Learning Lexical Inheritance Hierarchies with Maximum Entropy Models

Caroline Sporleder,
University of Edinburgh,
Edinburgh, EH8 9LW, UK
csporled@cogsci.ed.ac.uk

Abstract

Lexical inheritance hierarchies are used widely as a means of representing lexical information efficiently by capturing generalisations. But so far there have been few attempts to construct them automatically. This paper presents a two-step construction algorithm in which a Galois lattice is built from an input lexicon and then pruned into an inheritance hierarchy. The pruning step uses a maximum entropy model. This is compared to two other pruning methods: one which minimises the the number of attribute-value pairs in the hierarchy and one which prunes the hierarchy randomly. To asses the performance of the three pruning methods an automatic evaluation method has been implemented.

1 Introduction

Grammar development over the last decades has seen a shift away from large inventories of grammar rules to richer lexical structures. As a consequence, lexicons developed within modern grammar theories like *Head-Driven Phrase Structure Grammar* (Pollard & Sag 1994 [11]) and *Categorial Grammar* (Wood 1993 [16], Steedman 1996 [15]) can be very complex. Representing lexical information as a simple list of lexical entries results in an undesirable amount of redundancy. For example, the verbs *like*, *admire* and *take* all have in common that they usually require an object and that that object is in accusative case. Rather than specifying this information for each of the verbs individually it makes sense to represent information in an inheritance hierarchy. Properties of transitive verbs then only have to be specified at their superclass `transitive verb`[1] and can be inherited by individual transitive verbs.

Usually, inheritance hierarchies are constructed by hand but this is time-consuming and often impractical for big lexicons. In addition, automatic extraction of lexical information from corpora is becoming more and more popular and it makes sense to go a step further and automate the hierarchical organisation of lexical data too. Constructing hierarchies automatically or semi-automatically has the further advantage of facilitating a more systematic analysis of the lexical data and in the ideal case it might even lead to the discovery of trends in the data set that would otherwise go unnoticed.

This paper describes an approach to learning inheritance hierarchies from unstructured lexicons. That is, the input will be a list of lexical entries, where each entry is characterised by a set of attribute-value pairs. The output will be a lexical inheritance hierarchy. The output hierarchy can employ multiple inheritance, i.e. a node can inherit from more than one node. The paper deals only with monotonic

[1] In this paper I will use typewriter font for classes and types.

inheritance, i.e. a node is not allowed to override properties specified for its ancestors, but an extension to non-monotonic hierarchies is possible. The aim is to derive *linguistically plausible* hierarchies, i.e. the nodes in the hierarchy should correspond to linguistically meaningful classes.

The approach described here splits the task into two steps: First, a *Galois lattice* (also called *concept lattice*) is built for the lexicon. This encodes all possible word classes together with their properties. The lattice is then pruned into an inheritance hierarchy. This approach reduces the automatic construction of a hierarchy to a classification problem: Finding a good learning method amounts to finding a classifier that correctly predicts which nodes in the lattice should be pruned and which should be retained.

Using a Galois lattice to construct inheritance hierarchies is not new. Godin 1998 *et al.* [7] used it to built class hierarchies for object-oriented programming. For lexical inheritance hierarchies it has been proposed independently by Petersen 2001 [10]. However, Godin *et al.* and Petersen both prune the lattice into a hierarchy that is minimal redundant with respect to some notion of minimal redundancy. This paper describes a different approach. It is argued that focusing solely on minimal redundancy may not be beneficial if one aims to derive linguistically plausible hierarchies. It may be much more important to base the pruning on interdependencies in the data set. This requires a sophisticated pruning method that allows one to take several contextual properties into account. For this purpose a maximum entropy (see Ratnaparkhi 1998 [12]) based pruning method is suggested.

This paper presents some experiments in which a simple maximum entropy pruning method is compared to Petersen's minimal pruning and a random pruning method. To assess the performance of each pruning method an automatic evaluation method is needed. In previous approaches to lexical inheritance hierarchy learning, hierarchies have always been evaluated manually, i.e. by looking at the derived hierarchy and a manually-built hierarchy for the same lexicon. This is not feasible for big lexicons. Therefore I will also present an automatic evaluation method for the task. The evaluation method matches a derived hierarchy to a manually-built one, assesses their similarity and then calculates precision and recall.

The paper starts by giving an overview of previous approaches to the task. Section 3 gives a brief motivation why something more than simple redundancy measures might be needed to derive plausible hierarchies. Section 4 describes the learning architecture and evaluation method. Section 5 introduces the maximum entropy model. And finally, section 6 describes the experiments.

2 Background

So far there has been little work on the automatic construction of lexical inheritance hierarchies. The most comprehensive work to date is Barg 1996 [1]. Barg presents an algorithm for learning DATR hierarchies (Evans & Gazdar 1990 [5]) using a transformation-based approach. The search through the space of permissible transformations is guided by a set of evaluation criteria, which are ordered by priority. The criteria and their priority may vary for different linguistic domains. Most criteria refer to the size and complexity of a hierarchy.

Cahill [2] presents a semi-automatic method for building DATR hierarchies for multi-lingual lexicons. She focuses on inferring morphological and phonological generalisations across languages. These generalisations are extracted automatically by applying a pattern-matching algorithm to the orthographic and phonological forms of lexical entries while the general structure of the hierarchy (i.e. the partial order of nodes) is hard-wired.

Light 1994 [8] does not build hierarchies from scratch but investigates ways in which new entries can be inserted into an existing hierarchy. He uses a greedy algorithm that inserts entries by minimising a redundancy criterion based on the number of parents and attribute-value pairs of a node. Light's algorithm does not create new (non-terminal) classes and therefore cannot be used to learn a hierarchy from scratch.

Finally, Petersen 2001 [10] independently proposed an approach that is similar to the one taken here. She, too, builds a Galois lattice for the input lexicon and prunes it into a hierarchy. But she uses a non-stochastic pruning method which creates hierarchies where every attribute-value pair occurs only once.

3 Minimal Redundancy and Linguistic Plausibility

Most previous approaches focus on some notion of "minimal redundancy", i.e. they aim to derive hierarchies that are minimal with respect to some definition of redundancy. However minimal redundancy is not well defined in the context of inheritance hierarchies and different redundancy criteria often conflict with each other. For instance, reducing the number of attribute-value pairs often means increasing the number of nodes and *vice versa*. Previous approaches have addressed this problem by restricting the definition of minimal redundancy to one criterion (Petersen 2001 [10]), by weighting (Light 1994 [8]) or ordering potentially conflicting criteria (Barg 1996 [1]).

