In: Van Eynde, F. and D. Gibbon. Lexicon Development for speech and language processing
Kluwer Academic Publishers, 115-139, 2000

Inductive Lexica

Walter Daelemans and Gert Durieux

Abstract. Machine Learning techniques are useful tools for the automatic extension of existing lexical databases. In this paper, we review some symbolic machine learning methods which can be used to add new lexical material to the lexicon by automatically inducing the regularities implicit in lexical representations already present. We introduce the general methodology for the construction of *inductive lexica*, and discuss empirical results on extending lexica with two types of information: pronunciation and gender.

1. Introduction

Computational lexicology and lexicography (the study of the structure, organization, and contents of computational lexica) have become central disciplines both in language engineering and in theoretical computational linguistics. Most language engineering applications are in need of rich lexical knowledge sources, and in computational linguistics theory, the role of the lexicon has become increasingly important in linguistic formalisms, such as GPSG, HPSG, and TAG.

A lot of attention in the field has been directed towards issues in lexical knowledge representation: the design and evaluation of formalisms for the representation of lexical knowledge, e.g. Evans and Gazdar (1996) or Briscoe et al. (1993). Although adequate representation is important, paying too much attention to the issue of formalisms incurs a risk of throwing language engineering into a malaise similar to the "AI-winter" in expert systems technology during the eighties and early nineties. At that time, AI research was producing Knowledge System development shells using a wide range of formalisms, but neglected to fill them with useful knowledge. The lesson learned from the limited impact of these expert system shells on industry is that an expert system should first and foremost contain the knowledge necessary to solve the customers' problem, rather than relying on the users to provide that knowledge. Whether the formalism used to represent this knowledge is rule-based, first order predicate calculus or a semantic network is of less concern.

Similarly, in computational lexicography, lexica of language engineering applications should come with acceptable lexical coverage, and with the information necessary for the intended applications. They should also come equipped with methods for the automatic extension of the lexicon with new lexical entries. Whether these lexical entries are represented as DATR theorems, as typed feature structures, or as a record in a 'flat file' may be less crucial. The main research issue in computational lexicology is therefore to try to solve the following problem:

> All computational lexica are inherently incomplete because of (i) missing lexical entries, and (ii) missing information about lexical entries.

On closer inspection, though, missing lexical entries are not really a problem: either we don't need them in a particular application, and then we don't have to know that they exist, or we do need them, but then we will encounter some of their associated information (probably their spelling or pronunciation), and we will know some of the contexts they appear in. In that case, they are not missing, because the information present in the lexicon is sufficient to construct a surprising amount of additional lexical information, provided we have corpora and/or lexical databases available. As we shall see, this holds true even if the former contains only few lexical entries. The problem of missing lexical entries therefore reduces to the problem of extending existing lexical entries with additional information.

This paper addresses the automatic extension of lexica using symbolic machine learning techniques; in-depth discussion of alternative, quantitative methods such as neural networks or statistical approaches is beyond the scope of this paper. It is our belief that machine learning techniques allow the accurate prediction of lexical information associated with new lexical items on the basis of extracted regularities from the lexical information already present in a computational lexicon. First, we will define the place of this approach in the broader area of lexical acquisition (Section 2). Section 3 gives a short tutorial overview of relevant Machine Learning techniques, focusing on two approaches which we think are especially relevant for lexical acquisition: memory-based learning and decision tree induction. Section 4, finally, provides an overview of the general methodology of lexical extension proposed here, and presents two case studies: (i) the prediction of the pronunciation of a lexical item from its spelling, and (ii) the prediction of the gender of a Dutch noun on the basis of its phonological structure.

2. Approaches to Lexical Acquisition

To alleviate the task of hand-coding and extending large lexica, lexicographic environments have been designed, e.g. ONTOS and LUKE (see Wilks et al. (1996)) or WORD MANAGER (Domenig and ten Hacken, 1992). These environments can speed up acquisition by the semi-automatic computation of some information, i.e. algorithmic computation combined with manual checking, or by presenting the lexicographer with a set of contexts containing a new word, the grammaticality of which should be checked. On the basis of feedback from the lexicographer, lexical information about the word is then deduced. Useful as these environments may be, it will be intuitively clear that they do not constitute a cost-efficient solution to the enormity of the lexical acquisition and extension tasks. As noted in Wilks et al. (1996), there is a problem even with the very concept of hand-crafting lexical databases, as e.g. in WORDNET (Miller, 1990), since they can never be task nor theory independent.

A second approach, used from the mid-eighties onwards, makes use of Machine Readable Dictionaries (MRDs) to construct computational lexica (Wilks et al., 1996). The results of this approach have been criticized for being incomplete and inconsistent, because the base MRDs were developed with human users in mind (Ide and Véronis, 1995). We will show how machine learning techniques can nevertheless extend and refine computational lexica bootstrapped from MRDs.

The methodological context of this paper is the use of inductive techniques for the automatic extraction of lexical knowledge from corpora.[1] Recent work on corpus-based lexical acquisition (see Boguraev and Pustejowsky (1996) for a representative collection of recent research, and Zernik (1991) for older work) suggests that useful lexical information can be extracted from such corpora. In our opinion, the application of machine learning techniques to language learning, until recently a largely independent research activity (see e.g. Daelemans et al. (1997), and various links at the ACL SIGNLL home page[2]), is a powerful alternative or complementary approach to statistical lexical acquisition. This paper introduces the latter approach for lexical acquisition; Barg (1994) presents a different machine learning approach to lexical learning.

3. Machine Learning Crash Course

Machine Learning (ML) is the sub-discipline of Artificial Intelligence (AI) that studies algorithms that can learn either from experience or by reorganizing the knowledge they already have (see Mitchell (1997),

[1] With this term we mean raw text corpora, annotated text corpora, and existing lexical databases.

[2] Association of Computational Linguistics Special Interest Group in Natural Language Learning; URL: `http://signll.aclweb.org/~signll/`.

Langley (1996) and Carbonell (1990) for introductory material, Weiss and Kulikowski (1991) for methodological issues, and Natarajan (1991) for a formal-theoretical approach).

