A default logic based framework for context-dependent reasoning with lexical knowledge

Anthony Hunter
Department of Computer Science
University College London
Gower Street
London WC1E 6BT, UK
Email:a.hunter@cs.ucl.ac.uk

December 1, 1999

Abstract

Lexical knowledge is increasingly important in information systems — for example in indexing documents using keywords, or disambiguating words in a query to an information retrieval system, or a natural language interface. However, it is a difficult kind of knowledge to represent and reason with. Existing approaches to formalizing lexical knowledge have used languages with limited expressibility, such as those based on inheritance hierarchies, and in particular, they have not adequately addressed the context-dependent nature of lexical knowledge. Here we present a framework, based on default logic, called the dex framework, for capturing context-dependent reasoning with lexical knowledge. Default logic is a first-order logic offering a more expressive formalisation than inheritance hierarchies: (1) First-order formulae capturing lexical knowledge about words can be inferred; (2) Preferences over formulae can be based on specificity, reasoning about exceptions, or explicit priorities; (3) Information about contexts can be reasoned with as first-order formulae formulae; and (4) Information about contexts can be derived as default inferences. In the dex framework, a word for which lexical knowledge is sought is called a query word. The context for a query word is derived from further words, such as words in the same sentence as the query word. These further words are used with a form of decision tree called a context classification tree to identify which contexts hold for the query word. We show how we can use these contexts in default logic to identify lexical knowledge about the query word such as synonyms, antonyms, specializations, meronyms, and more sophisticated first-order semantic knowledge. We also show how we can use a standard machine learning algorithm to generate context classification trees.

Keywords: Knowledge representation; Lexical knowledge; Non-monotonic logic; Default logic; Reasoning under uncertainty

1 Introduction

There is a need for more sophisticated lexical knowledge in information systems. In particular, there is a need for richer semantic knowledge about words. Consider for example indexing documents using keywords, or dismbiguating words in a query to an information retrieval system, or a natural language interface.

Simple forms of semantic knowledge include semantic relations such as synonyms, related terms, antonyms, and specializations for a word [Cru86]. Semantic knowledge 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`.

However, formalizing semantic knowledge about words is challenging. Most words are ambiguous [Spa86, Hir87, Gut93, Gre96]. Obvious examples are `bank`, `plane`, and `train`. This includes ambiguity about category, e.g. `bank` is both a noun and a verb, and ambiguity about word meaning, e.g. for `bank`, a dictionary entry might be:

> `bank` *n* (1) raised shelf of ground, slope; (2) ground at edge of river; (3) mass of cloud; (4) establishment for custody of money; (5) money before keeper of gaming-table.

Difficulties with such a dictionary entry arise if we want to use automated reasoning since formalization is not straightforward. Each of the numbered statements gives a different meaning, or word sense, for the word. Furthermore, we may wish to sub-divide the definition. For example, sense (4) could be divided into (4a) a building you go to cash a cheque, and (4b) a company that holds your savings. This then means sense (4) is both a type of `company` and a type of `building`. Another difficulty is that the meaning of the word is partly dependent on the overlaps between the different senses of the word. So for example, (1) and (2) overlap and intuitively reinforce each other. Similarly for (4) and (5).

These problems call for a logic-based approach to representing and reasoning with lexical knowledge. In a logic-based approach, intermediate concepts can be defined as logical formulae, and used to formalize a range of inter-related word senses for a given word. This range of word senses may include more general, more specific, and overlapping wordsenses.

However, classical logic is not appropriate for lexical knowledge, as lexical knowledge is a form of context-dependent knowledge. For example, normally we can infer that `petroleum` is a synonym of `oil`, but in the context of `cooking`, this inference is defeated. Given the uncertainty involved in this context-dependent reasoning, we need to handle lexical knowledge using a form of default reasoning.

In this paper, we use default logic to provide a framework, called the dex framework, for context-dependent reasoning with lexical knowledge. The aim of the framework is to provide lexical knowledge, in particular semantic knowledge, for applications in information systems. The framework is a development of [Hun97].

2 Overview of default logic

Default logic was proposed by Reiter [Rei80], and good reviews are available (see for example [Bes89, Bre91, Ant97, BDK97, Sch98]).

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.

Default logic is an extension of classical logic. Hence, all classical inferences from the classical formulae in a default theory are derivable (if there is an extension). The default theory then augments these classical inferences by default inferences derivable using the default rules.

More formally, we introduce the operator Γ that indicates what conclusions are to be associated with a given set E of formulae, where E is some set of classical formulae. Let (D, W) be a default theory, where D is a set of default rules and W is a set of classical formulae. Let Cn be the function that for a set of formulae returns the set of classical consequences of those formulae. For this, $\Gamma(E)$ is the smallest set of classical formulae such that the following three conditions are satisfied.

1. $W \subseteq \Gamma(E)$
2. $\Gamma(E) = Cn(\Gamma(E))$
3. For each default in D, where α is the pre-condition, β is the justification, and γ is the consequent, the following holds:

$$\text{if } \alpha \in \Gamma(E) \text{ and } \neg\beta \notin E \text{ then } \gamma \in \Gamma(E)$$

Let us call E the satisfaction set, and $\Gamma(E)$ the putative extension. Once $\Gamma(E)$ has been identified, E is an extension of (D, W) iff $E = \Gamma(E)$. If E is an extension, then the first condition ensures that the set of classical formulae W is also in the extension, the second condition ensures the extension is closed under classical consequence, and the third condition ensures that for each default rule, if the pre-condition is in the extension, and the justification is consistent with the extension, then the consequent is in the extension.

We can view E as the set of formulae for which we are ensuring consistency with the justification of each default rule that we are attempting to apply. We can view $\Gamma(E)$ as the set of putative conclusions of a default theory: It contains W, it is closed under classical consequence, and for each default that is applicable (i.e. the precondition is in $\Gamma(E)$ and the justification is satisfiable with E), then the consequent is in $\Gamma(E)$. We ask for the smallest $\Gamma(E)$ to ensure that each default rule that is applied is grounded. This means that it is not the case that one or more default rules are self-supporting. For example, a single default rule is self-supporting if the pre-condition is satisfied using the consequent. The test $E = \Gamma(E)$ ensures that the set of formulae for which the justifications are checked for consistency coincides with the set of putative conclusions of the default theory. If $E \subset \Gamma(E)$, then not all applied rules had their justification checked with $\Gamma(E)$. If $\Gamma(E) \subset E$, then the rules are checked with more than is necessary.

