Influence Diagrams and Structured Retrieval: Garnata Implementing the SID and CID Models at INEX'06

Luis M. de Campos, Juan M. Fernández-Luna,
Juan F. Huete, and Alfonso E. Romero

Departamento de Ciencias de la Computación e Inteligencia Artificial
E.T.S.I. Informática y de Telecomunicación, Universidad de Granada,
18071 – Granada, Spain
{lci,jmfluna,jhg,aeromero}@decsai.ugr.es

Abstract. This paper exposes the results of our participation in INEX'06. Two runs were submitted to the Ad Hoc Thorough track obtained with Garnata, our Information Retrieval system for structured documents. We have implemented two different models based on Influence Diagrams, the SID and CID models. The result of this first participation has been very poor. In the paper, we describe the models, the system, and analyse the possible reason of such a bad performance.

1 Introduction

Although the research group "Uncertainty Treatment in Artificial Intelligence" at University of Granada has been participating in INEX since its beginnings, in this edition it is the first time that their members submit a run to the official tasks. Until now, our contribution to INEX had been the design of several topics and the assessments of relevance judgements, but now we have participated with a new experimental platform to perform structured retrieval using Probabilistic Graphical Models.

We have participated in the Ad Hoc Track (the Thorough subtask) with the results given by two models based on Influence Diagrams [5]: the Simple Influence Diagram Model (SID) and the Context-based Influence Diagram Model (CID) [2,3]. They have been implemented in the *Garnata Retrieval System* [4], a software specifically designed and implemented to work with Probabilistic Graphical Model-based structured retrieval models, like Bayesian Networks and Influence Diagrams [7]. As fas as we know, these models are the first attempts to apply Influence Diagrams to structured retrieval.

It also should be pointed out that the results of this first participation are not good, and in fact clearly disappointing. In fact, we are in the last positions of the ranking in the Thorough task. Perhaps, this bad behaviour could be due to a wrong implementation of the algorithm found in Garnata after studying the poor performance obtained by the SID and CID models, once the official results were published. However, after sharping the implementation fact, the results are

N. Fuhr, M. Lalmas, and A. Trotman (Eds.): INEX 2006, LNCS 4518, pp. 165–177, 2007.

still not good enough. Therefore, as future works we propose a list of possible improvements in the algorithms trying to identify the problems in them.

Another reason to get such a classification in the ranking could be the free unspecified parameters of the models. Since they are collection-dependent, those that we used in the experimentation were not the best, because no Wikipedia assessments were disposable at that time. We think that the behaviour of both models could be clearly improved with a more systematic experimentation finding an optimal configuration of the parameters.

In order to describe the models and the software that we have used, this paper is organised as follows: the next section will introduce Influence Diagrams to the reader (the formalism used in the models). Sections 3 and 4 will describe the SID and CID models, and Garnata, the Information Retrieval System, which implements them, respectively. The following section will discuss how the experimentation was performed, and try to explain the reasons of the unexpected performance of the models. This paper will finish with the conclusions and future works for our system.

2 Introduction to Influence Diagrams

An Influence Diagram [5,9] provides a simple notation for creating decision models by clarifying the qualitative issues of the factors which need to be considered and how they are related, i.e. an intuitive representation of the model. They also have associated an underlying quantitative representation in order to measure the strength of the relationships: we can quantify uncertain interactions between random variables and also the decision maker's options and preferences. The model is used to determine the optimal decision policy. More formally, an Influence Diagram is an acyclic directed graph containing three types of nodes (*decision*, *chance*, and *utility* nodes) and two types of arcs (influence and informative arcs).

Nodes in an Influence Diagram represent various types of variables.

- **Decision nodes:** Usually drawn as rectangles, these represent variables that the decision maker controls directly. These variables model the decision alternatives available for the decision maker.
- **Chance nodes:** Usually drawn as circles, these represent random variables, i.e. uncertain quantities that are relevant to the decision problem and cannot be controlled directly. They are quantified by means of conditional probability distributions, identical to those used in Bayesian networks[1]. Predecessors (parents) of chance nodes that are decision nodes act in exactly the same way as those predecessors that are chance nodes (they index the conditional probability tables of the child node).
- **Utility nodes:** Usually drawn as diamonds, they express the profit or the preference degree of the consequences derived from the decision process.