Furthermore, while it is one of the purposes of an inheritance hierarchy to reduce redundancy, this is not the whole story. Hierarchies which are minimally redundant with respect to some simple notion of redundancy are not necessarily "good" hierarchies, since hierarchies are also required to be linguistically plausible, i.e. the classes they contain should be linguistically plausible generalisations. Unfortunately minimal redundancy and linguistic plausibility do not necessarily coincide.

For example, look at the partial hierarchy in figure 1.[2] This hierarchy is not minimally redundant with respect to its attribute-value pairs. Thus the attribute-value pair SYN|SPR:<>[3] occurs three times: in the types PREP, PROPER NOUN and DET. Making this hierarchy minimally redundant with respect to attribute-value pairs would require the creation of a new supertype from which prepositions, proper nouns and determiners inherit their empty specifier list. But the class `preposition + proper noun + determiner` is hardly linguistically plausible.

Modelling linguistic plausibility without feeding in background knowledge[4] is not trivial, but properties of the data can give some hints as to whether a class is likely to be plausible or not. Thus plausible classes are usually fairly homogeneous, i.e. the terminal nodes inheriting from them often have more than one thing in common. For example, in the hierarchy in figure 2, transitive verbs have in common that they are not auxiliaries (SYN|HEAD|AUX:*false*),[5] that they have a direct object, that this direct object is a noun, needs to be in accusative case (SYN|COMPS|SYN|HEAD|CASE:*acc*), and is semantically

[2] All hierarchies in this paper are included for illustration only. While they are all based to some extent on "real", manually-built hierarchies they are usually only partially shown and omit crucial detail. For example, the fact that none of the hierarchies in this paper uses multiple inheritance is purely coincidental. Multiple inheritance is very important in linguistic classification and the system proposed here can built multiple inheritance hierarchies.
The hierarchy in figures 1 and 2 is loosely based on the so-called *pacifier* grammar, which is supplied with the LKB system (see Copestake [4]).

[3] The attribute SPR stands for *specifier* and is generally used in HPSG grammars to refer to the subject of a verb or the determiner of a noun etc. The attribute SYN abbreviates *syntax*.

[4] It would be possible to equip the system with some knowledge that some classes (e.g. *transitive verb*, *intransitive verb* etc.) are linguistically plausible but this would hamper the system's ability to (i) be applicable across languages (and across lexicons) and (ii) discover interesting new generalisations.

[5] Since the hierarchy is monotonic it is not possibly to specify AUX:*false* as a default at the verb level.