Conceptually, a learning system consists of a *performance component* which performs a specific task (given an input, it produces an output), and a *learning component* which modifies the performance component on the basis of its experience in such a way that performance of the system in doing the same or similar tasks improves (Figure 1). Experience is represented as a set of examples used to train the system. Examples usually take the form of a set of attribute/value pairs (the *predictor* attributes) together with their associated desired output (the *class* or *target* attribute). E.g., in mushroomology, the predictor attributes might describe a mushroom in terms of the shape, texture, and color of its parts, and its odor, and the desired output its edibility (edible or poisonous). In lexicology, the predictor attributes might be a description of a word in terms of its syllable structure and segmental material, and the class attribute its syntactic category. In the first case, we obtain the examples by collecting various mushrooms, describing their appearance, and testing their edibility; in the second case, we either provide the examples ourselves, or get them from corpora or existing lexical databases. Machine Learning algorithms can be successful in generalizing from these examples to new, previously unseen cases, i.e. new descriptions of mushrooms or nouns.

To perform its task, the performance component uses an internal representation. The task of the learning component may therefore be construed as a search in the space of possible representations for a representation that is optimal for performing the mapping. A large number of formalisms has been proposed for the internal representations of lexical acquisition systems: e.g. decision trees, case bases, taxonomies, and sets of probabilities. In most cases, finding *the* optimal representation given a set of examples and a representation language is computationally intractable. Some form of heuristic search is therefore used by all learning systems.

In Machine Learning, the concept of *bias* refers to domain- or algorithm-dependent constraints on the search process: knowledge about the task may be used to make the search simpler. There may also be bias in the way the experience presented to the learning component (the training examples) is preprocessed. The addition of linguistic bias to a learning system is the obvious way to let learning systems profit from linguistic knowledge about the task. A radically empiricist stance will of course strive for learning systems where linguistic bias is close to zero, and only domain-independent inductive methods are used.

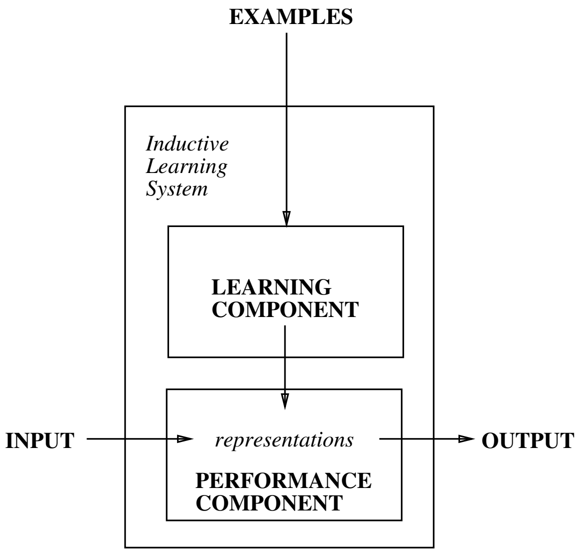

Figure 1. General architecture of an inductive learning system

3.1. Classification of Learning Methods

Given this very general model of inductive learning, a number of dimensions can be distinguished that should be considered in comparing and experimenting with these techniques.

- *Amount of Supervision.* In supervised learning, experience takes the form of examples, which consist of sets of attribute/value pairs describing some relevant properties of objects and a corresponding class attribute. These examples are presented to the system during a training phase. In *unsupervised learning*, examples are presented without information about their intended class. It is up to the system to exploit similarities within the examples in such a way that they can be used by the performance component to solve the task.[3]

- *Input Representation.* Commonly used representations for the predictor attributes include vectors of bits, ordered sets (vectors) of attribute/value pairs, where the values may be numeric or nominal (compare 'flat' feature structures in linguistics), or complex

[3] Several Machine Learning approaches have both supervised and unsupervised variants, e.g. the widely used back-propagation learning algorithm for neural networks is a supervised method, whereas the Self-Organizing Map implements an unsupervised variant.

recursive representations such as semantic nets (compare recursive feature structures in linguistics).

- *Output Representation.* The values for the class attribute may be a simple binary category (i.e. a yes/no decision), a symbolic category (a finite, discrete set of labels), a continuous category (a real number), or a vector of any of these.
- *Internal Representation.* The representation used by the performance component, and optimized by the learning component can be numeric (e.g. connection weights with neural networks) or symbolic (semantic nets, rules, decision trees, taxonomies, cases, ...).
- *Incremental Learning.* A learning system can be incremental. In that case, relevant information in additional examples can be integrated by the learning component into the performance component without re-learning everything from scratch. In non-incremental or *batch learning* systems, such as most neural networks, this is not possible. In batch learning, the complete set of examples has to be inspected—sometimes several times—before learning is completed, and the addition of new examples makes complete re-learning necessary.

3.2. Performance Evaluation

The success of a learning component in improving performance can be evaluated using a number of different quantitative and qualitative measures:

- *Generalization accuracy.* What is measured here, is the performance accuracy of the system on previously unseen inputs (i.e. inputs it was not trained on). This aspect of learning is of course crucial: it gives an indication of the quality of the *inductive leap* made by the algorithm on the basis of the examples. Good generalization accuracy indicates that the learning system has avoided *overfitting* on the training examples; this problem occurs mainly in noisy domains (cf. *infra*), when the learning component tries too hard to accommodate all idiosyncrasies of the training set, leading to overly specific representations which fail to capture the overall domain regularities. In order to get a good estimate of the real generalization accuracy, *cross-validation* techniques can be used, e.g. in 10-fold cross-validation an algorithm is tested on ten different partitions of the full data set available. In each run 90% of the data is used as training material, and 10% is set aside for testing. Each data item occurs once in one of the test sets. The average generalization accuracy over the ten test sets is then a good statistical estimate of the real accuracy.

– *Space and time complexity.* This covers both the amount of storage and processing involved in training the system and in performance, i.e. producing output given the input.

