Steffens (ed.) Machine Translation and the lexicon. Springer LNAI 898, 85-98, 1995

Memory-Based Lexical Acquisition and Processing

Walter Daelemans *

Institute for Language Technology and AI, Tilburg University
P.O.Box 90153, 5000 LE Tilburg, The Netherlands
Walter.Daelemans@kub.nl

Abstract

Current approaches to computational lexicology in language technology are knowledge-based (competence-oriented) and try to abstract away from specific formalisms, domains, and applications. This results in severe complexity, acquisition and reusability bottlenecks. As an alternative, we propose a particular performance-oriented approach to Natural Language Processing based on automatic memory-based learning of linguistic (lexical) tasks. The consequences of the approach for computational lexicology are discussed, and the application of the approach on a number of lexical acquisition and disambiguation tasks in phonology, morphology and syntax is described.

1 Introduction

In computational lexicology, three basic questions guide current research: (1) which knowledge should be in the lexicon, (2) how should this knowledge be represented (e.g., to cope with the problems of lexical gaps), and (3) how can this knowledge be acquired. Current lexical research in language

*Preprint of a paper to be published in Steffens (ed.) *Machine Translation and the Lexicon.* Springer Lecture Notes in Artificial Intelligence. I would like to thank my colleagues in the *Atila* project (Steven Gillis, Gert Durieux, and Antal van den Bosch) for their contributions to the approach described in this paper. The *Atila* (Antwerp-Tilburg Inductive Language Acquisition) project is a research corporation between the University of Antwerp and Tilburg University focusing on the application of Machine Learning techniques in linguistic engineering and in developmental psycholinguistics. Thanks also to the participants of the Heidelberg workshop on Machine Translation and the Lexicon for useful comments and suggestions.

technology is eminently *knowledge-based* in this respect. It is also generally acknowledged that there exists a natural order of dependencies between these three research questions: acquisition techniques depend on the type of knowledge representation used and the type of knowledge that should be acquired, and the type of knowledge representation used depends on what should be represented.

Also uncontroversial, but apparently no priority issue for many researchers, is the fact that the question which knowledge should be represented (which morphological, syntactic, and semantic *senses* of lexical items should be distinguished, [19]) depends completely on the natural language processing *task* that is to be solved. Different tasks require different lexical information. Also, different theoretical formalisms, domains, and languages require different types of lexical information and therefore possibly also different types of lexical knowledge representation and different acquisition methods. It makes sense to work on "a lexicon for HPSG parsing of Dutch texts about airplane parts" or on "lexicons for translating computer manuals from English to Italian", but does it make equal sense to work on "the lexicon"? Because it is uncontroversial that lexicon contents is a function of task, domain, language, and theoretical formalism, the *reusability problem* has been defined as an additional research topic in computational lexicology, an area that should solve the problem of how to translate lexical knowledge from one theory, domain, or application to the other. Unfortunately, successful solutions are limited and few.

In this paper, we propose an alternative approach in which a performance-oriented (behaviour-based) perspective is taken instead of a competence-oriented (knowledge-based) one. We try to automatically *learn* the language processing task on the basis of examples. The effect of this is that the priorities between the three goals discussed earlier are changed: the representation of the acquired knowledge depends on the acquisition technique used, and the knowledge acquired depends on what the learning algorithm has induced as being relevant in solving the task. This shift in focus introduces a new type of reusability: reusability of *acquisition method* rather than reusability of acquired knowledge. It also has as a consequence that it is no longer a priori evident that there should be different components for lexical and non-lexical knowledge in the internal representation of an NLP system solving a task, except when the task learned is specifically lexical.

The structure of the paper will be as follows. In Section 2 we will explain the differences between the knowledge-based and the behaviour-based approach to Natural Language Processing (NLP). Section 3 introduces *lazy* *learning*, the symbolic machine learning paradigm which we have used in experiments in lexical acquisition. In Section 4, we show how virtually all linguistic tasks can be redefined as a classification task, which can in principle be solved by lazy learning algorithms. Section 5 gives an overview of research results in applying lazy learning to the acquisition of lexical knowledge, and Section 6 concludes with a discussion of advantages and limitations of the approach.