[1] In fact, the subset of an Influence Diagram that consists only of chance nodes is a Bayesian network. Thus, an Influence Diagram can also be viewed as a Bayesian network enlarged with decision and utility nodes.

They are quantified by the utility of each of the possible combinations of outcomes of their parent nodes.

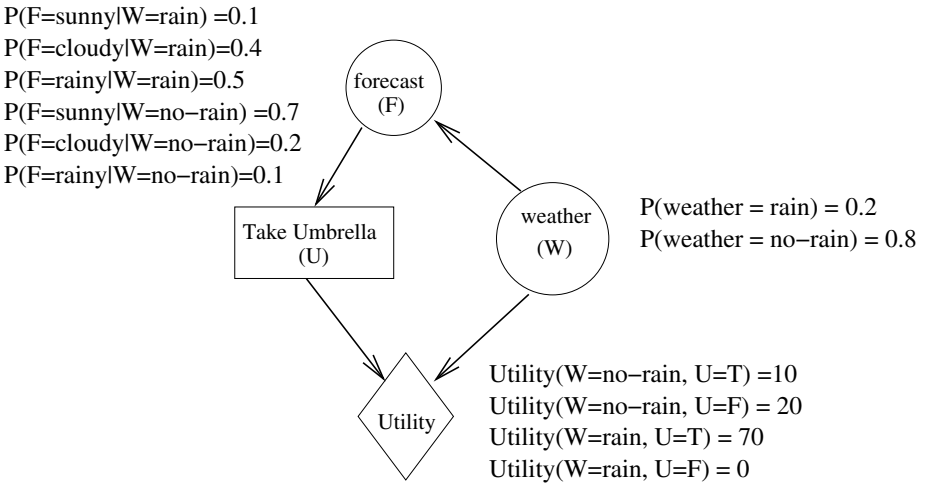

Fig. 1. An example of an Influence Diagram

There are also different types of arcs in an Influence Diagram, which generally represent influence. The arcs between chance nodes represent probabilistic dependences (as it occurs in Bayesian networks). The arcs from a decision node to a chance node or to a utility node establish that the future decision will affect the value of the chance node or the profit obtained, respectively. Arcs between a chance node and a decision node (also called *informative*) only say that the value of the chance node will be known at the moment of making the decision. Finally, arcs from a chance node to a utility node will represent the fact that the profit depends on the value that this chance node takes. The absence of an arc between two nodes specifies (conditional) independence relationships. It should be noted that the absence of an arc is a stronger statement than the presence of an arc, which only indicates the possibility of dependence.

Some arcs in Influence Diagrams clearly have a causal meaning. In particular, a directed path from a decision node to a chance node means that the decision will influence that chance node, in the sense of changing its probability distribution.

A simple example of an Influence Diagram appears in Figure 1. It has two chance nodes, F and W, representing, the weather forecast in the morning (sunny, cloudy or rainy), and whether it actually rains during the day (rain or no-rain), respectively. It has one decision node U, taking an umbrella (with possible values true or false). The utility node measures the decision maker's satisfaction.

With each chance node X in the graph, the quantitative part of an Influence Diagram associates a set of conditional probability distributions $p(X|pa(X))$,

one for each *configuration* $pa(X)$ from the *parent set* of X in the graph, $Pa(X)$, i.e. for each allocation of values to all the variables in the parent set of X. If X has no parents ($Pa(X) = \emptyset$), then $p(X|pa(X))$ equals $p(X)$. For each utility node V, a set of utility values $v(pa(V))$ is associated, specifying for each combination of values for the parents of V, a number expressing the desirability of this combination for the decision maker.