– *Explanatory Quality.* Usefulness of the representations found by the learning system as an explanation of the way the task is performed. When the system outputs e.g. a set of rules, these can be inspected by a human expert, and thus have potentially high explanatory quality. By contrast, the final set of connection weights reached by a neural network training algorithm is far more difficult to assess.

– *Noise Tolerance.* Different algorithms can be more or less sensitive to noise in the input. Noise can result from wrongly coded examples, missing values, or even from ambiguous examples, i.e. examples which have been assigned contradictory outputs in the training set. Algorithms dealing with linguistic data should be noise-resistant, if only for the simple reason that almost any linguistic domain is replete with sub-regularities and exceptions.

3.3. Overview of Methods

To sum up this introductory section on Machine Learning, we will give an intuitive description of how some of the algorithms work, using a prediction task where grammatical category is to be predicted from syllable structure and segmental content. This example application is typical of a large number of lexical acquisition and extension tasks: given a previously unseen word for which lexical information has to be acquired, this information can be induced in large part from the correspondences between form and category in known form-category pairs.

We discuss the algorithms in order of increasing abstraction of the internal representation. We start from storage and *table-lookup* of the 'raw' examples as a non-learning baseline.

– *Table Look-Up.* Store all examples (patterns of syllable structure of target words and their corresponding syntactic category) in a table. When a new input pattern is given to the performance system, look it up in the table, and retrieve the output of the stored example. In this approach, the system does not actually learn anything, and it fails miserably whenever an input pattern is not present in the table. In other words, there is no real generalization. However, surprising as it may seem, this approach sometimes shows performance accuracies similar to those of sophisticated inductive or statistical techniques. The reason for this is that—given a sizeable lexicon and a suitable input representation—the possibility of generalization comes to lie in large part with the input encoding: several words (including previously unseen ones) may be represented by the same input pattern and retrieval thus becomes a (rather crude) form of generalization. Some kinds of representation (e.g. windowing: sliding an imaginary fixed-width window over the input word, and assigning a new pattern to each 'snapshot') incorporate a marked generalization effect in this way.

– *Memory-Based Learning.* Store all examples in a table. When a new input pattern is given to the performance system, look up the most *similar examples* to the new pattern (in terms of the number of identical segments in identical positions in both the stored pattern and the new pattern, for example), and extrapolate from the categories assigned to these *nearest neighbors* of the new case. Various statistical and information-theoretic techniques can be used to design a suitable *similarity metric*. The definition of similarity is also a place where linguistic bias can be introduced in the learning algorithm. We could, for example, decide that the last syllable is more important than the other syllables of a word, and consider mismatches in the last syllable as more important than mismatches in other parts of the word.

– *Rule and Decision Tree Induction, Conceptual Clustering.* Use similarities and differences between examples to construct a decision tree or a rule set, and use this constructed representation to assign a category to a new input pattern. Forget the individual examples. In the unsupervised variant, examples do not come preclassified, but consist only of a set of attribute/value pairs. The unsupervised algorithms organize these examples into taxonomies, by creating, expanding and refining classes according to some measure of usefulness. Unlike the case of supervised algorithms, performance cannot be measured by comparing the system's predictions with the 'correct' categories. Instead, infer missing feature values by examining the node(s) in the induced taxonomy that the example is classified at.

– *Connectionism, Neural Networks.* Use the examples to train a network. In back-propagation learning, this training is done by repeatedly iterating over all examples, comparing for each example the output predicted by the network to the desired output, and changing connection weights between network nodes in such a way that performance increases. Keep the connection weight matrix, and forget the examples. In the unsupervised variant, neural network dynamics implementing some form of similarity computation on input patterns, self-organize a network of neurons (usually a two-dimensional grid) into a map where patterns are represented by neurons, and nearness of patterns on the map indicates closeness. In our example, e.g. verbs could tend to cluster together.

In terms of the amount of abstraction introduced during the learning phase, conceptual clustering and rule induction approaches are *eager learning* techniques.[4] These techniques abstract knowledge from the examples as soon as they are presented, and the examples themselves are forgotten. Memory-Based Learning is a *lazy learning* technique; generalization only occurs when a new pattern is offered to the performance component, and abstraction is therefore implicit in the way the contents of the case base and the similarity metric interact. We will continue this section by describing a typical lazy and a typical eager learning technique in somewhat more detail.

Memory-Based Learning

The memory-based learning paradigm is founded on the hypothesis that performance in cognitive tasks (in our case: language processing) is based on identifying analogies between new situations and stored representations of earlier experiences, and reasoning from those, rather than on the application of *mental rules* abstracted from representations of earlier experiences, as in rule induction and rule-based processing.

The concept has appeared several times in AI disciplines from computer vision to robotics, bearing such diverse labels as similarity-based learning, example- (or exemplar-) based learning, analogical reasoning, lazy learning, nearest-neighbor classifiers, instance-based learning, and case-based reasoning (Stanfill and Waltz, 1986; Kolodner, 1992; Aha et al., 1991; Salzberg, 1990).

Examples are represented as vectors of attribute values with an associated class label. Those attributes define a pattern space. During training, a set of examples (the training set) is presented in an incremental fashion to the learning algorithm, and added to memory. During processing, an input vector of attribute values, describing a previously

[4] The same applies to statistical models and neural network approaches.

unseen test pattern, is presented to the system. Its similarity (or *distance*) to all examples in memory is computed using a *similarity metric*, and the category of the most similar instance(s) is used as a basis to predict the category for the test pattern.

In this type of lazy learning, performance crucially depends on the similarity metric used. The most straightforward metric for linguistic problems with nominal (non-numeric) values would be an *overlap metric*: similarity is defined as the number of attribute values that are equal in two patterns being compared. In such a similarity metric, all attributes describing an example are interpreted as being equally important in solving the classification problem. However, this is not necessarily the case: in part of speech tagging e.g., the category of the word immediately before the word to be tagged is obviously more important than the category of the word three positions earlier in the sentence. We will call this problem the *feature relevance problem*. Various feature weighting and selection methods have been proposed to differentiate between the features on the basis of their relevance for solving the task (see Wettschereck et al. (1996) for an overview).