2 Knowledge-Based versus Behaviour-Based

One of the central intuitions in current knowledge-based NLP research is that in solving a linguistic task (like text-to-speech conversion, parsing, or translation), the more linguistic knowledge is explicitly modeled in terms of rules and knowledge bases, the better the performance.

As far as lexical knowledge is concerned, this knowledge is represented in a lexical knowledge base, introduced either by hand or semi-automatically using machine-readable dictionaries. The problem of reusability is dealt with by imposing standards on the representation of the knowledge, or by applying filters or translators to the lexical knowledge. Not only is there a huge and costly *linguistic engineering* effort involved in the building of a knowledge-based lexicon in the first place, the effort is duplicated for every translation module between two different formats of the lexical knowledge. In practice, most NLP projects therefore start lexicon construction from scratch, and end up with unrealistically few lexical items.

In this paper, we will claim that regardless of the state of theory-formation about some linguistic task, simple data-driven learning techniques, containing very little a priori linguistic knowledge, can lead to performance systems solving the task with an accuracy higher than state-of-the art knowledge-based systems. We will defend the view that all linguistic tasks can be formulated as a *classification* task, and that simple memory-based learning techniques based on a *consistency heuristic* can learn these classifications tasks.

> **Consistency Heuristic**. "Whenever you want to guess a property of something, given nothing else to go on but a set of reference cases, find the most similar case, as measured by known properties, for which the property is known. Guess that the unknown property is the same as that known property." ([28])

In this approach, reusability resides in the *acquisition method*. The same, simple, machine learning method may be used to induce linguistic mappings whenever a suitable number of examples (a corpus) is available, and can be reused for any number of training sets representing different domains, sublanguages, languages, theoretical formalisms, and applications. In this approach, emphasis shifts from knowledge representation (competence) to induction of systems exposing useful behaviour (performance), and from knowledge engineering to the simpler process of data collection. Fig. 1 illustrates the difference between the two approaches.

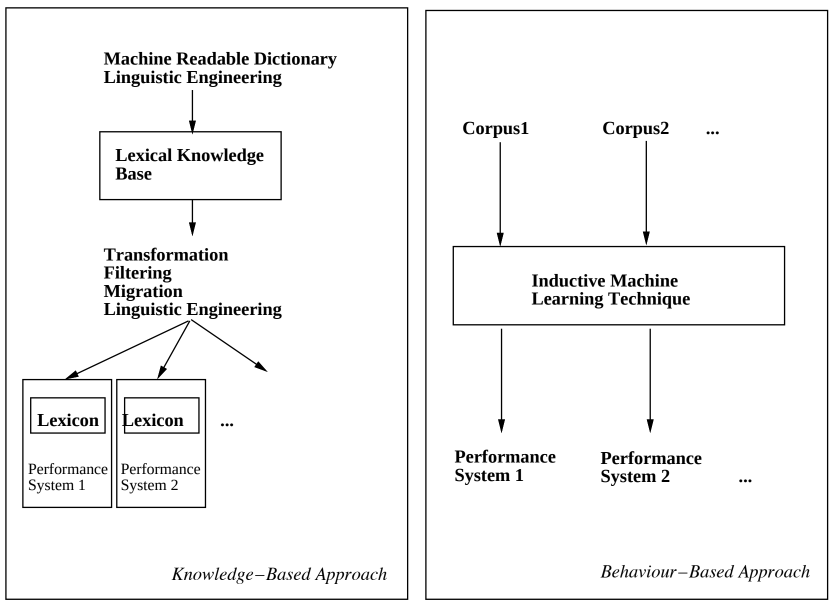

Figure 1: Knowledge-Based versus Behaviour-Based approaches to lexical acquisition

3 Supervised Machine Learning of Linguistic Tasks

In supervised Machine Learning, a learner is presented with a number of examples describing a mapping to be learned, and the learner should extract the necessary regularities from the examples and apply them to new, previously unseen input. It is useful in Machine Learning to make a distinction between a *learning component* and a *performance component*. The performance component produces an output (e.g., a syntactic category) when presented with an input (e.g., a word and its context) using some kind of representation (decision trees, classification hierarchies, rules, exemplars, ...). The learning component implements a learning method. It is presented with a number of examples of the required input-output mapping, and as a result modifies the representation used by the performance system to achieve this mapping for new, previously unseen inputs. There are several ways in which *domain bias* (a priori knowledge about the task to be learned) can be used to optimize learning. In the experiments to be described we will not make use of this possibility.

