A Galois Lattice based Approach to Lexical Inheritance Hierarchy Learning

Caroline Sporleder [1]

Abstract. Lexical inheritance hierarchies are used widely as a means of representing lexical information efficiently but there have been few attempts to construct them automatically. This paper presents a two-step construction algorithm in which a Galois lattice is built and then pruned into an inheritance hierarchy. The pruning step utilises a maximum entropy model. This is compared to a pruning method which prunes nodes in the hierarchy randomly. To asses the performance of the two methods an automatic evaluation method has been implemented which matches an automatically derived hierarchy to a manually built hierarchy for the same lexicon and defines precision and recall as a distance function between the two hierarchies.

1 INTRODUCTION

Inheritance networks have long been used as a means for knowledge representation in AI. In linguistics, their most prominent use is to encode lexical knowledge. 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 [14]) and *Categorial Grammar* (Wood 1993 [19], Steedman 1996 [18]) can be very complex. Representing lexical information as a simple list of lexical entries results in an undesirable amount of redundancy. For instance, in the lexicon in figure 1, the attribute-value pairs *subcat:transitive* and *past:+ed* occur 4 times. Lexical inheritance hierarchies reduce unnecessary redundancy by providing a way of capturing linguistic generalisations. Thus, it makes sense to create a class TRANSITIVE VERB and a class ED-VERB from which transitive and regular verbs can inherit (figure 2).

orth(ography)	past	past p(articiple)	subcat(egorisation)
beat	+0	+en	transitive
take	took	+en	transitive
hate	+ed	+ed	transitive
love	+ed	+ed	transitive
elapse	+ed	+ed	intransitive
expire	+ed	+ed	intransitive
rise	rose	+en	intransitive

Figure 1. Simple lexicon

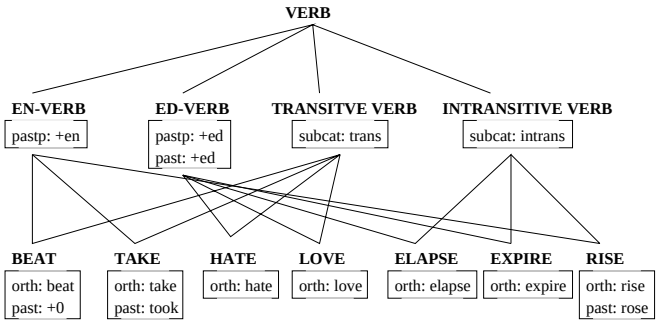

Figure 2. Inheritance hierarchy for the lexicon in figure 1

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.[2] 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 concepts contained in the data set, i.e. 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. The linguistic plausibility of a class is probably influenced by several factors. Therefore it seems that what is needed is a classifier that is able to combine many —not necessarily independent— contextual factors.

A maximum entropy classifier was chosen for this purpose since

[1] University of Edinburgh, Edinburgh, EH8 9LW, UK

[2] Lexical inheritance hierarchies can be monotonic or non-monotonic. In non-monotonic hierarchies, a class can override properties of a superclass. This paper only deals with monotonic hierarchies but an extension to non-monotonic hierarchies is possible.

maximum entropy classifiers can incorporate many contextual properties and these do not have to be independent. The classifier is trained in a supervised training step using training data provided by a manually built hierarchy.

2 BACKGROUND

The use of Galois lattices in machine learning and natural language processing is not new.

Formal Concept Analysis (FCA, Ganter & Wille 1999 [9]) uses Galois lattices (called *concept lattices* in FCA) to identify and analyse conceptual structures in data sets.

Carpineto and Romano 1993 [4] employ a Galois lattice to determine the classes of new objects. The class assigned to a new object is the most similar, consistent class in the Galois lattice. The lattice is always complete — no nodes get pruned. Carpineto and Romano 1996 [5] discusses an extension of this approach which allows for the incorporation of structural background knowledge.[3]

Basili *et al.* 1997 [2] use Galois lattices to learn verb subcategorization frames. After extracting sets of subcategorization frames from a corpus, a Galois lattice is constructed for each verb. The concepts in the lattice are (sets of) subcategorization frames. Some sets may be noisy or linguistically inadequate. Useful sets of subcategorization frames are chosen by applying two selection measures, which take into account the lattice structure and the linguistic probability of the individual subcategorization frames in a set.

In ontology construction, Galois lattices have been used by Godin *et al.* 1998 [10], who employ a Galois lattice to built and maintain class hierarchies in object-oriented design.