Another addition to the basic algorithm that has proved relevant for many natural language processing tasks is the introduction of a value difference metric (Stanfill and Waltz, 1986; Cost and Salzberg, 1993). Such a metric assigns different distances to pairs of values for the same attribute. In tagging e.g., such a metric would assign a smaller distance between proper nouns and common nouns than between proper nouns and adjectives, for example. These biases can of course also be manually added to the learner by a domain expert. Several other improvements and modifications to the basic memory-based learning scheme have been proposed and should be investigated for linguistic problems. Two promising further extensions are weighting the examples in memory, and minimizing storage by keeping only a selection of examples. In example weighting, examples are differentiated according to their quality as predictors for the category of new input patterns. This quality can be based on their typicality or on their actual performance as predictors on a held-out test set. In example selection, memory is pruned by deleting those examples which are bad predictors or which are redundant.

Decision Tree Learning and Rule Induction

The *decision tree learning* paradigm is based on the assumption that similarities between examples can be used to automatically extract decision trees and categories with both explanatory and generalization power. In other words, the extracted structure can be used to solve new instances of a problem, and to explain why a performance system behaves the way it does. In this paradigm, learning is *eager*, and abstraction occurs at learning time. There are systematic ways in which decision trees can be transformed into rule sets (the two representations are equivalent).

Decision tree induction is a well-developed field within AI, see e.g. Quinlan (1993) for a synthesis of major research findings. More ancient statistical pattern recognition work such as Hunt et al. (1966) and Breiman et al. (1984) also still makes for useful reading.

Decision tree learning works by repeatedly dividing the set of examples into subsets according to whether the examples in a particular subset have an attribute/value pair in common, until the subsets are homogeneous, i.e. all examples in the subset have the same class. The algorithm achieves this according to the simplified recursive scheme in Figure 2.

Given a set of examples T

If T contains only examples belonging to the same class C_j, then the decision tree for T is a leaf with category C_j.

If T contains different classes then

- Choose an attribute, and partition T into subsets that have the same value for the attribute chosen. The decision tree consists of a node containing the attribute name, and a branch for each value leading to a subset.
- Apply the procedure recursively to subsets created this way.

Figure 2. Recursive scheme for constructing decision trees

To classify new input patterns with a decision tree, start at the top node of the tree, and find the value in the input pattern for the corresponding attribute. Take the branch corresponding to that value, and perform this process recursively until a leaf node is reached. The category corresponding to this leaf node is the output.

Again, we are confronted with a *feature relevance problem* in this approach. In order to obtain a concise tree with good generalization performance (i.e. a tree reflecting the structure of the domain), we have to select at each recursion step in the above algorithm a test which is optimal for achieving this goal. The algorithm is non-backtracking, and considering all trees consistent with the data is an NP-complete problem, so a reliable heuristic feature selection criterion is essential. Usually, information-theoretic or statistical techniques are applied to maximize homogeneity of subsets. Several variants of and extensions to the basic algorithm have been developed, dealing with issues such as pruning (i.e. making the tree more compact by cutting off subtrees on the basis of a statistical criterion), grouping similar values of an attribute into classes, making tree building incremental, etc.

We have seen in this section that techniques developed in Machine Learning can in principle be used to predict unknown properties associated with linguistic objects such as lexical entries, on the basis of known properties (the attributes of the input pattern), and a set of examples. In the next section, we will see how these inductive techniques can be used to give lexica self-extending properties. We will first describe the general approach, and then go on to present two case studies illustrating the method.

4. Making Lexica Learn

In its most general formulation, a computational lexicon is a set of lexical entries, and a lexical entry a set of *lexical predicates* (propositions about some linguistic object). E.g. the lexical entry for a linguistic object labeled RED could be:

pronunciation(RED)	/'rEd/
spelling(RED)	red
syncat(RED)	(ADJ or N)

Lexical entries can correspond to various linguistic types of units: morphemes, base forms of words, word forms, idioms, phrases. The predicates can represent various types of linguistic knowledge: orthographic information may include spelling variants or hyphenation positions; phonetic and/or phonological predicates can describe pronunciation, word stress or syllable structure; morphological predicates can list component morphemes; syntactic predicates may provide information on argument structure, syntactic category, and agreement features, or even specify complete lexicalized syntactic trees (as in Tree Adjoining Grammar); semantic/pragmatic predicates, finally, may consist of case frames, selection restrictions, etc. Lexical predicates may also refer to extra-linguistic knowledge (e.g. domain concepts). Rules for the derivation of lexical properties would normally be taken as part of the different linguistic domains they refer to, but in some lexicon architectures, these rules can belong conceptually to the lexicon as well.

The basic idea behind *inductive lexica* is to use an available lexicon, however small, and, if available, a corpus, as a source to bootstrap lexical acquisition. Lexical predicates of newly encountered words are computed by reference to similar words previously encountered, for which the lexical information wanted is available. Depending on the

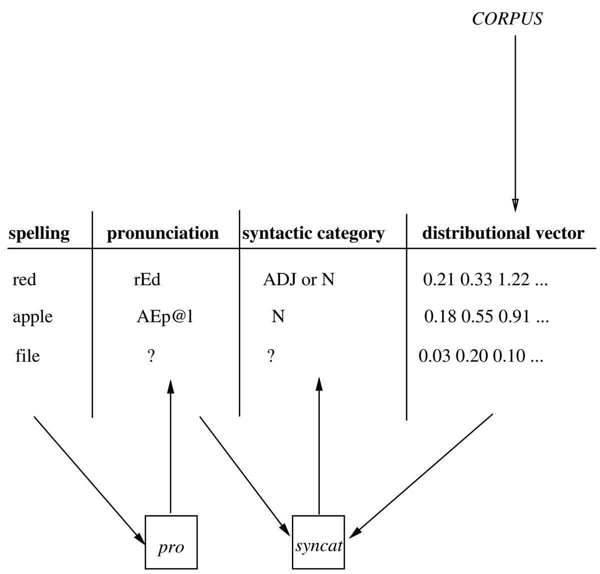

Figure 3. An inductive lexicon

lexical information to be predicted for the new word, different sources of information about the word are used as predictors.

