Learning Simple Phonotactics

Erik F. Tjong Kim Sang
Center for Dutch Language and Speech
University of Antwerp
erikt@uia.ac.be

John Nerbonne
Alfa-informatica, BCN
University of Groningen
nerbonne@let.rug.nl

Abstract

The present paper compares stochastic learning (Hidden Markov Models), symbolic learning (Inductive Logic Programming), and connectionist learning (Simple Recurrent Networks using backpropagation) on a single, linguistically fairly simple task, that of learning enough phonotactics to distinguish words from non-words for a simplified set of Dutch, the monosyllables. The methods are all tested using 10% reserved data as well as a comparable number of randomly generated strings. Orthographic and phonetic representations are compared. The results indicate that while stochastic and symbolic methods have little difficulty with the task, connectionist methods do.

1 Introduction

This paper describes a study of the application of various learning methods for recognizing the structure of monosyllabic words. The learning methods we compare are taken from three paradigms: stochastic learning (Hidden Markov Models), symbolic learning (Inductive Logic Programming), and connectionist learning (Simple Recurrent Networks using backpropagation). In each case we shall use the methods to build an acceptor for the monosyllables from positive data only, and we shall test it on held-back test data as well as randomly generated (negative data). We systematically compare results based on orthographic and phonetically encoded data. The data comes from Dutch but the results are expected to be similar for other related languages.

This study focuses on three questions. First, we ask which methods are able to learn the structure of monosyllabic words. Second, we seek to learn the influence of data representation on the performance of the learning algorithms and the models they produce. Third, we would like to see which the learning processes are able to create better models when equipped with basic initial knowledge, so-called innate knowledge.

2 Theoretical background

This section presents some theoretical background for the main problem of this paper, learning simple phonotactics.

2.1 Problem description

The phonotactic structure of a language determines which sequences of basic sounds are allowed in the language. Certain languages, such as Polish, allow words to start with *ml* but others, such as English and Dutch, do not. Phonotactics are directly reflected in the phonetic transcriptions of words, and indirectly in the orthography, i.e., the writing system. Different languages usually have different orthographies.

Some aspects of phonotactics, such as preference for consonant vowel sequences, are shared by almost all languages, but phonotactic structure varies from one language to another. No universal phonotactics exists. There are two possibilities for entering language-dependent phonotactics into a computer program. The first is to examine (by eye or ear) the language and create a list of rules reflecting phonotactic structure. This is labour-intensive and repetitive when many languages are involved. The second possibility is to have the machine *learn* the phonotactics by providing it with language data. People manage to learn phonotactic rules which

restrict phoneme sequences so it might be possible to construct an algorithm that can do the same. If we are able develop a model capable of learning phonotactics, we can use it to acquire the phonotactic structure of many languages.

2.2 Data representation

The input data for our learning methods can be represented in two ways. The first one is called the orthographic representation. Here words are represented by the way they are written down, for example: "the sun is shining". The second way of representing the words is phonetic. If we use the phonetic representation then the sentence "the sun is shining" is represented as [ðə sʌn Iz ʃeInIŋ].

We do not know which (if either) of the two representations will enable the learning process to generate the best word models. Acceptance decisions of words by humans may be based on the way the words are written but they may also be based on the pronounceability of the words. We are interested in finding out which representation way is most suitable for the learning methods. Therefore we perform two variants of some experiments: one with data in the orthographic representation and one with the same data in the phonetic representation.

2.3 Innate knowledge

A recurrent issue in modeling language acquisition is the amount of innate knowledge available or required. Linguists have emphasized that important aspects of language learning require some innate knowledge [Pin94]. Debates in the language acquisition literature have led to a general acceptance of the assumption that children use innate linguistic knowledge when they acquire their native language. Steven Finch's PhD thesis [Fin93] describes approaches to languages acquisition which assume no innate knowledge. Since we are interested in the influence of innate knowledge on language acquisition, we perform experiments with and without assumptions of (specific) linguistic knowledge. In the case of connectionist methods, this required some creativity, but we believe that a reasonable operationalization was found.

2.4 Positive and negative data

A further perennial question is whether negative information needs to be used—e.g., the information that 'mlod' is not a Dutch monosyllable. Early research in computational learning theory showed the need for negative learning if grammars are to characterize perfectly [Gol67]. Research in child language acquisition has had difficulties with finding negative language input from parents in conversations with young children, and has noted that children attend to it poorly. Here we have a problem: according to computational learning theory children need negative information for learning (perfectly), while children do not seem to receive this information even though they manage to learn natural languages.

