Phonotactics in Inductive Logic Programming

John Nerbonne and Stasinos Konstantopoulos

Alfa-informatica, Behavioral and Cognitive Neurosciences,
University of Groningen, 9700 AS Groningen, The Netherlands

Abstract. We examine the results of applying inductive logic programming (ILP) to a relatively simple linguistic task, that of recognizing monosyllables in one language. ILP is suited to linguistic problems given linguists' preference for formulating their theories in discrete rules, and because of ILP's ability to incorporate various background theories. But it turns out to be difficult to rival the performance that statistical theories achieve on the same task. Finally, we note that the theoretically preferred solutions are quite compact, but not optimally comprehensive. Perhaps this should better be interpreted as a reflection of what theoretical linguists prefer, rather than as a reflection of the learning technique.

1 Introduction

This paper reflects on the the results of applying a machine learning technique inspired by logic programming to a relatively simple problem in language description, the notion of syllable. We review INDUCTIVE LOGIC PROGRAMMING (ILP), a machine learning technique that produces logic programs describing a set of data in Section 2, arguing that this technique is particularly well-suited for problems that engage theoretical linguistics—either because the problem could benefit from existing partial solutions or because one wished to exam solutions for potential theoretical interest. Section 3 reviews the syllable learning problem, earlier studies and the set up for the experiments, whose results are presented and discussed in several subsections. A concluding section suggests an interpretation of the results thus far, including a reflection on the comparison of the learned results with linguistic theories.

2 Inductive Logic Programming and Aleph

Inductive Logic Programming (ILP) is a machine learning discipline which develops algorithms to construct Predicate Logic hypotheses that explain a set of empirical data or observations. The central idea is to INDUCE theories from data and theoretical primitives. For example from the background knowledge 'Socrates is human' and a set of observations (training data), e.g. 'Socrates is mortal,' we induce the hypothesis that 'All humans are mortal.' In more formal terms, given a logic program B modeling the background knowledge and a set of ground terms D representing the training data, ILP constructs a logic program H, such that $B \wedge H \models D$.

Sequential-covering See Mitchell, [1], p.276

- Learned-rules ← {}
- Rule ← LEARN-ONE-RULE(Target-attribute,Attributes,Examples)
- **while** Performance(Rule,Examples) > Threshold, **do**
  - Learned-rules ← Learned-rules + Rule
  - Examples ← Examples − {Examples covered by Rule}
  - Rule ← LEARN-ONE-RULE(Target-attribute,Attributes,Examples)
- Return Learned-Rules

Table 1. Sequential covering seeks best rules in a greedy fashion. Once a rule is adopted, examples covered by it are eliminated from further consideration.

A contribution of ILP is the construction of the search space within which hypotheses are sought. The basic algorithm used in the experiments reported on here is the SEQUENTIAL COVER ALGORITHM, which inputs a data on the one hand and background hypotheses and concepts on the other. The algorithm examines each data point in turn and tries to construct a theory which predicts the data (see Table 1). In doing this, it constructs a hypothesis which covers the data point under examination, and which is then evaluated for its ability to generalize to other data points. Once a rule is adopted, the data it explains is removed from further consideration.

Sequential covering is basically simple, but the space of hypotheses in which rules must be sought is daunting. The size of the search space is an exponential function of the number of background concepts (predicates) which may be used to describe instances, and which must be considered not only in all conjunctions, but likewise in combination with all interesting variable bindings. ILP uses Muggleton's [2] structuring of the hypothesis space using an inverse resolution operator mechanizing the notion 'induction.'

We repeatedly pick a positive example from the training data and construct the most specific, non-ground clause that entails it. We do this by repeatedly applying inverse resolution on the example, until we obtain a clause covering the original ground positive example and no other. This maximally specific clause the BOTTOM CLAUSE, and it provides a boundary for the rule search. Rule search then proceeds from the maximally general, empty-bodied clause and the maximally specific bottom clause, looking for a 'good' clause, i.e., one which allows the explanation of a large amount of data. The search traverses the lattice it defined by Plotkin's θ-subsumption [2].

It is interesting to include SYNTACTIC BIAS in constructing candidate rules (hypotheses), e.g., to enforce conformance to a particular theory, and to avoid traversing search paths that are known to be fruitless.