Consider the following example (Figure 3). We have a small lexicon of word forms with their spelling, their pronunciation, and their possible syntactic categories. For each lexical entry, we also have a distributional vector, based on indexes to positions in a corpus where realizations of that lexical entry occur (e.g. comparable to Schütze (1993)). Given a word for which no lexical information is available yet, we have its spelling and its distributional vector, representing its occurrences in a corpus, as information. To compute lexical predicates for the new word, we can bootstrap from the available lexical information: (i) to determine its possible syntactic categories: find known words which have a similar form (spelling, phonology) and a similar syntactic behavior (i.e. occur in similar syntactic contexts as their distributional vector is similar), and extrapolate from their category, (ii) to determine its pronunciation, extrapolate from known words in the lexicon with a spelling similar to the new word, to the pronunciation of that new word. In this approach, therefore, an unknown target predicate of a lexical entry is predicted on the basis of known lexical predicates of that lexical entry, known target predicates and other predicates of other lexical entries, and (sometimes) also from corpus information.

For each lexical predicate to be predicted (the target predicate), it is decided which sources of information (other lexical predicates or operationalizable corpus information) are relevant to its prediction. These sources of information are represented in terms of an attribute/value vector. The next step is the construction of a *classifier* using e.g. decision tree induction. In our example, we have two classifiers, one for predicting pronunciation (*pro*), and one for predicting syntactic category (*syncat*). The training material for this classifier is built from those lexical entries for which the target predicate is known. For each of these entries the input features and the associated output category (the target predicate) are collected, and this is used as training material for training the classifier. Inductive lexica are neutral as far as lexical representation formalisms are concerned. The only addition is the construction of a classifier for each lexical predicate (as far as it makes sense to try to predict that particular predicate). When using eager learning methods such as decision tree building or rule induction, this classifier is an actual data structure, when using a lazy learning method such as memory-based learning, the 'extracted' classifier is conceptual; the classification is done on the fly from the lexical entries themselves, rather than from a data structure extracted from them. Inductive lexica therefore fit a supervised learning paradigm, and can be either eager or lazy. In the case of lazy learning, they are also incremental, taking into account immediately any lexical entries added to the lexicon in predicting new lexical predicates, whereas most eager learning methods call for explicit retraining when new lexical entries are added.[5]

In the remainder of this section, we will illustrate the feasibility of this inductive lexicon architecture by means of two case studies.

4.1. Case Study 1: Word Pronunciation

Recently, the Flemish government funded a speech and language technology project called FONILEX which aimed at constructing a pronunciation lexicon for the Flemish variety of Dutch. The resulting lexical database contains the most frequent words of Dutch with their Flemish-Dutch pronunciations.[6] The inductive lexicon approach was applied in this project as one of several approaches used to build the desired lexicon.

Traditionally, grapheme-to-phoneme conversion (the computation of pronunciation representations on the basis of the spelling of words), is supposed to involve the formalization and application of different lev-

[5] But, as mentioned above, incremental versions of e.g. decision tree building exist, so the dichotomy is not absolute.

[6] The project was coordinated by the Centre for Computational Linguistics (University of Leuven), with participation from the Centre for Dutch Language and Speech (University of Antwerp) and the ELIS research group (University of Ghent).

els of linguistic description and knowledge (phonotactics, phonology, morphology, syntax). MITALK (Allen et al., 1987) is a classical example of a rule-based solution to the problem. It is, however, possible to achieve excellent grapheme-to-phoneme conversion accuracy using machine learning techniques (Daelemans and van den Bosch, 1996).

To make this problem suitable for machine learning algorithms, the following steps have to be taken:

- Automatic alignment. In order to make full use of the generalization possibilities implicit in splitting up the task into subtasks, the task is recast as the transcription of each letter in the word + its context to a phoneme. As similar words will contain similar letter + context combinations, their pronunciation will also be similar. However, this means that the letter string representing the spelling of the word and the phoneme string representing the pronunciation will have to be aligned. An algorithm was developed to do this automatically for the word-pronunciation pairs in the lexicon.

- Induction of a classifier. A decision tree structure (which is nevertheless memory-based or lazy because it remembers all information relevant for classification) is built on the basis of similarities among the letter + context to phoneme mappings. This tree represents both the regularities and the exceptions implicit in the spelling-pronunciation mappings of the existing lexical items.

- For automatic transcription, a new word is split up into letter + context representations, and the phoneme representation corresponding to this input is retrieved in case of an exact match; otherwise, a prediction is made based on similar cases in memory. The decisions for each letter are then combined to produce the final pronunciation representation.

The learning method which was used is a combination of decision tree induction and memory-based learning, for details see Daelemans and van den Bosch (1993); Daelemans and van den Bosch (1996) and van den Bosch and Daelemans (1993).

The method is applicable in the context of our inductive lexica approach because (i) it is corpus-based (it takes as training material the pairs of spellings and associated pronunciations already present in the lexicon), (ii) it is language-independent and reusable (the learning method works regardless of the type of phonetic alphabet, and of the language it is intended for), and (iii) its accuracy is as good as, or often even better, than alternative hand-crafted, knowledge-based approaches.

In the FONILEX project, the following procedure was used:

1. Initial Data. For the initial set of words, the 10,000 most frequent words from CELEX[7] were taken. The pronunciations contained therein represent the Dutch spoken in the Netherlands, similar to, but different from the Dutch spoken in Flanders. These pronunciations were adapted manually to the Flemish variant by a trained phonetician.

2. Bootstrapping. This 10,000 word pronunciation lexicon was then used to train the initial grapheme-to-phoneme converter according to the method described earlier in this section. In the context of our inductive lexicon approach, the lexical predicate *pronunciation* would be associated with this automatically trained converter for the transcription of spellings of new lexical entries. In the FONILEX project, we used the converter to transcribe the next batch of words and send them back for manual correction.

