# Filtering of Unrelated Answers in a Cooperative Query Answering System

Maheen Bakhtyar[1,2], Lena Wiese[3], Katsumi Inoue[4], and Nam Dang[5]

[1] Asian Inst. of Technology Bangkok, Thailand Maheen.Bakhtyar@ait.asia
[2] CSIT, University of Balochistan, Pakistan MaheenBakhtyar@um.uob.edu.pk
[3] Inst. of CS, University of Göttingen, Germany wiese@cs.uni-goettingen.de
[4] National Inst. of Informatics, Tokyo, Japan ki@nii.ac.jp
[5] Tokyo Inst. of Technology, Tokyo, Japan namd@de.cs.titech.ac.jp

**Abstract.** A database system may not always return answers for a query. Such a query is called a *failing query*. Under normal circumstances, an empty answer would be returned in response to such queries. Cooperative query answering systems produce generalized and relevant answers when an exact answer does not exist, by enhancing the query scope and including a broader range of information. Such systems may apply various generalization techniques, also referred to as generalization operators, to relax certain conditions and obtain related answers. These answers are not exact but informative answers that potentially contain some of the information that the user needs. Therefore, we propose a method to filter out unrelated answers and return only related answers to the user. We also propose a mechanism to have a restricted and optimized generalized query space by limiting the number of queries produced. We determine the similarity between user query and the answer produced. Unrelated answers are pruned out, and only the related and informative answers are returned to the user.

**Keywords:** Cooperative Query Answering, Query Relaxation, Semantic Filtering, Unrelated Answers, WordNet, Inductive Conceptual Learning

## 1 Introduction

A query with no resulting answer is called a *failing query. Cooperative query answering systems* produce generalized and relevant results when an exact answer does not exist by enhancing the query scope and including a broader range of information. These systems apply various generalization techniques; also referred to as generalization operators; to relax certain conditions and to obtain related answers. These answers are not exact but informative that potentially contain partial information that users need.

Various generalization techniques have been developed to give a wider range of answers [1–5]. Inoue et al., discuss and analyze the properties of three of the generalization/relaxation techniques [6]. They also discuss the iterative combination of the three techniques (called operators).

We can observe that sometimes a number of answers produced after query generalization are not related to what the user originally asked. The reason is often that new structures are added to the query when generalizing the query. Therefore, we propose a framework to determine the similarity between the user's original query and the answers produced after query expansion. Unrelated answers can be pruned out so that only the related and informative answers are returned to the user.

## 2 Techniques for Query Generalization in Cooperative Query Answering Systems

Cooperative query answering systems generalize queries by enhancing the query scope and including a broader range of information. If the answer to a query is *null*, then cooperative query answering relaxes the query, employing various techniques. Relaxation of the query results in some informative answers instead of an empty set.

Deductive generalization of queries assists in providing informative answers to failing queries. Gaasterland et al., provide a formal definition of deductive generalization [5]. We consider conjunctive queries and the generalization operators for them. Conjunctive queries contain both positive and negative conjuncts, but for simplicity, we initially target only queries with positive conjuncts/literals. See [6] for a definition of a conjunctive query.

A generalization operator is a mechanism to generalize a query to enhance the query scope. When applied to a set of queries, it produces a set of relaxed and more general queries. See [6] for the formal definition. We consider three generalization operators for conjunctive queries and their iterative execution combined with each other, as discussed by Inoue et al., [6].
The **DC operator** relaxes a conjunctive query by dropping one of the conjuncts from the query at a time, making the query less restrictive.
The **AI operator** adds a new variable in the query and therefore introduces a more general query. This results in coverage of different values for newly added variable.
The **GR operator** allows replacing a sub-part of the failing query with the head of a rule in the knowledge base $\Sigma$ (details in [6]). New constants and new conjuncts (but not variables) are potentially introduced in the generalized query, possibly removing some of the conjuncts, variables, and constants.

