Question-Answering Based on Virtually Integrated Lexical Knowledge Base

Abstract

This paper proposes an algorithm for causality inference based on a set of lexical knowledge bases that contain information about such items as event role, is-a hierarchy, relevant relation, antonymy, and other features. These lexical knowledge bases have mainly made use of lexical features and symbols in HowNet. Several types of questions are experimented to test the effectiveness of the algorithm here proposed. Particularly in this paper, the question form of “why” is dealt with to show how causality inference works.

1 Introduction

A virtually linked knowledge base is designed to utilize a pre-constructed knowledge base in a dynamic mode when it is in actual use.

An open-domain question answering architecture must consist of various components and processes (Pasca, 2001) that include WordNet-like resources, part of speech tagging, parsing, named entity recognition, question processing, passage retrieval, answer extraction, and answer justification. Consider a question like the following: “Why do doctors cure patients?”

The answer may be obtained by commonsense knowledge as follows:

1. A patient suffered from a disease.
2. A doctor cures the disease.
3. The doctor cures at hospital.
4. Doctor is an occupation.
5. So the doctor cures the patient.

These sentences are transformed into propositional forms, as illustrated below:

6. sufferFrom(patient,disease)
7. cure(doctor,disease)
8. cure(doctor,at-hospital)
9. occupation(doctor)
10. cure(doctor,patient)

Linguistic knowledge bases like WordNet (Miller, 1995), EDR dictionary (Yokoi, 1995) and HowNet (Dong, 1999) have been used to interpret these sentences.

Moldovan et al. (2002) generated lexical chains from WordNet in order to trace these topically related paths and thereby to search for causal explanations. A conceptual word $C_j$ inside of a gloss under a synset $C_i$ is linked to the synset $C_j$.

HowNet (Dong et al. 1999) is a linguistic knowledge base that is designed to have the definition of words and concepts as well as event role and role-filling entities. Commonsense knowledge like naive physics is also built up through event role relation like the relation of sufferFrom requiring cure.

HowNet is modularized into separate knowledge spaces for entity hierarchy, event hierarchy, antonymy, syntax, attributes, etc. Relations between various concepts (e.g., part-of, relevance, location) are defined implicitly in the definition of each concept.

This paper will focus on building an algorithm that allows for searching for some topical paths in order to find causal explanations for questions like “Why do doctors cure patients?” or “Why do patients pay money?” as illustrated in Figure 1.
In the following sections, issues on the virtual integration of knowledge bases, their algorithms and experimentations are presented.

2 Underlined Knowledge Bases and Virtual Integration

In Figure 1, each marked numbering has the following meaning:

1. Entity hierarchy: entity is the top node in the hierarchy of entities.

2. entity is the hypernym of patient, doctor, occupation, and money in the line (3).

3. Concepts or word entries are listed in this line. All concepts and word entries represent their definition by a list of concepts and marked pointers.

4. A concept (or word) in (3) features definitional relations to a list of concepts. For example, a doctor definition is composed of two concepts and their marking pointers: #occupation and *cure. Pointers in HowNet represent relations between two concepts or word entries, e.g., “#” means “relevant” and “*” does “agent”.

5. syn refers to the syntactic relation in the question “Why do patients pay money to doctors?”

6. converse refers to the converse relation between events, e.g., give and take.

7. Event hierarchy: For example, the hypernym for pay is give and the hypernym of give is event.

8. Event role: Now, event roles are partially filled with entities, e.g., patient and money.

9. Event role shift: The agent of give is equalized to the source of take.

An overview of each component of the knowledge base is in Figure 2, where three word entries why, patient, and money are in the dictionary. The four concept facets of entity, role, event, and converse are described in this example, mainly as part of linguistic knowledge.

Some issues on ontology integration have been discussed from various points of view. Pinto et al. (1999) classified the notions of ontology integration into three types: integration, merging and use/application. The term virtually integrated means the view of ontology-based use/application.