An EVALUATION FUNCTION determines how 'good' a clause under consideration is. The evaluation needs to strike a balance between OVERFITTING, i.e. covering data too tightly and making no essential generalizations and OVERGENERALIZING, i.e. covering data too loosely and accepting too many

negatives. Note that evaluation involves seeking a Prolog proof for each of the data still to be explained, a computationally complex task. Konstantopoulos [3] provides a parallel implementation of Aleph, in order to overcome this.

There is a trivial theory covering the positive instances of a concept while ignoring the negative examples, $C \leftarrow \{\}$, i.e., postulating that everything is a (positive) instance. For this reason, most versions of ILP require that data consist of both positive and negative instances. There are ways of avoiding the need for negative data, but they function in general less well.

Aleph [4] implements the Progol algorithm [5]. It allows for single-predicate learning only, without background theory revision or predicate invention. It incrementally constructs the clauses of a single-predicate hypothesis in a sequential covering fashion (see Table 1), using the bottom clause suggested by Muggleton and allowing the use of syntactic bias.

3 Syllable Structure

Phonotactics identifies what sequences of phonemes constitute a possible word or syllable in a given language. Since words are as sequences of syllables (with some adjustments allowed at boundaries), we reduce the problem to determining what constitutes a syllable, what sequences of phonemes constitute a possible syllable. Languages vary as to which sequences of sounds are used, so /ps/ is an impossible initial segment in English, but fine in Greek. We restrict our attention to monosyllables here to avoid the added complexities of the boundary phenomena noted above and also segmenting (dividing the word into syllables). The training data consists of 5095 monosyllabic words found in the Dutch section of the CELEX Lexical Database [6], with an additional 597 reserved for evaluation. CELEX contains a large number of loan words, which, however, are also found in common parlance. Including these makes the learning task more difficult, but also more realistic.

Tjong Kim Sang and Nerbonne [7] studied this same task, also using a logical technique. They adduce a bigram-inspired baseline which accepts 99.0% of positive data (precision) and rejects 76.8% of the negative data (recall). Where Tjong Kim Sang and Nerbonne restricted representations to concrete phonological segments, and expressed rules without variables, the present paper explores a variety of more abstract phonological representations and makes full use of the Horn-clause logic provided in ILP. This likewise entails changes in the problem set up (see Section 3.1). We compare our results to Tjong Kim Sang and Nerbonne's below (Section 4).

3.1 Representation

We construe the learning task as the learning of valid affixes to a partial syllable, beginning with a vowel. The CELEX data is transformed to instances

of a predicate matching syllable fragments with phonemes that can be affixed at that point.

The positive examples are constructed by breaking the phonetic transcriptions down to three parts: a prevocalic and a postvocalic consonant cluster (consisting of zero or more consonants) and a vowel or diphthong. The consonant clusters are treated as 'affixes' to the vowel, so that syllables are constructed by repeatedly affixing consonants, if the CONTEXT (the vowel and the pre- or post-vocalic material that has been already affixed) allows it. So, for example, from the word /ma:kt/ (*maakt* 'makes') the following positives would be generated:

```
prefix( m, [], [a,:] ).  suffix( k, [],[:,a] ).
prefix( ^, [m], [a,:] ). suffix( t, [k], [:,a] ).
                         suffix( ^, [tk], [:,a] ).
```

For example, the first two `suffix` rules read as follows: '/k/ can be suffixed to the /a:/ nucleus' and '/t/ can be suffixed to an /a:k/ syllable fragment'.

We reverse context lists in suffix rules so that the processes are symmetrical and use the same background predicates (manipulating lists).

The caret, `^`, is used to mark the beginning and end of a word. We need to explicitly license affix termination on the one hand in order to avoid errors. In Dutch, for example, a monosyllable with a short vowel has to be closed, which means that the null suffix is not valid. The end-of-word mark allows this to be expressed as a theory that does not have the following clause: `suffix( ^, [], [V] )`. We also prefer to require the syllable boundary to be learned in order to avoid over-informing the process, in effect assuming that all partial sub-affixes of a valid affix are necessarily valid as well. Tjong Kim Sang and Nerbonne [7] in fact made this assumption, and obtained good results, but they also provides the learner with too much information.

