Query Relaxation in AURA

Working Note 41

Peter Clark, Boeing Research & Technology

August 2010

Abstract

Query relaxation is a technique used in the database community to "broaden" a query to return more answers, in the event that no (or few) answers are returned from an initial search query. This working note discusses this technique in the context of AURA, how it might be applied, its potential relevance, and the results of a preliminary implementation and exploration of its use. Although relaxation appears to have some value when parts of question cannot be fully interpreted, in general it appears to have limited utility, primarily because AURA's task is question-answering (where zero answers may be completely valid) as opposed to search (where zero answers may be unhelpful to meeting the user's search goals). We provide illustrations and discussion of why this appears to be the case.

1. Introduction

Overview

Query relaxation is a technique used in the database community to "broaden" a query to return more answers, in the event that no (or few) answers are returned from an initial search query. This working note discusses this technique in the context of AURA, how it might be applied, its potential relevance, and the results of a preliminary implementation and exploration in AURA. We also discuss the distinction between relaxation and underspecification, another new technique used in AURA to improve question interpretation, and also discover, interestingly, that earlier AURA work previously described as "abductive answers" or "partial answers" can simply be viewed as query relaxation. Although relaxation appears to have some value when parts of question cannot be fully interpreted, in general it appears to have limited utility, primarily because AURA's task is question-answering (where zero answers may be completely valid) as opposed to search (where zero answers is unhelpful to meeting the user's search goals). We provide illustrations and discussion of why this appears to be the case.

Query Relaxation

The topic of query relaxation is most prominent in the database community, and informally, refers to the act of "broadening" or "generalizing" a query to retrieve a larger set of answers than those returned by the original query.[1] (We will give a formal characterization of this later). This is particularly useful if a user gets too few (e.g., zero) results to a query that he/she issues. A typical example in the database world (from [1]) is if I ask for flights to National airport, Washington DC, but get no results, a system might broaden the query to flights to any airport in Washington, DC, and as a result tell me (say) that although there are no flights to National, there are flights to Baltimore nearby -- useful information for someone wanting to make a trip. A

[1] Similarly, there is an inverse operation of query narrowing, useful if a query returns too many results. For this document, though, we will focus on relaxation, but bear in mind that there is a similar, inverse operation used in the database literature.

second example is if I do a Web shopping search for computers under $500, but get no results, a system might sensibly point out to me that there *are* some available for under $600, and list them.

It is important to note that the relaxed query is *not* the same as the original query, and hence the answer to the relaxed query is not the answer to original query. Because the original query returned too few answers to be useful (i.e., the condition typically used to trigger query relaxation), the system attempts to find a "more useful" or "neighboring" [2] alternative query by relaxing the original one to return more answers (i.e., a superset of the (possibly zero) original answers). Much of the work in the database literature on this topic thus concerns how to define and search this space of relaxed queries, including formally characterizing preference criteria to select the "best" or "most useful" relaxations. In particular, the notion of "best"/"most useful" is subjective and domain-specific, and hence DB techniques for relaxation focus quite heavily on mechanisms for encoding what this notion means in a particular domain-specific context. For example, when searching for flights it might be most useful to search for flights to nearby airports (rather than flights in a similar-sized airplane, for example), or when searching for products it might be most useful to search for products with a similar target price. Similarly, evaluation of a query relaxation mechanism is necessarily somewhat subjective, requiring some independent characterization of what "best/"most useful" means (e.g., from an end-user), and measuring the extent to which the relaxed query identifies such information from the database. For a more detailed survey and overview of some of the literature on query relaxation, the reader is referred to [4]; we will not repeat this survey here.

