Enhancing Product Search Engines Using Fuzzy Logic

CARLOS D. BARRANCO, JESÚS R. CAMPAÑA, JUAN M. MEDINA
Dept. Computer Science and Artificial Intelligence
University of Granada
Periodista Daniel Saucedo Aranda, s/n, 18071, Granada
SPAIN

Abstract: - The paper discuses the main problems which appear when traditional business, based in sales agents, moves to new e-commerce models. Customers, used to describe to sales agents their needs using natural language, are suffering from the lack of expressiveness using product query forms, which are too rigid. Replacing sales agents for automatic systems implies a loss of flexibility interpreting the customer query, narrowing the result set, which could imply a loss of business opportunities. The paper proposes solutions to these problems, using fuzzy logic to model customer requirements, process queries flexibly and ranking results.

Key-Words: - Fuzzy Sets, Fuzzy Databases, Product Search Engines, Real Estate Search, E-Commerce

1 Introduction

The Relational Database Model is widely used in the database world, due to its proved robustness and efficiency. It is not surprising that many proposals for Fuzzy Databases rely on this model to take advantage of its benefits. Thus Fuzzy Relational Database (FRDB) models as well as implementations based on them [1, 2, 3, 4, 5, 6] appear. These models allow a different approach to classic application fields and open new possibilities to find real life problems that can not be satisfactorily solved using classic models.

E-commerce is an area where fuzzy logic and fuzzy databases can provide solutions, offering more natural ways of communication with customers, especially in product search processes, which suffers from the lack of flexibility of classical query forms.

Our proposal is to ease customer requirements expression, allowing them to define their queries easily using flexible conditions. Also, we propose to improve systems flexibility in query processing, emulating sales agent behavior, in order to provide more intelligent query results, which could increase business opportunities.

This proposal will be approached using an example extracted from the real estate trading area. This area is chosen due to the high level of imprecision in real estate attributes, customer queries and query results.

The paper is organized as follows. Section 2 presents the problem of replacing sales agents by automatic systems. Section 3 describes representation and handling of fuzzy data. In Sect. 4, real estate attributes are modeled using fuzzy types. Section 5 introduces flexible conditions. Section 6 describes a flexible query example using a website. Finally, Sect. 7 discloses conclusions and outlines future works.

2 Product Searching Problem

Generally in e-commerce sites, customers have to express what they are looking for. In order to satisfy customer's needs, a set of suitable products are offered by the site. This process can be as simple or complex as the expression of customer needs is.

Our research is focused on real estate searching. This area is particularly appropriate for fuzzy treatment because of the complex expression of customer needs, and the imprecision in some real estate characteristic values. Customer needs are hard to represent using classical approaches, by introducing fuzzy logic into real estate searching processes they can be modeled more accurately.

In every real estate searching process a set of characteristics, describing the desired real estate, is defined. Usually these characteristics are not accurately defined, and it is not necessary that all of them are totally satisfied by the resulting real estate.

The customer determines a set of preferences, representing a general idea of what he is looking for. This idea not necessarily fits to a crisp – a classical – value (i.e. 4 rooms). It could be more properly represented by a more general description, as a value range (i.e. between 2 and 4 rooms), an approximate value (i.e. approximately 4 rooms) or even an upper or lower bound (i.e. up to 5 rooms). This freedom in representing these characteristics may allow to obtain results fitting user preferences on different degrees.

In the classical scenario, the imprecision in the description of customer preferences, and the imprecision evidenced by the different matching degrees of the resulting real estates, is managed by sales agents who can easily cope with fuzzy – imprecise – information. The real estate management process takes place between two humans, the

customer and the sales agent, both of them capable to handle fuzzy information naturally.

Problems expressing customer preferences appear when one of these entities, the sales agent in our case, is replaced by an automatic system, as is happening in all emerging e-business areas. These problems come from the strict interfaces, based in crisp logic, where customers have to express their needs. In order to succeed in this change, it is necessary to provide the systems with methods to handle fuzzy information in the same way the sales agent was doing before.

In this paper, a way to represent fuzzy information about real estates is proposed, in order to be able to design a system that can mimic the sales agent behavior, to interact fluidly with customers. This will reduce the difference between dealing with a human real estate agent and an automatic agent.

