Using default logic for lexical knowledge

Anthony Hunter

Department of Computer Science
University College London
Gower Street
London WC1E 6BT, UK

a.hunter@cs.ucl.ac.uk

Abstract. Lexical knowledge is knowledge about the morphology, grammar, and semantics of words. This knowledge is increasingly important in language engineering, and more generally in information retrieval, information filtering, intelligent agents and knowledge management. Here we present a framework, based on default logic, called Lexica, for capturing lexical knowledge. We show how we can use contextual information about a given word to identify relations such as synonyms, antinyms, specializations, and meronyms for the word. We also show how we can use machine learning techniques to facilitate engineering a Lexica knowledgebase.

1 Introduction

Lexical knowledge is knowledge about the semantics, morphology, and usage, of words. Handling words is central to many reasoning activities, and as a result lexical knowledge is increasingly important in language engineering, and more generally in information retrieval, information filtering, intelligent agents and knowledge management. Lexical knowledge can facilitate in the resolution of ambiguity in information.

For example, if we know that a newspaper article is about `oil`, it is usually reasonable to derive that it is about `petroleum`, with exceptions such as in contexts about `cooking`.

Lexical knowledge can also facilitate in the identification of synonyms, related terms, antinyms, and specializations for a word. It can also be used to identify meronymic relations, such as `engine` is `part-of` a `car`, and parts-of-speech such as relating actors with actions: For example, for the actor `terrorist` an appropriate action is `terrorism`.

In this paper, we briefly review the need for lexical knowledge — with particular emphasis on intelligent agents — and then show the need for a new approach to providing lexical knowledge. To address this need, we present a framework called Lexica for providing context-dependent lexical knowledge. The Lexica framework is based on default logic. We show how grammatical and semantic knowledge can be captured in Lexica. We also show how machine learning can be used to build Lexica knowledgebases.

1.1 An example of the need for lexical knowledge

Rapidly increasing amounts of information, particularly textual information, is being made available electronically, though the Internet, newsfeed, electronic databases, etc. This has created a pressing need to develop intelligent agents to search these sources for information that meets a user's needs.

Current search engines for the Internet (for example Yahoo, Lycos, and Alta Vista) use keywords given by the user to locate items that may be of interest to the user. Unfortunately, these search engines are limited in their ability to use background knowledge. They incorporate little knowledge on the meanings of keywords, morphology of words, nor on related terms for particular contexts. In particular, there is no way for users to provide background knowledge that could improve the precision and recall.

Statistical techniques in information retrieval and filtering offer some solutions for the Internet, but lack a systematic means for using background information, and are lacking in facilities for users to specify background knowledge. Statistical techniques are well suited to repositories of information where comprehensive statistical analyses can be undertaken (see for example [Cro93]), but are less well suited to heterogeneous distributed sources, such as the Internet, where the topics are so diverse, locally managed, and constantly evolving.

An increasingly important alternative is the approach of intelligent agents (for example [Mae94,GLC$^+$95,GF95,Lie95,BPK$^+$96,FJ96,Mou96]). Some incorporate a limited amount of background knowledge and allow users to input some background knowledge. However, they lack a systematic means for using large amounts of complex background knowledge. This raises the need to incorporate default knowledgebases as a repository for background knowledge. Using explicit defaults offers a lucid representation for users, and it aids maintainability and incrementality.

1.2 Lexicons

What is a lexicon? According to Trask [Tra93], a lexicon within the study of grammar has traditionally been used as a repository of miscellaneous facts, with little in the way of generalizations. This view has shifted and recent theories of language are using lexicons for significant proportions of linguistic knowledge [Bri91]. A wide variety of machine-readable lexicons have been developed (for reviews see [WSG96,GPWS96]), though many are oriented to specific approaches or tasks, such as EDR [Yok95] and Acquilex [Bri91] for machine translation.

