straightforward to choose some of the properties as *search criteria* and allow user to input their desired values.

Within a database with thousands of offer instances and tens of thousands of their properties, it is difficult for a user to find offers appropriate for him. The need for effective searching tool is obvious. A good searching mechanism should find not only offers exactly matching the user's preferences, but also offers which satisfy the preferences only partially, in particular when no exact match was found. However, the user should have a possibility to restrict the result set, e.g. in the form of marking some of the criteria as mandatory.

The user should be allowed to specify multiple values for each search criterion — either as alternatives of his preferences or as values required simultaneously. The choice between the two ways of interpreting multiple values of a search criterion depends on the nature of the criterion.

When the user specifies search criteria, a searching mechanism should find offers satisfying the criteria. However, this is not the sole role of the searching mechanism; it should also determine the found offers' measure of satisfaction. This measure (or *rate*) expresses the degree of similarity of found offers and the

preferences specified by the user (the ideal offer). The offers with highest rates should be proposed to the user at first.

The rate of the found offers should be calculated on the basis of required search criteria values. However, there are also some other aspects which should influence the searching. First, not all search criteria have the same relevance level to the user. Second, the difference between required value and the actual offer's value may have distinct weight for distinct search criteria. Finally, some criteria are crucial for the user in such a way, that their satisfaction is necessary (offers that do not satisfy such criteria should be ruled out).

We propose a searching mechanism aiming to satisfy these requirements. The searching mechanism, more specifically rating mechanism, is based on the notion of distance between property values. The first step is to calculate such distance and then convert it to the degree of similarity (rate) of the values. If we know the biggest possible distance, we can normalize the distance into the interval $\langle 0, 1\rangle$ of real numbers:

$$d = \frac{d_{act}}{d_{max}} \tag{1}$$

where d stands for normalized distance, d_{act} stands for actual distance of the two property values and d_{max} stands for maximal possible property value distance.

The greater the distance between the actual offer's property value and the preferred value, the lower the rate (here rate corresponds to the similarity of the values and distance to dissimilarity, see e.g. [4]), so the property value's rate (from the interval $\langle 0, 1\rangle$) could be directly calculated from its distance from the preferred value as:

$$r_{val} = 1 - d \tag{2}$$

Such calculation suffers from one problem: distances for various criteria should not be treated equally, because some criteria should satisfy requirements more precisely than others. In order to allow such distinction we propose a *precision* parameter for each criterion. This value reflects the user's subjective tolerance in difference from the ideal offer (the preferences).

Precision value (expressed as a number from $\langle 0, 1\rangle$), p, is used as a rate modifier in the equation (2):

$$\begin{aligned} r_{val} = {} & (1 - 2\,|0.5 - p|)(1 - d) \\ & + max(1 - 2p, 0)(1 - \tfrac{d}{(1-d)\alpha^2+1}) \\ & + max(2p - 1, 0)\tfrac{1-d}{d\alpha^2+1} \end{aligned} \tag{3}$$

In this equation, α is the impact of the precision to the resulting rate, $\alpha > 0$. The mean precision (0.5) ensures the equality of the equations (2) and (3).

This equation permits to adjust the resulting rate. In the case of a high precision, the rate is low even if the distance is small. On the other side the low level of precision does not penalize small distances (see figure 1).

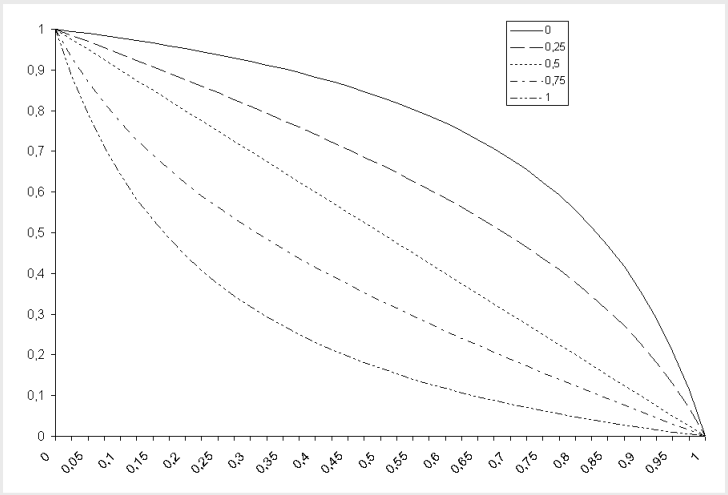

Fig. 1. The conversion function of distances to rates. Each line corresponds to some level of precision. α was set to 2.