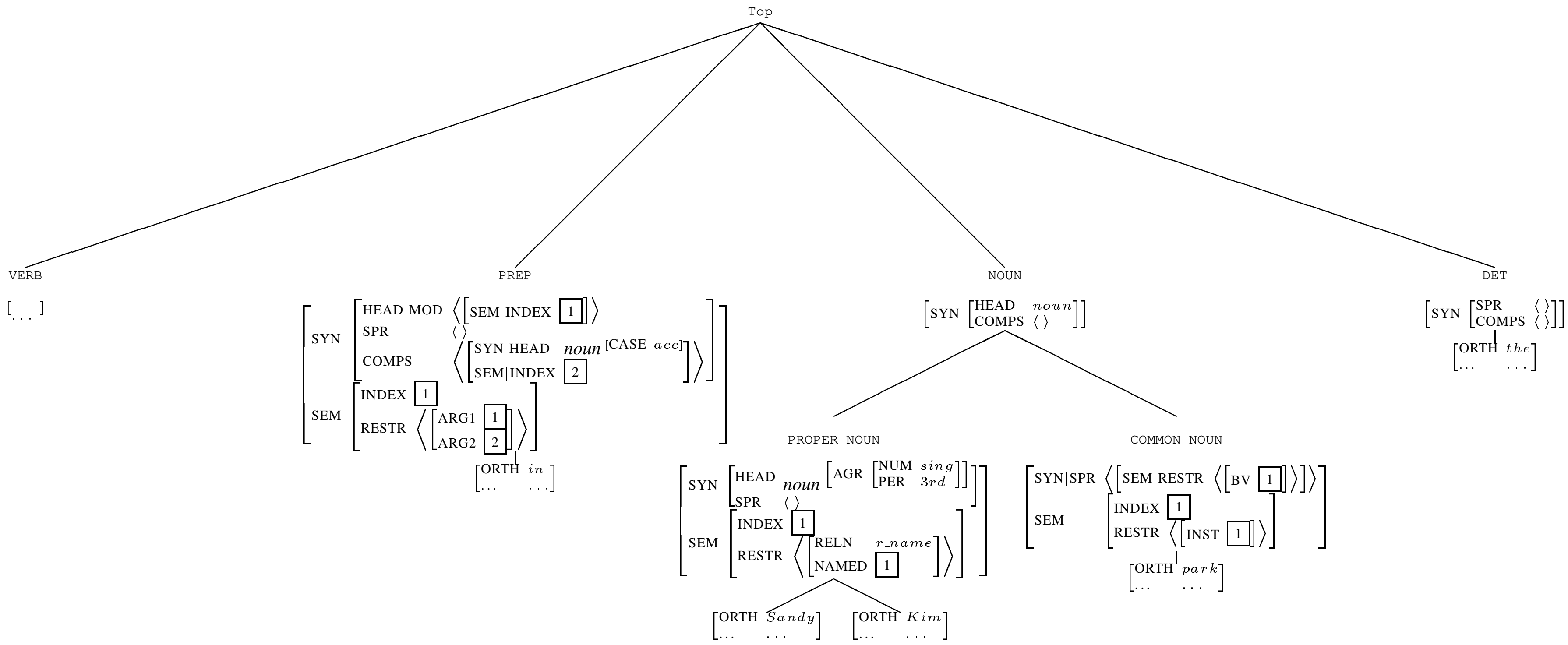

Figure 1: Part of an inheritance hierarchy

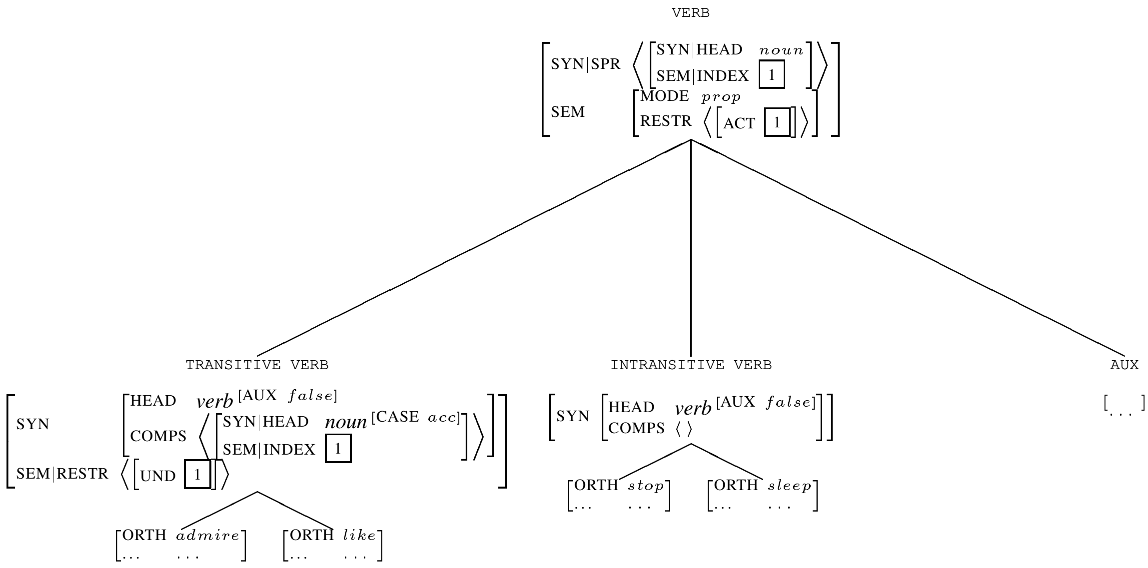

Figure 2: “Verb” Part of the hierarchy in figure 1

the “undergoer” of the action of the verb (SYN|COMPS|SEM|INDEX: $\boxed{1}$ and SEM|RESTR|UND: $\boxed{1}$). Likewise all verbs (in the hierarchy) have in common that their head is a noun, that their semantic mode is *proposition* and that the entity referred to by their subject is the “actor” of the action of the verb.

While not all linguistically plausible classes may specify more than one property that is common for their members, the number of attribute-value pairs is often a useful indicator for linguistic plausibility. Another indicator for linguistic plausibility is the dependence of a node’s attributes. Thus a node whose attributes are completely independent, i.e. co-occur in any combination of values, may generalise less well than a node whose attributes are not independent.

These indicators for linguistic plausibility are probably not statistically independent and interact in a complex fashion. For this reason it seems that maximum entropy models may be a good way of combining them.

4 Overview of the System

The system is implemented for LKB lexicons (Copestake 1999 [4]). Figure 3 gives an overview of the system. First, the manually-built lexical hierarchy is compiled out, i.e. inheritance is applied to all nodes, resulting in a set of fully-specified lexical entries. The compiled-out lexicon is then pre-processed. The main task of the pre-processing step is to *de-type* the feature structures (see section 4.1). Then a Galois lattice is constructed (see section 4.2). This lattice is pruned into an (untyped) inheritance hierarchy. To evaluate the automatically constructed hierarchy, it is matched to the manually-built hierarchy (see section 4.4). For the hierarchies to be comparable the manually-built hierarchy has to be de-typed as well. Once the two hierarchies have been matched, *precision* and *recall* can be calculated.

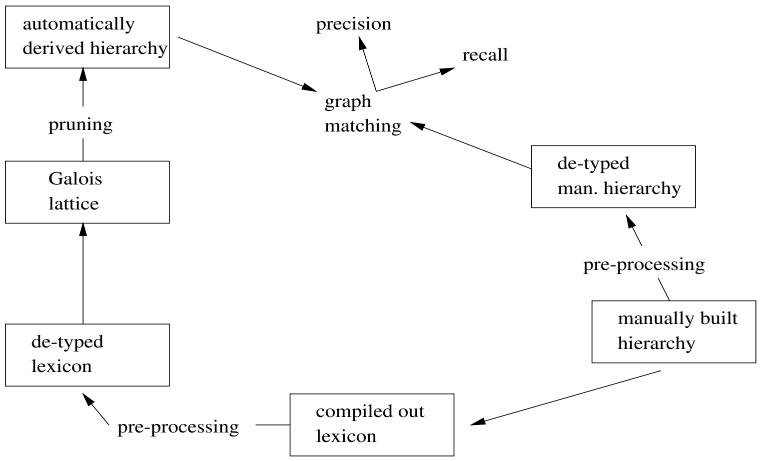

Figure 3: System overview

4.1 Pre-Processing

The pre-processing step is applied to both the compiled-out lexicon and the manually-built hierarchy (before it is matched to the automatically derived hierarchy). The main pre-processing task is to de-type all feature structures. This is necessary because the Galois lattice construction algorithm requires each lexical entry to be a set of attribute-value pairs, where the values have to be *atomic*.[6] Features structures in LKB lexicons, however, are typed and therefore potentially non-atomic. The pre-processing step converts typed features structures into atomic feature structures. This is done by unifying non-atomic values with the feature structures they abbreviate and then "flattening" the resulting feature-structure by removing references to non-atomic types. For example, in the lexical entry for *fish* (figure 4a)[7], the non-atomic value *noun* is unified with the feature structure in figure 4b and then flattened, resulting in the feature structure in figure 5.

$$\begin{bmatrix} \text{ORTH} & fish \\ \text{SYN} & {}_{\textit{gram\_cat}}\begin{bmatrix} \text{HEAD} & {}_{\textit{noun}}\begin{bmatrix} \text{AGR}|\text{PER} & 3rd \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

(a) *fish*

$${}_{\textit{noun}}\begin{bmatrix} \text{AGR} & {}_{\textit{agr\_cat}}\begin{bmatrix} \text{PER} & per \\ \text{NUM} & num \\ \text{GEND} & gend \end{bmatrix} \\ \text{CASE} & case\_cat \end{bmatrix}$$

(b) *noun*

Figure 4: Typed feature structures

$$\begin{bmatrix} \text{ORTH} & fish \\ \text{SYN}|\text{HEAD}|\text{AGR}|\text{PER} & 3rd \\ \text{SYN}|\text{HEAD}|\text{AGR}|\text{NUM} & num \\ \text{SYN}|\text{HEAD}|\text{AGR}|\text{GEND} & gend \\ \text{SYN}|\text{HEAD}|\text{CASE} & case\_cat \end{bmatrix}$$

Figure 5: De-typed and flattened feature structure for *fish*

[6] An *atomic* value is a value that is basic and not an abbreviation for a feature structure.
[7] Some details are omitted from the lexical entry.