Yet there is a need to general purpose information about words, such as for intelligent agents, information retrieval, information filtering, and information extraction. Perhaps the most significant example of such a general purpose system is WordNet [BFGM91,Mil95]. This is a semantic network containing lexical knowledge on over 90,000 word senses and it is now found to be an increasingly important resource on synonyms, generalizations, and specializations of words, for applications in information systems. In WordNet, each set of words that are strict synonyms (i.e. the words can be interchanged in a sentence) is called a synset. The following is an example of a synset.

{`Molotov-cocktail, petrol-bomb, gasoline-bomb`}

Whilst WordNet separates different meanings of the same word by putting the same word in more than one synset, there is no explicit machinery for determining in which context a particular wordsense should be used. Moreover, there is no logical reasoning with the relations in the semantic network.

Another kind of problem with WordNet is that the knowledge is very general. WordNet has been applied to information retrieval [Voo94] and information filtering [Eng96] and in these studies the utility of WordNet was limited by this generality. Furthermore, WordNet was limited by the inability of the user to be able to add context-dependent knowledge appropriate for the application domain.

1.3 Lexical knowledgebases

In order to build on the success of systems such as WordNet, and address some shortcomings, we need to develop more sophisticated systems that incorporate richer, more complex, knowledge about words. Essentially, we require knowledgebases containing structured knowledge about words. This calls for knowledge-based systems technology that can extend the approach of lexicons by allowing for automated identification of contexts for a word, by selecting knowledge on a context-dependent basis, and by supporting automated reasoning.

This need for lexical knowledge raises significant knowledge representation and reasoning questions. Lexical knowledge is default knowledge. This is knowledge that is usually correct, though can have some exceptions. Representing and reasoning with default knowledge in computing is difficult, and in anything other than small examples, it is necessary to adopt a logical approach in order to minimize these difficulties.

We therefore require a formal approach to knowledge representation and reasoning that can handle the context-dependent default knowledge. In this work, we explore the use of non-monotonic logics for building these more sophisticated lexical knowledgebases.

2 Contextual information

A context is a setting for a word. If a word is polysemous, then the word is a member of more than one context. Different contexts can denote different word senses for a word. In this way, a context can be viewed as a boundary on the meaning of a word. For example, the word `bank` can be described as being a member of contexts including `river` and `financial-institution`.

In language, the words surrounding a particular word can indicate the context for the word. For example, for a word in some text, the words in the same paragraph can usually indicate the context for the word.

In this section, we show how we can use a classification, or decision, tree to test whether a set of words is in a particular context. We then show how we can use machine learning techniques to generate such classification trees.

2.1 Context classification trees

Each classification tree is developed to test for a single classification. In this work, each classification is a context. Given a set of words, presence or absence of particular words in the set of words, is used by a classification tree to classify the set of words as either a positive or negative example for the classification. Hence, a classification tree determines whether the set of words is in a particular context.

A classification tree for a context C is a binary tree, where each node is a word, except the leaves which are labelled either POSITIVE or NEGATIVE. Given a set of words, start at the root: If the root is in S, then take the left subtree, otherwise take the right subtree. Upon taking the subtree, repeat the process, until reaching a leaf. If the leaf is POSITIVE then S is in context C.

Consider the example of a classification tree in Fig 1 for the classification `aircraft-accident`. Given the set $S = \{\texttt{crash}, \texttt{boeing}, \texttt{engine}, \texttt{runway}\}$, the tree classifies S as being in the context `aircraft-accident`.

Given a set of words S, there might be a number of classification trees with different contexts, and the set S is found to be in each of the contexts using the trees. To handle this see section 2.3.

2.2 Learning classification trees

In this work, we have used the ID3 inductive learning algorithm developed by Ross Quinlan [Qui86]. ID3 is an approach to machine learning based on constructing a classification, or decision, tree for a set of training examples. Training examples are presented as a table — each row is an example and each column is an attribute of the examples. The last attribute is the classification of the example. For example, in learning a decision tree for determining whether a patient has a particular disorder, we use a table of patients — some who have the disorder and some who do not. Each column refers to particular symptoms or tests, and the final column states whether the patient has the disorder. Once a decision tree has been constructed, and then tested successfully with examples not used in the training, it can be used to classify further examples.