We shall approach the acquisition of models for monosyllabic words from the research perspective of child language acquisition. We shall supply our learning methods with positive information only. In some learning experiments it might be theoretically necessary that negative examples are supplied in order to obtain a good result. We shall assume that in those learning experiments the negative information is supplied implicitly. One source of implicit information is innate knowledge. Another is the application of the closed world assumption which states that non-present information is false (but this would not meet Gold's objections).

3 Experiments and Results

This section describes the setup of the experiments. It also presents the results of the experiments with the three learning methods.

3.1 General information

Our goal is to perform four experiments with each learning method. Two experiments with data in orthographic representation and two with data in phonetic representation. For each representation we perform one experiment with random initialization and one in which the learning algorithm is supplied with some basic linguistic knowledge. We use ten-fold cross-validation which means that we divide our positive data in ten parts and use nine parts for training and one part for testing. Each part is used as testing part once. Additionally we use a constant test data set with non-words.

The data sets were derived from the CELEX cd-rom [BPvR93]. From the Dutch Phonology Wordforms directory (DPW) we extracted all monosyllabic word representation pairs. The

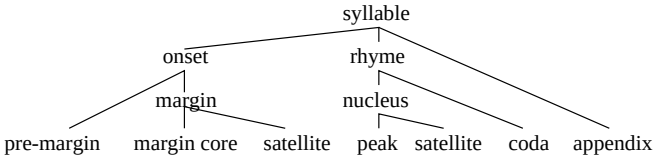

Figure 1: The syllable model of Cairns and Feinstein

first element of each pair was the orthographic representation of the word (field Head) and the second the phonetic representation of the word (field PhonolCPA). All characters in the orthographic representation of words were changed to lower case. We obtained a list of 6190 unique orthographic strings and 5695 unique phonetic strings. This was our positive data. The negative data sets consist of 1000 unique strings which do not appear in the positive data. This data was created by randomly generating strings with the same length distribution and character frequencies as the positive data.

In order to determine the complexity of the data we have subjected it to a baseline experiment. In this experiment we applied the simplest learning algorithm we could think of to this data. This algorithm accepted all and only strings that consist of character pairs (bigrams) which appear in the training data. This algorithm accepted $99.29 \pm 0.32\%$ of the positive orthographic data and rejected $55.72 \pm 0.90\%$ of the negative orthographic data. We use the notation $[99.29 \pm 0.32 | 55.72 \pm 0.90]$ for this performance. For the phonetic data these scores were $[99.00 \pm 0.46 | 76.83 \pm 0.49]$. This algorithm was good in accepting positive data but performed less well in rejecting negative data. The main task of the three learning algorithms is to improve the performance on negative data.

In the experiments with linguistic information we use the Cairns and Feinstein model as basic knowledge [CF82; Gil92]. This is a hierarchical syllable model consisting of a tree which contains seven leaves (see figure 1). Each leaf can either be empty or contain one phoneme. The leaves are restricted to a class of phonemes: the peak can only contain vowels and the other leaves may only contain consonants. The exact phonemes that are allowed in a leaf are language dependent. In the syllable model there are vertical lines between nodes and daughter nodes which are main constituents. A slanting line between two nodes indicates that the daughter node is dependent on the sister node that is a main constituent. A dependent constituent can only be filled if its main constituent is filled. For example, the margin satellite can only contain a phoneme if the margin core contains a phoneme.

In our experiments with initial knowledge we supply the learning algorithms with the syllable structure presented in figure 1. Two extra constraints are provided to the algorithms: the fact that the peak can only contain vowels while the other leaves are restricted to consonants. Furthermore the division of the phonemes in vowels and consonants is made explicit for the learning algorithms. Their task is to restrict the phonemes in each leaf to those phonemes that are possible in the language described by the learning examples. By doing this they convert the general Cairns and Feinstein model to a language-specific syllable model.

3.2 Hidden Markov Models

A Hidden Markov Model is a finite state automaton in which the state transitions and the string productions are probabilistic. Three important algorithms are associated with HMMs [Rab90; vA92]. The first one is the forward-backward algorithm. This algorithm computes the probability of a HMM production process that resulted in a specific string. The second important algorithm is the Viterbi algorithm. Given an HMM and an output string, it computes the sequence of the HMM production states which is most likely to underlie the output string. Finally there is the Baum-Welch algorithm which is able to compute the statistical HMM parameters that are required to make the HMM return high string scores for some collection of strings. This algorithm is the learning algorithm for HMMs, and it works by iteratively adjusting the transition and output probabilities in a way that makes the training corpus more probable. It belongs to the class of algorithms known as estimation maximization [Rab90], which are regarded as cognitively implausible because of the required iteration.