To illustrate the context-dependent reasoning with default logic, consider the default theory composed of the following set of default rules,

$$\frac{\texttt{bird(x)} \ : \ \neg(\texttt{ostrich(x)} \vee \texttt{penguin(x)})}{\texttt{fly(x)}}$$

$$\frac{\texttt{penguin(x) : }\neg\texttt{fly(x))}}{\neg\texttt{fly(x)}}$$

$$\frac{\texttt{ostrich(x) : }\neg\texttt{fly(x))}}{\neg\texttt{fly(x)}}$$

and the following set of atoms,

```
bird(fred)
bird(sid)
ostrich(sid)
```

From this default theory, we obtain just one extension which contains `fly(fred)` and ¬`fly(sid)`.

Basing the dex framework on default logic brings advantages. Default logic provides a rich (expressive) and lucid representation for context-dependent reasoning and handling of exceptions. It is an efficient representation in terms of space. Also, it is a well-understood formalism for representing uncertain information, and it has strong theoretical foundations. 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, NS98].

Other examples of using default logic in handling language include for reasoning about presuppositions [Mer91, Ger95], anaphoric resolution [Qua93], and for reasoning about the notion of "aboutness" [Hun96].

3 The dex framework

In the dex framework (for <u>d</u>efault l<u>ex</u>ical framework), we can represent morphological, grammatical, and semantic knowledge using default logic. The system is queried to find knowledge about a word. The knowledge can include synonyms, generalizations, specializations, definitions, meronyms, related terms, different lexical categories of the word, and so on.

A key feature of the dex framework is the identification of the context for a word being queried. The context is identified from words in the same text as the word being queried, such as the words in the same sentence.

In the dex framework, the use of a lexical knowledgebase, called a dex knowledgebase, 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. So if the lexical knowledgebase is being used to help index a document, then the query word would be a word in the document. Similarly, if the lexical knowledgebase is being used to help understand a query to a natural langauge interface, then the query word would be a word in the query.
- **Source.** A set of words including the query word that is used to identify the contexts for a query word. A source may be obtained in a number of ways. For example, it could be obtained from a sentence containing the query word. So if the lexical knoweldgebase is being used to help index a document, then the source would be some of the words in the document. Similarly, if the lexical knowledgebase is being used to help understand a query to a natural langauge interface, then the source would be the words in the query.

Output: Set of formulae providing lexical knowledge about the query word.

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

1. **A set of context classification trees:** Given a source S containing the query word `q`, the context classification trees are used to identify contexts that hold for S. Consider a tree T that tests whether S is in context ϕ: If the test is positive, then S is in context ϕ.
2. **A default theory:** This is based around a set of default rules representing context-dependent lexical knowledge. Given a query word and a set of contexts identified using the context classification trees, these default rules are used to provide lexical knowledge about the query word.

In the default theory, words and contexts are represented as constant symbols and lexical relations such as synonym, meronym, etc, are represented by predicate symbols, and so lexical knowledge about words is captured by first-order formulae. The two main types of default rules are:

Default context rules These default rules allow for the inference of further contexts given a set of contexts.

Default lexical rules These default rules allow for the inference of lexical knowledge according to which contexts hold.

So, from input to output, reasoning with a dex 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 default context rules. These further contexts are called **inferred contexts**. By reflexivity, the inferred contexts include the primary contexts.
3. From the inferred contexts, lexical knowledge about the query word is identified by reasoning with the default lexical 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[1] in this sentence is the following.

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

This leaves the following set as the source.

[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 [vR79].

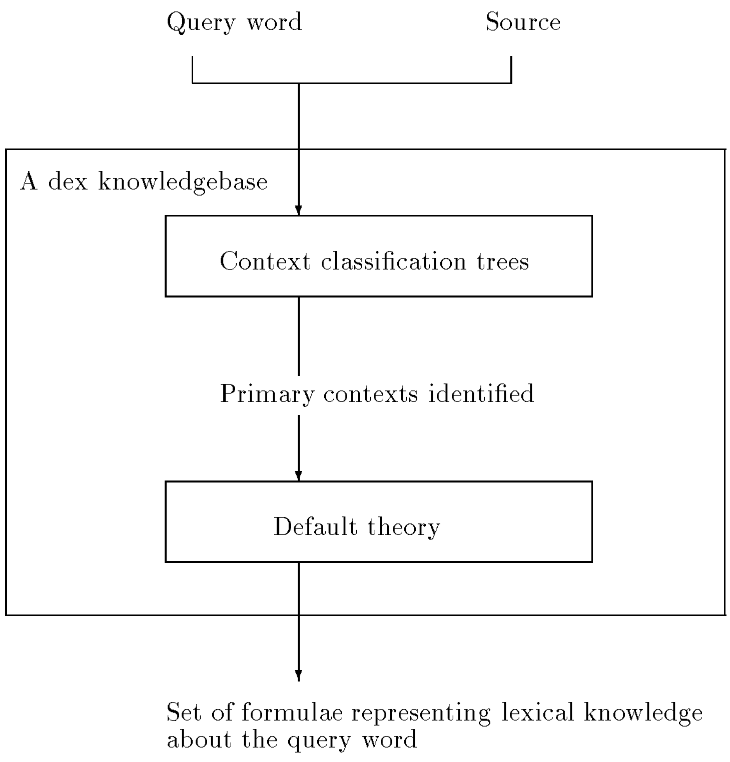
Query word
Source
A dex knowledgebase
Context classification trees
Primary contexts identified
Default theory
Set of formulae representing lexical knowledge about the query word

Figure 1: A schematic summary of using the dex framework. The dex knowledgebase is delineated by the outer box. The input to the dex knowledgebase is the query word and source, and the output is the set of first-order classical formulae representing lexical knowledge about the query word.