Inoue et al., state that it is sufficient to execute the three operators in a certain order when iteratively executing generalization steps. The authors apply the operators in breadth-first manner with the GR operator first, followed by DC and then AI. In some cases, GR is not applicable so we might apply DC then AI. When neither GR nor DC is applicable, we may apply only AI.

## 3 Similarity between User Query and the Returned Answer

We have discussed that the user may be disappointed if no answer is returned in response to a query. However, the user may be even more disappointed if unrelated answers are retrieved and returned in response. For example, if the query requires a *list of hotels in a particular town,* but the system provides a *list of hospitals in that town* after query generalization, the user would likely be unhappy, so such cases are best avoided.

Similarity is a general notion of a metric based on the relatedness of two target inputs (e.g., concepts, terms, or documents). Similarity values are usually between 0 and 1, where 0 means not similar at all. We propose a method to find and remove unrelated answers from a set of generalized and expanded results.

In our approach, we make use of syntactic as well as semantic constructs to compare and match the original query with generalized queries or with answers obtained in response to those generalized queries. We make use of WordNet[6]; in particular, we use the similarity between various concepts in WordNet to understand semantics and prune out unrelated queries or answers. There are various methods for measuring similarity between a pair of words, several based on WordNet. We use the similarity metric by Wu and Palmer [9] to measure similarity between a pair of words.

To better analyze, we express the operator tree by Inoue et al., in terms of the regular expression.

$$\underbrace{(AI)^+}_{I} \mid \underbrace{(DC)^+(AI)^*}_{II} \mid \underbrace{(GR)^+(DC)^*(AI)^*}_{III}.$$

We now examine each branch of the regular expression and develop similarity metrics for branches with expected dissimilar answers.

### 3.1 Anti-Instantiating Iteratively $(AI)^+$

The AI operator generates queries having new variables. Therefore, answers retrieved after introducing new variables according to a knowledge base may be unrelated to the original query. These answers might contain information entirely out of context for the query and the user's needs. To identify such situations, after AI execution, we match answers with the original query.

The AI operator and iterations of AI alone neither remove nor introduce any new predicate symbols. Therefore, the size (the sum of the arities of the predicates) of the query and the answer formulae should be equal, and it is possible to keep the answer predicates in the same order as in the query, with

[6] WordNet[7, 8] is a lexical database and a useful tool for computational linguistics and natural language processing(NLP) that contains a formal hierarchical arrangement of English vocabulary. The concepts are interlinked on the basis of some relationship such as synonym sets(called synsets).

one-to-one matching of the query predicates and answer predicates. To measure query-answer similarity for the AI$^+$ branch, we number the occurrences of variables and constants in a formula. Each such occurrence is called a position. For example, in the formula $\texttt{ill}(\texttt{mary}, X) \wedge \texttt{ill}(\texttt{peter}, X)$, `mary` occurs at position 1, $X$ occurs at positions 2 and 4, and `peter` occurs at position 3. In AI$^+$, the number of predicates and the size of the parameter tuples in the query and the answer is equal. Similarity measurement functions are shown in Equations 1 to 6.

The overall similarity $SimQA_A(q, a)$ between the generalized query $q$ and a candidate answer $a$ is show in Equation 1.

$$SimQA_A(q,a) = \frac{\sum_{i=1}^{m} Sim_p(i,q,a)}{m}, \tag{1}$$

$$Sim_p(i,q,a) = \begin{cases} 1, & V(p_i(q)) \wedge \neg A(p_i(q)). \\ Sim^h_{wn}(i,q,a), & V(p_i(q)) \wedge A(p_i(q)) \\ & \text{producing } p_i(a) \wedge \neg PN(p_i(a)). \\ Sim^h_{pn}(i,q,a), & V(p_i(q)) \wedge A(p_i(q)) \\ & \text{producing } p_i(a) \wedge PN(p_i(a)). \\ 1, & C(p_i(q)) \wedge p_i(q) = p_i(a). \\ 0.5, & PN(p_i(q)) \wedge p_i(q) \neq p_i(a). \\ Sim_c(p_i(q), p_i(a)), & \text{otherwise.} \end{cases} \tag{2}$$