This paper presents issues on and arguments for linguistic knowledge base and commonsense knowledge in (Lenat, Miller and Yokoi, 1995). One of the arguments was whether linguistic knowledge could be separated from commonsense knowledge, but it was agreed that both types of knowledge were essentially required for natural language processing.

This paper was motivated by the desire to make inferences using a lexical knowledge base, thus successfully carrying out a kind of commonsense reasoning.
3 Interpretation of Lexical Knowledge

Consider the following three sentences:

11. Doctors cure patients.
12. Doctors earn money.
13. Patients pay money.

One major concern is finding connectability among words and concepts. As shown in Figure 2, the following facts are derived:

14. Doctor is relevant to occupation.
15. Occupation allows you to earn money.

Because there exists a converse relation between give and take, their hyponyms earn and pay also fall under converse relation. It is something like the following social commonsense as shown in Figure 2: “If someone X pays money to the other Y, Y earns money from X.”

We humans now understand the reason for “why patients pay money.” The answer is that “doctors cure patients as their occupation allowing them to earn money.”

The following is a valid syllogism where Y is being instantiated to doctor:

If “X pays money to Y” is equivalent to “Y earns money from X”, and “a doctor earns money from X”, then “X pays money to the doctor.”

Consider the next syllogism: If “a doctor cures X” and “doctor is an occupation” and Axiom 1, then “the doctor earns money from X.”

Axiom 1 is needed to make such a syllogism that “If Y cures X and Y is an occupation, then Y earns money from X.” Then our challenge is to find out this Axiom 1 from the lexical knowledge bases. It is a commonsense and thus there is a gap in the lexical knowledge base.

The following is a list of questions derived from the three sentences of 11, 12 and 13 which are designed to discover such axioms (or rules) from a set of lexical knowledge bases: “Why do doctors cure patient?”, “Why do doctors earn money?”, and “Why do patients pay money to doctors?”

4 Connectability: Similarity Measure

Consider the query “Why do doctors cure patients?” Tracing Figure 2 back through Figure 1 leads to obtaining logical forms from 6 through 10. The best connectable path is planned from the first word of the question.

For each pair of words, the function called similar(*,*) will be estimated to choose the next best tracing concepts (or words). similar’s missions are summarized as (1) checking the connectability between two nodes, (2) selecting the best sense of the node, (3) selecting the best tracing candidate node in the next step. Finally, following the guidance by similar allows us to explain the question.

4.1 Observation and Evidence of Topical Relatedness

Let’s try to follow the steps 6-10 given in the logical forms. In the question “Why do doctors cure patients?” that focuses on three words doctor, cure, and patient, we can trace some key words given in example sentences as follows: patient ~ disease ~ cure ~ doctor ~ occupation ~ earn ~ pay ~ patient.

What kind of lexical relations are relevant to each pair of words or concepts? Their observation can be summarized as follows:

A) The relation between patient ~ disease is a role relation of “sufferFrom(patient, disease)”.

B) A sequence of cure ~ doctor ~ occupation ~ earn lets us infer the relation among cure ~ earn, which are closely linked by their relevance relation to occupation. Furthermore, earn and cure shares a common subject of these two events.

C) The sequence of earn ~ patient is the result of a converse event relation between earn and pay.

D) pay ~ patient: The agent of pay is a generic human. In other words, pay is a hy-

---

1 It is a converse relation.
2 A node means either concept or word.
3 It is similar with word sense disambiguation.
ponym for the act of human, one of whose hyponym is patient.

Consider again the match between the tracing sequences of concepts and the knowledge base. Going into more details, notations with footnotes will be given to each example. At this point, we will give names and formalization based on the observed characteristics.