The goal of Influence Diagram modeling is to choose the decision alternative that will lead to the highest expected gain (utility), i.e. to find the *optimal policy* [8]. In order to compute the solution, for each sequence of decisions, the utilities of its uncertain consequences are weighted with the probabilities that these consequences will occur.

3 The SID and CID Models

In this section, we shall briefly describe the SID and CID models for structured retrieval. A complete description of these models can be found in [2,3].

3.1 The Underlying Bayesian Network

In this model, we consider three different kinds of entities which will be represented by the means of three different kinds of random variables. Namely: *index terms*, *basic structural units*, and *complex structural units* (see below for definitions).

Because our models are Influence Diagrams, they are based in an underlying Bayesian networks which represents a structured document set. This Bayesian network will contain two kinds of *nodes*, representing the terms and the structural units. The former will be given by the set $\mathcal{T} = \{T_1, T_2, \ldots, T_l\}$. As stated before, there are two types of structural units: *basic structural units*, those which only contain terms (*leaf nodes* in XML), and *complex structural units* (*non-leaf nodes* in XML), that are composed of other basic or complex units. For those units containing both text and units (appearing often in Wikipedia), we consider them as complex units, and the text of that unit is assigned to a new unit called *virtual unit*[2], a non-retrievable basic unit (see figure 2). The notation for these nodes is $\mathcal{U}_b = \{B_1, B_2, \ldots, B_m\}$ and $\mathcal{U}_c = \{S_1, S_2, \ldots, S_n\}$, respectively. Therefore, the set of all structural units is $\mathcal{U} = \mathcal{U}_b \cup \mathcal{U}_c$. In this paper, T or T_k will represent a term; B or B_i a basic structural unit, and S or S_j a complex structural unit. Generic structural units (either basic or complex) will be denoted as U_i or U. Each node T, B or S has associated a *binary random variable*, which can take its values from the sets $\{t^-, t^+\}$, $\{b^-, b^+\}$ or $\{s^-, s^+\}$ (the term/unit is not relevant or is relevant), respectively. A unit is relevant for a given query if it satisfies the user's information need expressed by this query. A term is relevant in the sense that the user believes that it will appear in relevant units/documents.

[2] Of course this unit will not appear in the XPath route of its descendants, is only a formalism that allow us using the two different kinds of units explained above.

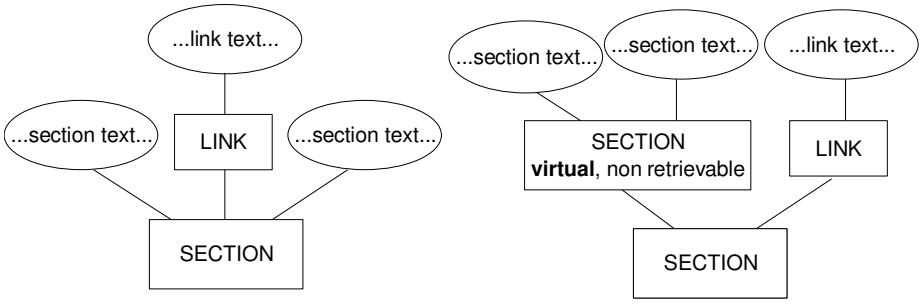

Fig. 2. An example of a virtual unit: after the change, the "section" does not contain text, only other units

Regarding the arcs of the models, there will be an arc from a given node (either term or structural unit) to the particular structural unit the node belongs to. It express the fact that the relevance of a given structural unit to the user will depend on the relevance values of the different elements (units or terms) that comprise it. It should be noted that with this criteria, term nodes have no parents.

Note that the hierarchical structure of the model determines that each structural unit $U \in \mathcal{U}$ has *only one* structural unit as its child: the unique structural unit containing U (except for the leaf nodes, i.e. the complete documents, which have no child). We shall denote indistinctly by $Hi(U)$ or $U_{hi(U)}$ the single child node associated with node U (with $Hi(U) = \text{null}$ if U is a leaf node).