Relaxation vs. Underspecification

It is worth also distinguishing relaxation from underspecification, a technique used earlier this year in AURA. Underspecification is a syntactic technique for allowing alternative interpretations (e.g., of ambiguous text) to be carried around as a single data structure, rather than one data structure per interpretation. Although both underspecification and relaxation concern searching a space of queries, these two spaces are very different. Underspecification is a technique for dealing with ambiguity, and the space of queries it explores are possible, valid interpretations of an original English query. In contrast, relaxation is a technique for dealing with an uninformative answer to an original query, and searches a space of *alternative* queries one that provides more information for the user. Thus underspecification concerns finding the meaning of the *original* query, while relaxation concerns searching *alternative* queries. Relaxation makes no claim that the relaxed query is a valid interpretation of the original English question; rather, it claims that the relaxed query is a "helpful alternative" to the original. In this sense, relaxation and underspecification are quite different.

Is Query Relaxation Useful for AURA?

Relaxation is useful in many database applications (e.g., the earlier examples of flights or shopping). However, it is less clear how useful it is in the context of AURA. In the earlier database examples, the user is performing a search task where broadening and narrowing a query is clearly useful, while in AURA, the user is (typically) not performing search but simply wanting an answer to a question, and an answer of "no values" is perfectly adequate (if that is indeed the correct answer). Even if we did want to relax a query when (for example) the original returned no values, defining the space of "helpful alternative" queries is challenging. We describe some preliminary investigation of these issues shortly.

In the remainder of this document we first give a formal characterization of relaxing a query, taken from the database field. We then describe how it can be applied in the context of queries in AURA, and explore the use of three relaxation operators:

(i) dropping a qualifier from the query
(ii) restricting the type of a univerally quantified variable
(iii) generalizing the type of an existentially quantified variable.

We summarize the findings arising from these.

2. A Framework for Query Relaxation

Relaxation as Generalization

Gaasterland [2,3] provides a useful framework for relaxation that we adopt (with minor extensions) here. Relaxation involves rewriting the query to cover a greater space of answers than the original, i.e., the query is "more general" than the original. This can be formally captured as follows. First, a query can be characterized as a formula p(x) with free variable x, i.e., the answer to the query is the set:[2]

$\{ x \mid p(x) \}$

Relaxation can then be described as a generalization operation: Given a query p(x) (with free variable x) and a query q(x) (with free variable x):

q(x) is a generalization (relaxation) of p(x) **if** $\forall x\ p(x) \rightarrow q(x)$ (1)

(Note that p(x) and q(x) can be arbitrary formulae, not only single predicates). It thus follows that the answer to a relaxed query is necessarily a superset of the answer to the original query.

This can be extended straightforwardly to also apply to is-it-true queries. An is-it-true query can be characterized as a sentence p with no free variables, and the answer to the query can be similarly expressed in terms of sets (for consistency) by defining the answer to be the singleton set {Yes} if p is true, or the empty set {} if p is false, i.e.,

$\{ x \mid p \rightarrow x=\text{Yes} \}$

Thus Gaaasterland's semantics of relaxation can be applied in the same way: is-it-true query q (e.g., "Does Fred own a house?") is a relaxation of p (e.g., "Does Fred own a big house?") if $p \rightarrow q$. It follows that the set of answers to the relaxed query q is necessarily a superset of the answers to the original query p.

Given this definition, and a query with no additional variables in (i.e., no quantifiers), several syntactic operators are relaxation operators based on the rules of logic, for example:

1. Drop a literal from p(x)
2. Replace a literal with its implication

This gives a straightforward, formal notion of relaxation. The main challenge, then, is to specify which operators should be applied, and with what preference.

Relaxation for Queries with Quantifiers

The earlier database queries we considered were queries for individuals in the database (e.g., a flight, a product). In contrast, AURA is primarily concerned with classes (types), resulting in queries containing quantifiers. For example:

"What organelles are part of cells?"
$\{ c \mid \forall x\ \text{isa}(x,\text{Cell}) \rightarrow \exists y\ \text{isa}(y,c) \wedge \text{is-subclass-of}(c,\text{Organelle}) \wedge \text{has-part}(x,y) \}$

[2] This can be naturally extended for queries for parts, lists, etc., but we will not do so here.