A) Feature comparison: To find the role relation among patient ~ disease, search the definition of entities (referring to patient and disease) in ways that two entities share the same event concept (referring to cure).\(^4\)

\[
\text{patient} \supset \text{human} \wedge \text{cure} \wedge \text{sufferFrom}.
\]

\[
\text{disease} \supset \text{medical} \wedge \text{cure} \wedge \text{undesired}.
\]

B) Interrelation: To find the event interrelation among cure ~ earn, two possible paths are presented as follows.

- First, inverse interrelation: Two event's role entities can be found by searching all of entities using *earn ~ *cure that share the same subject, and using *earn ~ Scure where the subject of earn is the object of cure.

- Second, sister interrelation: The following logical form can be derived from Figure 2.\(^5\)

\[
\text{doctor} \supset *\text{cure} \wedge \#\text{occupation}.
\]

Because cure and occupation is in the definition of doctor, a probable logical implication can be derived as follows.\(^6\)

\[
*\text{cure} \supset \sim \#\text{occupation}.
\]

C) Converse/antonymy: earn and pay have their respective hypernyms take and give. There exists a converse relation between these two hypernyms.

D) Inheritance: The relation among pay ~ patient is represented as follows.\(^7\)

\[
\text{pay} \prec \text{act} \\
\text{human} \Rightarrow *\text{act} \\
\text{patient} \prec \text{human}
\]

4.2 Rationale of Connectability

In the former section, we summarized four characteristics\(^8\) of causality (relatedness)-based path finding: feature comparison, interrelation, converse/antonymy in their hypernym's level, and inheritance. Among search spaces available, it is necessary to find out a measure of guiding the optimal\(^9\) path tracing.

We will call such a measure similar which will be defined according to the four characteristics just mentioned. Further details about the calculation formula will be presented again later.

A) For “feature comparison”, the measure feature similar(X,Y) defines the notion of similarity between the features in X and Y.

B) There are two interrelations in the last section.

- For “inverse interrelation”, inverse similar(X,Y) calculates how much similarity exists between X and Y in a manner that X = {Z | Z \(\subset\) \(\theta\) X}, where \(\theta\) X is an abstraction of role-marked concepts like *X, $X, #X, etc. Thus inverse similar(X,Y) = similar(X0,Y0).

- For “sister interrelation”, the measure sister similar(X,Y) means the following two situations: First, X and Y are features to define one concept (say, W). Second, one of them, say, Y’s definitional feature concepts (referring to Z) are similar with X such that X and Z are similar if W \(\supset\) X \(\wedge\) Y and Y \(\supset\) Z.

C) Converse or antonymy: The converse relation converse(X,Y) can be found by the measure feature similar. converse(X,Y) is formulated by \(X \subset \theta Y\) and \(Y \subset \theta X\) where \(\theta = \text{converse}\).

4 According to HowNet convention. “$” represents patient, target, possession, or content of an event, and “*” represents agent, experiencer, or instrument. “\(\supset\)” means implies or has features.

5 “#” means “relevant”.

6 “\(\sim\)” means “very probable”.

7 “\(X \prec Y\)” means “Y is hypernym of X”.

8 Their exhaustiveness should be discussed later.

9 “optimal” will not be discussed.
D) Using **inheritance** property in the concept hierarchy, relations between hypernym of concepts X and Y are inherited to X and Y in a way that X and Y is similar if there exist X’ and Z such that X ⊆ X’, Z ⊇ 0X’, and Y ⊆ Z. This inheritance tracing can be determined by how much similar X and Y are in terms of their path upward based on the relation of hypernym. We will define **path similar**. But tracing the path upward following hypernym links is to be described later according to the algorithm.

A measure called **similar** will be defined based on the discussion in this section. Then an algorithm is introduced through this measure with an example.

### 5 Measures

In the last section, we discussed four kinds of the measure **similar**:
- path similar,
- feature similar,
- inverse similar,
- sister similar.