{`bank, 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, `river-bank` could be an inferred context.

In the following subsections, we look more closely at how to identify contexts from the source, how to use default logic to represent lexical and context rules, how to obtain output from a dex knowledgebase, and how to use a standard machine learning algorithm to generate context classification trees from sets of training examples.

3.1 Identifying contexts from the source

A context is a setting for a word. If a word is ambiguous, 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 (non-strict) 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

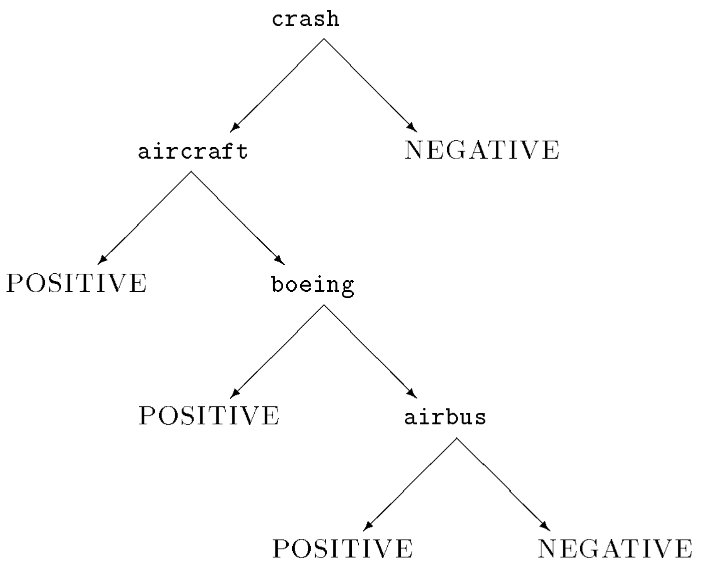

Figure 2: Classification tree for `aircraft-accident`.

for the word.

In this section, we show how we can use a classification tree (also called a decision tree) to test whether a set of words is in a particular context. In a later section, we show how we can use machine learning techniques to generate such 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 ϕ 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 ϕ. If the leaf is NEGATIVE, then we infer nothing from this tree about S.

Consider the example of a classification tree in Figure 2 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.

The input to a dex knowledgebase is a query word `q` and a source S. Each primary context that holds for S is entered into the default theory (D, W) as follows: If S is in context ϕ, then

$$\frac{\top \; : \; \texttt{context}(\phi)}{\texttt{context}(\phi)}$$

is a default rule in (D, W). So each primary context is represented by a default rule in the default theory.

Note, if S is not in context ϕ — in other words, if the context classification tree for ϕ classifies S as negative — then we do *not* enter

$$\frac{\top \ : \ \neg\texttt{context}(\phi)}{\neg\texttt{context}(\phi)}$$

into the default theory. The reason that we do not want this is that we want to maintain an "open world assumption". Even if we cannot show that S is in context ϕ using a context classification tree, we may obtain it using default context rules.

3.2 Using default logic to represent context rules

We use default context rules to reason with primary contexts to derive inferred contexts. So for example, given `context(finance)` and `context(business)`, and it is consistent to assume `context(corporate-finance)`, then derive `context(corporate-finance)`.

$$\frac{\texttt{context(finance)} \wedge \texttt{context(business)} \ : \ \texttt{context(corporate-finance)}}{\texttt{context(corporate-finance)}}$$

We may also incorporate constraints in the default theory: Suppose for a given source S and a query word q, we obtain both `context(finance)` and `context(river)` using a set of context classification trees. Normally, we would want to keep these contexts separate: In other words there is ambiguity about the context for the query word. So it would be useful to generate two extensions for the query word: the first with `context(river)` and the second with the `context(finance)`. We can ensure this by assuming the following classical formula in the set W in the default theory.

$$\texttt{context(river)} \leftrightarrow \neg\texttt{context(corporate-finance)}$$

This formula prohibits both `context(river)` and `context(corporate-finance)` to hold in the same extension. Now consider the following more complicated default context rule, where the precondition is a conjunctive normal form formula.

$$\frac{\texttt{context(trade)} \wedge \texttt{context(USA)} \wedge (\texttt{context(Mexico)} \vee \texttt{context(Canada)}) \ : \ \neg\texttt{context(GATT)}}{\texttt{context(NAFTA)}}$$

Another kind of default context rule precludes a context holding when some combination of other contexts hold:

$$\frac{\texttt{context(lisp)} \wedge \texttt{context(functions)} \ : \ \neg\texttt{context(transport)}}{\neg\texttt{context(transport)}}$$

We can consider a set of contexts C for a dex knowledgebase as being ordered by a specialization relation $\leq$, where for each $(\phi, \psi) \in \leq$, ϕ is a more specialized context than ψ. In general, $(C, \leq)$ is a directed acyclic graph. For some dex knowledgebases, it may be more restricted such as being a complete lattice, or a Boolean lattice.

$$\begin{tikzcd}
 & & \top \arrow[dl] \arrow[ddrr] & & \\
 & \texttt{transport} \arrow[dl] \arrow[dr] & & & \\
\texttt{motor} \arrow[dr] & & \texttt{rail} \arrow[dl] & & \texttt{lisp} \arrow[ddll] \\
 & \texttt{motorail} \arrow[dr] & & & \\
 & & \bot & &
\end{tikzcd}$$

Figure 3: A directed acyclic graph of contexts: Each node represents a context, and each arrow represents a specialization, where the arrow points to the more specialized context, and $\bot$ denotes the empty (inconsistent) context and $\top$ denotes the context for everything.

Since, there is some uncertainty involved in identifying the primary contexts for each source, the context classification trees cannot be guaranteed to be correct — sometimes they give a false POSITIVE and sometimes a false NEGATIVE. False POSITIVES can be viewed as a form of unsoundness and false NEGATIVES as a form of incompleteness.