For inheritance hierarchy induction, Galois lattices have been proposed independently by Petersen 2001 [13]. While she, too, builds a Galois lattice for the input lexicon and prunes it into a hierarchy, she uses a pruning method that is different from the one proposed in this paper in that it creates hierarchies where every attribute-value pair occurs only once. This pruning method is also used by Godin *et al.* 1998 [10] for the different task of class hierarchy induction.

There have been few other attempts to induce lexical inheritance hierarchies. The most comprehensive work to date is Barg 1996 [1]. Barg presents an algorithm for learning hierarchies for the DATR formalism (Evans & Gazdar 1990 [7]) 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 1998 [3] 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 over nodes) is hard-wired.

Light 1994 [11] 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.

[3] In lexical inheritance hierarchies structural background knowledge often occurs in the form of *types*. Types form the backbone of *typed inheritance hierarchies*.

3 LEARNING ARCHITECTURE

3.1 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 $R \rightarrow E \times E'$, a Galois lattice is a partial order over all pairs of the form (X, X'), where $X \subseteq E$, $X' \subseteq E'$ s.t.:

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

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 3 shows a small lexicon which contains information about the nativeness and the ability to carry stress for six German prefixes. Figure 4 shows the Galois lattice for the lexicon. Concepts contained in the lexicon and the lattice are, for example:

- 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 3. German prefixes: lexicon

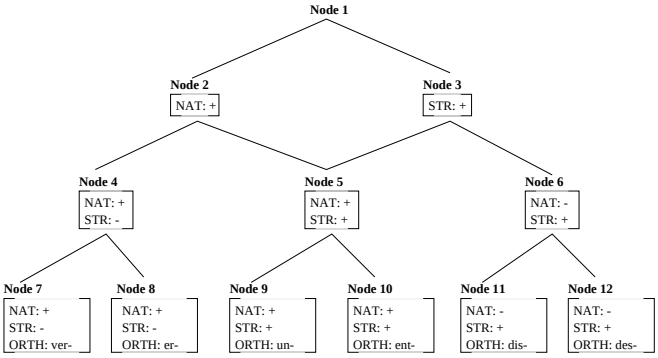

Figure 4. German prefixes: Galois lattice

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 5). 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*)[4] but the attribute-value pairs NAT:+ and STR:- are lacking from its intension because they can be inherited from Node 4.

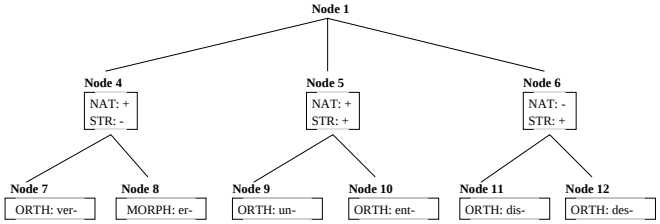

Figure 5. German prefixes: manually built hierarchy

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.[5]

3.2 Pruning

Once a Galois lattice has been built one has to decide which nodes (i.e. concepts) should be pruned and which should be retained. For example, in the German prefix example above (figure 4) the pruning method should retain nodes 4, 5, and 6 (cf. the manually built hierarchy in figure 5) rather than nodes 2 and 3.

A popular pruning method is based on the idea of minimal redundancy of attribute-value pairs. 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. This method is used by Petersen 2001 [13] to derive lexical inheritance hierarchies.

While the notion of minimal redundancy has been employed in some form in all previous approaches to inheritance hierarchy learning, it is not clear whether minimal redundancy is a particularly good criterion. Lexical inheritance hierarchies should be first and foremost linguistically plausible and it is not evident that some form of minimal redundancy necessarily coincides with linguistic plausibility. Therefore, the approach taken here uses a maximum entropy model to prune the lattice. This approach allows one to prune a lattice on the basis of many contextual properties not only on the basis of simple redundancy criteria.

Maximum Entropy models have been applied to quite a lot of tasks in natural language processing (see Ratnaparkhi 1998 [15] 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 ensures 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. These feature sets refer 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.

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 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.[6]

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

[5] 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.

[6] 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.

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 above functions range over continuous data, values 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*.

3.3 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 (Barg 1996 [1], Light 1994 [11]). 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.

The approach adopted here matches the derived hierarchies to a manually built hierarchy for the same lexicon. The graph matching algorithm is error-tolerant, i.e. two nodes do not have to be identical to be matched but sufficiently similar. Similarity between two nodes is defined in terms of extensional and intensional overlap and two nodes are matched if their similarity is above a user defined threshold, which was set to 0.5, i.e. two nodes were matched if their extensional and intensional overlap was more than 50%.