Note, that de-typing in this case means that information about the part-of-speech of *fish* is lost because the fact that *fish* is a noun is only encoded in the type of the feature-structure *head*. One way to avoid information loss is by adding the type of each feature-structure as a new attribute-value pair, e.g. SYN|HEAD|TYPE:*noun*. However, adding in this type information would significantly increase the number of attribute-value pairs in the lexicon, making it more difficult to construct a Galois lattice. And in many cases it is safe to remove reference to non-atomic types. Thus the fact that the feature structure *syn* is of type *gram_cat* or that the feature structure *per* is of type *per* is not particularly relevant in this context. For these reasons the pre-processing step currently does not keep reference to non-atomic types.[8]

Re-entrancies are also re-represented as atomic feature structures in the pre-processing step. Thus the feature structure in figure 6a is converted into the feature structure in figure 6b.

$$\begin{bmatrix} \mathrm{F} & \boxed{1}\ x \\ \mathrm{G} & \boxed{1} \\ \mathrm{H} & \boxed{1} \\ \mathrm{I} & \boxed{2}\ y \\ \mathrm{J} & \boxed{2} \end{bmatrix} \quad \Longrightarrow \quad \begin{bmatrix} \mathrm{F} & x \\ \mathrm{G} & x \\ \mathrm{H} & x \\ \mathrm{I} & y \\ \mathrm{J} & y \\ \mathrm{REENTR} & \{F, G, H\} \\ \mathrm{REENTR} & \{I, J\} \end{bmatrix}$$

(a) (b)

Figure 6: Re-representation of re-entrancies

Another pre-processing step involves the removal of lexical and grammar rules, which are often incorporated in HPSG lexicons. Rules are removed because they are not part of the core lexicon and dealing with them potentially requires different techniques.

4.2 Galois Lattices

Given a set of instances E (i.e. lexical entries), a set of features (i.e. the different attribute-value pairs describing the entries) E', and a binary relation $R \to E \times E'$, a Galois lattice is a partial order of all pairs of the form (X, X'), where $X \subseteq E$, $X' \subseteq E'$ such that:

$$X = \{x \in E | \forall x' \in X', xRx'\}$$
$$X' = \{x' \in E' | \forall x \in X, xRx'\}$$

The pair (X, X') is called a *concept*, where X is the *extension* of the concept (i.e. the set of lexical entries it comprises) and X' is its *intension* (i.e. the set of attribute-value pairs that are shared by all members of the concept's extension).

For instance, figure 7 shows a small lexicon which contains information about the nativeness and the ability to carry stress for six German prefixes. Figure 8 shows the Galois lattice for the lexicon. Concepts contained in the lexicon and the lattice are, for example:

[8] A possible extension for the future would be to try to distinguish between relevant and irrelevant non-atomic types and to keep the former. One clue for determining the relative relevance of a type is its position in the type hierarchy. Value types that are maximally general —like *gram_cat* and *per*— tend to be irrelevant. While important types —like *noun*— are usually not maximally general. The most general value type for the HEAD attribute is *pos* (*part-of-speech*).

- the concept of `stressed-native-prefixes` (Node 5):
({*un*, *ent*}, {NAT:+, STR:+})

- the concept of `stressed-prefixes` (Node 3):
({*un*, *ent*, *dis*, *des*}, {STR:+})

- the concept of `unstressed-native-prefixes-whose-orthography-is-ver` (Node 7):
({*ver*}, {NAT:+, STR:-, ORTH:ver-})

	er	*ver*	*un*	*ent*	*dis*	*des*
Orth	er-	ver-	un-	ent-	dis-	des-
Nat(ive)	+	+	+	+	-	-
Str(ess)	-	-	+	+	+	+

Figure 7: German prefixes: lexicon

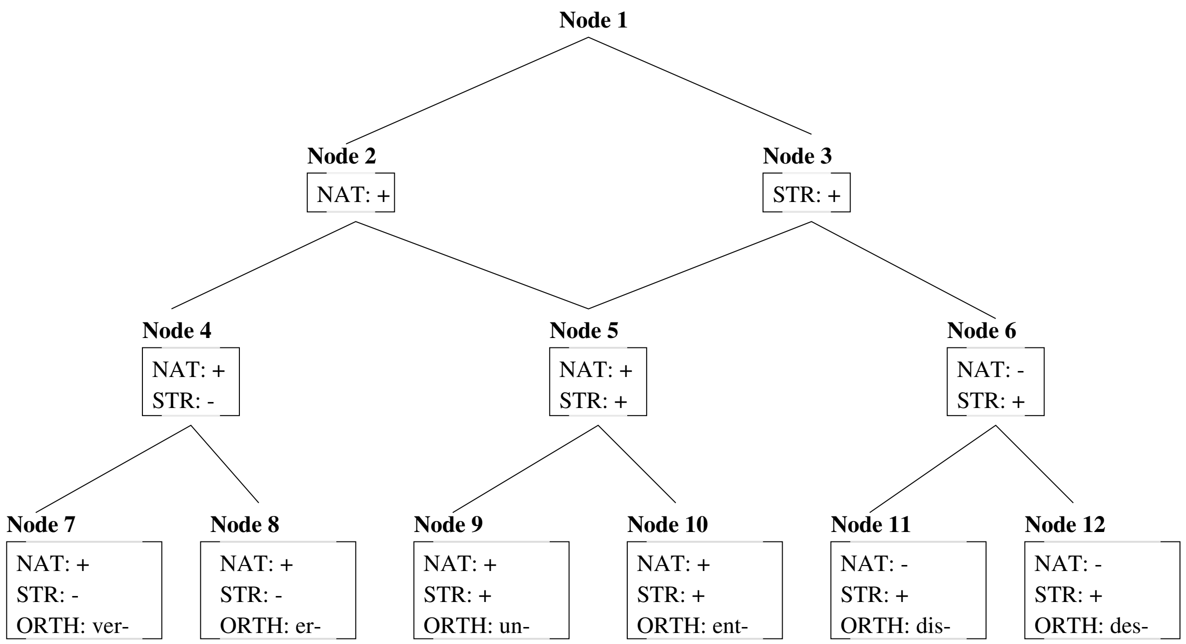

Figure 8: German prefixes: Galois lattice

Using Galois lattices for lexical inheritance construction is motivated by the assumption that every node of a (monotonic) inheritance hierarchy corresponds to a node in the Galois lattice derived from the same lexicon. Two nodes correspond if they have the same extension but they may differ in their intension because attribute-value pairs are not repeated within a given inheritance path in (monotonic) inheritance hierarchies, but are repeated in a Galois lattice.

This is illustrated by the manually-built hierarchy for the prefix lexicon (figure 9). Nodes 4, 5, and 6 are identical to their counterparts in the Galois lattice. Node 7 in the manually-built hierarchy has the same extension as Node 7 in the Galois lattice (namely *ver*)[9] but the attribute-value pairs NAT:+ and STR:- are lacking from its intension because they can be inherited from Node 4.