We have shown how the distance of two property values, namely the required value and the offer's property value, can be converted to rate. What remains unclear is the distance calculation.

We already mentioned the various types of ontology properties: texts, numbers and instances. The most straightforward distance calculation is definable for numbers. There must be clear for each numeric criterion whether the required value specified by the user means the least or the greatest limit, or some other constraint. In the case of the least value (e.g. for salary property in the domain of job offers) we use the following distance calculation:

$$d_{act} = \begin{cases} 0, & \text{if } v_o \geq v_s \\ 1, & \text{if } v_o \leq v_s\beta \\ \frac{v_o - v_s\beta}{v_s - v_s\beta}, & \text{otherwise} \end{cases} \tag{4}$$

where v_o stands for actual offer value, v_s stands for searched value (the value which the user prefers), and β (from $\langle 0, 1\rangle$) stands for the ratio of minimal acceptable value (all values below $v_s\beta$ are ruled out). Maximal distance (d_{max}) is equal to 1.

In the case of text criteria, we interpret the search value as a list of keywords which ought to be included in the value of the offer's property. The distance for text criteria is then calculated by:

$$d_{act} = 1 - \frac{w_{in}}{w_{all}} \tag{5}$$

where w_{in} stands for the number of keywords included in the property value and w_{all} stands for the number of all keywords. Maximal distance is equal to 1 again.

The case of properties with instances as values is the most difficult. There is almost nothing common in general which could be used as a basis for calculation of the distance between two instances. There are of course cases when some additional properties of the instances can help calculate the distances, e.g. earth coordinates in the case of location property, but in general, we have to solve the case when such additional properties are not at hand.

We can exploit general ontology characterizations only. It is a good practice (and often the real case) to choose an ontology class for each property to determine the type of the property's values. Within OWL ontology language, the property `rdfs:range` is used for exactly this aim. When the property range is defined as a class, all instances of the class and of all its subclasses are possible values of the property. Such hierarchy of classes and instances (taxonomy — see [5]) is one of the most common techniques to describe the problem domain. The properties `rdfs:subClassOf` and `rdf:type` serve for definition of taxonomies in OWL, although other kinds of hierarchies represented by some other property (e.g. `isPartOf` for a hierarchy of locations) can be defined. In the next, we assume that there exists a hierarchy of values for each offer property with ontological instances as values.

Assuming that instances are organized in some hierarchy, we can presuppose that two instances close in the hierarchy have short distance. So the distance could be proportional to the number of edges in the path between the instances:

$$d_{act} = \sum_{e} \delta(e) \tag{6}$$

where e runs over the edges from the path between the instances and $\delta(e)$ stands for the distance representing the distance of one edge, which could be same for all edges (1 for example):

$$\delta(e) = 1 \tag{7}$$

The maximal distance (d_{max}) can be defined as the length of the maximal path in the hierarchy.

It is clear that this definition of distance is overly simplified. The distance of distinct edges can vary and can depend also on the direction. We can expect that distances between resources on the top of the hierarchy are much bigger than distances on the bottom of the hierarchy. For example, regions like Canada and Mexico are much more distant than regions Ottawa and Calgary, which are lower in the region hierarchy.

The direction of the edge in the path between the two instances can also play an important role in the distance calculation. The distance should be considerable distinct if the user specifies location USA and the offer is located in Chicago and if the user prefers Chicago and the offer refers to USA. The first

case is almost the exact match; on the other hand, there is a high probability that the second case does not guarantee the desired location.