Again following Gaasterland, we can identify some syntactic relaxation operators for queries with this quantification pattern:

1. Drop a clause in the consequent, e.g.,

   "What are a part of cells?"
   { c | $\forall$x isa(x,Cell) $\rightarrow$ $\exists$y isa(y,c) $\wedge$ has-part(x,y) }

2. Specialize a class restriction in the antecedent, e.g.,

   "*What organelles are part of eukaryotic cells?"*
   { c | $\forall$x isa(x,Eukaryotic-Cell) $\rightarrow$
   $\exists$y isa(y,c) $\wedge$ is-subclass-of(c,Organelle) $\wedge$ has-part(x,y) }

3. Generalize a class restriction in the consequent, e.g.,

   "What subcellular entities are part of cells?"
   { c | $\forall$x isa(x,Cell) $\rightarrow$
   $\exists$y isa(y,c) $\wedge$ is-subclass-of(c,Subcellular-Entity) $\wedge$ has-part(x,y) }.

Again these are valid relaxation operators as, by the inference rules of predicate calculus, their application satisfies (1). Also note that this is not an exclusive list of possible operators, but instead a initial starting point of the most obvious ones. We investigate their use later.

The second operator above (specialize a class in the antecedent) is particularly interesting for two reasons. First, despite the fact it is specializing (not generalizing) a class, it is still a relaxation operation because the transformation in 2 satisfies (1), i.e., the answers to the query in 2 are necessarily a superset of the answers to the original query. Second, operator 2 is exactly the operation that we were previously describing as a "partial query" or an "abductive query" in AURA (e.g., “Do cells have a nucleus?” “No, but eukaryotic cells have a nucleus”) -- Gaarsterland's framework shows that in fact this transformation is simply a relaxation of the original query.

3. Relaxation in AURA

Introduction

The operators above represent some possible relaxation operators that might be applied to queries in AURA. Again drawing on the database literature, the space of possible relaxations is potentially large, and requires some domain-specific, heuristic design decisions about which parts of the space are worth searching and how to prioritize that search. For our purposes here, we have only made a preliminary exploration into this space - we only consider 3 relaxation operators (roughly corresponding to the first 3 above), and only consider them in isolation (we don't consider chaining them). Operators are applied if the original query returns a null answer, and preferred if the resulting relaxed query produces a non-null answer (all such relaxed queries and their answers are reported to the user).

Note that with any relaxation operation, the relaxed query is *not* the same as the original query. Thus the interface needs to clearly display to the user that the answers they see are not the answers to the original query, but to a modified query. Again, the degree to which users would find this helpful (or simply confusing) is open to debate in the context of AURA.

1. Dropping a Qualifier

Previously if AURA encounters an unrecognized noun modifier (typically an adjective or another noun) during question interpretation, the query would contain a vacuous (unknown) concept and the query would fail. With relaxation, however, AURA drops that qualifier from the query, the new query corresponding to a relaxation of the original as a restriction on part of the query expression has been removed. For example:

Q. Do prokaryotic cells have a rigid cell wall?

can be relaxed to:

Q. Do prokaryotic cells have a cell wall?

if "rigid" is not a known word. The resulting answer is not necessarily an answer to the original question, but may still be useful information to the user. In this situation, AURA warns the user that the original query has been modified:

```
Q. Do prokaryotic cells have a rigid cell wall?
A. I don't know (I don't know what "rigid" means).

However, you might find the following information of interest:
Q’: Do prokaryotic cells have a cell wall?
A’: Yes.
```

This mechanism is built into the current AURA system. Of course, it is clearly better if such modifiers were recognized by AURA in the first place, but given a failure, trivially relaxing the query to drop the unknown word is a possible way of handling it.