For **feature**, **inverse**, and **sister similar** functions, **path similar** is used as a basis of calculation. They are different with respect to both their search method and the depth of expanding features. **Feature similar** finds similar features by using **path similar**. inverse similar(X,Y) searches for entries that contain X and Y as features and then use the **path similar**. In the same way, **sister similar** finds sister concepts, expands them, and finally measures using the **path similar**.

Since **path similar** plays a key role in all these search and measure processes, its role will be explained in the next subsection. Other measures are only dealt with as part of the algorithm.

#### 5.1 Similarity Based on Hierarchy and Feature

The mission of the measuring function similar(X,Y) is to calculate their relevancy between two concepts or words whether they are of type entity, event, or of some other type.

If X and Y belong to different types of knowledge plane (e.g., entity and event), it is hard to compare their hypernym path upward to the top concept. However, if different types of concepts have any relevance to (connect) causality, we will use **feature similar** or **inverse similar** after finding the same type of concepts to calculate the **path similar**. Now we will explain the above by using two pairs of concept type: entity-entity and entity-event, without loss of generality.

First, pathsimilar(entity X, entity Y) is defined as follows:

\[
\text{pathsimilar}(X,Y) = \frac{2 \times |\text{path}'(X) \cap \text{path}'(Y)|}{|\text{path}'(X)| + |\text{path}'(Y)|}
\]

where |path'(X)| counts the number of nodes on the path.

Second, pathsimilar(entity N, event V) is defined as follows:

\[
\text{pathsimilar}(N,V) = \text{Max pathsimilar}(N\text{.feature}, V)
\]

where N.feature means the feature list in the definition of N. The following is an illustrative example for the definition:

\[
\text{money} \supset \text{$earn$,*buy,$sell$, $setAside$},
\]

it is equivalent to the following:

\[
\text{money.feature}=[\text{$earn$,*buy,$sell$, $setAside$}].
\]

So pathsimilar(money,earn)=pathsimilar(earn,earn) = 1. According to this Max function, the selection priorities for the path can be specified.

Third, pathsimilar(event V, entity N) is defined by **inverse similar** as follows: pathsimilar(V,N) = Max pathsimilar(V.inverse , N). For example,

\[
\text{pathsimilar}(|\text{cure.inverse}, \text{doctor}|) = \text{Max pathsimilar}(|\text{doctor}, \text{medical worker, medicine, patient}|, \text{doctor}).
\]

Fourth, pathsimilar(event X, event Y) shares the same formula with pathsimilar(entity X, en-
tity Y) shown before. But, we can give another 

inverse pathsimilar(event X, event Y) = Max 

pathsimilar(X.inverse, Y.inverse).

5.2 Logical Implication and Expansion Depth

All of the relations in Figure 2 are translated into 

logical form (see below). As shown in “Interpreta-

tion as Abduction” (Hobbs et al. 1988), “abductive 

inference is inference to the best explanation”. 

These relations showed “the interpretation of a text 

is the minimal explanation of why the text would 

be true” based on the abductive inference. By the 

same token, “the interpretation of a question is the 

minimal explanation of why the question would be 

true” based on a set of lexical knowledge bases.

Before proceeding to our algorithm, an example 

will be applied to abductive inference briefly as a 

set of logical forms as well as a diagram in Figure 

3.

16. doctor ⊃ human, #occupation, *cure, medical. 
17. medicine ⊃ *cure. 
18. disease ⊃ $cure. 
19. cure ⊃ medical, 

{agent, patient, content}. 
20. medical ⊃ #cure. 
21. converse(pay, earn) ⊃ 

agent=source, 

target=agent. 
22. patient ⊃ human, $cure. 
23. occupation ⊃ affairs, earn. 
24. cause(cure, sufferFrom) ⊃ 

patient=experiencer, 

content=content. 
25. possibleConsequence(cure, beRecovered) ⊃ 

patient=experiencer, 

content=stateIni.