Default context rules are used to manage this incompleteness by giving further contexts that hold for a set of primary contexts. In addition, they offer the opportunity for extending the set of contexts that can be identified for any source. In this way, the default context rules facilitate navigation of $(C, \leq)$, by supporting the inference of more general and more specialized contexts from a set of primary contexts.

More difficult are the false POSITIVES. Once a context is ascribed to a source, the context cannot be retracted. However, both the default context rules and the default lexical rules tolerate such erroneous information — in part by using multiple extensions.

3.3 Using default logic to represent lexical rules

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)
```

The 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 by the classical formula `focus(q)` in W.

We discuss propagating the focus relation in Section 3.4. In the following, we discuss some alternatives for representing lexical rules in default logic. In practice, the choice of default rules, and classical formulae depends on the application.

We assume some semantic relations as binary relations between pairs of words. Types of relation include synonymy, antonymy, 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`.

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

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

We now consider some defaults (represented below) for finding synonyms for `car`. The first says that `synonym(car,automobile)` holds if `context(road)` holds and that it is consistent to assume `synonym(car,automobile)` holds. The second says that in the more general situation where `context(transport)` holds, we also need `context(rail)` to not hold. In the dex framework, we have freedom as to whether we require a particular context (or negation of a context) as a precondition or justification.

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

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

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

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

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

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

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

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

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

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

So far with all these examples, we have handled inclusive lexical rules — i.e. rules for lexical relations that hold for a given focus in a given context. However, we can also handle exclusive lexical rules — i.e. rules for lexical relations that do not hold for a given focus in a given context. This is a form of preclusion. Consider, for example,

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

Representing preclusion using defaults means that this could be defeated by other information. For example, in the context of `motorail`. Using default rules for preclusion in an alternative to using a classical formula such as:

$$\texttt{synonym(car,wagon)} \leftrightarrow \neg\texttt{synonym(car,automobile)}$$

As another example of preclusion, consider the word `water`. In a normal context, `liquor` is not a synonym of `water`. However, in the context of a brewery, it is a synonym.

$$\frac{\texttt{focus(water)} \;:\; \neg\texttt{context(brewery)}}{\neg\texttt{synonym(water,liqour)}}$$

$$\frac{\texttt{focus(water)} \wedge \texttt{context(brewery)} \;:\; \texttt{synonym(water,liqour)}}{\texttt{synonym(water,liqour)}}$$

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 order to develop further semantic relations — e.g. “member/collection”, “portion/mass”, “place/area”, and “component/integral-object”.

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

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

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

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

3.4 Obtaining output from a dex knowledgebase

We now consider how we can reason with a dex knowledgebase in order to derive lexical information about a query word. We propagate focus words by axioms of the following form that are included in W in the default theory. These capture the transitivity of focus for particular relations such as `synonym`, `related-term`, and `meronym`.

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

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

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

We can also limit the number of inferences being generated:

$$\mathtt{focus(x,n)} \wedge \mathtt{synonym(x,y)} \wedge \mathtt{n} < \mathtt{m} \rightarrow \mathtt{focus(x,n+1)}$$

where `m` is the upper limit on the size of the network being produced.

We also need to assume some general dex knowledge, represented as a set of classical formulae. This includes formulae such as the following for generating further useful semantic relations.

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

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

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

Lexical relations, such as synonym, are only weakly transitive. In other words, after a few applications of transitivity, the words are no longer synonyms, and may indeed be quite unrelated. To address this, we can limit the number of applications of transitivity, as follows where i and j denote the number of applications of transitivity for each relation, and m is an upper limit.

$$\mathtt{synonym(x,y,i)} \wedge \mathtt{synonym(y,z,j)} \wedge \mathtt{(i+j)} < \mathtt{m} \rightarrow \mathtt{synonym(x,z,i+j)}$$

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

If (D, W) is a dex knowledgebase and E is an extension of (D, W), then E includes a set of semantic relations concerning the query word. Abstracting from E, we may obtain a semantic network where the nodes are words and the arcs are semantic relations. So for example, if E includes `synonym(happy,joyous)`, then `synonym` is an arc connecting the nodes `happy` and `joyous` in the corresponding semantic network. In this way, it is possible to define a default theory that would give an extension from which the semantic network in Figure 4 could be obtained.

As another example, consider the following set of four defaults:

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

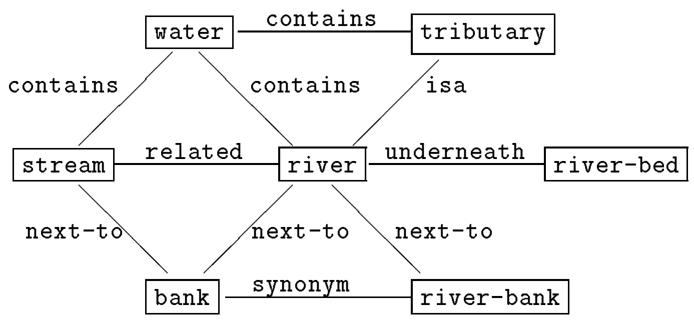