We have used ID3 to classify textual information. The methodology involves taking an item of text, removing the stop words[1], and then using the remaining

[1] Stop words are words that usually offer relatively little semantic information in a sentence, such as for example, `the`, `a`, `because`, and `what`. They normally constitute about 50% of the words in a sentence.

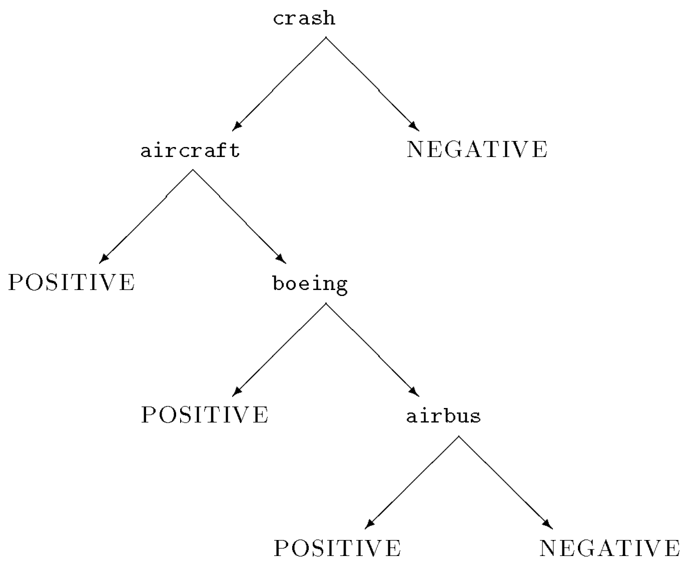

Fig. 1. Classification tree for `aircraft-accident`

words as a training (learning) example. Each item of text is a about at least one topic. Topics are used as the classifications for the examples. Each attribute in the table of training examples is a word. If a training example contains that word, then "yes" is entered into the corresponding position in the table, and "no" otherwise. Tables up to 236 attributes have been constructed containing up to 60 examples (rows) [Sha96]. We equate the notion of topic with that of context. So we can use these classification trees as described in section 2.1.

2.3 Reasoning with contextual information

We assume a set of atomic contexts. These are disjoint and exhaustive. They are the most focussed contexts we consider. We also allow contexts that are Boolean combinations of atomic contexts. These non-atomic contexts may have alternative names for lucidity. A classification tree can be trained for any Boolean combination of the atomic contexts. For example, if the language includes the non-atomic contexts `personal-finance` and `business-finance`, then the language includes the Boolean combinations such as

$$\texttt{personal-finance} \wedge \texttt{business-finance}$$

$$\texttt{personal-finance} \vee \texttt{business-finance}$$

The first is a more specialized context, whereas the second is a more generalized context. The second might have an alternative name such as `finance`.

In this way, we have a Boolean lattice of contexts built from the set of atomic contexts. The higher a context is in the lattice, the more general it is. Since we assume the set of atomic contexts is exhaustive, the top of the lattice is the context `anything` and the bottom is the context `nothing`. If for a given set of words S, we obtain contexts $\mathtt{x_1}, ..., \mathtt{x_n}$, using a set of classification trees, then $\mathtt{x_1} \wedge ... \wedge \mathtt{x_n}$ is a context for S.

We can reason with contexts in the lattice by assuming inferences that hold: If $\mathtt{x_1}$ and ... and $\mathtt{x_n}$ are contexts that hold for the source (according to the context classification trees), and $\mathtt{y}$ is a context higher in the lattice than $\mathtt{x_1} \wedge ... \wedge \mathtt{x_n}$, then $\mathtt{y}$ is a context for the source.

3 Reminder on default logic

In the following sections, we consider default logic as a formalism for lexical knowledge. Default logic was proposed by Reiter [Rei80], and good reviews are available (see for example [Bes89,Bre91]).