There are several ways we can measure the success of a learning method. The most straightforward way is to measure *accuracy*. We randomly split a representative set of examples into a training set and a test set[1], train the system on the training set, and compute the success rate (accuracy) of the system on the test set, i.e., the number of times the output of the system was equal to the desired output. Other evaluation criteria include learning and performance speed, memory requirements, clarity of learned representations, etc.

3.1 Lazy Learning

Recently, there has been an increased interest in Machine Learning for *lazy learning* methods. In this type of similarity-based learning, classifiers keep in memory (a selection of) examples without creating abstractions in the form of rules or decision trees (hence *lazy* learning). Generalization to a new input pattern is achieved by retrieving the most similar memory item according to some distance metric, and extrapolating the category of this item to the new input pattern (applying the consistency heuristic). Instances of this form of *nearest neighbour method* include instance-based learning ([2]), exemplar-based learning ([21], [5]), memory-based reasoning ([26]), and case-based reasoning ([17]). Advantages of the approach include an often surprisingly high classification accuracy, the capacity to learn polymorphous concepts, high speed of learning, and perspicuity of algorithm and classification (see e.g., [5]). Learning speed is extremely fast (it consists basically of storing patterns), and performance speed, while relatively slow on serial machines, can be considerably reduced by using k-d trees on serial machines ([13]),

[1] To have reliable results, this process is repeated 10 times with different partitions of 90% training and 10% test items, and the average success rate of these ten experiments is computed ([27]).

massively parallel machines ([26]), or Wafer-Scale Integration ([16]). In Natural Language Processing, lazy learning techniques are currently also being applied by various Japanese groups to parsing and machine translation under the names *exemplar-based translation* or *memory-based translation and parsing* ([16]).

Lazy learning has diverse intellectual dependencies: in AI techniques like memory-based reasoning and case-based reasoning, it is stressed that "intelligent performance is the result of the use of memories of earlier experiences rather than the application of explicit but inaccessible rules" ([26]). Outside the linguistic mainstream, people like Skousen, Derwing, and Bybee stress that "the analogical approach (as opposed to the rule-based approach) should receive more attention in the light of psycholinguistic results and new formalizations of the notion of analogy" ([25]; [12]), In cognitive psychology (e.g., [24]), exemplar-based categorization has a long history as an alternative for probabilistic and classical rule-based classification, and finally, in statistical pattern recognition, there is a long tradition of research on *nearest neighbour* classification methods which has been a source of inspiration for the development of lazy learning algorithms.

3.2 Variants of Lazy Learning

Examples are represented as a vector of feature values with an associated category label. Features define a pattern space, in which similar examples occupy regions that are associated with the same category (note that with symbolic, unordered feature values, this geometric interpretation doesn't make sense).

During *training*, a set of examples (the training set) is presented in an incremental fashion to the classifier, and added to memory. During *testing*, a set of previously unseen feature-value patterns (the test set) is presented to the system. For each test pattern, its distance to all examples in memory is computed, and the category of the least distant instance is used as the predicted category for the test pattern.

In lazy learning, performance crucially depends on the distance metric used. The most straightforward distance metric would be the one in equation (1), where X and Y are the patterns to be compared, and $\delta(x_i, y_i)$ is the distance between the values of the i-th feature in a pattern with n features.

$$\Delta(X,Y) = \sum_{i=1}^{n} \delta(x_i, y_i) \tag{1}$$

Distance between two values is measured using (2) for numeric features (using scaling to make the effect of numeric features with different lower and upper bounds comparable), and (3), an overlap metric, for symbolic features.

$$\delta(x_i, y_i) = \frac{|x_i - y_i|}{max_i - min_i} \tag{2}$$

$$\delta(x_i, y_i) = 0 \text{ if } x_i = y_i, \text{ else } 1 \tag{3}$$

3.3 Feature weighting

In the distance metric described above, all features describing an example are interpreted as being equally important in solving the classification problem, but this is not necessarily the case. Elsewhere ([7]; [10]) we introduced the concept of information gain (also used in decision tree learning, [20]) into lazy learning to weigh the importance of different features in a domain-independent way. Many other methods to weigh the relative importance of features have been designed, both in statistical pattern recognition and in machine learning (e.g., [1]; [15]; etc.), but the one we used is extremely simple and produced excellent results.