2. Specializing the Class in the Antecedent

The second relaxation operator explored is the one previously mislabeled as creating an "abductive" or "partial" query, namely specializing the universally quantified class in the question. For example:

Q: What organelles are in a cell?

can be relaxed to

Q: What organelles are in a eukaryotic cell?

and similarly:

Q: Do cells have a nucleus?

can be relaxed to

Q: Do eukyarotic cells have a nucleus?

(These are relaxation as the answers to the relaxed queries are necessarily a superset of the answers to the original queries, as discussed earlier in Section 2). Again, the results need to be presented carefully to the user, for example in the current system (when this mechanism is switched on), AURA behaves as follows:

```
Q. Do cells have a nucleus?
A. No (not always).

However, you might find the following information of interest:
Q’. Do eukaryotic cells have a nucleus?
A’. Yes.
```

There may be multiple ways that the universally quantified class can be specialized like this. The current implementation does a general-to-specific search until it finds class(es) that produce a non-NIL answer to the query.

3. Generalizing a Class in the Consequent

The third relaxation operator explored is to generalize a class in the consequent. For example,

Q: What organelles are part of cells?

$\{ c \mid \forall x \; isa(x,Cell) \rightarrow \exists y \; isa(y,c) \wedge is\text{-}subclass\text{-}of(c,Organelle) \wedge has\text{-}part(x,y) \}$

can be relaxed to

Q: What subcellular entities are part of cells?

$\{ c \mid \forall x \; isa(x,Cell) \rightarrow \exists y \; isa(y,c) \wedge is\text{-}subclass\text{-}of(c,Subcellular\text{-}Entity) \wedge has\text{-}part(x,y) \}$

To avoid generalizing to vacuous concepts, e.g.,

Q: What things are part of things?

the operator only generalizes a class to it's direct superclass, rather than all superclasses.

The presentation in AURA is as follows:

```
Q. What organelles are part of cells?
A. No values found.

However, you might find the following information of interest:
Q'. What subcellular entities are part of a cell?
A'. Chromosome Cytoplasm Plasma-membrane
```

As discussed earlier, these relaxation operators are some initial starting points, rather than a complete list, and we have not considered chaining such operators. The current implementation only considers relaxations if a null answer is returned by the original query, and uses a simple "prioritization" mechanism of preferring and reporting (all) relaxations that produce a non-null answer. By default, operator 1 is switched on in the main AURA build and operators 2 and 3 are switched off, for reasons that we now describe.

4. Empirical Investigation

We now describe some preliminary empirical investigation of these relaxation operators. Investigations were done on the current NewQF test suite, consisting of a mixture of a subset of the Intermediate Evaluation (IE) questions, Refinement Evaluation (RE) questions, NewQF test questions, and additional tests. Note that these questions are not questions "in the wild", but are (mostly) questions that have been authored and debugged to work with the reference KB. Despite this bias, they provide some useful feedback on using relaxation.

Of the 308 questions, 38 correctly result in a null ("no values"/"no") answer, and 16 cannot be answered as they include words not recognized by the language interpreter (i.e., part of the query cannot be mapped to AURA's ontology). These 54 are of interest, as it is only for these questions that query relaxation is tried (relaxation is triggered by a null ("no values"/"no") answer). For these 54, AURA attempts query relaxation to find a "neighboring" query that has a non-null answer, and present it (or them, if more than one is found) to the user.

As emphasized earlier, a relaxed query is not the same as the original query, and the answer to the relaxed query has a different purpose to that of the original answer: While the original answer is meant to provide a deductively correct response to the original question, the answer to the relaxed query is meant to provide "additional useful/neighboring information" for the user. As mentioned earlier, evaluating whether this has been achieved, and whether it is even desirable, is somewhat subjective. For our initial purposes here, we show the data produced by these mechanisms, and offer some initial thoughts on its value.