where, $Sim_p(i, q, a)$ is the similarity of the parameter (variables or constants) at position $i$ and $m$ is the total number of positions (sum of arities) in the query $q$ (or the answer). Let $p_i(q)$ be the parameter, i.e., variable or constant, at the $i^{th}$ position in the query (and for $a$ analogously). We must calculate the similarity $SimQA_A(q, a)$ for each answer $a \in A'$ with the original query $q$. The similarity value can be used to rank answers obtained in one AI step in the tree. Then irrelevant answers can be filtered based on a threshold. $Sim_p(i, q, a)$ is the similarity of the parameter at position $i$ in the query and the answer formula. A description of each of the Boolean functions in Equation 2 is provided in Table 1.

**Table 1.** Boolean Functions used in Equation 2.

| Function | Return Value |
|---|---|
| $V(arg)$ | $True$ if $arg$ is a variable, $False$ otherwise. |
| $A(arg)$ | $True$ if $arg$ is anti-instantiated, $False$ otherwise. |
| $PN(arg)$ | $True$ if $arg$ is a proper noun, $False$ otherwise. |
| $C(arg)$ | $True$ if $arg$ is a constant other than a proper noun, $False$ otherwise. |

$Sim^h_{wn}(i, q, a)$ is the similarity of the symbol at an answer position that is not a proper noun having the same variable in the corresponding positions of the query. The similarity between two concepts is measured according to WordNet (as discussed earlier). $Sim^h_{wn}(i, q, a)$ works by iterating through all the positions in the answer and calculating the similarities (using WordNet) at the appropriate positions. We refer to this similarity as *horizontal* similarity, because the AI operation breaks bonds inside the query. For example, af-

ter AI, $\texttt{ill}(X, \texttt{asthma}) \wedge \texttt{allergic}(X, \texttt{inhaler})$ may become $\texttt{ill}(X, \texttt{asthma}) \wedge \texttt{allergic}(Y, \texttt{inhaler})$; hence the bond created by common variable $X$ is broken and the similarity needs to be checked after the answer is obtained. Similarly, $Sim^h_{pn}(i, q, a)$ is the similarity of the answer positions that are proper nouns having the same variable in the corresponding positions of the query. We assign a moderate similarity of 0.5 in the case of proper nouns, because we neither want to completely suppress the significance of proper nouns nor to completely ignore the notion of generalization. $Sim_c(r, s)$ is either the semantic similarity between the constants or becomes undefined. A query-answer pair is rejected or pruned out if the constants' similarity value is below a certain threshold $T_c$ (0.5 in our case). This way we avoid having a huge cross product of two queries (leading to combinatory explosion) hence reducing and restricting the query space.

$$Sim_c(r, s) = \begin{cases} Sim_{wn}(r, s), & \text{if } Sim_{wn}(r, s) >= T_c \\ \perp, & \text{otherwise.} \end{cases} \quad (3)$$

$$Sim^h_{wn}(i, q, a) = \frac{\sum_{\substack{j=1,\ i \neq j \\ p_j(q) = p_i(q)}}^{m} Sim_{wn}(p_i(a), p_j(a))}{O\big(p_i(q)\big) - 1} \quad (4)$$

$$Sim^h_{pn}(i, q, a) = \frac{\sum_{\substack{j=1,\ i \neq j \\ p_j(q) = p_i(q)}}^{m} Sim_{pn}(p_i(a), p_j(a))}{O\big(p_i(q)\big) - 1} \quad (5)$$

$$Sim_{pn}(r, s) = \begin{cases} 1, & \text{if } r = s, \\ 0.5, & \text{if } r \neq s, \end{cases} \quad (6)$$

where $r$ and $s$ are two proper nouns.

For further optimization, we may get user feedback to decide if some variable is important and need not be anti-instantiated. This would reduce the query space by limiting the size of the cross product of two sub-queries.

Example 1 shows how the described functions can be used to measure similarity and how answers irrelevant to the original query can be filtered.