[9] Extensions are not shown in the lattice and hierarchy but should be obvious.

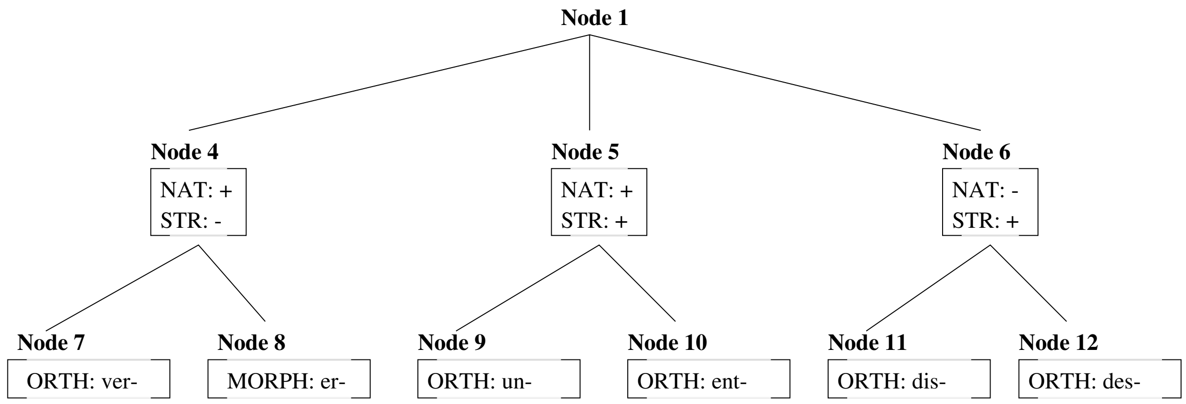

Figure 9: German prefixes: manually-built hierarchy

The assumption that each node of a hierarchy is contained in the corresponding Galois lattice is not strictly true. Thus while every class in the lattice has at least two subclasses this is not necessarily the case for inheritance hierarchies. A class in an inheritance hierarchy may sometimes only have one subclass. This is due to data sparseness, i.e. a class is included in the hierarchy because the linguist constructing the hierarchy knows that this class is a useful generalisation over several subclasses even if the lexicon contains only members of one of these subclasses. These classes are a problem for every data-driven machine learning technique but one would hope that reasonably big lexicons contain only a small proportion of single-child classes.[10]

4.3 Post-Pruning

Some pruning methods may result in attribute-value pairs occurring more than once in an inheritance path. These repetitions are redundant in (monotonic) inheritance hierarchies and are therefore removed in a post-pruning step. Post-pruning traverses the hierarchy top-down and deletes all attribute-value pairs of a given node that can be inherited from the node's ancestors. This may result in empty nodes, i.e. nodes which inherit all their attribute-value pairs from their ancestors. These empty nodes are subsequently removed from the hierarchy.

4.4 Evaluation

Assessing a pruning method involves evaluating the hierarchies derived by it. Previous approaches to lexical inheritance hierarchy learning used a manual evaluation technique, i.e. the derived hierarchies were inspected and manually evaluated for plausibility. The easiest way to evaluate hierarchies automatically is to count wrongly classified nodes (i.e. false positives and false negatives) and calculate precision and recall on this basis. However, it is possible that some wrongly classified nodes are worse than others. Even if few nodes are correctly retained, the derived hierarchy may still be relatively good if its nodes are very similar to the nodes in a good hierarchy.

[10]Non-monotonic inheritance poses another problem for Galois lattices since the latter are inherently monotonic. One way in which this problem can be addressed is by identifying a default value for every attribute-value pair and ensuring that every lexical entry contains the default value as well as the non-default value for each of its attribute-value pairs. A lattice derived from a lexicon modified in this way can be pruned into a non-monotonic hierarchy.

The approach adopted here uses a *error-tolerant* graph matching algorithm to match a derived hierarchy to the corresponding manually built one. Because the matching is error-tolerant it allows for imperfect matches. This requires (i) a measure of the similarity between two graphs or their components and (ii) a threshold which specifies how similar two components have to be before they can be matched.

Since hierarchies derived by the pruning methods discussed below are guaranteed to be *sound*, i.e. compiling them out will result in the same lexicon as compiling out the manually-built hierarchy, it is not necessary to test for soundness in the evaluation step.

The matching algorithm matches nodes. Node similarity (*ns*) is based on the extensional (*eol*) and intensional (*iol*) overlap between two nodes:[11]

$$ns = (0.5 \times eol) + (0.5 \times iol)$$

Where extensional and intensional overlap are defined as:

$$eol = \frac{\#\ of\ common\ terminal\ descendants \times 2}{\#\ tdesc\ n + \#\ tdesc\ n'}$$

($\#\ tdesc\ n$ and $\#\ tdesc\ n'$ are the number of terminal descendants of node n and node n', respectively)

$$iol = \frac{\#\ of\ shared\ attribute{-}value\ pairs \times 2}{\#\ avps\ n + \#\ avps\ n'}$$

($\#\ avps\ n$ and $\#\ avps\ n'$ are the number of attribute-value pairs of node n and node n', respectively)

The graph matching algorithm starts by matching the terminal nodes of the manually-built hierarchy to the terminal nodes of the derived hierarchy. This is fairly straightforward since every terminal node contains an attribute-value pair which encodes an ID number by which it can be uniquely identified. And ID numbers are identical in both hierarchies. The algorithm then traverses the manually-built hierarchy bottom-up, comparing each non-terminal node in the manually-built hierarchy to each non-terminal node in the derived hierarchy. The *n* best matches for a node are remembered, provided their similarity value lies above the user-defined threshold and neither their extensional nor their intensional overlap are zero.[12] The similarity threshold was set to 0.5 in the experiments discussed below.[13]

Once all nodes in the manually-built hierarchy have been considered the matching is made unique in the other direction, i.e. one has to ensure that no node in the automatically constructed hierarchy is

[11]The two different types of overlap are weighted. One might argue that it might make sense to assign a higher weight to the *extensional overlap* since a node's intension is influenced by non-local properties of the hierarchy. That is, a wrongly retained (or pruned) node will affect the intension of at least one of its descendants whereas this is not the case for the extension of a node. So in a way a difference in extension is more grave than a difference in intension since it means that the two nodes are derived from different Galois nodes. One might argue that the extensional overlap should be 100% for two nodes to be matched. However, I think this might be too strict because it may be that instead of retaining node *x* the pruning algorithm retained node *y*, one of *x*'s immediate descendants or ancestors in the Galois lattice. If nodes *x* and *y* are very similar the pruning method should still be given some credit. The weights associated with the two kinds of overlap allow one to make *node-similarity* more dependent on extensional overlap. At the moment, however, the weights are set to 0.5 on the basis that there is currently no real empirical evidence suggesting a different set of weights.