The main idea of *information gain weighting* is to interpret the training set as an information source capable of generating a number of messages (the different category labels) with a certain probability. The information entropy of such an information source can be compared in turn for each feature to the average information entropy of the information source when the value of that feature is known. Those features that reduce entropy most are most informative.

Database information entropy is equal to the number of bits of information needed to know the category given a pattern. It is computed by (4), where p_i (the probability of category i) is estimated by its relative frequency in the training set.

$$H(D) = -\sum_{p_i} p_i log_2 p_i \tag{4}$$

For each feature, it is now computed what the information gain is of knowing its value. To do this, we compute the average information entropy for this feature and subtract it from the information entropy of the database. To compute the average information entropy for a feature (5), we take the average information entropy of the database restricted to each possible value for the feature. The expression $D_{[f=v]}$ refers to those patterns in the database that have value v for feature f. V is the set of possible values for feature f. Finally, $|D|$ is the number of patterns in a (sub)database.

$$H(D_{[f]}) = \sum_{v_i \in V} H(D_{[f=v_i]}) \frac{|D_{[f=v_i]}|}{|D|} \tag{5}$$

Information gain is then obtained by (6), and scaled to be used as a weight for the feature during distance computation.

$$G(f) = H(D) - H(D_{[f]}) \tag{6}$$

Finally, the distance metric in (1) is modified to take into account the information gain weight associated with each feature.

$$\Delta(X, Y) = \sum_{i=1}^{n} G(f_i)\delta(x_i, y_i) \tag{7}$$

Even in itself, information gain may be a useful measure to discover which features are important to solve a linguistic task. Fig. 2 shows the information gain pattern for the prediction of the diminutive suffix of nouns in Dutch. In this task, features are an encoding of the two last syllables of the noun the diminutive suffix of which has to be predicted (there are five forms of this suffix in Dutch). Each part (onset, nucleus, coda) of each of the two syllables (if present) is a separate feature. For each syllable, the presence or absence of stress is coded as well. The feature information gain pattern clearly shows that most relevant information for predicting the suffix is in the rime (nucleus and coda) of the last syllable, and that stress is not very informative for this task (which conforms to recent linguistic theory about diminutive formation in Dutch).

3.4 Additional Extensions

Apart from the feature weighting solution, several other optimizations of the algorithm are possible. These concern, e.g., the use of symbolic features: when using the previous metric, all values of a feature are interpreted as equally distant to each other. This may lead to unsufficient discriminatory power between patterns. It also makes impossible the well-understood "Euclidean distance in pattern space" interpretation of the distance metric. Stanfill and Waltz ([26]) proposed a *value difference metric* which takes into

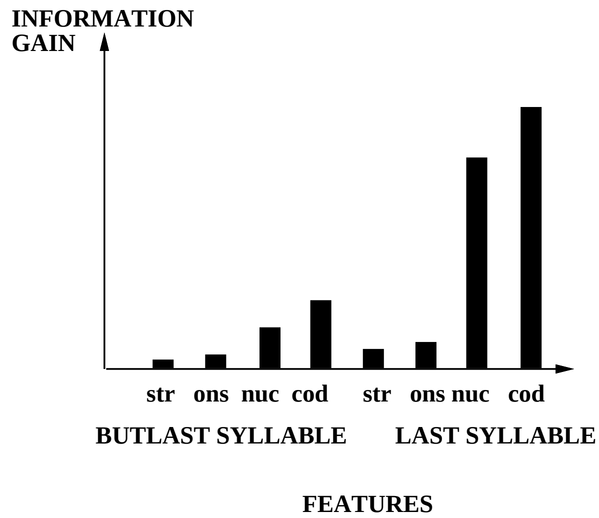

Figure 2: An example of an information gain pattern. The height of the bars expresses, for each feature describing an input word, the amount of information gain it contributes to predicting the suffix. Features are stress (str), onset (ons), nucleus (nuc), and coda (cod) of the last two syllables of the noun.

account the overall similarity of classification of all examples for each value of each feature. Recently, Cost and Salzberg ([5]) modified this metric by making it symmetric.