*Example 1.* $(\text{AI})^+$.
**Failing Query**: $q = ill(X, \texttt{asthma}) \wedge \texttt{ill}(X, \texttt{fever}) \wedge \texttt{allergic}(X, \texttt{inhaler})$
Now, we analyze some answers obtained using SOLAR [10], in response to a few generalized queries.
**Generalized Query**: $q' = \texttt{ill}(X, \texttt{asthma}) \wedge \texttt{ill}(X, \texttt{fever}) \wedge \texttt{allergic}(\textbf{V'}, \texttt{inhaler})$
**Generalized Answer**:

$$a' = \texttt{ill}(\underbrace{\texttt{lisa}}_{1}, \underbrace{\texttt{asthma}}_{1}) \wedge \texttt{ill}(\underbrace{\texttt{lisa}}_{1}, \underbrace{\texttt{fever}}_{1}) \wedge \texttt{allergic}(\underbrace{\texttt{john}}_{\frac{0.5+0.5}{2}}, \underbrace{\texttt{inhaler}}_{1})[SimQA_A = \textbf{.9}]$$

**Explanation**: The query is relaxed by replacing the third occurrence of $X$ with a new variable $V'$ as shown above. Breaking the horizontal bond results in a constant $john$ and the similarity for this position is computed using $Sim^h_{pn}(i, q, a)$. Since the overall similarity is still quite high, the answer may be related to user needs.

**Generalized Query**: $q'' = \texttt{ill}(X, \texttt{asthma}) \wedge \texttt{ill}(X, \texttt{fever}) \wedge \texttt{allergic}(V', \textbf{V"})$
**Generalized Answer**:

$$a'' = \texttt{ill}(\underbrace{\texttt{lisa}}_{1}, \underbrace{\texttt{asthma}}_{1}) \wedge \texttt{ill}(\underbrace{\texttt{lisa}}_{1}, \underbrace{\texttt{fever}}_{1}) \wedge \texttt{allergic}(\underbrace{\texttt{tonny}}_{\frac{0.5+0.5}{2}}, \underbrace{\texttt{fruit}}_{0.5}) [SimQA_A = \textbf{0.83}]$$

**Explanation**: The relaxed query above is generated by replacing the constant `inhaler` with a new variable $V''$. This newly generated query results in a new constant `fruit` in the answer $a''$. The similarity between the concepts `inhaler` and `fruit` is calculated using the similarity algorithm [9] based on WordNet (the similarity is 0.5.)
**Generalized Query**: $q'''' = \texttt{ill}(X, \texttt{asthma}) \wedge \texttt{ill}(\textbf{V""}, \textbf{V"'}) \wedge \texttt{allergic}(V', V'')$
**Generalized Answer**:

$$a'''' = \texttt{ill}(\underbrace{\texttt{lisa}}_{1}, \underbrace{\texttt{asthma}}_{1}) \wedge \texttt{ill}(\underbrace{\texttt{peter}}_{\frac{0.5+0.5}{2}}, \underbrace{\texttt{bipolar\_disorder}}_{0.3}) \wedge \texttt{allergic}(\underbrace{\texttt{tonny}}_{\frac{0.5+0.5}{2}}, \underbrace{\texttt{sunlight}}_{0.26})$$

$$[SimQA_A = \textbf{0.59}]$$

**Explanation**: The similarity based on WordNet is calculated for positions 4 and 6 as described in case $a''$. We reject this answer based on Equation 3, as `sunlight` is not related to `asthma`, which also makes the answer quite unrelated to the query. This query space reduction can be implemented beforehand during the generalization phase by only substituting constants that are related. Similarly, the value is calculated for position 3 as described previously in $a'$.