The positives are all the prefixes and suffixes that occur in context, so that all the monosyllables in the training data can be constructed: 11,067 and 10,969 instances of 1,428 and 1,653 unique examples, respectively.

The negative data is randomly generated words that match the template $\mathcal{C}_3\mathcal{V}\mathcal{C}_5$ and do not appear as positives. The random generator balances the number of examples at each affix length to avoid having large numbers of long, uninteresting sequences overwhelm the shorter, more interesting ones. Since the randomly generated negatives must also contain false negatives, we cannot expect even a good theory to fit perfectly. In order to avoid overfitting, the learning algorithm was set to only require an accuracy of 85% over the training data. The negative data is also split into evaluation and training data, and the negative examples are derived from the training negative data by an algorithm detailed by Konstantopoulos [8,9], omitted here.

3.2 Alternative Backgrounds

Since we are viewing the problem as the task of identifying the consonants that may be prefixed or suffixed to a partially constructed monosyllable, the

clauses of the target predicate must have a means to refer to various subsets of $\mathcal{C}$ and $\mathcal{V}$. This is achieved by specifying a (possibly hierarchical,) linguistically motivated description of $\mathcal{C}$ and $\mathcal{V}$. Each class described can be referred to as a feature-value pair, for example LAB+ to denote the set of the labials or VOIC+ for the set of voiced consonants. Intersections of these basic sets are allowed: the feature-value vector [VOIC+,LAB+] refers the voiced labials.

BACKGROUND KNOWLEDGE plays a decisive role in the quality of the constructed theory by providing a descriptive vocabulary in which to formulate hypotheses. This is operationalized as relations between segments and feature values, e.g. `labial(m,+)` or `voiced(m,+)`. Feature-value vectors are expressed as conjunctions, e.g., `labial(C,+)` $\wedge$ `voiced(C,+)`. The background knowledge also contains the `head/2` and `rest/2` list access predicates. These were preferred over direct list access with the `nth/3` predicate, as bias towards rules with more local context dependencies.

The theories described in Sections 3.3, 3.4 and 3.5 below, are based on background knowledge that encodes increasingly more information about Dutch phonology as well as Dutch phonotactics: for the experiment in 3.3 the learner has access to the way the various symbols are arranged in the International Phonetic Alphabet (IPA), a standard set of predicates designed to facilitate the description of arbitrary languages, whereas for the experiment in 3.4 a classification that is sensitive to features important in Dutch phonology was chosen. Finally, in Section 3.5 a scalar (as opposed to binary) sonority feature is implemented, which has been proposed with the explicit purpose of solving the problem of syllable structure [10].

The quantitative evaluation for each of the three experiments was carried out using the same 597 words and the same part of the randomly generated negative data that were reserved for this purpose.

3.3 The IPA segment space

For the first experiment the background knowledge reflects the system of the International Phonetic Alphabet (IPA, 1993 version): the phonological inventory consists of two disjoint spaces, one of consonants and one of vowels, with three and four orthogonal dimensions of distinction, respectively.

Consonants vary in PLACE, MANNER OF ARTICULATION, and VOICING. Manner can be PLOSIVE, NASAL, LATERAL APPROXIMANT, TRILL, FRICATIVE or APPROXIMANT. Place can be BILABIAL, ALVEOLAR, VELAR, LABIODENTAL, POSTALVEOLAR or PALATAL. Voicing can be present or absent. Vowels have four dimensions: PLACE (FRONT, CENTRE, BACK); HEIGHT (OPEN, MID-OPEN, MID-CLOSED, CLOSED); length and roundedness. The end-of-word mark has no phonological features whatsoever and it does not belong to any of the classes of either $\mathcal{C}$ or $\mathcal{V}$. This schema was implemented as one background predicate per dimension relating each segment with its value along that dimension: `manner(plosive, p).` etc.

We evaluated using the LAPLACE FUNCTION $\frac{P+1}{P+N+2}$, where P and N is the number of positive and negative examples covered, respectively.

The resulting hypothesis consisted of 199 prefix and 147 suffix clauses and achieved a recall rate of 99.3% with 89.4% precision. All the false negatives were rejected because of prevocalic material, typically from loan words could not be licensed. The /ʤ/ segment found in 'jeep' and 'junk', for example, was not permitted and so these words were rejected.