3 Imprecise Data Handling

It has been shown that in real life problems a great amount of fuzzy (imprecise) data is found, which can not be expressed using classical models.

Fuzzy set theory solves this problem, introducing mathematical methods for representing and handling vagueness. It extends the classical set theory, where an element can have a membership value of 0 (the element is not a member of the set) or 1 (the element is a member of the set), allowing intermediate degrees of membership (i.e. the element's membership value is 0.5, what means that the element is only a partial member of the set).

Fuzzy data handling, especially in databases, is accomplished by using several data types associated to different fuzzy data representations. We use the representation defined in [6], which describes the following data types:

1. *Precise data*. It uses the data representation provided by the host database, allowing fuzzy querying, using fuzzy conditions. This kind of data is named "*Type 1* data". For instance, a *Type 1* attribute could be the age of a group of people, and asking who is *old*, a fuzzy condition, because *old* is an imprecise concept.
2. *Imprecise data on an ordered underlying domain.* This kind of data is associated with a membership function. In this case we represent membership functions as trapezoidal possibility distributions, which let us represent imprecise data such as intervals, with and without margin, approximate values, boundaries and, of course, precise values. For the sake of simplicity, this kind of data is called "*Type 2* data". An example of a *Type 2* attribute could be the temperature of water measured by a human: cold, cool, warm, hot, boiling.
3. *Data with analogy on a discrete domain*. This type of imprecise data is built on discrete domains on which there are defined proximity relations. This group can represent data as simple scalars or as possibility distributions in discrete domains. We denominate this kind of data "*Type* 3 data". A typical example of a *Type 3* attribute is the hair color of people: blond, red, brunette, dark. There are some similarities between these hair colors, which can be defined using a proximity relation.

Also, the following fuzzy relational operators, which allow to define fuzzy conditions, are defined for these data types:

1. *Fuzzy Equal (FEQ)*: Based on possibility measures when applied on *Type 1* and *Type 2* data, and based on proximity relations when used on *Type 3* data.
2. *Fuzzy Less Than (FLT), Fuzzy Greater Than (FGT), Fuzzy Greater or Equal (FGEQ) and Fuzzy Less or Equal (FLEQ)*: Those relational operators are based on fuzzy extensions of classical relational operators, therefore they are only applicable on *Type 1* and *Type 2* data.

4 Modeling Real Estate Attributes

As explained before, real estate attributes are suitable of fuzzy modeling. From a wide variety of real estate attributes, the most representative ones have has been chosen and modeled using the fuzzy data types described before. Those ones are shown in Table 1.

Type 1 attributes are able to store precise data only. For instance, attribute *Rooms* stores number of rooms: 2, 3, 5, etc.

Type2 attributes are allowed to store imprecise data, in a trapezoidal possibility distribution representation. For instance, attribute *Price* stores real estate's price range: "between €100.000 and €150.000", "approximately €200.000" or "exactly €125.000". The trapezoidal representation of these values is shown in Figure 1.

All *Type 3* attributes have an associated scalar domain, and an associated proximity relation. This is, for instance, for *Kind* attribute: "Apartment", "Flat", "House", "Duplex" and "Attic". The associated proximity relation for attribute *Kind* is shown in Table 2, each value representing the similarity degree between two domain elements.

Geographic Type data type is a composite of a pair of *Type 1* attributes. It is designed to store geographical locations as a coordinate pair *(x,y)*,

which can be queried using fuzzy conditions, as we will see later. It is used to model the attribute *location* which stores the location of real estates.

Table 1: Real estate attributes and associated fuzzy types

Fuzzy Type	*Attributes*
Type 1	Rooms, Floors
Type 2	Price, Area, Age
Type 3	Kind, Orientation, Illumination, Views, Conservation
Geographic Type	Location

Table 2: Proximity relation for attribute Kind domain members

Flat	*House*	*Duplex*	*Attic*	***Kind***
0.75	0.3	0.2	0.75	*Apartment*
	0.3	0.3	0.75	*Flat*
		0.75	0.1	*House*
			0.1	*Duplex*

5 Modeling Customer Requirements

In previous sections, problems expressing customer needs, particularly in websites including real estate searching engines, are discussed. These problems are mainly originated by the lack of flexibility to express a query using crisp logic conditions, commonly offered in query interfaces of e-commerce websites.