The numerical values for the conditional probabilities have also to be assessed: $p(t^+), p(b^+|pa(B)), p(s^+|pa(S))$, for every node in $\mathcal{T}, \mathcal{U}_b$ and $\mathcal{U}_c$, respectively, and every configuration of the corresponding parent sets $pa(X)$. A canonical model proposed in [1] will be used to represent the conditional probabilities which supports a very efficient inference procedure. These probabilities are defined as follows:

$$\forall B \in \mathcal{U}_b, \quad p(b^+|pa(B)) = \sum_{T \in R(pa(B))} w(T, B)\,, \tag{1}$$

$$\forall S \in \mathcal{U}_c, \quad p(s^+|pa(S)) = \sum_{U \in R(pa(S))} w(U, S)\,, \tag{2}$$

where $w(T, B)$ is a weight associated to each term T belonging to the basic unit B and $w(U, S)$ is a weight measuring the importance of the unit U within S. In any case $R(pa(U))$ is the subset of parents of U (terms for B, and either basic or complex units for S) relevant in the configuration $pa(U)$, i.e., $R(pa(B)) = \{T \in Pa(B) \,|\, t^+ \in pa(B)\}$ and $R(pa(S)) = \{U \in Pa(S) \,|\, u^+ \in pa(S)\}$. These weights can be defined in any way with the only restrictions that

$$w(T, B) \geq 0, \quad w(U, S) \geq 0, \sum_{T \in Pa(B)} w(T, B) \leq 1, \text{ and } \sum_{U \in Pa(S)} w(U, S) \leq 1.$$

3.2 Constructing the Influence Diagram

Once the Bayesian network has been constructed, it is enlarged by including decision and utility nodes, and so transforming it into an Influence Diagram.

- **Decision nodes:** These nodes model the decision variables, representing the possible alternatives available to the decision maker. One decision node, R_i, for each structural unit $U_i \in \mathcal{U}$. R_i represents the decision variable related to whether or not to return the structural unit U_i to the user. The two different values for R_i are r_i^+ and r_i^-, meaning 'retrieve U_i' and 'do not retrieve U_i', respectively.
- **Utility nodes:** We shall also consider one utility node, V_i, for each structural unit $U_i \in \mathcal{U}$, $\forall i = 1, \ldots, |\mathcal{U}|$. V_i will measure the value of utility for the corresponding decision.

We shall also consider a *global utility node* representing the joint utility of the whole model. This node will be denoted by Σ, meaning we are assuming an additive behavior of the model. In addition to the arcs between chance nodes (those present in the Bayesian network), a set of arcs pointing to utility nodes are also included, employed to indicate which variables have a direct influence on the desirability of a given decision, i.e. the profit obtained will depend on the value of these variables. We shall consider two different set of arcs, which will consistently generate *two different Influence Diagrams models*:

1. *Simple Influence Diagram (SID)*: We shall only take into account arcs from chance nodes U_i and decision nodes R_i to the utility nodes V_i, $\forall i = 1, \ldots, |\mathcal{U}|$. These arcs mean that the utility function of V_i depends obviously only on the decision made and the relevance value of the structural unit considered. Finally, the utility node Σ has all the utility nodes $V_{i,j}$ as its parents. These arcs represent the fact that the joint utility of the model will depend on the values of the individual utilities of each structural unit.
2. *Context-based Influence Diagram (CID)*: In order to represent that the utility function of V_i obviously depends on the decision made and the relevance value of the structural unit considered, we use arcs from each chance node U_i and decision node R_i to the utility node V_i. Another important set of arcs are those going from $Hi(U_i)$ to V_i, which represent that the utility of the decision about retrieving the unit U_i also depends on the relevance of the unit which contains it (of course, for those units U where $Hi(U) = \text{null}$, this arc does not exist).
   Again, the utility node Σ will have the same set of parents as in the SID model.

Figure 3 shows an example of both Influence Diagram models: the SID (left-hand side) and the CID (right-hand side).