### 3.2 Zero or More AI Iterations after Iterative Dropping Conditions $(\textbf{DC})^+(\textbf{AI})^*$

This branch can be further divided into the following sub-branches:

**$(\textbf{DC})^+$ (Dropping conditions iteratively)** : Dropping a condition does not introduce any new variable or conjunct while generalizing a query. Therefore, no new constants/answers will be introduced. Only informative answers related to some cropped part of the original query are returned. However, the information lost with each generalization step depends on the constants and variables being dropped with the dropped literal. A function returning the similarity depending on each generalization step is given in Equation 7.

$$SimQA_D(q, a) = \frac{m'}{m} \tag{7}$$

where $q$ is the original user query, $a$ is one of the answers produced against some generalized query, $m$ is the sum of arities of the predicates in the query, and $m'$ is the sum of arities of the predicates in the answer (or in the generalized query). Example 2 shows how similarity is calculated when literals are dropped in generalization steps. We also suggest to take optional feedback from the user if one predicate is more important than the others. Then we can decide which literal to drop first. This optimization will help drop literals efficiently.

*Example 2.* $(\text{DC})^+$.
**Failing Query**: $q = \texttt{patient}(X, \texttt{tokyo}, Y) \wedge \texttt{ill}(X, \texttt{asthma}) \wedge \texttt{allergic}(X, \texttt{inhaler})$
The generalized queries, their respective answers (obtained using SOLAR [10]), and the similarity values are:
**Generalized Query**: $q'_1 = \texttt{ill}(X, \texttt{asthma}) \wedge \texttt{allergic}(X, \texttt{inhaler})$
**Generalized Answer**: $a'_1 = \texttt{ill}(\texttt{peter}, \texttt{asthma}) \wedge \texttt{allergic}(\texttt{peter}, \texttt{inhaler})$
$[\frac{4}{7} = \mathbf{0.57}]$
**Explanation**: Three positions dropped; hence, more information loss

We can see from the similarity values as well as the answers obtained that the similarity decreases with each step of generalization. We also notice that the similarity can be calculated before the actual answer is extracted in case of $(DC)^+$ because in this case, the similarity is not obtained on the basis of semantics but only considering the syntax of the query. This enables supervised and controlled answer generation. Supervised and controlled query generalization improves the efficiency of the system by omitting some queries for which an answer need not be calculated.

**$(\text{DC})^+(\text{AI})^+$(Iterative dropping conditions followed by iterative AI)** : We already discussed that the DC operator alone does not introduce any new variables. However, if AI is applied after DC, then we do expect new variables in the generalized query. Therefore, we may obtain very dissimilar answers. DC will always reduce the size of the query, therefore reducing the answer size. In this case, one-to-one matching with the original query is not possible. Additionally, iterations over DC operator followed by AI increases the possibility of having queries with more dissimilar answers.

We propose a mechanism to first determine the amount of information retained after iteration over DC operations and then find out how similar the anti-instantiated part is when some information has already been cropped out during the DC iterations.

A function returns the similarity between query and the answer is shown in Equation 8.

$$SimQA_{DA}(q, a) = SimQA_A(q^{DC^+}, a^{DC^+AI^+}) \times \frac{m}{m'}, \tag{8}$$

where $SimQA_A$ is passed the query which is the result of the DC operator, along with the final answer obtained. The cropped query after DC is treated as the original query so that one-to-one matching is possible. We multiply this factor with the information retained after applying the DC operator. $m$ is total number of positions in the original query, and $m'$ is the number of positions remaining after DC operations have been carried out. This function first determines the information retained in the query after applying iterations of the DC operator and then finds how similar the retained information is to the query. Example 3 uses this function to find the similarity between the query and the answer with a single AI operation applied after multiple DC operations.

*Example 3.* Multiple DC followed by AI.
**Failing Query**: $q = \texttt{patient}(X, \texttt{tokyo}, Y) \wedge \texttt{ill}(X, \texttt{fever}) \wedge \texttt{allergic}(X, \texttt{inhaler})$
**Explanation**: $q^{DC.DC.AI} = \texttt{allergic}(X, Y)$ is the relaxed query produced after applying DC twice followed by single AI operation. Here, $\frac{m}{m'} = 0.28$ (Information Retained) and $a^{DC.DC.AI} = \texttt{allergic}(\texttt{peter}, \texttt{bronchodilator})$ is the generalized answer.
**Similarity**: $SimQA_{DA}(q, a) = SimQA_A(q^{DC.DC}, a^{DC.DC.AI})$ of $0.28 = (1 + 0.3)/2$ of $0.28 = 0.18$