In this section, new conditions, based in fuzzy logic, are proposed. These new conditions provide more flexibility to search engines, enabling them to offer customers a new way to define naturally their preferences.

5.1 Flexible Numerical Conditions

Flexible Numerical Conditions are a group of user definable conditions that can be applied on *Type 1* and *Type 2* attributes. They are designed to allow the definition of flexible value ranges in numerical attributes (i.e. price). They are the following:

1. *"Flexible between" condition:* This condition is the flexible variant of the classical "between" condition, adding a margin. This kind of condition must be defined with a lower *(l)* and an upper *(u)* limit and a margin *(m)*, and it is applied using the fuzzy condition *a FEQ [l-m,l,u,u+m]*, where *a*, in next references, is the attribute value on which the condition is applied, and *[α,β,γ,δ]* represents a generic trapezoidal possibility distribution. Using this kind of conditions customers can express conditions like "the size of the real estate must be *between 100m² and 120m², accepting deviations up to 10m²*" setting *l=100*, *u=120* and *m=10*.
2. *"Approximate to" condition:* The condition extends the classical "equal to" condition adding some flexibility. This condition is built setting a central value *(v)*, to which we want to approximate, and a margin *(m)* to define the flexibility of the approximation. Actually, the flexible condition applied, using fuzzy logic, is the condition *a FEQ [v-m,v,v,v+m]*. This condition is useful for customers to define restrictions like "the price must be around €100.000 allowing a maximum oscillation of 10%" setting *v=100.000* and *m=10.000*.
3. *"Flexible greater than" condition:* This condition gives to the classical "greater than" condition more flexibility. The condition is defined by a lower limit *(l)* and a margin *(m)*, and it is applied employing the fuzzy condition *a FGT [l-m,l,l,l]*. A customer restriction like "the real estate must have 6 rooms minimum, however 5 rooms will be accepted if there is no other offers" can be expressed, setting *l=6* and *m=4*, which results in 0.5 as condition fulfillment degree when the number of rooms is *5*, and 0 for lower values.
4. *"Flexible lower than" condition:* The condition is the flexible version of classical "lower than" condition. This condition is defined by an upper limit *(u)* and a margin *(m)*. The flexible condition is applied by using the fuzzy condition *a FLT [u,u,u,u+m]*. This condition can be used to express restrictions like "prices *under €100.000* will be accepted, with a *limit of €110.00*", setting *u=100.000* and *m=10.000*.

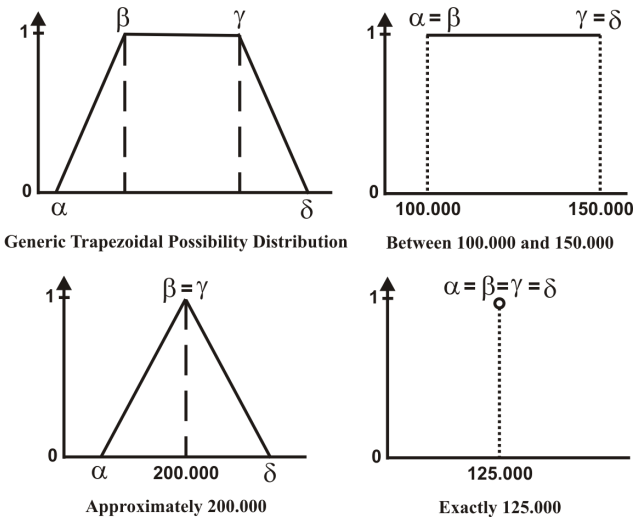

Figure 1: Price ranges modeled as trapezoidal possibility distributions

5.2 Nearness Conditions Qualifiers

This group of user qualifiers is designed to be employed on nearness conditions on *Type 3* attributes. They are used to model semantic concepts employed by humans to specify to what extent the result's value, for a concrete attribute, must match the value set by the customer as condition.

These nearness conditions are specified, using fuzzy logic, as a fuzzy equality condition like *a FEQ v THOLD t*, where *a* is the attribute on which is applied the condition, *v* is the user desired value and *t* is the threshold value for the chosen qualifier.