The most generic rules found were:

```
prefix(A,B,C) :- A= '^'.     suffix(A,B,C) :-  A= '^'.
prefix(A,[],C).              suffix(A,[],C).
```

meaning that (a) the inner-most consonant can be anything, and (b) all sub-prefixes (-suffixes) of a valid prefix (suffix) are also valid. We also noted pairs of rules which might have collapsed given a richer background vocabulary.

3.4 Booij's Feature Classes

The second experiment made a richer (but more language-specific) background knowledge available to the inductive algorithm, by implementing the feature hierarchy suggested by Booij [11] and shown in figure 1.

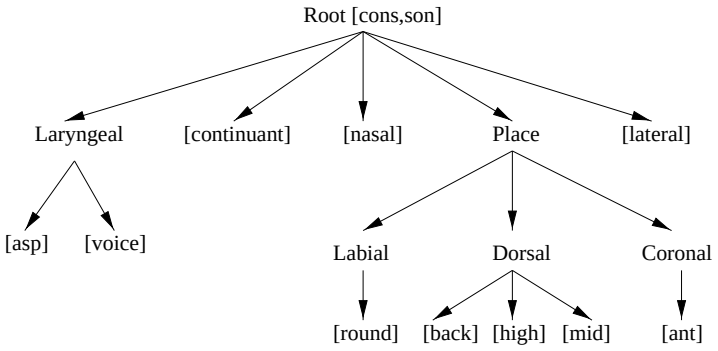

Fig. 1. Booij's [11] feature geometry for Dutch (simplified).

The most generic features are the MAJOR CLASS FEATURES (CONSONANT and SONORANT) on the root, which divide the space into vowels [CONS-,SON+], obstruents [CONS+,SON-] and sonorant consonants [CONS+,SON+]. Since all vowels are sonorous, [CONS-,SON-] is invalid.

Most of the features bundled together under two FEATURE CLASSES, LARYNGEAL and PLACE. Booij postulates these classes because they group features that behave the same in Dutch phonology. Laryngeal features mark the voiced-voiceless distinction, while the ASPIRATION feature distinguishes only

/h/ from the rest. Some derived or redundant features such as Glide, Approximant and Liquid are defined, but not shown in Figure 1. The vowels do not include the schwa, which is set apart and only specified as Schwa+.

Again using Laplace evaluation the theory adduced consisted of 13 prefix and 93 suffix rules, accepting 94.2% of the test positives and under 7.4% of the test negatives. Among the rejected positives are loan words ('jeep' and 'junk' once again), but also all the words starting with perfectly Dutch /s/ - obstruent - liquid clusters. The prefix rule with the widest coverage is:

```
prefix(A,B,C) :-  head(C,D), sonorant(D,plu), rest(B,[]).
```

≈ 'prefix anything before a single consonant before a non-schwa nucleus.'

The suffix rules rejected only 3 positives, 'branche', 'dumps' and 'krimpst' (the first two are loan words) which failed to suffix /ʃ/ or /s/. Some achieve wide coverage (although never as wide as the prefix rules), but others make reference to individual phonemes:

```
suffix(A,B,C) :-  rest(C,D), head(D,E), rest(B,[]), A=t.
```

or, 'suffix a /t/ after exactly one consonant, if the nucleus is a long vowel'.

The end-of-word marking rules (see Section 3.2), are interesting because open, short monosyllables are very rare in Dutch (there are four in CELEX). This suggests that these are exceptions to the general rule disallowing open, short monosyllables. The learner instead adduced 29 rules for suffixing `^`, the most general of which is:

```
suffix(A,B,C) :-  head(B,t), larynx(t,E), rest(B,F),
                    head(F,G), larynx(G,E), A= '^'.
```

or 'suffix an end-of-word mark after at least two consonants, if the outer-most one is a /t/ and has the same values for all the features in the Laryngeal feature class as the consonant immediately preceding it'.

This experiment also exposed ILP's computational complexity. A more flexible background vocabulary contains more predicates and thus provides for more interesting hypotheses, but this results in longer bottom clauses and a larger search space.