### 3.3 DC followed by AI after Iterative Goal Replacement (GR)$^+$(DC)$^*$(AI)$^*$

This branch can be further divided into sub-branches but we only focus on one of the branches to start with.
**(GR)$^+$ (Iterative goal replacement)**: Execution of the GR operator potentially adds new conjuncts, new variables, and new constants to create generalized queries. It replaces a sub-part of the failing query with the head of a matching *single-headed range-restricted (SHRR) rule* in the knowledge base ($\Sigma$). The answers returned against these queries might be extremely dissimilar.

Generalized queries in this case consist of two parts, a replaced part (we call it the body of the rule $B$) and an actual existing/preserved part $E$, which is not replaced. Logically speaking, the replaced part of the query should not semantically affect similarity directly because it is database dependent, reflecting how the knowledge base is defined. On the other hand, we also notice that some variables or constants may be dropped or new ones may be introduced; therefore, we have to analyze accordingly.

We realize that the newly introduced constants, variables, and conjuncts are database-dependent and are placed as a rule in the knowledge base; therefore, we do not consider them as irrelevant or dissimilar to the original query construct. For the same reason, we do not compare the body with the head of the rule so that the sense of generalization is retained. However, we do consider the constants and variables missing in the generalized queries because that is the information lost during generalization steps. We consider the inter-relationship $R_{BE}$ between the body $B$ and existing part $E$ and then relate it with the inter-relationship $R_{HE}$ between the head of rule $H$ and the existing part $E$. Finding and analyzing the relevance $R_{BE}$ and $R_{HE}$ shows how much information has been lost during the relaxation process, and we also see how the bond between the body replaced and the existing part in the query is broken when the repeating variables or repeating constants are removed from the query.

We present a supervised and controlled answer generation mechanism for GR. We first assign a weight to each repeating variable and constant in $E$ and $B$. By repeating variable (or constant), we mean a variable (or constant) that is present in both $E$ and $B$. These variables and constants represent the bond or link between the body and the existing part, and we need to analyze the effects of breaking these links on query answer similarity. The weight for each repeating variable/constant is calculated using Equation 9.

$$w_e = \frac{O(e)}{m}, \quad (9) \qquad w^t = \sum(O(e^t) \times w_e), \quad (10) \qquad SimQA_G(q,a) = \frac{w^{q'}}{w^q} \quad (11)$$

where $w_e$ is the weight for variable (or constant) $e$ in $E$ and $B$. $O(e)$ is the total number of occurrences of $e$, and $m$ is the total number of positions in the original query. Once the weight for each repeating variable (or constant) is calculated, we find the total weight of the orignal query as well as that of the generalized query by Equation 10, where $t \in \{q, q'\}$ indexes the weight based on the number of repeating variables or constants in the original or relaxed query. The total similarity is calculated using Equation 11.

We know that $w^q$ and $w^{q'}$ are the total weights of the repeating arguments in the original query and generalized query, respectively. $w^q$ is the actual weight carried inside the query, whereas $w^{q'}$ is the weight retained after generalization. Therefore, the retained weight is calculated in Equation 11.

*Example 4.* $(\text{GR})^+$.
**Failing Query**:

$$q = \overbrace{\texttt{ill}(X, \texttt{asthma}) \wedge \texttt{allergic}(X, \texttt{inhaler})}^{B} \wedge \overbrace{\texttt{gender}(X, \texttt{male}) \wedge \texttt{history}(X, \texttt{asthma})}^{E}$$