In the spirit of the above considerations we propose the adjusted distance calculation for properties with instances as values:

$$\delta(e) = \gamma(e)\ \varepsilon^{depth(e)} \tag{8}$$

$$\gamma(e) = \begin{cases} \gamma^{+}, & \text{if the edge is oriented from child to parent} \\ \gamma^{-}, & \text{if the edge is oriented from parent to child} \end{cases} \tag{9}$$

where $depth(e)$ represents the depth of the edge from the top of the hierarchy and ε denotes the discount of the edge distance from the top to the bottom of the hierarchy and should be slightly less than 1. Orientation of the edge is determined by the transition from the searched value to the actual offer's value. γ^{+} should be distinctly greater than γ^{-}. The situation is illustrated in the figure 2.

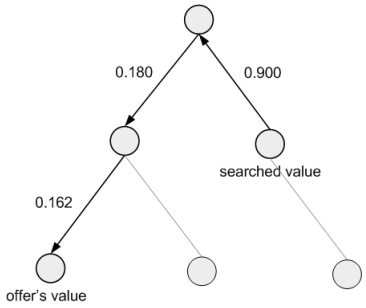

Fig. 2. The distance calculation for two hierarchy values. γ^{+} is 1.0, γ^{-} is 0.2 and ε is 0.9. The whole distance in this case is 1.242.

Up to now, we have defined the rate of an offer's property value relative to a user specified search value. Now we have to determine the whole offer's rate. It is done in two steps. First, we determine the rate for each search criterion individually. Second, we calculate the whole offer's rate based on the criteria' rates.

If the user has specified only one value for a criterion, the rate for the criterion is the same as the rate for the sole criterion's value. If the user has input multiple values, the algorithm has to know whether the criterion allows to input alternative requirements (connected by the logical operator *or*) or simultaneously held requirements (connected by the logical operator *and*), and combine the values' rates distinctly for these cases.

The simplest way to determine criterion's rate is to use the best and the worst rates:

$$r_{crit} = \begin{cases} max(r_{val}), & \text{if the connective is or} \\ min(r_{val}), & \text{if the connective is and} \end{cases} \tag{10}$$

This works, but it ignores values other than maximal value in one case or minimal value in the other. If two alternative values of a criterion are satisfied up to 75% and 74%, this case should be more rewarded as the case of 75% and 3% satisfaction. Thus the actual calculation we use takes into account also the average of the rates:

$$r_{crit} = \begin{cases} \eta \ \frac{\sum r_{val}}{|r_{val}|} + (1-\eta) \ max(r_{val}), \text{ for or} \\ \\ \eta \ \frac{\sum r_{val}}{|r_{val}|} + (1-\eta) \ min(r_{val}), \text{ for and} \end{cases} \tag{11}$$

The parameter η (from $\langle 0, 1 \rangle$) represents the influence of the rates' average and $|r_{val}|$ stands for the count of values specified by the user for this particular criterion.

The last step is to determine the whole offer's rate. It could be the simple average of the criteria' rates, but that does not exactly correspond with the intuition. Some criteria are more important for the user than the others and so each criterion in the algorithm has its *importance* value. Using this value the resulting rate could be calculated as weighted average of criteria' rates:

$$r = \frac{\sum_{crit}(i_{crit} \ r_{crit})}{\sum_{crit}(i_{crit})} \tag{12}$$

where i_{crit} designates the criterion *crit*'s importance.

3 Conclusion

The described searching algorithm was implemented as a tool *CriteriaSearch* for a job offer portal (the works have been accomplished within the project of development of methods and tools for process heterogeneous information — see [6]). The algorithm was implemented independently of domain of job offers; it could be used for offers of any kind.

Our tests confirmed that the search results correspond to search criteria specified by the user and that the mechanism fulfills the intuitions described in the beginning of the section 2. The more of the criteria were satisfied by the offer, the better position it gained. The mandatory criteria caused restriction of the result set (found offers). It was also confirmed that the found offers need not exactly satisfy search criteria.