Finally, for each node V_i, the associated utility functions must be defined:

1. *Utility nodes in SID*: For each node V_i, we need to assess a numerical value for the possible combination of the decision node R_i and the chance node

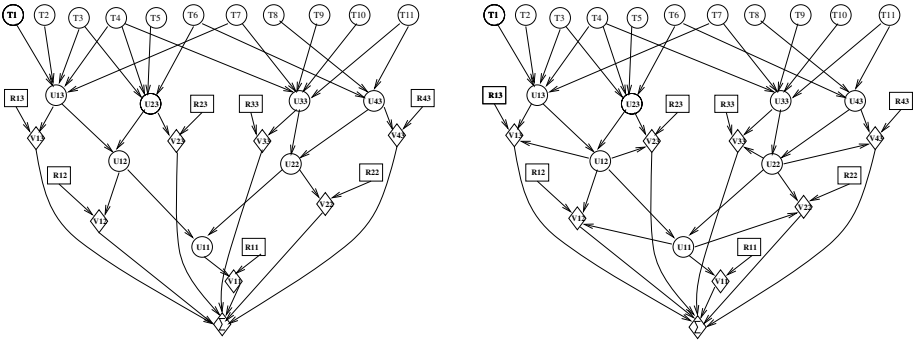

Fig. 3. Influence diagrams for the SID and CID models

representing the structural component U_i. The *four values* are $v(r_i^+|u_i^+)$, $v(r_i^-|u_i^+)$, $v(r_i^+|u_i^-)$ and $v(r_i^-|u_i^-)$.

2. *Utility nodes in CID*: For each utility node V_i we have *eight parameters*, one for each combination of values of the decision node R_i and the chance nodes U_i and $Hi(U_i)$ (except for the leaf nodes, which only require four values). These values are represented by $v(r_i, u_i, u_{hi(U_i)})$, with $r_i \in \{r_i^-, r_i^+\}$, $u_i \in \{u_i^-, u_i^+\}$, and $u_{hi(U_i)} \in \{u^-_{hi(U_i)}, u^+_{hi(U_i)}\}$.

3.3 Inference and Decision Making

To solve an Influence Diagram, the expected utility of each possible decision (for those situations of interest) has to be computed, thus making decisions which maximize the expected utility. In our case, the situation of interest corresponds to the information provided by the user when he/she formulates a query. Let $\mathcal{Q} \subseteq \mathcal{T}$ be the set of terms used to express the query. Each term $T_i \in \mathcal{Q}$ will be instantiated to either t_i^+ or t_i^-; let q be the corresponding configuration of the variables in $\mathcal{Q}$. We wish to compute the expected utility of each decision given q. As we have assumed a global additive utility model, and the different decision variables R_i are not directly linked to each other, we can process each one independently. The expected utilities for each U_i can be computed for each model by means of:

– *SID Model:*

$$EU(r_i^+ \mid q) = \sum_{u_i \in \{u_i^-, u_i^+\}} v(r_i^+, u_i,)\, p(u_i|q)\,, \tag{3}$$

$$EU(r_i^- \mid q) = \sum_{u_i \in \{u_i^-, u_i^+\}} v(r_i^-, u_i)\, p(u_i|q)\,. \tag{4}$$

– *CID Model:*

$$EU(r_i^+ \mid q) = \sum_{\substack{u_i \in \{u_i^-, u_i^+\} \\ u_{hi(U_i)} \in \left\{u_{hi(U_i)}^-, u_{hi(U_i)}^+\right\}}} v(r_i^+, u_i, u_{hi(U_i)})\, p(u_i, u_{hi(U_i)}|q)\,, \quad (5)$$

$$EU(r_i^- \mid q) = \sum_{\substack{u_i \in \{u_i^-, u_i^+\} \\ u_{hi(U_i)} \in \left\{u_{hi(U_i)}^-, u_{hi(U_i)}^+\right\}}} v(r_i^-, u_i, u_{hi(U_i)})\, p(u_i, u_{hi(U_i)}|q)\,. \quad (6)$$

In the context of a typical decision making problem, once the expected utilities are computed, the decision with greatest utility is chosen: this would mean to retrieve the structural unit U_i if $EU(r_i^+|q) \geq EU(r_i^-|q)$, and not to retrieve it otherwise. However, our purpose is not only to make decisions about what to retrieve but also to give a ranking of those units. The simplest way to do it is to show them in decreasing order of the utility of retrieving U_i, $EU(r_i^+|q)$.

