

Machine Learning of Lexical Inheritance Hierarchies: Linguistic Plausibility vs. Minimal Redundancy

Caroline Sporleder

Division of Informatics

University of Edinburgh

csporled@cogsci.ed.ac.uk

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 modular architecture which makes it possible to evaluate different formal criteria that may guide a learning algorithm.

1 Introduction

Grammar development over the last decades has seen a shift away from large inventories of grammar rules to richer lexical structures. Many modern grammar theories like *Head-Driven Phrase Structure Grammar* (Pollard and Sag, 1994) and *Categorical Grammar* (Wood, 1993) (Steedman, 1996) are highly lexicalised. But simply listing lexical entries results in an undesirable amount of redundancy. Lexical inheritance hierarchies address this problem by providing a way of capturing linguistic generalisations.

Inheritance hierarchies are commonly constructed by hand but this is time-consuming and often impractical if a lexicon is very large. Constructing hierarchies automatically or semi-automatically facilitates a more systematic analysis of the lexical data. In addition, lexical information is often extracted automatically from corpora and this is likely to increase over the coming years. Therefore it makes sense to go a step further and automate the hierarchical organisation of lexical data too.

This paper presents an approach to learning inheritance hierarchies from unstructured lexicons. The

task is split into two steps: First, a so-called *Galois 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 which is intended to capture interesting linguistic generalisations. Thus the automatic construction of a hierarchy is reduced to a classification problem: Finding a good learning method amounts to finding a classifier that correctly predicts when a node in the lattice should be pruned and when it should be retained.

The set-up allows one to experiment with different classifiers but this presupposes an evaluation method. So far, automatically constructed hierarchies have been evaluated by comparing them “by hand” to manually constructed hierarchies. To facilitate the evaluation of big hierarchies, I will propose an automatic evaluation technique based on a graph matching algorithm and a distance measure.

Two different pruning methods are evaluated: one based on a simple concept of minimal redundancy and the other using a basic maximum entropy model. For comparison, a randomly pruned lattice has also been evaluated.

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). Barg presents an algorithm for learning hierarchies for the DATR lexical representation language (Evans and Gazdar, 1996) 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

included in the set and their priority may vary for different linguistic domains. Most criteria refer to the size and complexity of a hierarchy.

Cahill (1998) 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) investigates ways in which new entries can be inserted into an existing hierarchy. He uses a greedy algorithm that inserts entries in such a way that the sum of the number of parent nodes and the number of attribute-value pairs that have to be listed at the entry itself is minimised. Light’s algorithm does not create new (non-terminal) classes and therefore cannot be used to learn a hierarchy from scratch.

Petersen (2001) independently proposed an approach that is similar to the one presented here. She also constructs a *Galois* (or *Concept*) *lattice*. This lattice is then pruned into a hierarchy that is minimally redundant with respect to the number of its attribute-value pairs, i.e. each attribute-value pair occurs exactly once.

Most of these approaches focus on some notion of “minimal redundancy”, i.e. they aim to derive hierarchies which 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), by weighting (Light, 1994) or ordering potentially conflicting criteria (Barg, 1996). So far, however, there has not been any work on assessing which formal criteria are useful for the task of constructing hierarchies automatically or how these criteria should be combined, that is how much weight conflicting criteria should carry. It is also not clear that focusing more or less exclusively on criteria that aim to minimise redundancy is bene-

ficial. If the aim is to derive linguistically plausible hierarchies then it may be better to focus more on interdependencies in the data than on the number of nodes or attribute-value pairs in a hierarchy.

3 System Architecture

The architecture of the system utilises a Galois lattice (Godin et al., 1995) (Ganter and Wille, 1999). Given a set of instances (i.e. lexical entries) E , 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 $X \subseteq E$, $X' \subseteq E'$ 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).

Under the (simplifying) assumption that the “ideal” hierarchy for a lexicon is contained in the lattice,¹ building a hierarchy can be reduced to a classification task: one has to decide which nodes (i.e. concepts) should be pruned and which should be retained. Finding a good algorithm for constructing lexical inheritance hierarchies then amounts to finding a suitable pruning method.

Assessing a pruning method involves evaluating the hierarchies derived by it. Barg (1996) and Light (1994) evaluate their hierarchies by comparing them “by hand” to a manually built hierarchy for the same lexicon. I follow these approaches in assuming that manually built hierarchies are the “gold-standard” – they represent what linguists believe are important generalisations. But I propose an automatic evaluation method which uses a graph matching algorithm to match a derived hierarchy to the corresponding manually built one. Since hierarchies derived by the

¹In practice there is not an “ideal hierarchy” but rather a set of more or less plausible hierarchies. Not every of these hierarchies is necessarily contained in the Galois lattice. For instance, in the Galois lattice every non-terminal node will have at least two children while there may be a plausible hierarchy that –due to data sparseness– contains a node with only one child.

pruning methods discussed below are guaranteed to be *sound*, i.e. compiling them out² 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 graph-matching has to be *error-tolerant*, i.e. allow 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.