The criteria parameters of precision and importance also accomplish the intuitions. High precision caused decrease of offers' rates which vary from an ideal offer a little and high importance increases the impact of the satisfaction of the criterion to the overall rate.

The searching mechanism requires knowing some user's subjective characteristics of search criteria other than the searched values — namely precision,

importance and obligation of each criterion. We argue that in an adaptive system, these characteristics could be deduced from the user's actions during his work with the searching tool. However, it requires also some kind of explicit or implicit evaluation of found offers by the user and user interaction time enough for gaining the plausibility of the deduced characteristics.

A similar approach of ontology concept comparison was used in [1], where Andreasen et al. formulated two definitions of similarity of atomic or compound concepts based on paths, which corresponds approximately to our evaluation of rates of two hierarchical property values.

Their first approach is based on the shortest path between two ontology concepts and the authors claimed similarity is not reflexive relation within hierarchies, which exactly corresponds to our viewpoint. What is different, is that our approach uses distance as primary measure and similarity as derived — in opposite to their approach, where the primary measure attached to each edge in the path between the concepts represents similarity. These two approaches behave distinctly in the case of refinement of the concept hierarchy. If the refinement is performed on the shortest path of the concepts being compared, it has large influence on the similarity level, even though the meaning of two concepts remains the same. Changes of the similarity in our approach are not so high in such case.

The other main contribution of our approach lies on distincting between upper and lower distances in hierarchy, which is not included in the first approach in [1].

The other measure of similarity of ontology concepts defined in [1] and developed further in [2] and [3], which is based on shared upwards-reachable nodes approach, distinguish indirectly between similarity in the upper and lower levels of concept hierarchy, and is subject of comparison with our approach.

Our approach also extends the concept of similarity to multi criterial searching mechanism, similarly to what Andreasen et al. have carried out in [3]. In accordance with the searching mechanism in [3], we allow the determination of importance for each search criterion. In addition to that, we introduce user required precision of each search criterion, which influence the impact of distance from the ideal value to offer evaluation. We also define (although simple) similarity measures for text and numeric criteria.

In the near future, we plan to verify the algorithm's properties and test and improve the performance of the algorithm.

Acknowledgements

This work was partially supported by the Slovak State Programme of Research and Development "Establishing of Information Society" under the contract No. 1025/04.

References

1. Andreasen, T., Bulskov, H., Knappe, R.: On Ontology-based Querying. In: Stuckenschmidt, H. (Eds.): 18th International Joint Conference on Artificial Intelligence, Ontologies and Distributed Systems, IJCAI 2003, Acapulco, Mexico, August 9, 2003. pp. 53–59.
2. Andreasen, T., Bulskov, H., Knappe, R.: Similarity from Conceptual Relations. In: Walker, E. (Eds.): 22nd International Conference of the North American Fuzzy Information Processing Sociaty, NAFIPS 2003, Chicago, Illinois USA, July 24–26, 2003. pp. 179–184.
3. Andreasen, T., Knappe, R., Bulskov, H.: Domain-Specific Similarity and Retrieval. 11th International Fuzzy Systems Association World Congress, IFSA 2005, Beijing, China, July 28–31, 2005.
4. Barla, M.: Interception of User's Interests on the Web. In: Wade, V., Ashman, H., Smyth, B. (Eds.): 4th Int. Conf. on Adaptive Hypermedia and Adaptive Web-Based Systems, AH'06, Dublin, Ireland, Springer, LNCS 4018, June 2006. pp. 435–439.
5. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities. Scientific American, May 17, 2001.
6. Návrat, P., Bieliková, M., Rozinajová, V.: Methods and Tools for Acquiring and Presenting Information and Knowledge in the Web. In: Rachev, B., Smrikarov, A. (Eds.): CompSysTech 2005, Varna, Bulgaria, June 2005. pp. IIIB.7.1–IIIB.7.6.
7. Smith, M.K., Welty, C., McGuinness, D.L. (Eds.): OWL Web Ontology Language Guide. W3C Recommendation. February 10, 2004. `http://www.w3.org/TR/2004/REC-owl-guide-20040210/`