While pursuing the path tracing enabling mini-

mal explanation, now we are going to propose 

a connectability measure similar such as 

“weighted abduction” (Hobbs et al. 1988). As 

“likelihood estimation” is useful to consider a 

“bounded conditioning” (Russell & Norvig, 1995) 

in a belief network, the “expansion depth” of simi-

lar will be useful for the explanation path tracing 

for the purpose of the minimal explanation of the 

question.

Figure 3: Virtual Linking for Causality

The “expansion depth level” of similar has two 

kinds of utilities: one is to find the minimal expla-

nation, and the other is to be dynamically adapt-

able to the level of interaction. This level of 

similar is defined as a function similar(Level)(X,Y) for X and Y, concepts or words in the following manner:

- similar(0)=pathsimilar: they use only them-


selves and their hypernym path from X and 

Y.

- similar(1)=feature_similar: they use their 

features that are expanded one more than 

similar(0).

- similar(2)=inverse_similar

- similar(3)=sister_similar 

=inverse_similar • feature_similar.

Depending on what level of similar is chosen, 

the search paths may be changed. A snapshot up to 
similar(2) is given in Figure 4.

Figure 4: Snapshot for similar(2).
6 Tracing Algorithms

6.1 Algorithm Crossover
The overall algorithm flow depends on similar(Level) as in the next program.

**Algorithm Crossover**
For Level=0...N until stopping condition is satisfied:
Expand the trace by similar(Level)

For example, when Level=1, the algorithm crossover finds a very primitive answer to the question "Why do doctors cure patients?" We will expand other features of doctor except for cure because cure has a syntactic relation between doctor and patient.

As shown in the logical forms (16~24) introduced in the previous section, this algorithm in Level=1 can find the following concepts as a result: medical, human, cure (Scure, *cure).

When Level=2, the algorithm crossover will seek higher-order relations (like the hypothesis) from the concept (by inverse_similar), converse/antonymy relations (by feature_similar), and event relations (if any, for use in knowing the cause or consequence relation). Consider again our example "Why do doctors cure patients?" by using the previous section's logical forms. The results are as follows:

* cure = {doctor, medicine}
$ cure = {patient, disease}
* sufferFrom = {patient}
$ sufferFrom = {disease}

Its generated meaning may be "If a doctor cures a patient, the patient is recovered from disease. Because patients suffer from diseases, doctors cure the patients. Patients are recovered after getting cured."

6.2 Stopping Condition
Stopping conditions for the algorithm crossover are as follows:
1. Event roles are filled up.
2. If no event is found in the feature definition, increase similar level.
3. [weak stopping condition] When there is no event, one of the other features is commonly shared between two concepts. For example, medical is a common feature between doctor and cure.

6.3 Hypernym Climbing
In section 4.2, inheritance was discussed for the purpose of finding a relation among pay ~ patient. After trying to make Level=2 in section 5.2, we have been motivated to find the interrelation between hypernyms. The algorithm crossover is updated.

**Algorithm Crossover+**
For Level=0..N until stopping condition is satisfied:
Expand the trace by similar(Level)
If Level >= 2, then repeat climb up hypernym until it matches with the higher relation.

6.4 Algorithm Crossover++
Consider again the question "Why do patients pay money to doctors?" As shown in Figure 1, the best trace is Scure ~ *cure ~ *earn ~ Spay. It provides an explanation for the statement that "patients are cured by doctors ~ doctors earn money ~ patients pay money to doctors". This minimal explanation is observed by switching over the role pointers θ whenever tracing is performed. For example, Scure was switched over to *cure. This extended version of algorithm is called Crossover++.

7 Evaluation
By the algorithm Crossover’s, the behavior of "why"-type questions are investigated by extracting the answer paths as follows.