In addition, the exemplars themselves can be weighted, based on typicality (how typical is a memory item for its category) or performance (how well is an exemplar doing in predicting the category of test patterns), storage can be minimized by keeping only a selection of examples, etc.

4 Lazy Learning of Linguistic Tasks

Linguistic tasks (including lexical tasks) are context-sensitive mappings from one representation to another (e.g., from text to speech, from spelling to parse tree, from parse tree to logical form, from source language to target language etc.). These mappings tend to be many-to-many and complex because they can often only be described by conflicting regularities, subregularities, and exceptions.

In current NLP, these different levels of generalization have been the prime motivation for research into inheritance mechanisms and default reasoning ([6]; [4]), especially in research on the structure and organisation of the lexicon.

To illustrate the difference between the traditional knowledge-based approach with the lazy learning approach, consider Fig. 3. Suppose a problem can be described by referring to only two features (a typical problem would need tens or hundreds of features). In a knowledge-based approach, the computational linguist looks for dimensions (features) to describe the solution space, and formulates rules which in their condition part define areas in this space and in their action part the category or solution associated with this area. Areas may overlap, which makes necessary some form of rule ordering or "elsewhere condition" principle.

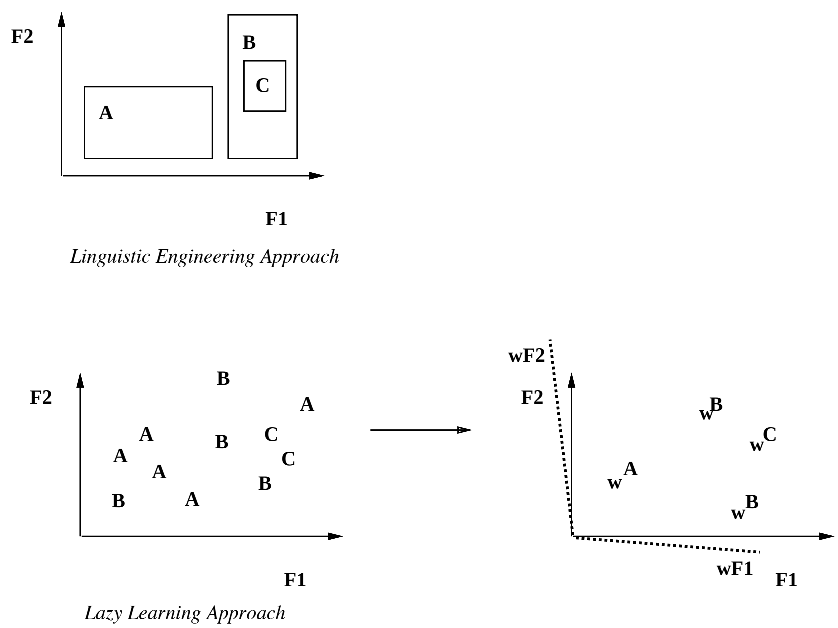

Figure 3: A graphical view of the difference between linguistic engineering (top, knowledge-based) and lazy learning (bottom, behaviour-based)

For example, the two dimensions might be case and number of adjectives in some language, and the three categories might be different suffixes associated with different combinations of values for the case and number features.

In a lazy learning approach, on the other hand, knowledge acquisition is automatic. We start from a number of examples, which can be represented as points in feature space. This initial set of examples may contain noise, misclassifications, etc. Information-theoretic metrics like information gain basically modify this feature space automatically by assigning more or less weight to particular features (dimensions). In constructive induction, completely new feature dimensions may be introduced for separating the different category areas better in feature space. Exemplar weighting and memory compression schemes modify feature space further by removing points (exemplars) and by increasing or decreasing the "attraction area" of exemplars, i.e., the size of the neighbourhood of an exemplar in which this exemplar is counted as the nearest neighbour. We are finally left with a reorganized feature space that optimally separates the different categories, and provides good generalization to unseen inputs. In this process, no linguistic engineering and no handcrafting were involved.