We train HMMs by presenting them a list of valid Dutch monosyllabic words and applying

the Baum-Welch algorithm until the scores they assign to the training words become stable. Here we define a stable score as a score in some phase of the training algorithm that do not differ more than 1% from the same score in the previous phase of the training algorithm.

When the HMM becomes stable we test it by applying it to the test data. We define that test strings that receive a score which is lower than the minimal score for a training data item are rejected by the HMM. We assume that all other strings are accepted by the HMM. Our HMMs process character bigrams rather than character unigrams. This means that they interpret words as sequences of two characters such as *splash=sp-pl-la-as-sh*. By working this way the HMMs are forced to take the context of a character in account.

In the experiments without initial knowledge, the HMMs were initialized with random weights and the production probabilities were set at random values. The HMMs contained seven states and they were allowed to change every weight and every production probability during the training phase. They performed well, for orthographic data the scores were [99.10±0.38|82.19±3.27] and for phonetic data [98.68±0.54|91.60±1.06]. The scores for the positive data were not significantly different from the baseline scores but the negative data figures were significantly better than the baseline scores.

The model of Cairns and Feinstein was inserted in the HMMs by disabling certain state connections and production possibilities. The remaining model parameters were initialized with random values. The performance of these HMMs did not change a lot: [99.16±0.34|77.35±0.77] for orthographic data and [98.93±0.52|92.88±0.27] for phonetic data. The figures were not significantly different from the previous figures. For orthographic data the rejection figure was slightly lower, possibly because the phonetic initialization model did not fit very well for this representation type. The HMMs with initial knowledge trained faster: they needed on average 30.3±12.3 rounds for orthographic data (was 93.3±36.3 rounds) and 42.60± 24.4 rounds for phonetic data (was 57.8±24.0, this difference was not significant).

3.3 Inductive Logic Programming

Inductive Logic Programming (ILP) is a logic programming approach to machine learning ([Mug92]). The term induction in the name is a reasoning technique which can be seen as the reverse of deduction. In ILP theory one makes a distinction between three types of knowledge namely background knowledge, observations and hypotheses ([Mug92]). Background knowledge is that knowledge that a learner already has about the domain of the learning problem. Observations are the input patterns for the learning method with their classifications. The hypotheses contain the model of the domain that ILP should build.

Our ILP models build words by adding prefix characters and suffix characters to a nucleus. They assume that every correct word can be broken down to a correct nucleus by stripping of prefix and suffix characters and they assume that every intermediate word in this break-down process is correct as well. The models are built by deriving three type of hypotheses from training data:

> BASIC WORD HYPOTHESIS
> A basic word hypothesis BWH($w_1...w_n$, s_i) defines that the string $w_1...w_n$ can be produced in state s_i.
> SUFFIX HYPOTHESIS
> A suffix hypothesis SH(w_{n-1},w_n,s_i) defines that when a string $w_1...w_{n-1}$ can be produced in a predecessor state of state s_i then the string $w_1...w_{n-1}w_n$ can be produced in state s_i.
> PREFIX HYPOTHESIS
> A prefix hypothesis PH(w_1,w_2,s_i) defines that when a string $w_2...w_n$ can be produced in a predecessor state of state s_i then the string $w_1w_2...w_n$ can be produced in state s_i.

For example, if both *clan* and *clans* are in the training data then SH(s_i,*n*,*s*) might be derived. The state s_i is a processing state which is similar to an HMM state. The states are used to constrain the models that use initial linguistic knowledge.

In the experiments without initial knowledge the number of states was limited to one. This state was connected to itself and it could produce every character. The performance of these ILP models was almost the same as for the baseline model: [99.33±0.33|55.71±0.90] for orthographic data and [99.07±0.43|74.80±0.16] for phonetic data. The only difference with the baseline figures lies in the rejection of negative phonetic data: ILP performs worse than the baseline method (76.83±0.49).