In default logic, knowledge is represented as a **default theory**, which consists of a set of first-order formulae and a set of default rules for representing default information. A **default rule** is of the following form, where α, β and γ are classical formulae,

$$\frac{\alpha : \beta}{\gamma}$$

The inference rules are those of classical logic plus a special mechanism to deal with default rules: Basically, if α is inferred, and $\neg\beta$ cannot be inferred, then infer γ. For this, α is called the pre-condition, β is called the justification, and γ is called the consequent. Informally, an **extension** is a maximally consistent set of inferences (classical formulae) that follow from a default theory.

Basing the framework on default logic brings advantages. Default logic provides an efficient representation for context-dependent reasoning and handling of exceptions, and it is a well-understood formalism for representing uncertain information. In addition, there are prototype implementations of inference engines for default logic that can be used for developing default logic knowledge-bases [Nie94,LS95,Sch95].

4 Outline of Lexica framework

The Lexica framework is based on default logic. In this framework we represent morphological, grammatical, and semantic relations using default logic. The user queries the system to find information about a word. The information can include synonyms, generalizations, specializations, meronyms, related terms, different lexical categories of the word, and so on.

A key feature of the Lexica framework is the identification of the context for a query. The context is identified from the input that the user provides. The interaction between a user and the system can be summarized as follows:

Input: A query word plus a source, defined as follows.

Query word. A word for which further information is required.

Source. A set of words used to help identify the contexts for a query word. A source may be obtained in a number of ways. For example, it could be given directly by a user seeking information about a query word, or it could be obtained from a sentence containing the query word.

Output: Set of relations providing further information about the query word.

A Lexica knowledgebase is composed of the following three sets of knowledge that are used to provide the output from the system.

1. **A set of context classification trees.** Given a source S and a query word `q`, the context classification trees are used to identify contexts that hold for $S \cup \{\texttt{q}\}$. Consider a tree T that tests whether $S \cup \{\texttt{q}\}$ is in context `x`: If the test is positive, then $S \cup \{\texttt{q}\}$ is in context `x`.
2. **A context lattice.** A Boolean lattice generated from a set of contexts. These atomic contexts are disjoint and exhaustive. Every context that can be identified by the set of context classification trees is present in the context lattice.
3. **A set of default rules.** These default rules represent context-dependent lexical knowledge. Given a query word and a set of context propositions derived from the source, these default rules are used to provide further information about the query word.

From input to output, reasoning with a Lexica knowledgebase is a three-stage process.

1. From the source and query word, contexts are found using the set of context classification trees. These contexts are called **primary contexts**.
2. From the primary contexts, further contexts are inferred from the context lattice. These further contexts are called **inferred contexts**. By reflexivity the inferred contexts include the primary contexts.
3. From inferred contexts, semantic relations that hold for the query word are identifed by reasoning with the default rules.

As an example, consider the following sentence.

```
The bank of a river in a flood plain is usually low.
```

Suppose the query word is `bank`, and the set of stop words in this sentence is the following.

{`The, of, a, in, is` }

This leaves the following set as the source.

{`river, flood, plain, usually, low` }

Assuming that `river` can be identified as a context by a context classification tree, and that `valley` can be identified as a context by a context classification tree, then `river` and `valley` are primary contexts containing `bank`. As a result, the following is an inferred context. We may choose to use an alternative name such as `river-bank` for it.

$$\texttt{river} \wedge \texttt{valley}$$

In the next subsection, we show how we represent the input to the system. We then show how we can represent lexical knowledge using default logic. Finally, we show how we query the system.

4.1 Representing inputs

The input to a Lexica system is a query word `q` and a source S. Let C denote the set of inferred contexts that hold for $S \cup \{\texttt{q}\}$. From the inputs, we form a set Q of formulae that we use as part of the default theory to derive the output. We now show how we form Q from the inputs.

$S \cup \{\texttt{q}\}$ is in context `x`
iff `context(x)` is in Q.