Nearness conditions qualifiers are linguistic labels representing a threshold (*t*), a minimum nearness level, for the nearness degree between the scalar value specified in the user condition (*v*) and the scalar value stored in the attribute (*a*) on which the condition is applied. These qualifiers and their associated threshold values, are shown in Table 3.

The described qualifiers can be used by customers to model conditions like "I am looking for a flat, but other similar kind of real estates will be considered". To model this condition, the qualifier "Fairly" is chosen, defining the condition "flats and fairly similar". Taking into account the proximity relation defined in Table 2, the results will be *flats*, and with lower relevance degree, *apartments* and *attics*.

Table 3 :Nearness condition qualifiers

Qualifier	Nearness degree threshold
Only	1
Very	0.9
Fairly	0.75
Moderately	0.5
Not Much	0.25
Any	0

5.3 Context Adaptive Conditions

Context adaptive conditions are a special group of fuzzy numerical conditions designed to be context sensitive. This kind of conditions are used to set restrictions on results' attribute values, employing humans concepts like, for instance, "cheap", "moderate" or "expensive" – if we are talking about a house price –, or concepts like "small", "middle-sized" or "big" – if we are talking about house size –.

These conditions are described as "adaptive" because the human concepts modeled by them are not mathematically constant, they can not be described as a constant range of values. Let us analyze the problem using an example: A "*cheap* detached house in suburbs" does not cost the same as a "*cheap* flat in city centre". Of course both are "*cheap*", but the prices are not in the same range, both prices ranges representing the concept "*cheap*" are different. As it can be seen, the context in which the real estate is involved must be considered (i.e. "a *flat* in *city centre*" or "*detached house* in *suburbs*") to obtain the numerical representation of the range represented by the concept "*cheap*".

Regardless of the attribute on which the condition is applied (i.e. price, area, etc.), the following main abstract degree concepts can be defined:

1. *Low:* Includes specific concepts like "cheap" (price) or "small" (size).
2. *Medium:* Includes specific concepts like "moderate" (price) or "middle-sized" (size).
3. *High:* Includes specific concepts like "expensive" (price) or "big" (size).

These main abstract degree concepts split the underlying domain in three fuzzy regions. The boundaries of these regions are defined basing on the average value (*M*) of each affected attribute, as Figure 2 shows. This average is calculated taking into account only the set of records which comprise the *context* of the condition.

The following steps are necessary to create a concrete trapezoidal representation for each adaptive condition in the query. This representation is built based on the average value – in the context – of the attribute associated with each adaptive condition.

1. *Obtaining the context:* Apply all conditions defined in the query, except *context* adaptive conditions, on all records. The obtained result set is named the *context* defined in the query.
2. *Obtaining average value:* Calculate attribute's average value (*M*) on *context* result set, for each attribute affected by a context adaptive condition.
3. *Creating trapezoidal representations:* Build the concrete trapezoidal representation of each context adaptive condition present in the query, using *M* value of the attribute on which the condition is applied, as Figure 2 shows.
4. *Obtaining query result:* Once all context adaptive conditions are defined as a concrete trapezoid, all conditions present in query are applied on all records to obtain query results.

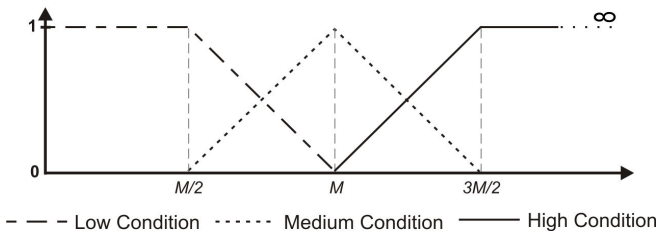

Figure 2 : Partition of underlying domain by context adaptive conditions

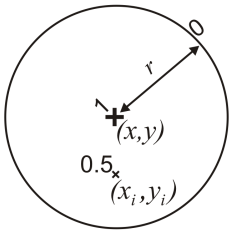

Figure 3 : Membership function for a circular area

5.4 Fuzzy Geographical Conditions

This kind of conditions make possible to set restrictions on results using geographical criteria.

Usually, customers look for real estates near to a specific location (i.e. highway exits, train stations, etc.). The nearer the real estate to the location the better, limiting results to a maximum distance.