Since the hierarchies derived by the pruning methods described below are guaranteed to be *sound*, i.e. compiling them out[7] 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.

To assess the overall distance between two hierarchies, *precision*, *recall* and *f-score* have to be calculated. The overall precision (*prec*) of the learning algorithm is 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:[8]

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

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

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

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*):

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

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

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

In general, it is much easier for a construction algorithm to do well on terminal nodes than 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 terminal node corresponding to the relevant lexical entry) and consequently every pruning algorithm will get them right. To avoid precision and recall being influenced by the terminal to non-terminal ratio, the evaluation metrics only take non-terminals into account.

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 usually cannot 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 highly subjective, one would expect the upper bound for this task to be noticeably below 100%.

4 EXPERIMENTS

The system is implemented to work with LKB grammars (Copestake 1999 [6]). For a first experiment with the maximum entropy pruner, the (manually constructed) inheritance hierarchies supplied with Sag and Wasow 1999 [16] and Quirino Simões [17] have been used. The former encodes an English lexicon, the latter a Spanish lexicon.[9]

The hierarchies were compiled out and a Galois lattice was constructed for the compiled out lexicons. The Galois lattices were then used to train a maximum entropy model. Maximum entropy training is only done on intermediate nodes (i.e. not on terminals or the root node) because only those can be pruned. The root and terminals have to be retained to ensure that the pruned hierarchy is sound, i.e. does not add any information to or delete any information from the original lexicon, and single-rooted. 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*). Each of the two Galois lattices was pruned using the maximum entropy model trained on the other lattice. After the nodes were pruned, a second pruning step removed redundant attribute-value pairs, i.e. attribute-value pairs that could be inherited, from the retained nodes. If this resulted in empty nodes,

[7] A hierarchy is compiled out by having each lexical entry inherit all properties from its ancestors, resulting is a set of fully specified lexical entries.

[8] 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.

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

these were removed, too. Finally the pruned lattices were matched to the original hierarchies and evaluated.

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 6. 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. 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.

	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 6. Number of nodes in manual hierarchies and Galois lattices

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.

The results for pruning the two grammars with the *n*-best maximum entropy models are shown in figure 7. The pruning method works much better on the English lexicon than on the Spanish lexicon. This is due to the fact that the English lexicon supplies less positive training examples than the Spanish lexicon. Thus training on the English grammar and testing on the Spanish grammar leads to worse results than the other way round. It seems that the number of training examples in the English grammar is too small to train the model sufficiently well.

	f-score	precision	recall	interm. nodes
English	22.16%	18.59%	27.44%	51
Spanish	0.29%	0.62%	0.19%	25

Figure 7. Results for n-best maximum entropy pruning

The last column in figure 7 shows the number of intermediate nodes retained by the pruning method. The parameter *n* was set to 45 (English) and 102 (Spanish). Thus one would expect 45 retained nodes in the English hierarchy and 102 in the Spanish hierarchy. Interestingly, this is not the case. For the English grammar more than *n* nodes were retained. The reason for this is that $P(retain)$ was identical for some nodes, which suggests that the maximum entropy model is not discriminating well enough yet. For the Spanish grammar the number of intermediate nodes is significantly lower than *n*. While 102 nodes were initially retained the selection was so bad that many of them had to be removed afterwards because they inherited all their attribute-value pairs from their ancestors. This is another indicator that the model applied to the Spanish lattice was not very good.

For comparison, the lattices were also pruned randomly according to one of two probability distributions. The first method uses a uniform distribution, i.e.: $P(prune) = P(retain) = 0.5$. The second method retains *n* intermediate nodes randomly, where *n* is the number of intermediate nodes in the relevant manually built hierarchy.[10] Random pruning was performed 100 times for each lexicon and the results were averaged.

Figure 8 shows the results of uniform random pruning. The f-score is lower than it is for maximum entropy pruning (at least for the English lexicon). While the random pruning method retains many more nodes and therefore has a higher recall the precision is lower than for maximum entropy pruning.

	f-score	precision	recall	interm. nodes
English (avg.)	18.37%	12.21%	37.19%	287
Spanish (avg.)	16.90%	12.01%	28.59%	556

Figure 8. Results for uniform random pruning

Figure 9 shows the results of *n*-best random pruning. For the English lexicon the f-score improves but still is slightly lower than the maximum entropy f-score. Somewhat surprisingly, the f-score drops for the Spanish lexicon. This 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.

	f-score	precision	recall	interm. nodes
English	21.93%	23.65%	20.65%	43
Spanish	9.54%	11.36%	8.28%	100

Figure 9. Results for *n*-best random pruning

On the whole the very crude maximum entropy pruning method shows some promise. While an f-score of about 20% may not look

[10] Note, that in both cases the actual pruning rate will be higher than this if pruning leads to nodes that can inherit all their attribute-value pairs from their ancestors because this nodes will later be removed.