We can restate this as follows, where in a Boolean lattice, the downset of `x` is the set of all elements less than or equal to `x` in the lattice.

`context(x)` is in Q
iff the least element for the inferred contexts of $S \cup \{\texttt{q}\}$
is in the downset of `x` in the context lattice for $S \cup \{\texttt{q}\}$.

We also require the complement for the `context` relation.

$\neg$`context(x)` is in Q
iff the least element for the inferred contexts of $S \cup \{\texttt{q}\}$
is not in the downset of `x` in the context lattice for $S \cup \{\texttt{q}\}$.

The query word is the word for which further information is sought. Via the relations that hold for the query word, we also seek information about further words. For example, if the query word is `bank`, and the following relation holds, we then seek further information about `river-bank`.

`synonym(bank,river-bank)`

These words for which we seek further information are called **focus words**, and we denote this by the relation `focus`.

If `q` is the query word,
then we represent this as `focus(q)` in Q.

We propagate focus words by axioms of the following form. These capture the transitivity of focus for particular relations such as `synonym`, `related-term`, and `meronym`.

$$\texttt{focus(x)} \wedge \texttt{synonym(x,y)} \rightarrow \texttt{focus(y)}$$

$$\texttt{focus(x)} \wedge \texttt{related-term(x,y)} \rightarrow \texttt{focus(y)}$$

$$\texttt{focus(x)} \wedge \texttt{meronym(x,y)} \rightarrow \texttt{focus(y)}$$

The exact combination of axioms required in Q depends on which relations are used in the knowledgebase.

4.2 Representing semantic and grammatical relations

We assume a semantic relation is a binary relation between a pair of words. Types of relation include synonymy, antinymy, specialization, and meronymy. We qualify semantic relations according to context. For example, in the context of `river`, `bank` is a synomyn of `river-bank`, whereas in the context of `corporate-finance`, `bank` is a synonym of `merchant-bank`.

$$\frac{\texttt{focus(bank)} \ : \ \texttt{context(river)}}{\texttt{synonym(bank, river-bank)}}$$

$$\frac{\texttt{focus(bank)} \ : \ \texttt{context(corporate-finance)}}{\texttt{synonym(bank, merchant-bank)}}$$

We now consider some defaults for finding synonyms for `car`. The first says that `synonym(car,automobile)` holds if `context(road)` holds. The second says that in the more general situation where `context(transport)` holds, we also need `context(rail)` to not hold. The third rule is a weaker alternative to the second option: In the general situation where `context(transport)` holds, we also need to check $\neg$`context(road)` does not hold. In the framework, we have freedom as to whether we require a particular context (or negation of a context) is needed as a precondition or justification.

$$\frac{\texttt{focus(car)} \wedge \texttt{context(road)} \ : \ \top}{\texttt{synonym(car, automobile)}}$$

$$\frac{\texttt{focus(car)} \wedge \texttt{context(transport)} \ : \ \neg\texttt{context(rail)}}{\texttt{synonym(car, automobile)}}$$

$$\frac{\texttt{focus(car)} \wedge \texttt{context(transport)} \ : \ \texttt{context(road)}}{\texttt{synonym(car, automobile)}}$$

In some situations, `automobile` is not an appropriate synonym for `car`, such as in the case of `wagon`.

$$\frac{\texttt{focus(car)} \land \texttt{context(rail)} \ : \ \neg\texttt{context(road)}}{\texttt{synonym(car, wagon)}}$$

Another word sense for `car` is in the context of `lisp`. Here we consider the `specialization` relation as consequent.

$$\frac{\texttt{focus(car)} \land \texttt{context(lisp)} \ : \ \top}{\texttt{specialization(car,lisp-function)}}$$

If the context `lisp` cannot be determined, then the following default may be appropriate.

$$\frac{\texttt{focus(car)} \land \texttt{context(computing)} \ : \ \neg\texttt{context(transport)}}{\texttt{specialization(car,lisp-function)}}$$