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

$$ns = \frac{eol + iol}{2}$$

Where extensional and intensional overlap are defined as:

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

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

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

(# *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.³ Once all nodes in the manually built hierarchy have been considered the matching is made

²A hierarchy is compiled out by having each lexical entry inherit all properties from its ancestors, resulting is a set of fully specified lexical entries.

³*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 (which was 0.5).

unique in the other direction, i.e. one has to ensure that no node in the automatically constructed hierarchy is 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.

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:⁴

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

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

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

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*):

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

⁴The 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.

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

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

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:

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

The experimental set-up then is as shown in Figure 1. First, a manually built hierarchy is compiled-out. Then a Galois lattice is constructed for the compiled-out lexicon. The lattice is then pruned and the resulting hierarchy is matched to the manually built hierarchy.

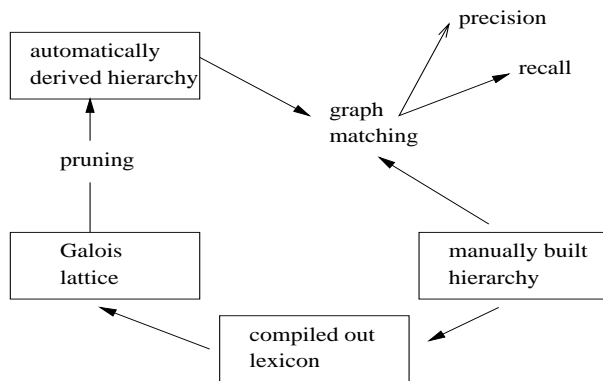


Figure 1: Experimental Set-Up

4 Minimal vs. Random Pruning

The experiment was conducted using the lexical inheritance hierarchy supplied with (Sag and Wasow,

1999). After removing lexical rules and inconsistent entries, the hierarchy contained 501 terminal nodes (i.e. lexical entries). It was compiled-out and a Galois lattice was built for the compiled-out lexicon. Then two different pruning methods were tested: In *minimal-pruning*, the lattice is pruned in such a way that every attribute-value pair occurs only once. For each attribute-value pair there is guaranteed to be a unique highest introduction point and for minimal-pruning attribute-value pairs are only retained at their highest introduction point and pruned from every node further down the inheritance path. Nodes which do not function as highest introduction points for any attribute-value pair will consequently be stripped of all their attribute-value pairs and will subsequently be removed from the hierarchy. The root and terminal nodes, however, are retained even if they do not contain attribute-value pairs to ensure that the hierarchy is well-defined (Carpenter, 1992). This pruning method is equivalent to Petersen’s (2001) approach.

In *random-pruning*, non-terminal nodes are pruned using a random process $X : \Omega \rightarrow \{prune, retain\}$:

$$P(X = retain) = P(X = prune) = 0.5$$

Random-pruning is done top-down. If a node is retained, all of its attribute-value pairs are removed from nodes further down the inheritance path. This guarantees that no attribute-value pair will be specified twice on any inheritance path. Again, resulting empty nodes are pruned (with the exception of the root and terminals). Therefore the actual pruning rate may be higher than $P(X = prune)$. Random-pruning forms a baseline for the task.

The automatically constructed hierarchy was then matched to the original, manually built, hierarchy (using a threshold of 0.5) and evaluated. Random-pruning was performed 100 times and the results were averaged (figure 2).

	Minimal	Random (avg.)	Manual
f-score	22.14%	18.37%	n/a
prec.	15.79%	12.21%	n/a
rec.	37.05%	37.19%	n/a
nodes	740	789	547

Figure 2: Minimal vs. Random Pruning

As expected, minimal pruning performs better than random pruning. While an f-score of 22.14% may not look very impressive, it has to be remembered that the upper bound for this task is certainly below 100%. This follows from the fact that linguists do not always agree on the most plausible hierarchy.

A visual inspection of the hierarchy derived by minimal pruning revealed that it contains several nodes with only one attribute-value pair. This is not surprising as attribute-value pairs will only occur together at a node if they always occur together in the original lexicon. However, nodes which contain only one attribute-value pair are unlikely to be plausible generalisations of the lexical data and hardly ever occur in manually built hierarchies. Thus it is likely that minimal-pruning is too aggressive in this respect.

5 A Maximum Entropy Classifier

It seems that it might be beneficial to experiment with an alternative classifier that aims for a global optimisation over different contextual criteria that can effect the construction of lexical inheritance hierarchies. This allows for a more fine-grained modelling of linguistic plausibility and can be done by making use of the maximum entropy framework.

Maximum entropy models have been successfully applied to several NLP problems (Ratnaparkhi, 1998). They use weighted *features* to determine the probability of a class. Features represent the context of an entity to be classified. The weights are set automatically by running a parameter-estimation algorithm, enforcing a uniform distribution while still obeying the constraints specified by the features.

Because feature weights can be set automatically in a supervised training step, the maximum entropy framework makes it easy to combine features into one classifier. Consequently, maximum entropy models can be very fine-grained, taking several aspects of the context into account. This makes them ideal for pruning the Galois lattice. Furthermore, the features do not have to be statistically independent. This is useful because some contextual properties are not independent (cf. the redundancy criteria: number of attribute-value pairs and number of nodes).