[12]*n* was set to 10 in the experiments, but in practice a node could maximally be matched to 2 other nodes. For all other potential matches the node similarity was below the threshold.

[13]The value 0.5 was chosen somewhat arbitrarily. In general, the threshold should not be too low since this would mean that nodes are matched in an relatively arbitrary fashion. Choosing a threshold below 0.5 therefore does not seem like a good idea. On the other hand the threshold should not be too high either as this makes the matching less error-tolerant and might lead to some relatively well learned nodes remaining unmatched. 0.5 seemed like a good initial guess but it may be that a higher threshold might be better in the long run.

matched twice. A greedy search is employed for this optimisation: Conflicts are resolved by comparing the best match for every node to the best match for every other node. If the best match for two different nodes in the manually-built hierarchy involves the same node in the automatically constructed hierarchy, the algorithm chooses the match with the higher node similarity and backs off to the second best match for the other node if possible. If backing off creates a new conflict this new conflict is resolved in a similar fashion. The algorithm stops when there are no conflicts left. In practice, conflicts occurred in less than 1% of the nodes if a threshold of 0.5 was used.

The overall precision (*prec*) of the learning algorithm is then defined as the average of two basic precision measures: *attribute-value pair precision* (*avp-prec*), which measures how well the attribute-value pairs in the derived hierarchy represent the attribute-value pairs in the manually-built hierarchy, and *node precision* (*node-prec*), which measures the percentage of nodes in the derived hierarchy that could be matched:[14]

$$\textit{avp-prec} = \frac{\#\ matched\ avps}{\#\ avps\ in\ derived\ hierarchy}$$

$$\textit{node-prec} = \frac{\#\ matched\ nodes}{\#\ nodes\ in\ derived\ hierarchy}$$

$$\textit{prec} = \frac{(avp\text{-}prec)+(node\text{-}prec)}{2}$$

Attribute-value pair precision and *node precision* have different strengths and weaknesses, which can be levelled out by combining the two measures to form an overall precision measure. Thus, node precision depends largely on the matching threshold and once two nodes have been matched the quality of the match is disregarded, while attribute-value pair precision depends to a lesser degree on the threshold as it takes the overall quality of matches into account but it disregards the number of nodes in the hierarchy.

Overall recall (*rec*) is calculated in the same fashion, based on *attribute-value pair recall* (*avp-rec*) and *node recall* (*node-rec*):

$$\textit{avp-rec} = \frac{\#\ matched\ avps}{\#\ avps\ in\ manually-built\ hierarchy}$$

$$\textit{node-rec} = \frac{\#\ matched\ nodes}{\#\ nodes\ in\ manually-built\ hierarchy}$$

$$\textit{rec} = \frac{(avp\text{-}rec)+(node\text{-}rec)}{2}$$

Precision and recall only take non-terminal nodes into account since it is in general much easier for a construction algorithm to do well on terminal nodes than it is to do well on non-terminals. This is because there are usually several attribute-value pairs that apply uniquely to one lexical entry, for example attribute-value pairs referring to the orthography or phonology of an entry. These attribute-value pairs will only occur in one place in the Galois lattice (namely in the node corresponding to the relevant lexical entry) and consequently every pruning algorithm will get them right. If terminal nodes were taken into account, precision and recall would depend on the ratio of terminal to non-terminal nodes.

Once precision and recall have been defined they can be combined into a single measure, the *f-score*, which is defined as:

$$\textit{f-score} = \frac{2 \times prec \times rec}{prec + rec}$$

Note, that the evaluation method assumes that there is one "ideal" hierarchy, namely the manually-built hierarchy for the lexicon. This is of course a gross simplification: linguists may find it difficult to

[14]The term *matched avps* describes the number of attribute-value pairs that could be matched, i.e. the number of shared attribute-value pairs in a matched node summed over all matched nodes.

agree on one ideal hierarchy per lexicon. But it is possible to identify a subset of hierarchies which are regarded as relatively plausible. Because lexical inheritance hierarchy construction is fairly subjective, one would expect the upper bound for this task to be well below 100%.

5 Maximum Entropy Pruning

As was argued above a simple notion of minimal redundancy may not be sufficient if one wants to derive linguistically plausible hierarchies. This work therefore takes a different approach: it uses *maximum entropy* models to prune a Galois lattice into an inheritance hierarchy.

Maximum Entropy models have been applied to quite a lot of tasks in natural language processing (see Ratnaparkhi 1998 [12] for an overview). A maximum entropy model can be used to assign an object to a class based on the object's context. The context is represented in the form of *features*. The features can be quite complex and do not have to be statistically independent. Each feature is assigned a weight in a supervised learning step using a parameter estimation algorithm. Weights are chosen in such a way that the entropy of the model is maximised, i.e. maximum entropy models preserve maximal uncertainty and do not assume anything beyond the data. Once a model has been trained the probability of an object *o* belonging to a class *c* can be calculated as follows:

$$P(o,c) = \frac{1}{Z}\prod_{i=1}^{n}\alpha_i^{f_i(o,c)}$$

where *n* is the number of features (*f*), α_i is the weight of feature f_i, and *Z* is a normalisation constant which ensure that the values of P(o,c) lie between 0 and 1.

In the task of pruning Galois lattices, the objects to be classified are the nodes in the lattice and the possible classes are *prune* and *retain*.

So far, 14 contextual feature sets have been implemented, referring to the following properties of a node:

- **Immediate Ancestors and Descendants:** features referring to the number of immediate ancestors and descendants.
- **Terminal Descendants:** features referring to the number of terminal descendants.
- **Attribute-Value Pairs:** features referring to the number of attribute-value pairs at the node, the average number of values permissible for each attribute of the node, the number of times that each of the node's attributes occurs in the lexicon (average), and the number of attribute-value pairs for which the node serves as highest introduction point.
- **Level of Node:** the level of the node in the hierarchy.[15]

The features have been deliberately kept fairly unspecific, i.e. there are no features that refer to specific attributes. Reference to particular attributes has to be avoided if the model is to work across lexicons and possibly also across languages.