1. Dropping a Qualifier

Of the 308 questions, 16 contain an unknown word and thus AURA is unable to answer them without additional processing. For 14 of these 16, the unknown word is a adjective/adverb modifier, allowing AURA to relax the query simply by dropping that word and try the relaxed query. These 14 are as follows (the modifier that was dropped shown in bold):

5. Is the nucleus **closely** associated with the transcription of genes?
6. Is the nucleus **closely** associated with the transcription of DNA?
7. Does a prokaryotic cell have a **rigid** cell wall?
59. The **actual** separation of the chromosomes during mitosis occurs in what event?
60. What event is the **actual** separation of the chromosomes during mitosis?
64. What is the **thread-like** material, containing DNA and protein, that comes together during mitosis?
65. Some **thread-like** material comes together during mitosis. The material contains DNA and protein. What is the material?
66. Some **thread-like** material containing DNA and protein comes together during mitosis. What is the material?
2a. Is it true that a prokaryotic cell has a **semi-fluid** region consisting of cytosol?
2e. is it true that a prokaryotic cell has a **rigid** cell wall?
5abc-2. Is it true that mitochondria can provide **cellular** energy?
37a. Is it true that a eukaryotic cell has a **membrane-bound** nucleus?
74c. Is it true that in DNA replication, the DNA polymerase builds a **new** strand from the 5 prime end to the 3 prime end?
70. What is the **thread-like** material containing DNA and protein that assembles during mitosis?

Thus for these 14, although AURA is unable to answer the original question, it can answer the relaxed question (without the modifier). For some queries, this is clearly helpful (namely when the modifier is largely superfluous), e.g.,:

```
Q. The actual separation of the chromosomes during mitosis
      occurs in what event?
B. I don't know (I don't know what "actual" means).

However, you might find the following information of interest:
Q': The separation of the chromosomes during mitosis
      occurs in what event?
A': Anaphase.
```

For other queries, its utility is a little more case-specific, depending on whether the dropped modifier included important information. For example, for this query:

```
Q. Does a prokaryotic cell have a rigid cell wall?
A. I don't know (I don't know what "rigid" means).

However, you might find the following information of interest:
Q'. Does a prokaryotic cell have a cell wall?
A'. Yes.
```

AURA relaxed the query by dropping "rigid", but it is unclear whether the relaxed query and its answer is useful to the user -- it depends whether "rigid" is critical or incidental to the question. By presenting the relaxed query and its answer, however, the user has the choice of whether to use the additional information or not, depending on whether the relaxation dropped something critical to the question.

2. Specializing the Class in the Antecedent

Of the 308 questions, 38 correctly have a null answer. For these 38, AURA also searches relaxed versions of the query generated by specializing the class in the query antecedent, looking for one producing a non-null answer, in an attempt to find useful, neighboring information to also show the user. Of these 38, a relaxed query producing a non-null answer was found for 8 of them (7 were true/false questions, 1 was a find-the-value question), as follows:

23. Does a cell contain chloroplasts?
37. Is a chromosome the recipient of a transfer?
40. A cell has a chloroplast. Is the cell a eukaryotic cell?
44. Is it true that an organelle with a photosynthetic pigment is a chloroplast?
36e. Is it true that a lysosome is inside a membrane?
201. Do organelles provide energy?
202. What fluids are parts of cells?

In these 8 cases, with this relaxation operator switched on, AURA is able to offer an answer to an alternative, relaxed query after the original query produces a "no values"/"null" answer. For example:

```
Q. 23. Does a cell contain chloroplasts?
A. No.

However, you might find the following information of interest:
Q'. Does a plant cell contain chloroplasts?
A'. Yes.
```

and

```
Q. 202. What fluids are part of cells?
A. No answers found.

However, you might find the following information of interest:
Q'. What fluids are part of eukaryotic cells?
A'. Cytosol.
```