3.5 Sonority Scale

The most popular linguistic theory on the syllable is that a syllable is defined by a sequence of segments rising monotonically in sonority to a vowel and then falling monotonically to the end of the syllable [12,13], even though exceptions to this general principle are known and discussed [14,15]. We implement and test van der Hulst's [10] syllabic model here. We do not present the results of a machine learning experiment, but rather a hand-crafted theory, which we use for comparison. The Dutch syllable is analysed as having three prevocalic

phoneme	obstruents	m	n	l	r	glides	vowels
sonority	1	2	2.25	2.5	2.75	3	4

Table 2. The Sonority Scale

and 5 postvocalic positions (some of which may be empty), and constraints are placed on the consonants that can occupy each.

The most prominent constraint stipulates a high-to-low SONORITY progression from the nucleus outwards. Each phoneme is assigned a sonority value (as in table 2) and syllables are then built from the nucleus outwards, by stacking segments of decreasing sonority.

Part of the sonority scale is based on language-independent characteristics of the segments. For example, vowels are always more sonorant than consonants, and obstruents are the least sonorant of the consonants. The scale shown here has been further refined to account for particular idiosyncrasies of Dutch. For example, in the original (language-independent) version nasals and liquids are not distinguished. In the final theory, however, there are four distinctions (see Table 2). The justification for this refinement is to explain, e.g., why /karl/ is acceptable while /kalr/ is not. We emphasize that the scale is not only language specific, but also "problem"-specific: it was developed to solve the very problem under investigation. It represents a best account of the phenomenon currently available.

In addition to the high-to-low sonority level progression from the nucleus outwards, there are both FILTERS and explicit licensing rules. Filters are restrictions referring to sonority (e.g. 'the sonority of the three left-most positions must be smaller than 4') or other phonological features (e.g. the 'no voiced obstruents after the vowel' filter, p. 92) and are applicable in conjunction with the sonority rule. Licensing rules are typically restricted in scope and take precedence over the sonority-related constraints mentioned so far. The left-most position, for example, may be /s/ or empty, regardless of the contents of the rest of the prevocalic material. We left some constraints out of the implementation that were too long when translated from their fixed-position perspective to the affix-licensing one used here, or that appeared to be fine tuning the theory to individual consonant clusters.

The sonority progression rule together with the most widely applicable filters and rules yielded impressive compression rates matched with results between those of the two previous experiments: 93.1% recall, 83.2% precision.

4 Results and Discussion

As can be seen in Table 3, the ILP-constructed rules compare favorably (in both performance and hypothesis compactness) with those constructed by the deductive approach employed in [7]. The most popular linguistic theory

	Bigrams	Tjong (no ling.)	Tjong C & F	IPA	Booij's Features	Sonority
Recall	99.0%	99.1%	99.0%	99.3%	94.2%	93.1%
Precision	76.8%	74.8%	91.9%	79.8%	92.6%	83.2%
|Clauses|		577 + 577	674 + 456	145 + 36	13 + 93	3+8

Table 3. Results. The first column reports the bigram baseline, the second the linguistically unbiased result in Tjong Kim Sang and Nerbonne (2000), and the third Tjong Kim Sang and Nerbonne's results using a biased inspired by Cairns and Feinstein's theory of the syllable [16]. The last three columns report on the experiments in Sections 3.3–3.5. Recall is the percentage of correct data accepted, precision the percentage of incorrect data reject, and |Clauses| reports on the number of prefix and suffix clauses in the theory learned.

on the syllable (Table 3, right column) is included as a point of comparison. It is notable for its compression, but less so for its coverage (recalling, however, that very specific details were omitted). Tjong Kim Sang and Nerbonne [7] note that experiments with (stochastic) Hidden Markov Models performed slightly better than the best discrete models here, achieving 98.9% recall and 92.9% precision.

The experiments demonstrate that ILP can identify sequential patterns, and that its ability to accommodate different background theories leads to an interesting variety of results depending on background theory. ILP does not deal well with the fact that natural languages have irregular and semiregular patterns, many of which are also infrequent, however, and the rather costly learning algorithm also makes it a poor candidate for complex tasks. See, however, Džeroski et al [17] for further ideas on applying ILP to language.