4.1 Linguistic Tasks as Classification

Lazy Learning is fundamentally a *classification* paradigm. Given a description in terms of feature-value pairs of an input, a category label is produced. This category should normally be taken from a finite inventory of possibilities, known beforehand[2]. It is our hypothesis that *all* useful linguistic tasks can be redefined this way. All linguistic problems can be described as context-sensitive mappings. These mappings can be of two kinds: *identification* and *segmentation* (identification of boundaries).

- **Identification**. Given a set of possibilities (categories) and a relevant context in terms of attribute values, determine the correct possibility for this context. Instances of this include *part of speech tagging*, *grapheme-to-phoneme conversion*, *lexical selection in generation*, *morphological synthesis*, *word sense disambiguation*, *term translation*, *stress assignment*, etc.

- **Segmentation**. Given a target and a context, determine whether and which boundary is associated with this target. Examples include *syllabification*, *morphological analysis*, *syntactic analysis* (in combination with tagging), etc.

[2] This restriction can be circumvented by having multiple classifiers predict a different part of the output pattern, see [18] for this approach in learning decision trees.

An approach often necessary to arrive at the context information needed is the *windowing* approach (as in [22] for text to speech), in which an imaginary window is moved one item at a time over an input string where one item in the window (usually the middle item or the last item) acts as a target item, and the rest as the context. An alternative possibility is to use *operators* as categories, e.g., shift and different types of reduce as categories in a shift-reduce parser (see [23] for such an approach outside the context of Machine Learning).

5 Examples

The approach proposed in this paper is fairly recent, and experiments have focused on phonological and morphological tasks rather than on tasks like term disambiguation. However, we hope to have made clear that the approach is applicable to all classification problems in NLP. In this section we briefly describe some of the experiments and hope the reader will refer to the cited literature for a more detailed description.

5.1 Syllable Boundary Prediction

Here the task to be solved is to decide where syllable boundaries should be placed given a word form in its spelling or pronunciation representation (the target language was Dutch). In a knowledge-based solution, we would implement well-known phonological principles like the *maximal onset principle* and the *sonority hierarchy*, as well as a *morphological parser* to decide on the position of morphological boundaries, some of which overrule the phonological principles. This parser requires at least lexical knowledge about existing stems and affixes and the way they can be combined.

In the lazy learning approach ([7]; [8]), we used the windowing approach referred to earlier to formulate the task as a classification problem (more specifically, a segmentation problem). For each letter or phoneme, a pattern was created with a target letter or phoneme, a left context and a right context. The category was *yes* (if the target letter or phoneme should be preceded by a syllable boundary) or *no* if not. The lazy learning approach produced results which were more accurate than both a connectionist approach (backpropagation learning in a recurrent multi-layer perceptron) and a knowledge-based approach. The information gain metric also "discovered" an interesting asymmetry between predictive power of left and right context (right context turned out to be more informative).

5.2 Grapheme-to-Phoneme Conversion

Grapheme-to-phoneme conversion is a central module in text-to-speech systems. The task here is to produce a phonetic transcription given the spelling of a word. Again, in the knowledge-based approach, the lexical requirements for such a system are extensive. In a typical knowledge-based system solving the problem, morphological analysis (with lexicon), phonotactic knowledge, and syllable structure determination modules are designed and implemented.

In a lazy learning approach ([9]; [3]), again a windowing approach was used to formulate the task as a classification problem (identification this time: given a set of possible phonemes, determine which phoneme should be used to translate a target spelling symbol taking into account its context). Results were highly similar to the syllable boundary prediction task: the lazy learning approach resulted in systems which were more accurate than both a connectionist approach and a linguistically motivated approach. The results were replicated for English, French, and Dutch, using the same lazy learning algorithm, which shows its reusability.

5.3 Word Stress Assignment

Another task we applied the lazy learning algorithm to, was stress assignment in Dutch monomorphematic, polysyllabic words ([10], [11]). A word was coded by assigning one feature to each part of the syllable structure of the last three syllables (if present) of the word (see the description of the diminutive formation task described earlier). There were three categories: final stress, penultimate stress, and antepenultimate stress (an identification problem).

Although this research was primarily intended to show that an empiricist learning method with little a priori knowledge performed better than a learning approach in the context of the "Principles and Parameters" framework as applied to metrical phonology, the results also showed that even in the presence of a large amount of noise (from the point of view of the learning algorithm), the algorithm succeeded in automatically extracting the major generalizations that govern stress assignment in Dutch, with no linguistic a priori knowledge except syllable structure.