Fuzzy geographical conditions are useful to model this kind of user requirements. These conditions are defined by the parameters *x*, *y* and *r*, from which a circular area is defined, centered in the point *(x,y)* – the specific location – and *r* radius length – the maximum distance limit –, as Figure 3 shows.

Equation 1 determines the membership degree of each real estate in the defined area. Where *(x,y)* is the center of the area defined by the condition, *r* is the radius of this area, (x_i,y_i) are the location coordinates of each evaluated real estate, and *dist(a,b)* is the Euclidean distance between *a* and *b* points.

5.5 Real Estate Fulfillment Degree

The Real Estate Fulfillment Degree (REFD) is a value, in the interval [0,1], that reflects de matching degree between a real estate and a user query. A value of 0 means that the real estate do not fulfill any query conditions, a value of 1 means that the real estate is a perfect match that fulfill all query conditions. Values between 0 and 1 are interpreted as partial fulfilling of query conditions. The REFD is computed – for each query result – as a weighted average, aggregating the fulfillment degrees obtained for each query condition. The weight associated to each condition represents its significance set in the condition priority ranking defined by the user.

The suggested calculus of weights given a user priority order (PO) is defined as shown in equations (2) and (3), where *t* is the number of conditions defined in the query, *n* the number of conditions included in the PO, *m* the number of conditions without associated PO, $w_{p(i)}$ the weight of a condition with a PO *i* – considering *i=1* the highest PO, and *i=n* the lowest PO – and w_u the weight for conditions without an assigned PO.

$$CDEG((x,y),(x_i,y_i)) = \frac{r - dist((x,y),(x_i,y_i))}{r} \quad (1)$$

$$w_p(i) = \begin{cases} \frac{1}{2^i} & i \neq t \\ \frac{1}{2^{i-1}} & i = t \end{cases} \quad i = 1...n \quad (2)$$

$$w_u = \frac{1}{m2^n} \quad (3)$$

6 Fuzzy Querying Example

ImmoSoftWeb (ISW) [7] is a real estate management application that provides a web interface to a FRDB, allowing flexible storage and retrieval of fuzzy data. ISW is built on a Fuzzy SQL server [8] based on the GEFRED (a GEneralized model for Fuzzy RElational Databases) model [5]. The FRDB provides a fuzzy query language, Fuzzy SQL (FSQL), for interaction between the application and the database.

The example formulates a query in natural language and shows its construction in the Web Form. The query is the following: "Search for a *Flat and fairly similar*, with an *area between 85 and 100 m² allowing others with a maximum difference of 20 m²*, and *Price lower than €60000 with an upper limit of €5000*, and must have *at least 2 rooms*".

Using the Web Form, the user can define different conditions for each of the real estate attributes. First, a condition type is selected and then, if necessary, numerical values are introduced to fully characterize the condition. There are different kinds of conditions, as seen previously, that contribute to better express user requirements. Once filled with the values and conditions present in the query, the Web Form should look like the one in Fig. 4.

The application makes possible to rank the query conditions defined, in order to give more significance to those conditions that are more important to the user, and that he wants to be fulfilled even at the expense of other less important conditions. To do this, each condition has an associated weight that measures its significance to the user. The sum of the weights of all the conditions present in the query must be 1. In the example all conditions have the same weight, the same significance to the user.

Once the query is defined, real estates fitting the query are searched in the database. The matching among query and real estates is performed attribute by attribute. For each attribute compared, a resemblance degree is obtained for that attribute. After matching query and real estate, a resemblance degree for each compared attribute is obtained. Then a global value of fulfillment degree is computed aggregating the resemblance degrees obtained for the attributes. In this aggregation process the different

weights set by the user are taken into account. Real Estate Fulfillment Degree (REFD) is a value between 0 and 1, which is showed as a percentage to the user.

Figure 5 shows resulting real estates ordered by the computed REFD. Obtained REFD reveal that there is only one result that fits completely to the query. The rest of the results are fairly similar to that requested in the query, but they are not a perfect match.

In a crisp system this example would have found only one resulting real estate, the one that fits the query with REFD 100%. Therefore the search has been improved, returning not only real estates that fit perfectly to the query, but those that are very similar. In cases where real estates fitting completely the query were not found, it could be possible to obtain similar results that could be of interest.