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. And the fact that maximum entropy pruning performs slightly better than the *n*-best random pruning method (for the English lexicon) seems to show that the maximum entropy model represents some useful generalisations.

5 CONCLUSION AND FUTURE WORK

This paper presented a first approach to using maximum entropy models 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 was 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.
- **Using a Prior:** Maximum entropy models have problems with highly unbalanced training data (Osborne 2002 [12]). The use of a prior might counteract unbalanced 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. 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 [8]). In addition, 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.

While the approach presented here was implemented and tested for lexical inheritance hierarchies it should also be applicable to other ontology learning tasks. Since a supervised learning technique is used this would normally require the existence of suitable training data. However, it may be possible to apply maximum entropy models that have been trained on lexical inheritance hierarchies directly to other ontology learning problems, since the maximum entropy features used for lexical inheritance hierarchies are deliberately kept knowledge-poor, i.e. they do represent general properties of concept representativeness etc. and not properties that are specific to linguistics. However, whether this is indeed possible remains a bit speculative at this stage.

Acknowledgements

This research was funded by a University of Edinburgh Faculty of Science and Engineering Scholarship. I would like to thank three anonymous reviewers for their comments on an earlier version of this paper.

REFERENCES

[1] Petra Barg, *Automatischer Erwerb von linguistischem Wissen. Ein Ansatz zur Inferenz von DATR-Theorien*, Max Niemeyer, Tübingen, 1996.
[2] Roberto Basili, Maria Teresa Pazienza, and Michele Vindigni, 'Corpus-driven unsupervised learning of verb subcategorization frames', in *Proceedings of the 5th Conference of the Italian Association for Artificial Intelligence (AI*IA-97)*, pp. 159–170, (1997).
[3] Lynne J. Cahill, 'Automatic extension of a hierarchical multilingual lexicon', in *2nd Workshop on Multilinguality in the Lexicon (ECAI-98)*, pp. 16–23, (1998).
[4] Claudio Carpineto and Giovanni Romano, 'Galois: An order-theoretic approach to conceptual clustering', in *Proceedings of the 10th International Conference on Machine Learning (ICML-93)*, pp. 33–40, (1993).
[5] Claudio Carpineto and Giovanni Romano, 'A lattice conceptual clustering system and its application to browsing retrieval', *Machine Learning*, **24**, 95–122, (1996).
[6] Ann Copestake, *The (new) LKB System*, CSLI, 1999.
[7] Roger Evans and Gerald Gazdar, 'The DATR papers: February 1990', Cognitive science research paper, University of Sussex, (1990).
[8] 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*, pp. 1022–1027. Morgan Kaufmann, (1993).
[9] Bernhard Ganter and Rudolf Wille, *Formal Concept Analysis. Mathematical Foundations*, Springer, Berlin, 1999.
[10] 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).
[11] Marc Light, 'Classification in feature-based default inheritance hierarchies', in *Proceedings of KONVENS-94*, (1994).
[12] 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).
[13] Wiebke Petersen, 'A set-theoretical approach for the induction of inheritance hierarchies', *Electronic Notes in Theoretical Computer Science*, **47**, 1–12, (2001).
[14] Carl Pollard and Ivan A. Sag, *Head-Driven Phrase Structure Grammar*, University of Chicago Press, Chicago, 1994.
[15] Adwait Ratnaparkhi, *Maximum Entropy Models for Natural Language Ambiguity Resolution*, Ph.D. dissertation, Computer and Information Science, University of Pennsylvania, 1998.
[16] Ivan A. Sag and Thomas Wasow, *Syntactic Theory: A Formal Introduction*, CSLI Publications, Stanford, 1999.
[17] Ana Paula Quirino Simões, *Spanish Clitics. A Computational Model*, Master's thesis, Universität Bielefeld, 2001.
[18] Mark Steedman, *Surface Structure and Interpretation*, MIT Press, Cambridge, Mass., 1996.
[19] Mary Wood, *Categorial Grammars*, Linguistic Theory Guides, Routledge, London and New York, 1993.