**Weight of each Repeating Variable/Constt.**: $w_X = \frac{4}{8} = .5, w_{\texttt{asthma}} = \frac{2}{8} = .25$
**Generalized Query**:
$q' = \underbrace{\texttt{treat}(X, \texttt{injection})}_{H} \wedge \underbrace{\texttt{gender}(X, \texttt{male}) \wedge \texttt{history}(X, \texttt{asthma})}_{E}$
**SHRR rule**: $\texttt{ill}(X, Y) \wedge \texttt{allergic}(X, Z) \rightarrow \texttt{treat}(X, \texttt{injection})$
**Substitution**: $\theta = \{\texttt{asthma}/Y, \texttt{inhaler}/Z, X/X\}$
**Total Weight**: $w^q = 4 \times 0.5 + 2 \times 0.25 = 2.5$
$w^{q'} = 3 \times 0.5 + 1 \times 0.25 = 1.75$
**Total Similarity**: $SimQA_G(q,a) = \frac{1.75}{2.5} = 0.7$

We have discussed and proposed a similarity metric for the similarity between a user query and generalized informative answers produced by a cooperative query answering system SOLAR [10].

We explained our similarity metrics for all three operators executed iteratively (i.e., DC, AI and GR) and the execution of AI followed by DC. We do not discuss the case of combination with GR yet in detail. We now briefly discuss the remaining operator combinations with GR.

DC applied after GR will keep the same variables, conjuncts, and constants as in the initial query, with new ones added by the GR operator. In iterative GR followed by iterative AI, the AI operator adds more variables by replacing constants and some variables. Therefore, similarity must be calculated for the answers generated against this case.

We conclude that we need to check the similarity between the query and the answer only in case of AI and GR operators. Whenever these operators are

applied in any iteration, they might introduce new variables or conjuncts and may result in unrelated answers.

Once the similarity between the original query and the obtained answer is calculated, we filter out the answers with low similarity. A trial-and-error approach is required to define a threshold for deciding whether the answer is deemed related or not.

## 4 Conclusion, Limitation, and Future Work

We propose an approach to filter out unrelated answers in a cooperative query answering system and to return relevant answers to the user. We provide a similarity metric for all combinations of operators executed iteratively, except the GR operator combined with DC and AI. In the future, we intend to extend our approach and develop a similarity function for the remaining cases that are not covered yet. We also plan to evaluate our complete approach on a benchmark dataset.

## References


1. Chu, W.W., Yang, H., Chiang, K., Minock, M., Chow, G., Larson, C.: Cobase: A scalable and extensible cooperative information system. Journal of Intelligent Information Systems **6** (1996) 223–259 10.1007/BF00122129.
2. Halder, R., Cortesi, A.: Cooperative query answering by abstract interpretation. In: Proceedings of the 37th international conference on Current trends in theory and practice of computer science. SOFSEM'11, Berlin, Heidelberg, Springer-Verlag (2011) 284–296
3. Shin, M.K., Huh, S.Y., Lee, W.: Providing ranked cooperative query answers using the metricized knowledge abstraction hierarchy. Expert Systems with Applications **32**(2) (2007) 469–484
4. Pivert, O., Jaudoin, H., Brando, C., HadjAli, A.: A method based on query caching and predicate substitution for the treatment of failing database queries. In: ICCBR'10. LNCS, Springer (2010) 436–450
5. Gaasterland, T., Godfrey, P., Minker, J.: Relaxation as a platform for cooperative answering. Journal of Intelligent Information Systems **1**(3) (December 1992) 293–321
6. Inoue, K., Wiese, L.: Generalizing conjunctive queries for informative answers. In: Proceedings of the 9th International Conference on Flexible Query Answering Systems. Lecture Notes in Artificial Intelligence, Springer-Verlag (2011)
7. Miller, G.A.: Wordnet: A lexical database for english. Communications of the ACM **38** (1995) 39–41
8. Fellbaum, C., ed.: WordNet: An Electronic Lexical Database (Language, Speech, and Communication). The MIT Press (1998)
9. Wu, Z., Palmer, M.: Verbs semantics and lexical selection. In: Proceedings of the 32nd annual meeting on Association for Computational Linguistics. ACL '94, Stroudsburg, PA, USA, Association for Computational Linguistics (1994) 133–138
10. Nabeshima, H., Iwanuma, K., Inoue, K., Ray, O.: Solar: An automated deduction system for consequence finding. AI Commun. **23** (April 2010) 183–203