This operator was used in AURA for several months earlier this year. However, the informal report from users was that these additional answers were more confusing than helpful -- users were not sure what to do with this additional information, or how to score it when evaluating AURA's performance. Note that, unlike in the database search context, a "No"/"No answers found" answer by AURA to the original question is in general completely acceptable (provided that this is indeed the correct answer). However, finding and displaying a relaxed query with its (non-null) answer may possibly be useful in some situations, for example if the user accidentally asked about a more general concept than he/she intended (e.g., wrote "cell", but was thinking "plant cell"). In practice, though, this situation has not occurred very often. Given the sometimes confusing nature of the above relaxed queries and answers, we currently have disabled this relaxation operator in AURA.

3. Generalizing a Class in the Consequent

Finally we explored relaxation by generalizing a class in the consequent of the query expression. With this operator enabled, of the 38 questions with null answers, a relaxed query producing a non-null answer was found for 20 of them. Unfortunately, though, the relaxed queries typically seem of low utility, for example:

```
Q. 77-c-2. Is it true that the plant cell has a cell plate?
A. No.

However, you might find the following information of interest:
Q'. Is it true that the plant cell has a subcellular entity?
A'. Yes, a plant cell has mitochondria.
```

In this example, the relaxed query generalizes "cell plate" to its immediate superclass ("subcellular entity"), which then produces a non-null answer ("yes") for mitochondria (a subclass of subcellular entity). Although logically correct, the relaxed query is substantially different to the original one, and does not appear particularly useful concerning the original query. Another example is:

```
Q. 93. What organelles are part of a nucleus?
A. No values found.

However, you might find the following information of interest:
Q'. What subcellular entities are part of a nucleus?
A. Nucleolus. (A nucleolus is a subcellular entity).
```

Again organelle has been generalized to subcellular entity, and an answer (nucleolus) to the generalized query found. As with the previous example, it is not clear that such additional information is particularly useful.

5. Summary and Conclusions

Relaxation is a method for generalizing a query such that more answers are returned. However, although it has been found useful in the context of search/retrieval with databases where no/few answers is undesirable, it appears less relevant in the context of question-answering where no/few answers is a completely acceptable result. The original motivation for looking at relaxation for AURA was the use case of a query failing because an element in the query should have been understood, but was not -- either due to a missing word-to-concept mapping or missing knowledge in the KB -- and the relaxed query was considered potentially helpful as it showed the alternative result if that mismatched element was ignored. This particular use case, operator 1 ("Dropping a Qualifier") above, does indeed appear potentially useful in these cases, providing that the user carefully inspects the alternative query and answer to check the dropped element was not a critical component of the original question. The two other operators explored, however, based on relaxation by specializing/generalizing concept, appear to have less obvious utility. Although the relaxed query may be answered correctly, the answer's relevance to the original question is often unclear, in particular when the null answer to the original question is correct. In some cases the relaxed query can illuminate special cases of the original query (e.g., Q. "Do cells contain a nucleus?" A. "No, but eukaryotic cells contain a nucleus."), but in other cases the relaxed query can seem strange, as it is answering a different question to the one originally asked. The overall conclusion from this exploration, then, is that the first operator, for relaxation by dropping qualifiers, should be retained in AURA, but the other two operators should be disabled for now, to be resurrected if/when AURA expands to support search-style tasks in addition to its current Q/A capabilities.

References

[1] W. Kiebling. Foundations of Preferences in Database Systems. Tech Report 2001-8, October 2001, Univ Augsburg.
[2] T. Gaasterland. Cooperative answering through controlled query relaxation. IEEE Expert, 1997.
[3] T. Gaasterland, P. Godfrey, J. Minker. Relaxation as a platform for cooperative answering. Journal of Intelligent Information Systems, 1992 - Springer.
[4] P. Clark. A Brief Overview of Some Datapoints in the Query Relaxation Literature. Working Note 40. http://www.cs.utexas.edu/users/pclark/working_notes/ 2009.