A detailed description of how to compute the posterior probabilities required in these previous equations can be found in [2,3], but only to mention that the specific characteristics of the canonical model used to define the conditional probabilities will allow us to efficiently compute the posterior probabilities in the following way:

$$\forall B \in \mathcal{U}_b, \qquad p(b^+|q) = \sum_{T \in Pa(B) \setminus Q} w(T, B)\, p(t^+) + \sum_{T \in Pa(B) \cap R(q)} w(T, B)\,, \quad (7)$$

$$\forall S \in \mathcal{U}_c, \qquad p(s^+|q) = \sum_{U \in Pa(S)} w(U, S)\, p(u^+|q)\,. \quad (8)$$

4 Garnata: An Information Retrieval System for Structured Documents

Garnata was born as an implementation completely adapted to the models based on the above Probabilistic Graphical Models to retrieve structured documents, although other models following the same philosophy could be easily implemented in it. Written in C++, following the object-oriented paradigm, it offers a wide range of classes and a complete set of utility programs. It implements the SID and CID models.

It is able to manage different collections, and different indexes over the same collection. It can choose among different stopword lists (previously inserted into the system) and use (if desired) Porter's stemming algorithm.

In our models, several valid weighting schemes could exist because of its experimental nature. As a consequence, in Garnata, the process of indexing does not compute the weights (setting all of them to be zero). Instead of that, we have added the possibility to calculate weights (following a certain weighting scheme)

for previously built indexes without inserting into them, and store them in files, the so-called *weight files*. Thus, records of that precomputed weight files are kept in order to provide a fast way to insert one into the index itself in order to retrieve with it.

To store *textual information* (terms and identifiers of the final units where they appear), we use inverted indexes [6]. While the lexicon is kept entirely in memory (both while indexing and querying), the list of occurrences is read from disk. We use another file to write the list of relative positions of each term inside a unit in order to answer queries containing proximity operators or phrases (although in the current stage of Garnata, they are not used to formulate a query).

To maintain *information about the structural units*, we use one direct access file, except for the XPath routes, which are stored separately. Other files keep relations among units, being accessible with only two disk reads. So, a large file contains data of each unit itself (identifier, tag, container, position, ...) and besides, we can easily manage the following relationships with two disk accesses (essential for our models):

- Given a non-final unit, returning the list of identifiers of the units that it contains.
- Given a final unit, returning the container unit and, recursively, all the containers until a root unit is found.
- Given a final unit, returning the list of contained terms (using a *direct index*).

The Garnata's indexing subsystem also implements file compression to speed up query processing.

Querying subsystem is the most critical part of an IR system. In our case, we have built structures at indexing time to reduce at maximum the amount of disk accesses while processing a query, in order to save time and give a short response time. The algorithm for achieving this task comprises the following steps (not necessarily in this order):

1. The query is parsed, and occurrences of the component terms are retrieved from disk.
2. For each occurrence, implied final units are read into memory (if not already there).
3. For each final unit, its descendants are read into memory (if not already there).
4. Propagation is carried out, units are sorted by its probability of relevance, and the result is returned.

The first big bottleneck to be minimized is due to the reading from disk of the unit objects (containing information about each unit). We will keep two unit caches in memory: the first one, containing final units, and the second one, containing complex units. Both will be *static caches*, meaning that they will not change the unit stored in each cache slot. Cache is accessed doing a hash function-like scheme, so for each cache slot, we shall have several candidates (those identifiers being the hash inverse of the slot identifier).