3. The corrected transcriptions were added to the training material of the classifier and used to generate a new version of the converter, which in its turn was used to convert the next batch of words. In the inductive lexicon context, this would correspond to the occasional retraining of the classifier whenever a suitable number of new lexical entries has been added, or, in the case of an incremental learning technique, to immediate accommodation of new training examples.

4. Step (3) was repeated a number of times with increasingly larger sets of words.

In the FONILEX project, this approach added considerably to the flexibility of lexical acquisition. As the system did not make use of hand-made rules, it did not matter that the specifications of the target transcription were continually revised and extended during the project: if the changes were present in the training material, they were picked up automatically by the learning method. The manual adaptation of rule sets would probably cost considerably more time.

We estimated the accuracy of the approach by 10-fold cross-validation on each input dataset. These experiments show a gradual improvement of accuracy with the size of the training data, from 94% to 98% accuracy at phoneme level, corresponding with 80% to 90% at word level. A similar grapheme-to-phoneme converter for Dutch spoken in the Netherlands achieves an accuracy of 99% at phoneme level. The difference is due to the fact that FONILEX uses a richer phonetic transcription which includes archiphonemes.

[7] CELEX is a lexical database for Dutch, English and German, developped at the Max Planck Institute, Nijmegen, and distributed on CD-ROM by LDC.

Table I. Agreement targets within singular NPs

	article	*demonstrative*		*adjective*
M	de	deze	die	-e
F	de	deze	die	-e
N	het	dit	dat	-e/$\emptyset$

Although in this project, the extension of the lexicon was done offline, in different stages of retraining and applying the trained system to new words, it is easy to imagine how the induced system could be associated with the *pronunciation* lexical predicate in a computational lexicon to predict the pronunciation of newly attested words in corpora.

4.2. Case Study 2: Gender Prediction

The previous case study showed how iterative application of machine learning techniques can be instrumental in constructing and extending large pronunciation dictionaries; this case study will focus on a rather surprising use of such phonological information in a very different, syntactic problem domain: gender assignment in Dutch.

Gender is a grammatical category used for the analysis of word classes displaying such contrasts as masculine/feminine/neuter or animate/inanimate (Crystal, 1997). In contrast to a category such as number, most words have (or belong to) only a single gender, which is lexically determined. Genders thus form an important part of lexical structure and can be distinguished syntactically by the agreements nouns take; agreeing elements (or *agreement targets*) are e.g. articles, demonstratives, adjectives or verbs. Under a sufficiently broad definition of agreement[8], control of anaphoric pronouns by their antecedent is covered as well, which is not without importance for Dutch. Historically, Dutch had a three-gender system, distinguishing the traditional categories of *masculine*, *feminine* and *neuter* (Dekeyser, 1980). Currently, the system is shifting towards a two-gender system, where the distinction between masculine and feminine is lost, and only the neuter/non-neuter opposition persists, as can be witnessed from Table I. Remnants of the three gender system, however, are still observed with pronominal anaphora, as Table II shows. Although in the Netherlands the masculine/feminine distinction is only preserved when the antecedents

[8] E.g. "some systematic covariance between a semantic or formal property of one element and a formal property of another" (Steele, 1978).

Table II. Pronominal agreement targets (singular)

	personal	*possessive*	*relative*
M	hij	zijn	die
F	zij	haar	die
N	het	zijn	dat

denote persons (male/female respectively)[9], in Flanders the opposition extends to non-human antecedents as well. Thus, gender identification, as exemplified by Dutch above, is ultimately a syntactic matter. Nevertheless, syntax may not always provide the necessary cues: consider e.g. a Natural Language Understanding system for Dutch, where the pronoun resolution component is faced with a feminine pronoun, while possible antecedents can only be diagnosed as non-neuter on the basis of agreement evidence. Clearly, proper assignment of the relevant items to their respective genders would be an important step towards disambiguation. Appropriate gender information in computational lexica would therefore be an asset.

This problem of *gender assignment* is, of course, well-known and has traditionally been handled by the formulation of gender assignment rules (Corbett, 1991), which draw upon a number of different information sources: in *semantics-based* gender systems, meaning is sufficient to determine gender; here, oppositions such as animate/inanimate, human/non-human, etc. assign words to their respective genders. In predominantly *morphological* systems, word structure (both derivational and/or inflectional) is an important factor in gender assignment. In *phonological* systems, finally, the sound shape of a single wordform reliably indicates gender. The rule-based approach, however, is not without problems. First, although all assignment systems are taken to have at least a semantic core, most languages employ different combinations of assignment criteria, which renders the identification of adequate rules difficult. Second, most assignment rules cover only specific portions of the lexicon, and complete coverage of the lexicon by the whole rule *set* is often not attained. Finally, varying numbers of exceptions exist, and having to list them separately begs the question of lexicon extension. For Dutch, a number of gender assignment rules have been formulated (Haeseryn et al., 1997), but none of them are entirely satisfactory. This has led some researchers to flatly deny the possibility

[9] For non-human antecedents, the masculine forms are used.

of solving the gender assignment problem for Dutch: "The relationship between article and noun in Dutch is, except for a few exceptions, more or less arbitrary: the form the article takes is not systematically determined by any phonological, morphosyntactic, semantic, or conceptual features of the noun." (Deutsch and Wijnen, 1985).

To take up the challenge within the context of Inductive Lexica, we conducted some exploratory experiments with a memory-based learning algorithm. The only assumptions made in constructing the classifier, were that gender and phonological information are available (or can be obtained) for a sizeable part of the noun lexicon. Building on the observation that, cross-linguistically, there is often considerable overlap among various types of assignment criteria, the expectation was that —*pace* Deutsch and Wijnen (1985)—phonological information should make at least some headway in supplying gender information for unknown lemmas.