Figure 4: A semantic network: nodes denote words and labelled arcs denote binary relations. Note that this semantic network is restricted to a single (coherent) context. It could be viewed as a subgraph of a larger semantic network where in the larger semantic network multiple word senses are represented.

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

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

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

Suppose we have the query word `car`, we can obtain an extension from which we can obtain the semantic network composed of the following arcs.

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

Note, the definition of a dex knowledgebase 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.

3.5 Exploiting the expressibility of default logic

A wider variety of predicates can be used in addition to capture object-level and meta-level information about a query word and the situation in which it is used. Object-level information might include co-locational information such as identification of the word(s) to the left. For example, in the context of `fisheries`, the word `bed` could refer to `river-bed`, or to `oyster-bed`. However, if the word to the left is of the occurence of `bed` is either `river` or `oyster`, then the ambiguity is resolved. Consider the following default rule, where `wordtoleft(hairpin)` denotes that `hairpin` is the word to the immediate left of the focus word:

$$\frac{\texttt{focus(bend)} \wedge \texttt{wordtoleft(hairpin)} \ : \ \texttt{context(road)}}{\texttt{synonym(bend,switchback)}}$$

We can use this approach to resolve more problematical ambiguities that arise with noun-noun phrases such as `fruit flies` in the sentence.

`Fruit flies like plants.`

For this, we can use the following default rules.

$$\frac{\texttt{focus(flies)} \land \texttt{wordtoleft(fruit)} \;:\; \texttt{context(biology)}}{\begin{array}{c}\texttt{specialization(flies, fruit-flies)} \\ \land\, \texttt{category(flies, noun)} \\ \land\, \texttt{category(fruit-flies, noun-noun-phrase)}\end{array}}$$

Further relations that can capture co-locational information include the binary relation `insamesentence` that can provide more useful information on disambiguation when the identified contexts are too weak to draw sufficient inferences for a particular source. For example,

$$\frac{\texttt{focus(plant)} \land \texttt{insamesentence(plant,manufactoring)} \;:\; \texttt{context(business)}}{\texttt{specialization(plant, manufactoring-plant)}}$$

Meta-level information might include the type of article or type of publication so for example an article in the company reports section of the *Financial Times* is almost certainly about `companies`, `business`, etc. Similarly, the words in the first sentence of a *Financial Times* article are much more significant in determining the context of an article than words that occur in later sentences. As another example, proper nouns are particularly important in determining the context of an article in publications such as the *Financial Times* and *The Economist.*

Richer lexical knowledge, and associated world knowledge, can be captured using the first-order knowledge representation and reasoning available in default logic. Consider for example the following default rule:

$$\frac{\texttt{focus(sentence)} \land \texttt{context(law)} \;:\; \neg\texttt{context(writing)}}{\exists \texttt{x}, \texttt{y}, \texttt{t}\, [\texttt{judge(x)} \land \texttt{defendant(y)} \land \texttt{sentences(x, y)} \land \texttt{sentence(y, t)}]}$$

Once we consider a richer langauge for inferencing with words in context, we can harness other formalisms for knowledge representation, such as for example episodic logic [HS93]. In this way, we can use default logic as a bridge between handling words in text and reasoning with implied word senses.

4 Learning context classification trees

We have used the ID3 inductive learning algorithm developed by Quinlan [Qui86] for learning context classification trees.

4.1 Using the ID3 to learn context classification trees

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, and then using the remaining words as a training (learning) example. The stop list we used was similar to that in [vR79]. Each item of text implies one or more contexts. Contexts 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.

The ID3 algorithm is a very simple approach to learning classification trees. There are developments, such as C4.5 [Qui93] that include handling missing or noisy data, and avoiding overfitting using pruning of classification trees.

4.2 A case study of learning context classification trees

In this case study, we used a set of 139 news summaries taken from *The Economist* newspaper in the period 1996-98. We focussed on articles for the contexts of `m&a` (corporate mergers and acquisitions), `collaboration` (collaborative deals, alliances, and joint ventures), `military` (terrorist activities, peace keeping, military manoeuvres, and wars), and `trade` (international trade agreements), `business` and `paramilitary-incident`.

The ID3 algorithm was written in Prolog. For learning context classification trees, the methodology for the case study is summarized as follows: A training example was generated from each article by removing proper nouns[2], stop words, and punctuation. For the remaining words, each was rewritten into just lower case characters, plural nouns were rewritten as singular nouns, and verbs were rewritten as base verbs. The resulting set of words was the training example. Each training example is classified by hand as being either a POSITIVE or NEGATIVE example for each context. We therefore assume that for each article, the sense of each word in the article is constant — so multiple occurrences of the word in the article have the same meaning. A context classification tree was obtained for each of the contexts. Two examples are given in Figures 5 and 6.

Given the context classification trees, we can analyse further articles of the kind found in *The Economist.* Consider the following example:

> Seagram, the Canadian drinks multinational, has made an agreed bid for the Polygram records and film business that is part-owned by the Dutch electronics group Philips. It is unlikely that the acquisition will attract any interest from any regulator.

After removing stop words and rewriting the remaining words, we have the following set of words for the source:

```
drinks, multinational, make, agreed, bid, record, film, business, part, own,
electronic, group, unlikely, acquisition, attract, any, interest, regulator
```

[2]Proper nouns seem to be a very good indicator of a context. This is particularly so for articles in *The Economist.* The main reason they were excluded in this case study was that we wanted to demonstrate the viability of the approach without the advantage of them. A disadvantage of proper nouns is that they make the classification trees very specialized — even over specialized.

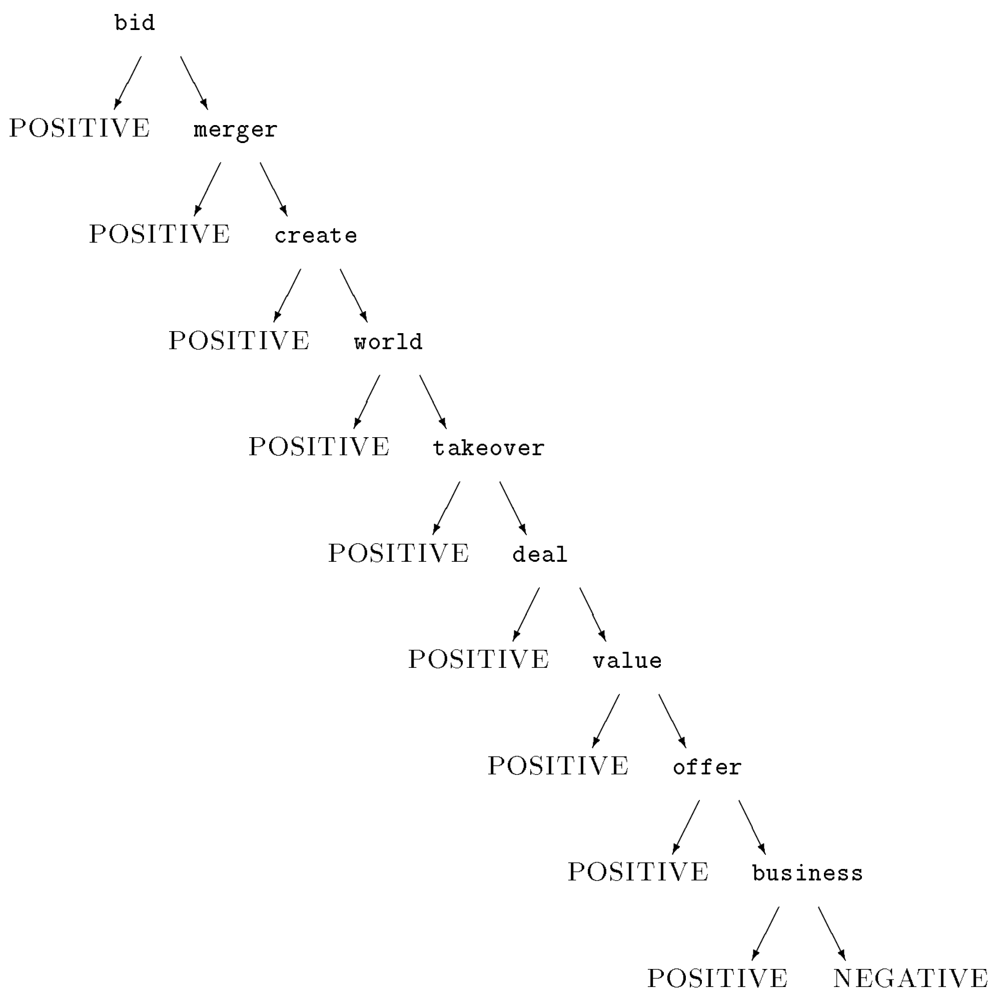

Figure 5: A context classification tree for the context `m&a`. The tree was generated from a training set of 81, where 51 were positive examples and 30 were negative examples, taken from articles on business from the *Business this week* section of *The Economist*. Each article had around 50 words. The set of articles used over 750 different words after removing stop words. Each example was tabulated on the basis of the presence or absence of each of 30 attributes. Each attribute is a word that occurs in 5 or more of the articles. The attributes for this training set were `alliance`, `airline`, `agree`, `bank`, `business`, `bid`, `biggest`, `buy`, `company`, `create`, `deal`, `firm`, `form`, `group`, `high`, `market`, `merge`, `merger`, `offer`, `own`, `pay`, `plan`, `purchase`, `serice`, `share`, `takeover`, `up`, `value`, `venture`, and `world`.

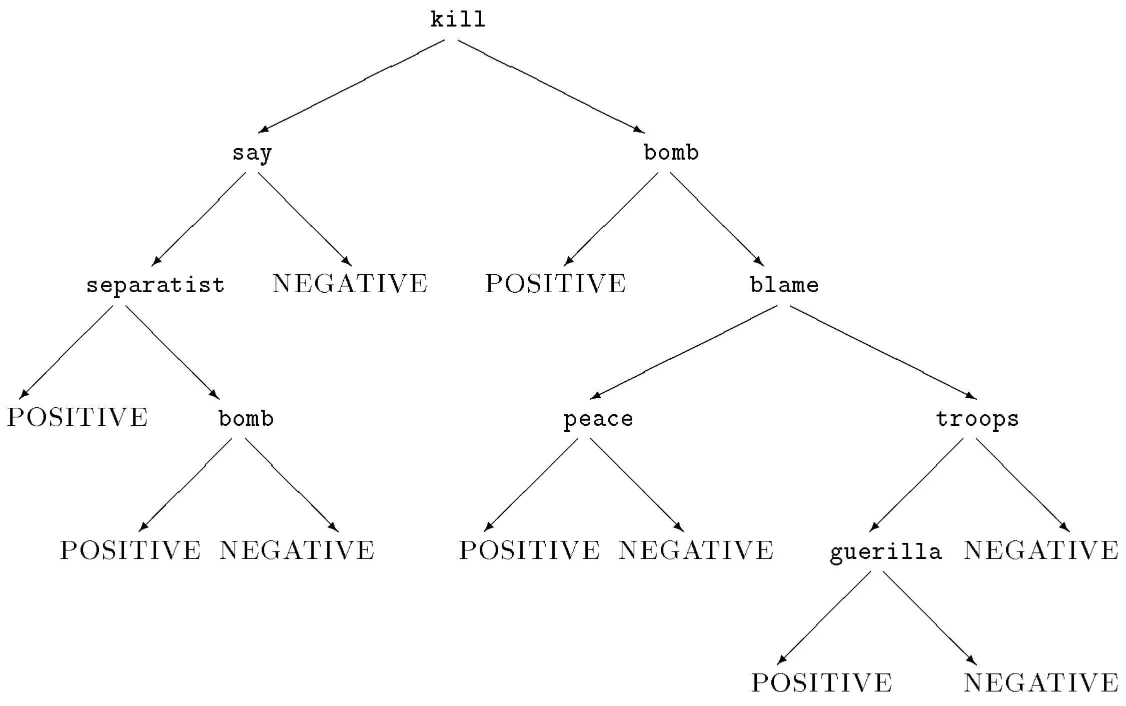

Figure 6: A context classification tree for the context `paramilitary-incident`. The tree was generated from a training set of 47, where 16 were positive examples and 31 were negative examples, taken from articles on military incidents and issues from the *Politics this week* section of *The Economist*. Each article had around 50 words. The set of articles used over 400 words after removing stop words. Each example was tabulated on the basis of the presence or absence of each of 20 attributes. Each attribute is a word that occurs in 5 or more of the articles. The attributes for this training set were `after`, `army`, `blame`, `bomb`, `country`, `government`, `group`, `guerilla`, `kill`, `last`, `military`, `month`, `more`, `peace`, `people`, `say`, `separatist`, `soldier`, `talk`, and `troops`.

From this set, we obtain the context `m&a` using the context classification tree in Figure 5.

From this case study, we can see that a relatively large numbers of training examples can be used to generate the context classification trees and the resulting trees can be quite lucid.

5 Comparison with other approaches to lexical knowledge

The simplest computer-oriented approaches to lexical knowledge are machine-readable lexicons that may be viewed as relational databases (for reviews see [WSG96, GPWS96]). These are large repositories of information on words with little or no structure beyond providing some attributes for each word. These approaches therefore do not harness context-dependent reasoning, default reasoning, or uncertainty management.

Perhaps the most significant example of a general purpose system for lexical knowledge 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 language engineering applications, such as information retrieval [Voo94]. In WordNet, each set of words that are regarded as 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 word sense should be used. Moreover, there is no logical reasoning with the relations in the semantic network. This problem led to the notion of plausible inference with WordNet [HM98]. The dex framework can be directly applied to formalize this notion of plausible inference.

A more sophisticated system being developed with a deeper knowledgebase on several thousand word senses is FrameNet [FA98]. This is a frame-based representation with logical reasoning limited to inheritance, where one frame is an elaboration of another, and composition, where one frame is built-up of other frames as the parts. As with WordNet there is no explicit machinery for determining in which context a particular word sense should be used,

A richer knowledge representation and reasoning framework is the generative lexicon [PB93, Pus95]. This is a form of frame-based representation that is based on specifying the following features for a word:

- Argument structure: providing conventional information about the syntax of a word.
- Event structure: describing a word in terms of states, processes, and transitions.
- Qualia structure: describing a word's meaning (or qualia) in terms of the following aspects of the concept it captures:
  - Constitutive: the relation between the concept and its constituent parts.