For the final units cache, in each slot, we shall store the unit containing greater number of terms (among the candidates). For the complex units cache, in each slot, we shall store the unit containing more final units. These two heuristics has shown very good time performance in our experiments.

The paper [4] contains a more detailed and technical description of Garnata.

5 Experimental Setting for INEX'06. Analysis of the Results

5.1 Parameter Setting and Official Runs

In this section, we shall describe the conditions under which we have performed the two runs submitted to the Thorough task.

First of all, the weighting scheme used in equations 7 and 8 to compute posterior probabilities has basically been a normalized tf-idf scheme for weights of terms in units. On the other hand, the weights of units included in a complex unit, U_i, measure to a certain extent, the proportion of the content of the unit U_i which can be attributed to each one of its components. A detailed explanation of how they are computed is shown in [2].

With respect to the prior probabilities of relevance of the terms, $p(t^+)$, they can also be defined in any reasonable way, for example an identical probability for all the terms, $p(t^+) = p_0$, $\forall T \in \mathcal{T}$, as proposed in [2], specifically, $\frac{1}{|\mathcal{T}|}$.

Because we are ranking units by the expected utility of retrieving them, $EU(r_i^+ \mid q)$, we only need to assess half of the parameters for each model (those utilities with r_i^+), being two in the SID and four in the CID. Regarding the utilities for the SID model, for a given unit U_i, the best situation is clearly for a relevant unit to be retrieved, and the worst situation, for a relevant unit to be hidden. We therefore fix $v(r_i^+|u_i^+) = 1$ and $v(r_i^-|u_i^+) = 0$. As stated before, $v(r_i^-|u_i^+)$ and $v(r_i^-|u_i^-)$ are not used. In the context of the CID model, and following the idea of trying to return a unit only when its container is not relevant, we shall fix these two parameters: $v(r^+|u^+, w^-) = 1$ and $v(r^-|u^+, w^-) = 0$, and set the rest to 0. Therefore, the utility values are those in Table 1.

Table 1. Utility configuration for the CID model

v^+_{--}	v^+_{-+}	v^+_{++}	v^+_{+-}
0.0	0.0	0.0	1.0

In the first row of this table, the superscript is related to the value that the node R_i is taking. The subscripts refer to the value that units U and W are taking respectively at the same time.

The choice in the case of the SID model is very clear: we retrieve a unit when it is relevant. In the CID model, we retrieve a unit when it is relevant and the unit where it is contained is not relevant. We could say that these values

are the default vectors of utilities for both models, which is the reason why we selected them to run our experiments. We have to recognise that these are really strong restrictions and could be relaxed obtaining more appropriate values of the different utilities, but we could not do it due to the lack of time as we were finishing the development of Garnata when the submission deadline went off.

For the case of the CID model in the Thorough task, this setting is clearly counterproductive reducing the RSV of lots of units whose container is relevant. This fact will be discussed in the next subsection.

We should note that we finally compute RSV as $EU(r_i^+ \mid q) \cdot \mathrm{nIdf}_Q(u_i)$, being $\mathrm{nIdf}_Q(u_i)$ the proportion of the idf of the query terms contained in this unit or in their ancestors, that is

$$\mathrm{nIdf}_Q(u_i) = \frac{\sum_{p \in Q \cap u_i} \mathrm{idf}(p)}{\sum_{t \in Q} \mathrm{idf}(t)}.$$

Thus, $\mathrm{nIdf}_Q(u_i) \in [0, 1]$, and acts as a correcting factor for the utility (the more query terms a units contains, the more interesting for the user it is).

With these experimental settings, according to the results published in the evaluation section of the INEX'06 website [3], considering "Metric:ep-gr, Quantization: gen, Overlap=off", we have obtained a mean of effort-precision of 0.0004 with the runs of obtained by Garnata for the two models, occupying the disappointing 97^{th} and 98^{th} positions in the Thorough task.