5.4 Part of Speech Tagging

In this as yet unpublished research, a slightly more complex learning procedure was applied to the problem of part of speech tagging (an identification problem). First, a *lexicon* was derived from the training set. The training set consists of a number of texts in which each word is assigned the correct part of speech tag (its category). To derive a lexicon, we find for each word how many times it was associated with which categories. We can then make an inventory of *ambiguous categories*, e.g., a word like *man* would belong to the ambiguous category *noun-or-verb*. The next step consists of retagging the training corpus with these ambiguous categories. Advantages of this extra step are (i) that ambiguity is restricted to what actually occurs in the training corpus (making as much use as possible of sublanguage characteristics), and (ii) that we have a much more refined measure of similarity in lazy learning: whereas non-ambiguous categories can only be equal or not, ambiguous categories can be *more or less* equal. For the actual tagging problem, a moving window approach was again used, using patterns of ambiguous categories (a target and a left and right context). Results are only preliminary here, but suggest a performance comparable to hidden markov modeling approaches.

6 Conclusion

There are both theoretical and practical aspects to the work described in this paper. First, as far as linguistic engineering is concerned, a new approach to the reusability problem was proposed. Instead of concentrating on linguistic engineering of theory-neutral, poly-theoretic, multi-applicable lexical representations combined with semi-automatic migration of lexical knowledge between different formats, we propose an approach in which a single inductive learning method is reused on different corpora representing useful linguistic mappings, acquiring the necessary lexical information automatically and implicitly.

Secondly, the theoretical claim underlying this proposal is that language acquisition and use (and a fortiori lexical knowledge acquisition and use) are behaviour-based processes rather than knowledge-based processes. We sketched a *memory-based lexicon* with the following properties:

- The lexicon is not a static data structure but a set of lexical processes of identification and segmentation. These processes implement lexical performance.

- Each lexical process is represented by a set of exemplars (solved cases) in memory, which act as models to new input.

- New instances of a lexical process are solved through either memory lookup or similarity-based reasoning.
- There is no representational difference between regularities, subregularities, and exceptions.
- Rule-like behaviour is a side-effect of the operation of the similarity matching process and the contents of memory.
- The contents of memory (the lexical exemplars) can be approximated as a set of rules for convenience.

In a broader context, the results described here argue for an empiricist approach to language acquisition, and for exemplars rather than rules in linguistic knowledge representation (see [11] and Gillis et al. [14] for further discussion of these issues).

There are also some limitations to the method. The most important of these is the *sparse data problem.* In problems with a large search space (e.g., thousands of features relevant to the task), a large amount of training patterns is necessary in order to cover the search space sufficiently. In general, this is not a problem in NLP, where for most problems large corpora are available or can be collected. Also, information gain or other feature weighting techniques can be used to automatically reduce the dimensionality of the problem, sometimes effectively solving the sparse data problem.

Another problem concerns *long-distance dependencies*, especially in syntax. The methods described often make use of a moving window approach in which only a local part of an input representation is used. Whenever important factors determining a category decision are outside the scope of a pattern, the category assignment cannot be learned. A possible solution for this problem is the cascading of different lazy learning systems, one working on the output of the other. For example, a learning system for part of speech tagging could be combined with a learning system taking patterns of disambiguated tags as input, and producing constituent types as output. Taking patterns of constituent types as input, a third learning system should have no problem assigning "long-distance" dependencies: given the right representation, all dependencies are local.

References

[1] Aha, D.: A study of Instance-Based Algorithms for Supervised Learning Tasks. University of California at Irvine technical report 90-42, 1990.

[2] Aha, D., Kibler, D. and Albert, M.: Instance-Based Learning Algorithms. *Machine Learning* 6, (1991) 37–66.

[3] Van den Bosch, A. and Daelemans, W.: 'Data-oriented methods for grapheme-to-phoneme conversion.' Proceedings of the Sixth conference of the European chapter of the ACL, ACL, (1993) 45–53.

[4] Briscoe, T., de Paiva, V. and Copestake, A.: *Inheritance, Defaults and the Lexicon.* Cambridge: Cambridge University Press, 1993.