At the moment, the features are still very simple. In particular, there are no features that refer to interdependencies between attribute-value pairs. Eventually one would like to take dependence between

[15] Each node has a unique level in the hierarchy, defined to be the level of its highest level descendant plus 1. Terminals are defined to have a level of 1.

attribute-value pairs into account. For example, the existence of some kind of dependence between the attribute-value pairs of a node should increase the probability of the node being retained. Similarly one would also like to take more dependencies between nodes into account. In particular, a node should be more likely to be pruned if it inherits all or most of its attribute-value pairs from its ancestors. But this also has not been implemented yet.[16]

To avoid the size of a hierarchy influencing the value of the contextual predicates, features have been normalised, where appropriate, by dividing them by an average value for the hierarchy, e.g. the contextual predicate *relative-level* returns the level of the node divided by the overall number of levels in the hierarchy. Also, since maximum entropy features usually require contextual predicates that return nominal data and most of the features are numeric,[17] features have been quantised by breaking them up into 6 or 12 intervals. For example, the *relative-level* contextual predicate returns 0, 20, 40, 60, 80 or 100 and a relative level of 0.10 is translated to *relative-level=20*.

Maximum entropy pruning requires two manually-built hierarchies: a training hierarchy and a test hierarchy. Intermediate nodes that occur in the Galois lattice but not in the corresponding manual hierarchy are negative training examples (i.e. belong to the class *prune*) while intermediate nodes that do occur in the manual hierarchy are positive training examples (i.e. *retain*).

6 Experiments

Two manually-built lexicons have been used for the experiments described below: one for English (Sag & Wasow 1999 [13]) containing 501 entries and one for Spanish (Quirino Simões 2001 [14]) containing 405 entries.[18] The hierarchies were compiled out and a Galois lattice was constructed for the compiled out lexicons. The Galois lattices were then pruned into inheritance hierarchies.

Unfortunately, the two data sets provide very little positive training data. This becomes evident when looking at the sizes of the Galois lattices and manually-built hierarchies shown in Figure 10. The first column gives the number of terminal nodes (i.e. lexical entries), the second the number of intermediate nodes and the third the overall number of nodes. The Galois lattices are considerably larger than the corresponding manually constructed hierarchies. The relationship between the number of nodes in the manually-built hierarchy and the number of nodes in the Galois lattice depends to a large extend on the lexicon. Thus, while the manually-built hierarchies for the two lexicons are comparable in size the Galois lattice for the Spanish lexicon is much bigger than its English counterpart. But the relationship between the size of the manual hierarchy and the Galois lattice is probably potentially exponential.

Since the maximum entropy models are only trained on intermediate nodes and there are relatively few positive intermediate nodes, 98% of the training data will be negative. This is further worsened by the fact that the two lexicons contain a relatively large amount of single-child nodes. This is probably due to the fact that both lexicons are relatively small. Since these nodes are not contained in the Galois lattice they are currently disregarded by the training algorithm. Thus, the number of positive training instances is 29 for the English lexicon and 47 for the Spanish lexicon.

[16]Interdependencies between nodes can only be used on a relatively small scale, since taking node interdependencies across the hierarchy into account is computationally expensive. Another problem is that large contexts are likely to cause data sparseness problems. One reason for training on nodes rather than on larger graph fragments is that there are relatively few suitable manually-built hierarchies around and training on larger graph fragments would significantly decrease the amount of available training data.

[17]Not all features are numeric. Some features are boolean, e.g. “is the level of the node higher than the average node level” etc.

[18]Since the maximum entropy features were deliberately kept knowledge-poor, having lexicons for different languages should not cause any problems.

	terminal	intermediate	all
English manual	501	45	547
English Galois	501	1,852	2,354
Spanish manual	405	102	508
Spanish Galois	405	10,738	11,144

Figure 10: Number of nodes in manual hierarchies and Galois lattices

Apart from the maximum entropy pruning that was described above, two other pruning methods have been used: *minimal pruning* and *random pruning*. Pruning was only performed on intermediate nodes since the root node and the terminal nodes have to be retain to ensure that the derived hierarchy is single-rooted[19] and well-defined. Finally, the post-pruning step (see section 4.3) removed redundant attribute-value pairs (and nodes) if necessary. The derived hierarchies were then evaluated against the original hierarchies.

Minimal Pruning Petersen 2001 [10] also proposed a Galois lattice based approach to inheritance hierarchy learning. But she uses a pruning technique which results in every attribute-value pair occurring exactly once in the hierarchy. For each attribute-value pair there is guaranteed to be a unique highest introduction point and a hierarchy is minimal with respect to attribute-value pairs if attribute-value pairs are only retained at their highest introduction point and pruned elsewhere. Nodes which do not function as highest introduction points for any attribute-value pair will be stripped of all their attribute-value pairs and then removed.

Random Pruning This pruning method prunes the lattices randomly according to one of two probability distributions. The first method uses a uniform distribution, i.e.: $P(prune) = P(retain) = 0.5$.[20] The second method retains *n* intermediate nodes randomly, where *n* is the number of intermediate nodes in the relevant manually-built hierarchy. In both cases, random pruning was performed 100 times for each lexicon and the results were averaged. Random pruning was used as a baseline for the other pruning methods.

Maximum Entropy Pruning The straightforward way to use maximum entropy models is to retain a node if $P(retain) > P(prune)$. However, since the hierarchies used in this experiment are relatively small and since there are many more negative training examples than positive ones, this classification method leads to derived hierarchies in which nearly all intermediate nodes are pruned. As an alternative an *n*-best approach was taken in which the n nodes with the highest retain probability are retained, where n is the number of intermediate nodes in the relevant manually-built hierarchy. Of course, this approach is not feasible in the general case because the ideal number of intermediate nodes in a hierarchy will not be known, but it may be possible to calculate an average number of nodes (in relation to the number of lexical entries) across several hierarchies.

[19]Insisting on single-rootedness follows the specification in Carpenter 1992 [3].

[20]Note, that due to the post-pruning step the actual pruning rate may be higher than this.

6.1 Results and Discussion

The results for the English lexicon are given in figure 11. As expected uniform random pruning has the lowest f-score. Maximum entropy and minimal pruning outperform n-best random pruning by a small fraction and maximum entropy pruning is minimally better than minimal pruning.

	f-score	precision	recall	interm. nodes
maximum entropy	22.16%	18.59%	27.44%	51
avp minimal	22.14%	15.79%	37.05%	238
random, n-best (avg.)	21.93%	23.65%	20.65%	43
random, uniform (avg.)	18.37%	12.21%	37.19%	287

Figure 11: Results for the English lexicon

The last column in the table shows the number of retained nodes. Minimal pruning retained more than four times as many nodes as maximum entropy pruning. This is to be expected since there is a trade-off between the number of nodes and the number of attribute-value pairs. Minimising the latter will usual result in increasing the former. The high number of retained nodes has the effect that recall is higher for minimal pruning.

Interestingly the number of retained nodes for maximum entropy pruning is higher than 45. The algorithm was forced to retain more than 45 nodes because $P(retain)$ was identical for some nodes. This suggest that the maximum entropy model is not discriminating well enough yet.