Fuzzy real estate search is a useful tool for customers and sales agents, mainly due to the increased number of results, including alternative results that increase business opportunities by detecting more accurately which products could be of interest for customers.

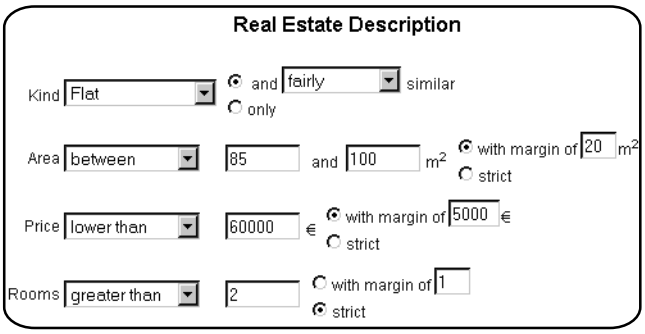

Figure 4 : Real Estate Description Web Form

Results :: Buy

ID	REFD	Kind	Price €	Area m^2	Rooms
43	100%	Flat	38.000 €	90.0 - 100.0 ±9% m^2	3
30	94%	Flat	60.000 €	80.0 ±10% m^2	4
1	93%	Flat	65.000 €	90.0 - 120.0 ±9% m^2	4
45	92%	Flat	63.000 €	90.0 ±10% m^2	4
42	90%	Apartment	40.000 €	85.0 - 90.0 ±9% m^2	3
44	90%	Apartment	43.000 €	90.0 - 100.0 ±9% m^2	3
29	85%	Attic	63.000 €	100.0 ±10% m^2	4
62	82%	Flat	45.000 €	70.0 ±10% m^2	5

Figure 5 : Resulting Real Estates

7 Concluding Remarks and Future Works

In this paper a solution for enhancing expression capabilities in search forms of e-commerce sites – focusing on real estate search engines – is proposed. This enhanced expression capabilities contribute to a better and easier communication between customers and product search engines, allowing customer to express more naturally their needs, using flexible conditions.

As a result of this flexibility, the search engine can offer to customers a wide range of results, fitting completely and partially customer requirements, empowering business opportunities.

Future works will lead to employ Soft Computing, Data Mining and Knowledge Discovery to obtain customer profiles, shopping profiles and trend analysis. This information will help to accurately detect customer needs in order to improve precision and recall of product search engines.

Acknowledgments:
This work has been partially supported by the Spanish "Ministerio de Ciencia y Tecnología" (MCYT) under grant TIC2002-00480.

References:

[1] H. Prade, C. Testemale, Generalizing Database Relational Algebra for the Treatment of Incomplete or Uncertain Information and Vague Queries, *Information Sciences* Vol. 34, 1984, pp. 115-143.

[2] M. Zemankova-Leech, A. Kandel, *Fuzzy Relational Databases -- A Key to Expert Systems*, Köln, Germany, TÜV Rheinland, 1984.

[3] S. Fukami, M. Umano, M. Muzimoto, H. Tanaka, Fuzzy Database Retrieval and Manipulation Language, *IEICE Technical Reports*, Vol. 78, N. 233, pp. 65--72, AL-78-85 (Automata and Language) 1979.

[4] M. Umano, Freedom-O: A Fuzzy Database System, *Fuzzy Information and Decision Processes*. Gupta-Sanchez edit. North-Holand Pub. Comp. 1982.

[5] J.M. Medina, O. Pons, M.A. Vila, GEFRED. A Generalized Model of Fuzzy Relational Data Bases. *Information Sciences*, 76(1-2), pp. 87-109 1994.

[6] J.M. Medina, M.A. Vila, J.C. Cubero, O. Pons, Towards the Implementation of a Generalized Fuzzy Relational Database Model. *Fuzzy Sets and Systems*, 75, pp. 273-289. 1995.

[7] ImmoSoftWeb: http://idbis.ugr.es/immosoftweb

[8] J. Galindo, J.M. Medina, O. Pons, J.C. Cubero, A Server for Fuzzy SQL Queries, *Flexible Query Answering Systems*, eds. T. Andreasen, H. Christiansen and H.L. Larsen, Lecture Notes in Artificial Intelligence (LNAI) 1495, pp. 164--174. Ed. Springer, 1998.