As another example, consider the polyseme `case`. Here we a provide default for the `baggage` word sense.

$$\frac{\texttt{focus(case)} \ : \ \texttt{context(transport)} \land \neg\texttt{context(legal)}}{\texttt{synonym(case,baggage)}}$$

We now consider other semantic relations, including `located` and `made-of`, that can hold for a given word.

$$\frac{\texttt{focus(knife)} \ : \ \texttt{context(cooking)}}{\texttt{located(knife, kitchen)}}$$

$$\frac{\texttt{focus(hull)} \ : \ \texttt{context(ship)}}{\texttt{made-of(hull,steel)}}$$

$$\frac{\texttt{focus(hull)} \ : \ \texttt{context(sailing-ship)}}{\texttt{made-of(hull,wood)}}$$

We can draw on a richer taxonomy of meronymic relations, in particular [Cru86,WCH87], in order to develop further semantic relations. For example, "member/collection", "portion/mass", "place/area", and "component/integral-object".

Semantic information is also important in applying morphological and grammatical rules. Consider, for example, the following rules.

$$\frac{\texttt{focus(bank)} \ : \ \texttt{context(finance)}}{\texttt{category(bank, verb)} \lor \texttt{category(bank, noun)}}$$

$$\frac{\texttt{focus(bank)} \ : \ \texttt{context(river)}}{\texttt{category(bank, noun)}}$$

Since many morphological and grammatical rules are context-dependent, these can also be usefully presented in a Lexica knowledgebase.

4.3 Obtaining output

We now consider how we can reason with a Lexica knowledgebase in order to derive lexical information about a query word. First we need to assume some general Lexica knowledge, represented as a set of classical formulae, denoted G. This includes formulae such as the following for generating further useful semantic relations.

$$\texttt{synonym}(\texttt{x},\texttt{y}) \wedge \texttt{synonym}(\texttt{y},\texttt{z}) \rightarrow \texttt{synonym}(\texttt{x},\texttt{z})$$

$$\texttt{synonym}(\texttt{x},\texttt{y}) \rightarrow \texttt{synonym}(\texttt{y},\texttt{x})$$

$$\texttt{specialization}(\texttt{x},\texttt{y}) \wedge \texttt{specialization}(\texttt{y},\texttt{z}) \rightarrow \texttt{specialization}(\texttt{x},\texttt{z})$$

A Lexica knowledgebase is a default theory (D, W), where D is a set of semantic rules (discussed in section 4.2), and W is the union of G (discussed above) and Q (discussed in section 4.1). If E is an extension of (D, W), then E contains a set of semantic relations concerning the query word.

Given an extension E, we need to extract the semantic relations concerning E. Let R denote the set of semantic and grammatical relations in E. So for example, if E includes `synonym(happy,joyous)`, then `synonym(happy,joyous)` is in R.

4.4 Using the Lexica framework

Reasoning with a Lexica knowledgebase is non-monotonic with respect to the source: Taking a superset of the source may cause lexical inferences to be retracted. This gives the context-dependent reasoning that is necessary for lexical knowledge.

The definition of the Lexica framework does not exclude multiple extensions. For a given query word and source, the generation of multiple extensions implies that with respect to the source, the query word is ambiguous. This may be because the context is underdetermined.

Abstracting from a Lexica knowledgebase (D, W), we can obtain a semantic network — where the nodes are words and the arcs are semantic relations. We obtain this semantic network by taking the consequents of all the default rules in D. Call this network G. So $G = (N, A)$ is a directed graph where N is a set of nodes and A is a set of directed arcs. G is not necessarily a connected graph. For example, consider the following set of three defaults:

$$\frac{\texttt{focus}(\texttt{car}) : \texttt{context}(\texttt{road})}{\texttt{synonym}(\texttt{car},\texttt{automobile})}$$

$$\frac{\texttt{focus}(\texttt{automobile}) : \texttt{context}(\texttt{road})}{\texttt{synonym}(\texttt{automobile},\texttt{motor-car})}$$

$$\frac{\texttt{focus}(\texttt{road}) : \neg\texttt{context}(\texttt{sea})}{\texttt{synonym}(\texttt{road},\texttt{street})}$$