  - Formal: information that distinguishes the concept within a larger domain.
  - Telic: information on the purpose and function of the concept.
  - Agentive: factors involved in the origin or generation of the concept.
- Lexical inheritance: determines the relation of a word to other words in a lexicon.

The generative lexicon framework does offer context-dependent reasoning — though predominantly this is classical — so for a word disambiguation different word senses may be determined using the extra constraints in the qualia structure. Whilst the generative lexicon is a rich and powerful framework, the approach is limited in context-dependent reasoning, supporting only a form of inheritance, and in particular it does not support uncertainty management.

Some machine-readable lexicons have been developed in formal knowledge representation frameworks, supported by logical reasoning. Key formal frameworks are DATR [EG96, EG89], Laurel [Cop92, BCP94], and persistent default unification (PDU) [LBAC95]. These are forms of inheritance hierarchy that offer an efficient and lucid representation of lexical knowledge. However, they are limited formalisms in terms of expressibility and inferential capability, in particular with respect to context-dependent reasoning, and reasoning with inferences. They don't provide any mechanisms for determing the context for a given ambiguous word or for reasoning with context information. Also preferences used in the default reasoning are restricted to a form of specificity. There is no facility for preferences based on reasoning about exceptions or for explicit priorities.

There are some proposals to use logics for lexical knowledge representation and reasoning. An epistemic logic has been used as the basis of a framework for reasoning about context for resolving lexical ambiguity [Buv96]. Unfortunately, the reasoning is monotonic and so offers no context-dependent default reasoning.

A new variant of default logic has been proposed for reasoning about ambiguity according to context [Poe96]. However, there is no mechanism for determining context for a given ambiguous word, and the approach is limited in reasoning about contexts. There are also open questions about the behaviour of the default logic with regard to automated reasoning.

Another modal logic for disambiguation has been proposed in [Fer97] that does provide default reasoning. This framework is interesting as an abstract framework. But, there are open questions with regard to its practical application. Again the reasoning about contexts is limited and there are no implicit or explicit prioritization mechanisms over context-dependent rules for disambiguation. There is a mechanism for determining a context for a given ambiguous word, but this is also limited to explicitly listing the presence or absence of individual words in the sequence of words to the left and right of the ambiguous word for each disambiguation rule. This does not take advantage of the structure that exists in the lexical knowledge.

So to conclude this comparison, despite the extensive developments in lexical knowledgebases, there is a pressing need for a formal approach to knowledge representation and reasoning that can handle the context-dependent default knowledge, and in particular for addressing uncertainty management with contexts. To address the shortcomings of existing approaches, we believe that using default logic together with context classification trees is a promising direction for developing lexical knowledgebases.

6 Discussion

In this paper, we have presented the dex framework: 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. Reasoning with a dex 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.

Elsewhere machine learning techniques have also been successfully applied to text categorization. In particular, rule induction with rule refinement techniques has been been applied to generating text categorization rules using large training and test sets (around 8000 examples) of news articles from the Reuters newswire [ADW94]. Given the close relationship between the function of text categorization rules, and context classification trees, the results indicate how the generation of context classification trees could be scaled-up using refinement techniques.

Our next goal in developing the dex framework is to support 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, NS98].

We also want to refine the notion of a source to allow better scoping. To illustrate the need, consider the following extreme example that is a zeugma taken from [LCB96].