It is likewise noteworthy that *all* techniques adducing rules—including the stocahstic techniques mentioned above—seem limited to about 99% precision and 93% recall, and moreover that they rise to this level of accuracy only at the cost of including many very specific rules. This suggests that sheer memory plays a very significant role in learning even this very simple sort of structure, a theme emphasized repeatedly by Daelemans [18] and his colleagues. In this same vein, it is noteworthy that leading theoretical models do not appear to fare better. The need to accommodate very substantial amounts of exception penetrates even into carefully constructed models.

Applying machine learning to language can inform theories of language acquisition by showing what information is implicit in the data and also by operationalizing the "innateness" always emphasized in discussions of child language acquisition. Although computational modeling is sometimes dubbed "simulation," our experiments have not attempted to model children's language acquisition in any specific way, e.g., by enforcing an incremental scheme, by limiting training data to a small set of initial experiences, or limiting working memory. We note, nonetheless, the increasing interest in

the INPUT HYPOTHESIS, i.e. that the information in children's experience is rich enough to inform learning in more detail than "innateness" advocates might expect [19].

References

1. Mitchell, T.M.: Machine Learning. McGraw-Hill, Singapore (1997)
2. Muggleton, S., De Raedt, L.: Inductive Logic Programming: Theory and methods. Journal of Logic Programming **19** (1994) 629–679 Updated technical report CW 178, May 1993, Computing Science, K.U. Leuven.
3. Konstantopoulos, S.T.: A data-parallel version of Aleph. In Camacho, R., Srinivasan, A., eds.: Proc. of the Workshop on Parallel and Distributed Computing for Machine Learning Workshop, ECML/PKDD 2003. (2003)
4. Srinivasan, A.: The Aleph Manual, `www.comlab.ox.ac.uk/oucl/research/areas/machlearn/Aleph/` (2002)
5. Muggleton, S.: Inverse entailment and Progol. New Generation Computing **13** (1995) 245–286
6. Burnage, G.: CELEX: A guide for users. Documentation, Centre for Lexical Information, University of Nijmegen, Nijmegen (1990)
7. Tjong Kim Sang, E.F., Nerbonne, J.: Learning the logic of simple phonotactics. In Cussens, J., Džeroski, S., eds.: Learning Language in Logic. Volume 1925 of Lecture Notes in Artificial Intelligence., Springer Verlag (2000)
8. Konstantopoulos, S.T.: Learning phonotactics using ILP. Special Issue of WEB-SLS On-line Journal: Student Language Section of ESSLLI-01 (2002)
9. Konstantopoulos, S.: Using Inductive Logic Programming to Learn Local Linguistic Structures. PhD thesis, Rijksuniversiteit Groningen (2003)
10. van der Hulst, H.: Syllable Structure and Stress in Dutch. PhD thesis, Rijksuniversiteit Leiden (1984)
11. Booij, G.: The Phonology of Dutch. The Phonology of the World's Languages. Clarendon Press, Oxford (1995)
12. Kiparsky, P.: Metrical structure assignment is cyclical. Linguistic Inquiry **10** (1979) 421–441
13. Harris, J.: English Sound Structure. Blackwell, Oxford (1994)
14. Fudge, E.: Syllables. Journal of Linguistics **5** (1969) 253–286
15. Bolognesi, R.: The Phonology of Campidanian Sardinian: A Unitary Account of a Self-Organizing Structure. PhD thesis, University of Amsterdam (1998)
16. Cairns, C.E., Feinstein, M.H.: Markedness and the theory of syllable structure. Linguistic Inquiry **13** (1982)
17. Džeroski, S., Cussens, J., Manandhar, S.: An introduction to inductive logic programming and learning language in logic. In Cussens, J., Džeroski, S., eds.: Learning Language in Logic. Springer, Berlin (2000) 3–35 Springer Lecture Notes in Artificial Intelligence, Vol. 1925.
18. Daelemans, W., van den Bosch, A., Zavrel, J.: Forgetting exceptions is harmful in language learning. Machine Learning **34** (1999) 11–43 Special Issue on natural language learning.
19. Saffran, J.R., Aslin, R.N., Newport, E.L.: Statistical language learning by 8-month olds. Science **274** (1996) 1926–1928