By abstracting from this Lexica knowledge, we obtain the semantic network composed of the following arcs. This network does not form a connected graph.

```
synonym(car,automobile)
synonym(automobile,motor-car)
synomym(road,street)
```

However, observe that given a source and a query word, an extension E of the corresponding Lexica knowledgebase will be such that the set of semantic relations R in E form a connected subgraph. This results from the propagation of the `focus` relation using the axioms in the Q subset of the default theory. To continue the above example, suppose the source just contains the word `car` and so the query word is `car`, then the extension contains the semantic relations `synonym(car,automobile)` and `synonym(automobile,motor-car)`, but not `synomym(road,street)`.

5 Discussion

In this paper, we have presented an important problem — reasoning with lexical knowledge — where default logic has much to offer. We have shown how we can use classification trees to identify contexts for a word, and shown how we can use the identified contexts to reason with lexical knowledge about the word in default logic. We have also shown that machine learning techniques can be used to generate context classification trees.

Our immmediate goal in developing the Lexica framework is to develop the Lexica framework for goal-directed reasoning. For simplicity, we chose Reiter's version of default logic. But, for efficiency, a goal-directed form of default reasoning is more appropriate. In particular, we are investigating the use of the XRay query answering system for default logics [Sch95].

For an application, it is possible that a relatively large number of default rules would be required for an acceptable level of performance. To address this viability problem, we aim to investigate a number of avenues: (1) Using the framework in restricted domains that require a limited number of default rules; (2) Using inductive logic programming ([Mug92]) to generate default rules for a domain; (3) Using co-locational data for knowledge engineering; and (4) Using machine-readable dictionaries and thesauri for knowledge engineering [Mei93].

The Lexica framework is complementary to formalizations of the notion of "aboutness" such as [BH94,Buv95,Hun96]. A Lexica knowledgebase could potentially be used in such frameworks to allow identification of, and reasoning with, relations such as "article A is about topic T".

Finally, we can consider the Lexica approach as a move towards reusable knowledgebases or general knowledge systems. CYC is perhaps the best known, and certainly the most intensively developed example of a general knowledge system [Len95]. There are many problems with reuse of knowledge, as highlighted during the development of CYC. It is likely that initial success will eminate from more highly structured systems with constrained querying, such as in the Lexica approach, than in more general systems, such as CYC, that have a wider range of knowledge and querying.

Acknowledgements

I would like to thank an anonymous referee for some helpful suggestions.

References

[Bes89] Ph Besnard. *An Introduction to Default Logic*. Springer, 1989.

[BFGM91] R Beckworth, C Fellbaum, D Gross, and G Miller. WordNet: A lexical database organized on psycholinguistic principles. In U Zernik, editor, *Lexical Acquisition: Exploiting On-line Resources to Build a Lexicon*, pages 211–226. Lawrence Erlbaum Associates, 1991.

[BH94] P Bruza and T Huibers. Investigating aboutness axioms using information fields. In *Proceedings of the 18th ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'94)*, pages 112–121. Springer, 1994.

[BPK$^+$96] U Borghoff, R Pareschi, H Karch, M Nohmeier, and J Schlichter. Constraint-based information gathering for a network publication system. In *Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi-agent Technology.* Pratical Applications Company, 1996.

[Bre91] G Brewka. *Common-sense Reasoning*. Cambridge University Press, 1991.

[Bri91] T Briscoe. Lexical issues in natural language processing. In E Klein and F Veltman, editors, *Natural Language and Speech*, pages 39–68. Springer, 1991.

[Buv95] S Buvac. Resolving lexical ambiguity using a formal theory of context. In K van Deemter and S Peters, editors, *Semantic Ambiguity and Underspecification*, pages 101–124. CSLI Publications, 1995.