```
John banked the money and then the plane.
```

The context of the first occurrence of `banked` could be described as `finance` and of the second (implicit) occurrence could be described as `aviation`. Whilst this is an extreme example, it does indicate that the larger the source, the more likely that the set of words will refer to disjoint contexts. Managing the size of the source is therefore an important issue. It raises interesting questions like how to differentiate noise in determining a context from more complex structure in the contexts that pertain to a piece of text. So for example, how does context normally change in reading a sentence or set of sentences, and how does it move between more specialized and more generalized contexts.

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 inductive logic programming ([Mug92, Cus97, Cus98, TM98]) and statistical techniques (such as [CHS93]), to generate formulae for a default theory for a domain; (2) Using co-locational data with decision tree learning based on ID3 or C4.5, or co-locational data with unsupervised learning techniques such as in [Yar95]; and (3) Using machine-readable dictionaries and thesauri [NFE90, Mei93]. Whilst none of these techniques use a hypothesis language that is as expressive as default logic, they are potentially very useful in identifying relationships and generating rules that could be incorporated within default logic.

Acknowledgements

The author would like to thank Beatrice Duval, Roger Evans, Ruth Kempson, Rodger Kibble, Lutz Marten, Wilfred Viol-Meyer, Pascal Nicolas, and Torsten Schaub, for some very helpful discussions that led to the work reported in this paper. The author is also grateful to the two anonymous referees for some useful suggestions.

References

[ADW94] C Apte, F Damerau, and S Weiss. Automated learning of decision rules for text categorization. *ACM Transactions on Information Systems*, 12(3):233–251, 1994.

[Ant97] G Antoniou. *Non-monotonic Reasoning*. MIT Press, 1997.

[BCP94] T Briscoe, A Copestake, and V de Paiva, editors. *Inheritance, Defaults and the Lexicon.* Studies in Natural Language Processing. Cambridge University Prees, 1994.

[BDK97] G Brewka, J Dix, and K Konolige. *Non-monotonic Reasoning: An Overview.* CLSI Publications, 1997.

[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.

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

[Buv96] 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, 1996.

[CHS93] J Cussens, A Hunter, and A Srinivasan. Generating explicit orderings for non-monotonic logics. In *Proceedings of the Eleventh National Conference on Artificial Intelligence (AAAI'93)*, pages 420–425. MIT Press, 1993.

[Cop92] A Copestake. The representation of lexical semantic information. Technical report, University of Sussex, 1992. CSRP 280.

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

[Cus97] J Cussens. Parts-of-speech tagging using progol. In *Proceedings of the Inductive Logic Programming Conference(ILP'97)*, 1997.

[Cus98] J Cussens. Using prior probabilities and density estimation for relational classification. In *Proceedings of the Inductive Logic Programming Conference(ILP'98)*, 1998.

[EG89] R Evans and G Gazdar. Inference in DATR. In *Proceedings of the 4th Conference of European Chapter of the Association of Computational Linguistics (EACL89)*, pages 66–71, 1989.

[EG96] R Evans and G Gazdar. DATR: A language for lexical knowledge representation. *Computational Linguistics*, 22:167–216, 1996.

[FA98] C Fillmore and B Atkins. Framenet and lexicongraphic relevance. In *Proceedings of the First International Conference on Language Resources and Evaluation*, 1998.

[Fer97] T Fernando. A modal logic for non-determinsitic disambiguation. In *Proceedings of the Amsterdam Colloquium*, pages 121–126, 1997.

[Ger95] P Gervas. *Logical considerations in the interpretation of presuppositional sentences.* PhD thesis, Department of Computer Science, Imperial College, 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.

[Gre96] G Green. Ambiguity resolution and discourse interpretation. In K van Deemter and S Peters, editors, *Semantic Ambiguity and Underspecification*, pages 1–26. CSLI Publications, 1996.

[Gut93] L Gutherie. A note on lexical disambiguation. In C Souter and E Atwell, editors, *Corpus-based computational linguistics*, pages 227–237. Rodopi, 1993.

[Hir87] G Hirst. *Semantic Interpretation and the Resolution of Ambiguity.* Studies in Natural Language Processing. Cambridge University Press, 1987.

[HM98] S Harabagiu and D Moldovan. Knowledge processing on an extended wordnet. In C Fellbaum, editor, *WordNet: An Electronic Lexical Database*, pages 379–405. MIT Press, 1998.

[HS93] C Hwang and L Schubert. EL: A formal, yet natural, comprehensive knowledge representation. In *Proceedings of the Eleventh National Conference on Artificial Intelligence (AAAI'93)*, pages 676–682. MIT Press, 1993.

[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.

[Hun97] A. Hunter. Using default logic for lexical knowledge. In *Qualitative and Quantitative Practical Reasoning*, volume 1244 of *Lecture Notes in Computer Science*, pages 86–95. Springer, 1997.

[LBAC95] A Lascarides, T Briscoe, N Asher, and A Copestake. Order independent and persistent typed default unification. *Linguistics and Philosophy*, 19(1):1–90, 1995.

[LCB96] A Lascarides, A Copestake, and T Briscoe. Ambiguity and coherence. *Journal of Semantics*, 13(1):41–65, 1996.

[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.

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

[Mer91] B Mercer. Presuppositions and default reasoning: A study in lexical pragmatics. In J Pustejovsky and S Bergler, editors, *Lexical Semantics and Knowledge Representation*, volume 627 of *Lecture Notes in Artificial Intelligence*, pages 321–340. Springer, 1991.

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

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

[NFE90] J Nutter, E Fox, and M Evens. Building a lexicon from machine-readable dictionaries for improved information retrieval. *Literary and Linguistic Computing*, 5(2):129–137, 1990.

[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.

[NS98] P Nicolas and T Schaub. The XRay system: An implementation platform for local query answering in default logics. In A Hunter and S Parsons, editors, *Applications of Uncertainty Formalisms*, Lecture Notes in Computer Science, pages 354–378. Springer, 1998.

[PB93] J Pustejovsky and B Boguraev. Lexical knowledge and natural language processing. *Artificial Intelligence*, 63:193–223, 1993.

[Poe96] M Poesio. Semantic ambiguity and perceived ambiguity. In K van Deemter and S Peters, editors, *Semantic Ambiguity and Underspecification*, pages 159–202. CSLI Publications, 1996.

[Pus95] J Pustejovsky. *The Generative Lexicon.* MIT Press, 1995.

[Qua93] J Quantz. A preferential default description logic for disambiguation in natural language processing. In G Brewka and C Witteveen, editors, *Proceedings of the Dutch-German Workshop on Nonmonotonic Reasoning Techniques and their Applications*, pages 1–13. RWTH Aachen, 1993.

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

[Qui93] J Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 1993.

[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.

[Sch98] T Schaub. *The Automation of Reasoning with Incomplete Information: From Semantic Foundation to Efficient Computation.* Springer, 1998.

[Spa86] K Spark-Jones. *Synonym and Semantic Classification.* Edinburgh Information Technology Series. Edinburgh University Press, 1986.

[TM98] C Thompson and R Mooney. Semantic lexicon acquisition for learning natural language interfaces. Technical Report TR AI98-273, AI Lab, University of Texas at Austin, 1998.

[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.

[vR79] C van Rijsbergen. *Information Retrieval.* Cambridge University Press, 1979.

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

[Yar95] D Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 32nd Annual Meeting of the Association of Computational Lingustics*, pages 189–196, 1995.