Q: Why does patient pay money?
Path: patient ~ Scure ~ doctor ~ #occupation ~ $earn ~ money

Q: Why does researcher read textbook?
Path: researcher ~ #knowledge ~ #information ~ readings ~ textbook

Paths between two concepts can now be found by simply checking the presence of a path among the concepts reached from an initial concept. Table
1 and Table 2 show examples of the number of paths as a function of path size.

| Source concept | Reached concepts path size |   |   |   |
|----------------|---------------------------|---|---|---|
|                | 1  | 2  | 3  |    |
| cure           | 275| 593| 24854|
| eat            | 268| 605| 24903|
| study          | 276| 358| 23172|
| food           | 532| 650| 18066|
| human          | 6713| 3686| 51171|
| money          | 328| 1312| 19827|

Table 1: Examples of destination concepts reached starting from one source concept

| Concept1 | Concept2 | Paths number length |   |   |   |
|-----------|----------|---------------------|---|---|---|
|           |          | 1       | 2    | 3    |    |
| cure      | human    | 0      | 78   | 26   |    |
| pay       | money    | 0      | 7    | 3    |    |
| human     | money    | 0      | 3    | 7    |    |
| food      | human    | 0      | 0    | 28   |    |
| read      | write    | 0      | 4    | 6    |    |
| earn      | pay      | 0      | 0    | 7    |    |

Table 2: The number of paths between pairs of concepts

8 Discussion

HowNet (Dong et al. 1999-2003) has already defined the words and concepts using the features of concepts. Each event role is also defined under the notion of feature. On the other hand, WordNet (Miller, 1995) consists of synsets and their glosses. Moldovan et al. (2002) showed a lexical chain to use words in glosses in order to trace the topically related paths.

Their search boundary is restricted to the shapes: V, W, VW, and WW. In this paper, crossover* is shown to be flexible and search for a more probable explanation.

9 Conclusion

In this paper, we have attempted to show how to link pre-existing lexical knowledge bases to one another. The major issue was to generate a path to give explanation paths for answering the “why”-type question. While observing the causality path behavior, we proposed the measure similar and also the algorithm crossover. It is compared with the “weighted abduction” (Hobbs et al. 1988) and “lexical chain” (Moldovan et al. 2002).

With the ability to provide explanations depending on the level of the measure similar, our proposed algorithm adapts itself to the user knowledge level and well as to the type of interactive questions to enable more detailed level of question-answering.

References

Zhen Dong and Q. Dong. 1999-2003. Hownet, http://www.keenage.com/

Jerry R. Hobbs, Mark Stickel, Douglas Appelt and Paul Martin. 1988. Interpretation as Abduction, Proceedings of the Conference on 26th Annual Meeting of the Association for Computational Linguistics.

Doug Lenat, George Miller, and Toshio Yokoi. 1995. CYC, WordNet, and EDR: Critiques and Responses, Communications of the ACM, 38(11):45-48.

Bernardo Magnini and Manuela Speranza. 2002. Merging Global and Specialized Linguistic Ontologies, Proceedings of Ontolex 2002 (Workshop held in conjunction with LREC-2002), Las Palmas.

George Miller. 1995. WordNet: a lexical database. Communications of the ACM, 38(11):39-41.

Dan Moldovan and Adrian Novischi. 2002. Lexical Chains for Question Answering, Proceedings of COLING 2002, Taipei.

Takanoa Ogino and Masahiro Kobayashi. 2000. Verb Patterns extracted from EDR Concept Description, IPSJ SIGNotes Natural Language Abstract, No.138 – 006:39-46.

Alexandru Marius Pasca. 2001. High-Performance, Open-Domain Question Answering from Large Text Collections. Ph.D Dissertation, Southern Methodist University.

H. Sofia Pinto, Asunciön Gómez-Pérez and João P. Martins. 1999. Some Issues on Ontology Integration, Proceedings of the IJCAI-99 workshop on Ontologies and Problem-Solving Methods (KRR5), Stockholm.

Stuart Russell and Peter Norvig. 1995. Artificial Intelligence: A Modern Approach. Prentice-Hall.

Toshio Yokoi. 1995. The EDR Electronic Dictionary. Communications of the ACM, 38(11).