5.2 Analysis of the Results

Studying the reasons of this bad behaviour, we discovered several bugs in the software, by which, in some cases, instead of retrieving a certain unit, we were returning a neighbour. Basically, the software returned a completely different unit to that which should have been returned. This has been fixed now.

About possible improvements, on the one hand, we have shown that the set of parameters used for the CID model should not perform well in this task. That set would clearly perform better choosing the best entry point for an XML file, but it is not productive in the Thorough task. We have carried an unofficial run changing the parameters in 1 to $(0, 1, 1, 1)$ and we have obtained a MAep of 0.0015. This is a 375% better than our previous result, but clearly not enough.

On the other hand, using nIdf_Q as a correcting factor sets the RSV of a unit which does not contain query terms to be zero. Since we are using a representation called *virtual unit* for units containing both text and units, the contained units will get a RSV of 0 if they do not contain query terms. However this is not what is supposed in the evaluation procedure, which assigns a positive RSV to a unit without query terms contained into other unit with relevant text. That fact is very common in the Wikipedia collection (assessments contain plenty of "Collectionlinks" without relevant text) but we are not managing that fact correctly.

[3] http://inex.is.informatik.uni-duisburg.de/2006/adhoc-protected/results/thorough/Thorough.html

6 Conclusions and Future Works

In this paper, we have presented the SID and CID models, and Garnata (the Information Retrieval System which implements them). These are the retrieval tools that we have used to participate in this our first edition of INEX, in the Thorough task. The results are very disappointing as we are at the bottom of the ranking. We have explained several reasons of this situation that we shall solve in future editions of the INEX workshop.

With respect to the future works, we intend to have a detailed experimentation with the IEEE and Wikipedia collections, in order to find automatically those values for the utility configurations which perform best. We think that selecting correctly the utilities should make some improvement.

Also, for next edition of INEX, we plan to participate in other tasks, such as 'Focused' or 'Best in Context'. In the case of this last task, we think that our models, using the underlying formalism of Influence Diagrams, are specially designed to make decisions considering the context, and it could perform acceptably.

Also, we are developing new models based in Probabilistic Graphical Models, which improve the performance of the SID and CID models, avoiding the limitations of the current. With these actions, we hope to have a better performance in the future.

Acknowledgments. This work has been jointly supported by the Spanish Ministerio de Educación and Ciencia, and Junta de Andalucía, under projects TIN2005-02516 and TIC276, respectively.

References

1. de Campos, L.M., Fernández-Luna, J.M, Huete, J.F.: The BNR model: foundations and performance of a Bayesian network-based retrieval model. Int. J. Appr. Reason. 34, 265–285 (2003)
2. de Campos, L.M., Fernández-Luna, J.M., Huete, J.F.: Using context information in structured document retrieval: An approach using Influence Diagrams. Inform. Process. Manag. 40(5), 829–847 (2004)
3. de Campos, L.M., Fernández-Luna, J.M., Huete, J.F.: Improving the Context-based Influence Diagram for Structured Retrieval. In: Losada, D.E., Fernández-Luna, J.M. (eds.) ECIR 2005. LNCS, vol. 3408, pp. 215–229. Springer, Heidelberg (2005)
4. de Campos, L.M., Fernández-Luna, J.M., Huete, J.F., Romero, A.E.: Garnata: An Information Retrieval System for Structured Documents based on Probabilistic Graphical Models. In: Proceedings of the Eleventh International Conference of Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU), pp. 1024–1031 (2006)
5. Jensen, F.V.: Bayesian Networks and Decision Graphs. Springer, Heidelberg (2001)
6. Witten, I.H., Moffat, A., Bell, T.C.: Managing Gigabytes. Morgan Kaufmann, San Francisco (1999)

7. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, San Mateo. Morgan Kaufmann, San Francisco (1988)
8. Shachter, R.: Evaluating Influence Diagrams. Oper. Res. 34, 871–882 (1986)
9. Shachter, R.: Probabilistic Inference and Influence Diagrams. Oper. Res. 36(5), 527–550 (1988)