[Cro93] B Croft. Knowledge-based and statistical approaches to text retrieval. *IEEE Expert*, pages 8–12, 1993.

[Cru86] D Cruse. *Lexical Semantics*. Cambridge University Press, 1986.

[Eng96] B Engleder. *Filtering News Articles.* MSc Thesis, Department of Computing, Imperial College, London, 1996.

[FJ96] A Falk and I Jonsson. PAWS: An agent for WWW-retrieval and filtering. In *Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi-agent Technology.* Practical Applications Company, 1996.

[GF95] B Grosof and D Foulger. Globenet and RAISE: Intelligent agents for networked newsgroups and customer service support. Technical report, IBM Research Division, T J Watson Research Center, New York, 1995.

[GLC$^+$95] B Grosof, D Levine, H Chan, C Parris, and J Auerbach. Reusable architecture for embedding rule-based intelligence in information agents. Technical report, IBM Research Division, T J Watson Research Center, New York, 1995.

[GPWS96] L Gutherie, J Pustejovsky, Y Wilks, and B Slator. The role of lexicons in natural language processing. *Communications of the ACM*, 39(1):63–72, 1996.

[Hun96] A Hunter. Intelligent text handling using default logic. In *Proceedings of the IEEE Conference on Tools with Artificial Intelligence*, pages 34–40. IEEE Computer Society Press, 1996.

[Len95] D Lenat. CYC:a large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11):33–38, 1995.

[Lie95] H Lieberman. Letizia: An agent that assists web browsing. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence.* Morgan Kaufmann, 1995.

[LS95] T Linke and T Schaub. Lemma handling in default logic theorem provers. In *Symbolic and Qualitative Approaches to Reasoning and Uncertainty*, volume 946 of *Lecture Notes in Computer Science*, pages 285–292. Springer, 1995.

[Mae94] P Maes. Agents that reduce work and information overload. *Communications of the ACM*, 37(7):31–40, 1994.

[Mei93] W Meijs. Exploring lexical knowledge. In C Souter and E Atwell, editors, *Corpus-based Computational Linguistics*, pages 249–260. Rodopi, 1993.

[Mil95] G Miller. WordNet: A lexical database for English. *Communications of the ACM*, 38(11):39–41, 1995.

[Mou96] A Moukas. Amalthaea: Information discovery and filtering using a multiagent evolving ecosystem. Technical report, MIT Media Laboratory, Cambridge MA, 1996.

[Mug92] S Muggleton. *Inductive Logic Programming.* Academic Press, 1992.

[Nie94] I Niemelä. A decision method for non-monotonic reasoning based on autoepistemic reasoning. In *Proceedings of the Fourth International Conference Principles of Knowledge Representation and Reasoning*, pages 473–484. Morgan Kaufmann, 1994.

[Qui86] J Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[Rei80] R Reiter. Default logic. *Artificial Intelligence*, 13:81–132, 1980.

[Sch95] T Schaub. A new methodology for query-answering in default logics via structure-oriented theorem proving. *Journal of Automated Reasoning*, 15:95–165, 1995.

[Sha96] A Shaikh. *Data Mining Using Inductive Logic Programming.* MSc Thesis, Department of Computing, Imperial College, London, 1996.

[Tra93] R Trask. *A Dictionary of Grammatical Terms in Linguistics.* Routledge, 1993.

[Voo94] E Voorhees. Query expansion using lexical-semantic relations. In W Croft and C van Rijsbergen, editors, *Proceedings of the Seventeenth International ACM-SIGIR Conference on Research and Developement in Information Retrieval*, pages 61–69, 1994.

[WCH87] M Winston, R Chaffin, and D Herrman. A taxonomy of part-whole relations. *Cognitive Science*, 11:417–444, 1987.

[WSG96] Y Wilks, B Slator, and L Guthrie. *Electric Words: Dictionaries, Computers, and Meanings.* MIT Press, 1996.

[Yok95] T Yokoi. The EDR Electronic Dictionary. *Communications of the ACM*, 38(11):42–44, 1995.