The results for the Spanish lexicon are given in figure 12. The performance of maximum entropy pruning drops dramatically. This is probably due to the fact that the model was trained on the English lexicon which only contains 29 positive training examples. It seems that this number of nodes is too small to train the model sufficiently well.

	f-score	precision	recall	interm. nodes
maximum entropy	0.29%	0.62%	0.19%	25
avp minimal	23.99%	19.84%	30.32%	330
random, n-best (avg.)	9.54%	11.36%	8.28%	100
random, uniform (avg.)	16.90%	12.01%	28.59%	556

Figure 12: Results for the Spanish lexicon

The fact the the maximum entropy pruned hierarchy contains only 25 intermediate nodes instead of 102 indicates that many of the originally retained nodes were removed in the post-pruning step. This was not the case for the English lexicon where no node had to be post-pruned.

Interestingly, the performance of random, n-best pruning also drops significantly and even falls below random uniform pruning. The performance drop is probably related to the fact that the ratio between intermediate nodes in the manually-built hierarchy and intermediate nodes in the Galois lattice is much smaller than in the English lexicon: in the English lexicon, out of 100 Galois nodes 2.4 are contained in the manually-built hierarchy, in the Spanish lexicon only 0.9 nodes out of 100 are contained in the manually-built hierarchy. Consequently, selecting the "right" n nodes is more difficult for the Spanish lexicon.

On the whole, maximum entropy pruning performs reasonably well, at least on the English lexicon. While an f-score of about 20% may not look impressive it has to be remembered that the upper score for the task is probably well below 100% as linguists do not always agree on the best hierarchy for a given lexicon. Also, linguists tend to make their decisions on the basis of linguistic background knowledge and intuition, things that are not available to automatic learning methods. Consequently the task of learning linguistically plausible inheritance hierarchies is probably fairly hard.

The fact that, the very crude maximum entropy model fares better than random pruning (for the English lexicon) despite the fact that the training set was very small is quite encouraging, even though it has to be admitted that the difference between n-best random pruning and maximum entropy pruning is very small.

7 Conclusion and Future Work

This paper presented a first approach to using maximum entropy models and Galois lattices to induce lexical inheritance hierarchies. So far, the model is still very simplistic but it shows some promise for the future. To improve the performance of the model future research has to focus on the following areas:

- **More Training Data:** The results could probably be substantially improved if more (positive) training data were available. Constructing a complete Galois lattice is infeasible for big lexicons but it should be possible to approximate the lattice construction or interleave it with the pruning step, e.g. by pruning away some of the less promising nodes while the Galois lattice is constructed.
- **Using a Prior:** The training data is highly unbalanced: there are many more negative examples than positive. Maximum entropy models have problems with highly unbalanced training data (Osborne 2002 [9]). The use of a prior might counteract imbalance in the data.
- **Maximum Entropy Features:** The feature set is still very small and in particular it does not contain features that take interdependencies in the data set into account. Likewise inter-node interdependencies are only partly considered. However, these interdependencies are probably the most important factor in deciding whether a node should be pruned.
- **Continuous Values:** Representing continuous data as intervals is not optimal. Selecting a good interval size is difficult. An interval that is too small will lead to data sparseness problems while an interval that is too big may prevent the model from being discriminative enough. Maybe techniques that have been used for decision trees might be helpful (e.g. Fayyad & Irani 1993 [6]). Also, the fact that intervals are treated like nominal data means that every sense of distance is lost. But it may be important that the interval "40" is closer to "20" than to "80". Therefore it might be beneficial to look at ways of determining good interval sizes automatically or look for an alternative representation. Maybe some other learning technique, like memory-based learning, might yield better results for continuous data.

Acknowledgements

This research was funded by a University of Edinburgh Faculty of Science and Engineering Scholarship. I am grateful to Alex Lascarides, Miles Osborne and Steve Finch for many helpful discussions and to four anonymous reviewers for their comments on the paper. I would also like to thank Steve Finch for supplying the Galois lattice construction code and Jason Baldridge and Gann Bierner for their maximum entropy package.

References

[1] Petra Barg. *Automatischer Erwerb von linguistischem Wissen. Ein Ansatz zur Inferenz von DATR-Theorien*. Max Niemeyer, Tübingen, 1996.

[2] Lynne J. Cahill. Automatic extension of a hierarchical multilingual lexicon. In *2nd Workshop on Multilinguality in the Lexicon (ECAI-98)*, pages 16–23, 1998.

[3] Bob Carpenter. *The logic of typed feature structures : with applications to unification grammars, logic programs and constraint resolution*. Cambridge University Press, Cambridge, 1992.

[4] Ann Copestake. *The (new) LKB System*. CSLI, 1999.

[5] Roger Evans and Gerald Gazdar. The DATR papers: February 1990. Cognitive science research paper, University of Sussex, 1990.

[6] Usama M. Fayyad and Keki B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pages 1022–1027. Morgan Kaufmann, 1993.

[7] Robert Godin, Hafedh Mili, Guy W. Mineau, Rokia Missaoui, Amina Arfi, and Thuy-Tien Chau. Design of class hierarchies based on concept (Galois) lattices. *Theory and Practice of Object Systems*, 4(2):117–134, 1998.

[8] Marc Light. Classification in feature-based default inheritance hierarchies. In *Proceedings of KONVENS-94*, 1994.

[9] Miles Osborne. Using maximum entropy for sentence extraction. In *Proceedings of the ACL 2002 Workshop on Automatic Summarization, Philadelphia, Pennsylvania, USA, July 11-13*, to appear.

[10] Wiebke Petersen. A set-theoretical approach for the induction of inheritance hierarchies. *Electronic Notes in Theoretical Computer Science*, 47:1–12, 2001.

[11] Carl Pollard and Ivan A. Sag. *Head-Driven Phrase Structure Grammar*. University of Chicago Press, Chicago, 1994.

[12] Adwait Ratnaparkhi. *Maximum Entropy Models for Natural Language Ambiguity Resolution*. PhD thesis, Computer and Information Science, University of Pennsylvania, 1998.

[13] Ivan A. Sag and Thomas Wasow. *Syntactic Theory: A Formal Introduction*. CSLI Publications, Stanford, 1999.

[14] Ana Paula Quirino Simões. Spanish clitics. A computational model. Master's thesis, Universität Bielefeld, 2001.

[15] Mark Steedman. *Surface Structure and Interpretation*. MIT Press, Cambridge, Mass., 1996.

[16] Mary Wood. *Categorial Grammars*. Linguistic Theory Guides. Routledge, London and New York, 1993.