As a first experiment a simple maximum entropy model has been trained on a Spanish lexicon (Simões, 2001). Nodes contained in both the Galois lattice and the manual hierarchy are positive training examples (i.e. *retain*) and nodes only contained in the Galois lattice are negative training examples (i.e. *prune*). The trained model has then been used to prune the Galois lattice derived from the English lexicon mentioned above (Sag and Wasow, 1999) and the resulting hierarchy has been evaluated against the original (English) hierarchy.

To be able to train on one lexicon and test on another means that the maximum entropy features have to be kept relatively knowledge-poor, i.e. they should not refer to particular attribute-value pairs. So far, 14 maximum entropy features sets have been implemented, which refer, for example, to the ancestors and descendants of a node, to its level in the hierarchy and to properties of its attribute-value pairs (see Sporleder (2002)). However, so far no interdependencies between attribute-value pairs have been taken into account. The results are shown in figure 3.

f-score	prec.	rec.	nodes
22.16%	18.59%	27.44%	553

Figure 3: Simple Maximum Entropy Model

The f-score is comparable to the one for minimal pruning. Given that the maximum entropy model is still very crude and the training set supplied by the Spanish lexicon is quite small, this is encouraging. It is hoped that with a more sophisticated model and a bigger training set this will further improve.

6 Conclusions and Future Work

This paper presented an approach to constructing (monotonic) multiple lexical inheritance hierarchies automatically by first constructing a Galois lattice for a lexicon and then pruning it. To evaluate the resulting hierarchies an automated evaluation method was proposed, which assesses the similarity between an automatically derived and a manually built hierarchy.

A pruning method using a simple minimal redundancy criterion was evaluated and compared to pruning randomly and to a simple maximum entropy pruning method. As expected, both the minimal

pruning and the maximum entropy pruning yield better results than random pruning. The fact that the results for the relatively crude maximum entropy model are already comparable to the minimal pruning method is quite encouraging.

For the future it is planned to refine the maximum entropy model by incorporating more features and training on a bigger training set and to evaluate it on a set of different hierarchies. However, constructing a complete Galois lattice is infeasible for most non-trivial lexicons. So a first step will involve approximating the Galois lattice construction or interleaving it with the pruning step.

Acknowledgements

This research was funded by a University of Edinburgh Faculty of Science and Engineering Scholarship. I would like to thank Alex Lascarides, Steve Finch and Miles Osborne for many helpful discussions. Many thanks also to Julia Hockenmaier, James Curran and three anonymous reviewers for their comments on this paper.

References

- Petra Barg. 1996. *Automatischer Erwerb von linguistischem Wissen. Ein Ansatz zur Inferenz von DATR-Theorien*. Max Niemeyer, Tübingen.
- Lynne J. Cahill. 1998. Automatic extension of a hierarchical multilingual lexicon. In *2nd Workshop on Multilinguality in the Lexicon (ECAI-98)*, pages 16–23.
- Bob Carpenter. 1992. *The logic of typed feature structures : with applications to unification grammars, logic programs and constraint resolution*. Cambridge University Press, Cambridge.
- Roger Evans and Gerald Gazdar. 1996. DATR: A language for lexical knowledge representation. *Computational Linguistics*, 22(2):167–216.
- Bernhard Ganter and Rudolf Wille. 1999. *Formal Concept Analysis. Mathematical Foundations*. Springer, Berlin.
- Robert Godin, Rokia Missaoui, and Hassan Alaoui. 1995. Incremental concept formation algorithms based on Galois (concept) lattices. *Computational Intelligence*, 11(2):246–267.
- Marc Light. 1994. Classification in feature-based default inheritance hierarchies. In *Proceedings of KONVENS-94*.
- Wiebke Petersen. 2001. A set-theoretical approach for the induction of inheritance hierarchies. *Electronic Notes in Theoretical Computer Science*, 47:1–12.
- Carl Pollard and Ivan A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. University of Chicago Press, Chicago.
- Adwait Ratnaparkhi. 1998. *Maximum Entropy Models for Natural Language Ambiguity Resolution*. Ph.D. thesis, Computer and Information Science, University of Pennsylvania.
- Ivan A. Sag and Thomas Wasow. 1999. *Syntactic Theory: A Formal Introduction*. CSLI Publications, Stanford.
- Ana Paula Quirino Simões. 2001. Spanish clitics. A computational model. Master's thesis, Universität Bielefeld.
- Caroline Sporleder. 2002. Learning lexical inheritance hierarchies with maximum entropy models. In *Proceedings of the ESSLLI Workshop on Machine Learning Approaches in Computational Linguistics*, Trento, Italy.
- Mark Steedman. 1996. *Surface Structure and Interpretation*. MIT Press, Cambridge, Mass.
- Mary Wood. 1993. *Categorial Grammars*. Linguistic Theory Guides. Routledge, London and New York.