[5] Cost, S. and Salzberg, S.: A weighted nearest neighbour algorithm for learning with symbolic features. *Machine Learning* 10, (1993) 57–78.

[6] Daelemans, W. and Gazdar, G.: (guest eds.) Special Issue *Computational Linguistics* on Inheritance in Natural Language Processing, 18 (2) and 18 (3), 1992.

[7] Daelemans, W. and van den Bosch, A.: Generalization Performance of Backpropagation Learning on a Syllabification Task. In: M.F.J. Drossaers and A. Nijholt (eds.) *Connectionism and Natural Language Processing.* Proceedings Third Twente Workshop on Language Technology, (1992) 27–38.

[8] Daelemans, W. and van den Bosch, A.: 'A Neural Network for Hyphenation.' In: I. Aleksander and J. Taylor (eds.) *Artificial Neural Networks II: Proceedings of the International Conference on Artificial Neural Networks.* Elsevier Science Publishers, (1992) 1647–1650.

[9] Daelemans, W. and van den Bosch, A.: 'TABTALK: Reusability in Data-oriented grapheme-to-phoneme conversion.' *Proceedings of Eurospeech*, Berlin, (1993) 1459–1466.

[10] Daelemans, W., Gillis, S., Durieux, G., van den Bosch, A.: Learnability and Markedness in Data-Driven Acquisition of Stress. In: T. Mark Ellison and James M. Scobbie (eds) *Computational Phonology.* Edinburgh Working Papers in Cognitive Science 8, (1993) 157–178.

[11] Daelemans, W., Gillis, S., and Durieux, G.: 'The Acquisition of Stress, a data-oriented approach.' *Computational Linguistics* 20 (3), (1994) forthcoming.

[12] Derwing, B. L. and Skousen, R.: Real Time Morphology: Symbolic Rules or Analogical Networks. *Berkeley Linguistic Society* 15: (1989) 48–62.

[13] Friedman, J., Bentley, J., and Finkel, R., an algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, (1977) 3 (3).

[14] Gillis, S., Daelemans, W., Durieux, G. and van den Bosch, A.: 'Learnability and Markedness: Dutch Stress Assignment.' In: *Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society*, Boulder Colorado, USA, Hillsdale: Lawrence Erlbaum Associates, (1993) 452–457.

[15] Kira, K. and Rendell, L.: A practical approach to feature selection. Proceedings International Conference on Machine Learning, 1992.

[16] Kitano, H.: Challenges of massive parallelism. Proceedings IJCAI 1993, 813–834.

[17] Kolodner, J.: Case-Based Reasoning. San-Mateo: Morgan-Kaufmann. 1993.

[18] Ling, C.: Learning the past tense of English verbs: The symbolic Pattern Associator vs. Connectionist Models. *Journal of Artificial Intelligence Research* 1, (1994) 209–229.

[19] Pustejovsky, J.: Dictionary/Lexicon. In: Stuart C. Shapiro (ed.), *Encyclopedia of artificial intelligence*, New York: Wiley, 1992, 341–365.

[20] Quinlan, J. R.: Induction Of Decision Trees. *Machine Learning* 1, (1986) 81–106.

[21] Salzberg, S.: A nearest hyperrectangle learning method. *Machine Learning* 6, (1990) 251–276.

[22] Sejnowski, T. and Rosenberg, C.: NETtalk: a parallel network that learns to read aloud. *Complex Systems* 1, (1986) 145–168.

[23] Simmons, R. and Yu, Y.: The acquisition and use of context-dependent grammars for English. *Computational Linguistics* 18 (3) (1992), 391–418.

[24] Smith, E. and Medin, D.: *Categories and Concepts.* Cambridge, MA: Harvard University Press, 1981.

[25] Skousen, R.: *Analogical Modeling of Language.* Dordrecht: Kluwer, 1989.

[26] Stanfill, C. and Waltz, D.L.: Toward Memory-based Reasoning. *Communications of the ACM* (1986) 29: 1213–1228.

[27] Weiss, S. and Kulikowski, C.: *Computer systems that learn.* San-Mateo: Morgan Kaufmann, 1991.

[28] Winston, P.: Artificial Intelligence. Reading Mass.: Addison-Wesley, 1992.