1. Data was extracted from the CELEX lexical database. Two series (A and B) of three experiments were carried out, one for each relevant gender distinction. Experiments A1–A3 involved 6090 noun lemmas; target classes were `M`(asculine), `F`(eminine), `N`(euter). Experiments B1–B3 involved 7651 noun lemmas; here, target classes were `DE` and `HET`, for non-neuter and neuter resp. For each of the two series, the number of features was gradually increased over the three experiments: the simplest encoding (Experiments A1 and B1) only used onset, nucleus, and coda of the final syllable as features. For Experiments A2 and B2, onset, nucleus and coda of the initial syllable was added. Finally, for Experiments A3 and B3, the stress pattern and number of syllables were included as well, yielding a total of eight features per input example. An overview of the different encodings for the Dutch word *tafel* ('table') is given in Table III. The column labels OF, NF and CF denote the Onset, Nucleus and Coda of the Final syllable, OI, NI and CI stand for Onset, Nucleus and Coda of the Initial syllable, Stress denotes the stress pattern, and Syls the number of syllables.

2. All tests were run with IB1-IG (Daelemans and van den Bosch, 1992), which is the basic memory-based learning algorithm, augmented with *information gain* for feature weighting. Predictions were based on a single nearest neighbor, and the test regime was *leaving-one-out*. Results for the experiments are displayed in Tables IV and V.

   From Table IV it can be seen that the three-way gender distinction remains fairly well predictable, even though agreement marking

Table III. Encodings for 'tafel'

Exp	*Class*	*OF*	*NF*	*CF*	*OI*	*NI*	*CI*	*Stress*	*Syls*
A1	`F`	`f`	`@`	`l`					
A2	`F`	`f`	`@`	`l`	`t`	`a`	`-`		
A3	`F`	`f`	`@`	`l`	`t`	`a`	`-`	`10`	`2`
B1	`DE`	`f`	`@`	`l`					
B2	`DE`	`f`	`@`	`l`	`t`	`a`	`-`		
B3	`DE`	`f`	`@`	`l`	`t`	`a`	`-`	`10`	`2`

Table IV. Success rates for Experiments A1–A3

target	*Exp A1*	*Exp A2*	*Exp A3*
`M`	79.99%	80.26%	81.54%
`F`	89.03%	88.91%	91.97%
`N`	81.58%	81.96%	80.75%
total	83.15%	83.35%	84.30%

for this distinction is disappearing from the language. The overall success rates are situated around 84%, which is significantly better than the claims of "arbitrariness of the Dutch gender system" would lead one to suspect. For the individual target categories, `F` is predicted best, with success scores around 90%, while the other two target categories reach scores of about 80%. Augmenting the number of features increases predictive accuracy.

The results from Table V for the two-way distinction confirm the previous finding that augmenting the number of features yields

Table V. Success rates for Experiments B1–B3

target	*Exp B1*	*Exp B2*	*Exp B3*
`DE`	90.25%	90.49%	91.00%
`HET`	76.17%	77.26%	78.04%
total	86.37%	86.84%	87.65%

Table VI. Confusion matrix for Experiment A3

target	*predicted*		
	M	F	N
M	—	77	339
F	74	—	68
N	311	87	—

higher success rates. Overall success rates are higher than for the previous experiment, with about 87% correct predictions; success rates for the individual target categories are comparable: around 90% for DE and (slightly less than) 80% for HET.

Even though these experiments were largely exploratory in nature, and little effort was made to maximize performance, the results suggest that an Inductive Lexicon approach to this problem is feasible. Whether these results are good enough to warrant practical application remains to be seen, although a glance at the confusion matrix for Experiment A3 (Table VI) might be instructive: Returning to our pronoun resolution problem from the introduction to this section, the main difficulty resided in the masculine/feminine distinction, for which agreement evidence within NPs is lacking. It is precisely for this distinction that the classifier makes relatively few errors.

5. Conclusion

In this paper we introduced a machine learning solution to the problem that computational lexica are never complete, and that to be useful, they should have self-extending properties. Inductive Lexica associate with each lexical predicate in the lexicon a classifier, which makes it possible to compute this predicate for new lexical entries. Inductive Lexica bootstrap on the knowledge implicit in the lexical entries already present in the lexicon (however small it may be), and if present, on information from corpora. We have shown the feasibility of the approach on the basis of two case studies.

We would like to conclude with an alternative idea about the role of computational lexica. Although some Machine Learning techniques are eminently suited for the Inductive Lexicon approach discussed, they also suggest a radically different approach to computational lexicography. The holy grail of computational lexicology has been the concept of reusable, explicit, knowledge-oriented, theory-neutral, polytheoretic computational lexica, useful in a large number of natural language processing tasks. Machine Learning of Natural Language research suggests a radical *performance-oriented* view, in which the idea of generic lexica is abandoned. Different language processing tasks may need different categories and structures to solve the task. These categories may be lexical, grammatical, or a combination thereof. When shifting attention to acquisition, the task dictates the acquisition method, and the acquisition method dictates which information (lexical and contextual) is needed to solve the task. There is therefore a shift from the reusability of the lexical knowledge to the reusability of the acquisition method (e.g. memory-based learning, Daelemans (1995)).

E.g. in word sense disambiguation, both lexical and contextual information is needed for acceptable performance. By providing a learning algorithm with a sufficient amount of examples of word sense disambiguation instances in context, the learning algorithm extracts the necessary information and categories (some of them lexical, some of them contextual, some of them combined) to solve the task. These categorizations need not, and in most cases will not, coincide with categorizations induced for other tasks, such as part of speech tagging. The linguistic view inherent in this approach is therefore task-relativistic: different tasks need different linguistic category systems, including combined lexical-contextual categorizations, and the concept of a unitary, central, reusable lexicon therefore may not be universally applicable.

Acknowledgments

This research was partially funded by a grant to the Tilburg ILK (Induction of Linguistic Knowledge) project from the Dutch National Science Foundation (NWO, geesteswetenschappen), and partially belongs to a concerted research action on computational psycholinguistics of the University of Antwerp.

References

Aha, D. W., D. Kibler, and M. Albert: 1991, 'Instance-based learning algorithms'. *Machine Learning* **7**, 37–66.

Allen, J., S. Hunnicut, and D. H. Klatt: 1987, *From Text to Speech: The MITalk System*. Cambridge, UK: Cambridge University Press.

Barg, P.: 1994, 'Automatic acquisition of PATR theories from observations'. Technical Report 59, Theorie des Lexicons: Arbeiten des Sonderforschungsbereichs 282.

Boguraev, B. and J. Pustejowsky (eds.): 1996, *Corpus Processing for Lexical Acquisition*. Cambridge, MA: MIT Press.

Breiman, L., J. Friedman, R. Ohlsen, and C. Stone: 1984, *Classification and regression trees*. Belmont, CA: Wadsworth International Group.

Briscoe, T., V. de Paiva, and A. Copestake (eds.): 1993, *Inheritance, Defaults, and the Lexicon*. Cambridge, UK: Cambridge University Press.

Carbonell, J. G.: 1990, *Machine learning: paradigms and methods*. Cambridge, MA: MIT Press.

Corbett, G.: 1991, *Gender*, Cambridge Textbooks in Linguistics. Cambridge, UK: Cambridge University Press.

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

Crystal, D.: 1997, *A Dictionary of Linguistics and Phonetics.*, The Language Library. Oxford, UK: Blackwell Publishers Ltd., 4 edition.

Daelemans, W.: 1995, 'Memory-Based Lexical Acquisition and Processing'. In: P. Steffens (ed.): *Machine Translation and the Lexicon*, No. 898 in Springer Lecture Notes in Artificial Intelligence. Springer, pp. 85–98.

Daelemans, W. and A. van den Bosch: 1992, '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.* Twente, The Netherlands, pp. 27–38.

Daelemans, W. and A. van den Bosch: 1993, 'TABTALK: Reusability in Data-oriented grapheme-to-phoneme conversion'. In: *Proceedings of Eurospeech*. Berlin, Germany, pp. 1459–1466.

Daelemans, W. and A. van den Bosch: 1996, 'Language-Independent Data-Oriented Grapheme-to-Phoneme Conversion'. In: J. P. H. van Santen, R. W. Sproat, J. P. Olive, and J. Hirschberg (eds.): *Progress in Speech Synthesis*. New York, NY: Springer Verlag, pp. 77–90.

Daelemans, W., A. Weijters, and A. van den Bosch (eds.): 1997, 'ECML'97 Workshop Notes on Empirical Learning of Natural Language Processing Tasks'. Prague, Czech Republic: Laboratory of Intelligent Systems.

Dekeyser, X.: 1980, 'The diachrony of the gender systems in English and Dutch'. In: J. Fisiak (ed.): *Historical Morphology*, No. 17 in Trends in Linguistics: Studies and Monographs. The Hague, The Netherlands: Mouton, pp. 97–111.

Deutsch, W. and F. Wijnen: 1985, 'The article's noun and the noun's article: explorations into the representation and access of linguistic gender in Dutch'. *Linguistics* **23**, 793–810.

Domenig, M. and P. ten Hacken: 1992, *Word Manager: A system for morphological dictionaries*. Hildesheim, Germany: Olms.

Evans, R. and G. Gazdar: 1996, 'DATR: A Language for Lexical Knowledge Representation.'. *Computational Linguistics* **22**(2), 167–216.

Haeseryn, W., K. Romijn, G. Geerts, J. de Rooij and M.C. van den Toorn: 1997, *Algemene Nederlandse Spraakkunst*. Groningen, The Netherlands: Martinus Nijhoff.

Hunt, E., J. Marin, and P. Stone: 1966, *Experiments in induction*. New York, NY: Academic Press.

Ide, N. and J. Véronis: 1995, 'Knowledge Extraction from Machine-Readable Dictionaries: An Evaluation'. In: P. Steffens (ed.): *Machine Translation and the Lexicon*, No. 898 in Springer Lecture Notes in Artificial Intelligence. Springer, pp. 19–34.

Kolodner, J. D.: 1992, *Case-Based Reasoning*. San Mateo, CA: Morgan Kaufmann.

Langley, P.: 1996, *Elements of Machine Learning*. Los Altos, CA: Morgan Kaufmann.

Miller, G.: 1990, 'Special Issue. WordNet: an on-line lexical database'. *International Journal of Lexicography* **3**(4).

Mitchell, T. M.: 1997, *Machine Learning*. New York: McGraw-Hill Companies, Inc.

Natarajan, B.: 1991, *Machine learning: a theoretical approach*. San Mateo, CA: Morgan Kaufmann.

Quinlan, J. R.: 1993, *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann.

Salzberg, S.: 1990, 'A nearest hyperrectangle learning method". *Machine Learning* **6**, 251–276.

Schütze, H.: 1993, 'Word space'. In: C. S.J.Hanson, J.D.Cowan (ed.): *Advances in Neural Information Processing Systems*, Vol. 5. Morgan Kaufmann, pp. 895–902.

Stanfill, C. and D. Waltz: 1986, 'Toward memory-based reasoning'. *Communications of the ACM* **29**, 1212–1228.

Steele, S.: 1978, 'Word order variation: a typology study'. In: J. H. Greenberg, C. A. Ferguson, and E. A. Moravcsik (eds.): *Universals of Human Language*, Vol. 4. Stanford: Stanford University Press, pp. 585–623.

van den Bosch, A. and W. Daelemans: 1993, 'Data-oriented methods for grapheme-to-phoneme conversion'. In: *Proceedings of the Sixth conference of the European chapter of the ACL*. pp. 45–53.

Weiss, S. and C. Kulikowski: 1991, *Computer systems that learn*. San Mateo, CA: Morgan Kaufmann.

Wettschereck, D., D. W. Aha, and T. Mohri: 1996, 'A review and comparative evaluation of feature weighting methods for lazy learning algorithms'. Technical Report AIC-95-012, Naval Research Laboratory, Navy Center for Applied Research in Artificial Intelligence, Washington, DC.

Wilks, Y., B. Slator, and L. Guthrie: 1996, *Electric Words. Dictionaries, Computers, and Meanings*. Cambridge, MA: MIT Press.

Zernik, U. (ed.): 1991, *Lexical acquisition: exploiting on-line resources to build a lexicon*. Hillsdale, NJ: Lawrence Erlbaum.