Initial knowledge was inserted to the models by constraining the links between the states and restricting the characters that could be produced in a state. The models contained nine states: seven representing the Cairns and Feinstein model and two extra for producing strings without vowels. States that correspond with basic word hypotheses were allowed to produce strings instead of characters. The performance of the models improved compared with the previous experiments: [98.56±0.26|84.86±0.27] for orthographic data and [99.03±0.48|91.93±0.33] for phonetic data. The acceptance score for orthographic data was worse than the baseline in return the rejection score was better than the baseline score. For the phonetic data there was no difference for the acceptance score but the rejection score was better than the baseline score. The two rejection scores also improved the non-initialized rejection scores.

3.4 Simple Recurrent Networks

In 1990 Jeffrey Elman introduced Simple Recurrent Networks (SRNs) as an extension of standard backpropagation networks [Elm90]. The basic version of such a network contains three layers of cells: an input layer, an output layer and a hidden layer. It is able to process time-dependent patterns because of the extra backward connections from the hidden layer to the input layer.

In experiments described in [CSSM89], [SSCM91] and [Cle93], Axel Cleeremans, David Servan-Schreiber and James McClelland trained a network to recognize strings which were generated using a small grammar that was originally used by [Reb76]. They trained an SRN to predict the next character in a sequence of 60,000 strings which were randomly generated by the grammar. This prediction task is non-deterministic, and the size of the network was too small to memorize the complete sequence, so some error was to be expected. The learning was deemed successful when it could distinguish which characters are valid successors. Cleeremans, Servan-Schreiber and McClelland tested their network using both positive and negative data and obtained effectively perfect results. We replicated their results: SRNs learn the Reber grammar perfectly.

In an earlier study we have applied SRNs to a related data set of Dutch monosyllabic words in orthographic representation [Tjo98]. The results were discouraging. The SRN without initial knowledge accepted all positive test data but it was only able to reject 8.3% of the negative data. Linguistic knowledge was added to the experiments by supply data in order of increasing phonotactic complexity. The resulting SRNs did not perform better than the previous ones. They accepted all positive test data but rejected only 4.8% of the negative data (baseline scores for this data set were: [99.2|60.2]). The experiments with our main data set show similar figures: trained SRNs accept all positive test data and a large part of the negative data.

The complexity of our data sets is larger than the complexity of the data sets used by Cleeremans et.al. In the valid strings processed by their networks, characters could only be followed by two characters. In Dutch monosyllabic words, this number is larger. We tested the performance of SRNs on increasingly complex data and discovered a relation between data complexity and performance [Tjo95; Tjo98]. When characters in valid strings can have two successors, a trained SRN accepts all positive data and rejects all negative data [100.0|100.0]. However the performance degrades for three successors [100.0|92.2] and becomes even worse for four successors [100.0|80.0]. Many characters in words in Dutch phonotactic data can have more than four successors. SRN seem to be unable to handle such complex data and therefore we conclude that SRNs are unfit for processing our data set.

4 Concluding remarks

The problem of phonotactics as it has been tackled here is basically the problem of sequencing. The results show that both stochastic and symbolic machine learning techniques which perform credibly, if not perfectly on this task. They also indicate that further advancements in the application of neural networks to sequencing are needed, and this is indeed the subject of ongoing work. The results further indicate that linguistic knowledge is a useful starting point for learning algorithms since this turns up in speed and accuracy. Finally, results show that learning from written symbols is more difficult (or less easy) than learning from phonetic representation.

There are numerous natural extensions and refinements of the work presented here, not only seeking improved performance in these techniques, extending the study to other learning techniques, but also refining the task so that it more closely resembles the human task of language learning. This would involve incorporating frequency information, noisy input, and coding input for phonetic properties, and naturally

orthographic data	without initial knowledge		with initial knowledge	
learning algorithm	% accepted positive data	% rejected negative data	% accepted positive data	% rejected negative data
HMM	99.10±0.38	82.19±3.27	99.16±0.34	77.35±0.77
ILP	99.33±0.33	55.71±0.90	98.56±0.26	84.86±0.27
baseline	99.29±0.32	55.72±0.90		

phonetic data	without initial knowledge		with initial knowledge	
learning algorithm	% accepted positive data	% rejected negative data	% accepted positive data	% rejected negative data
HMM	98.68±0.54	91.60±1.06	98.93±0.52	92.88±0.27
ILP	99.07±0.43	74.80±0.16	99.03±0.48	91.93±0.33
baseline	99.00±0.46	76.83±0.49		

Figure 2: A summary of the performance of the stochastic and the symbolic methods on learning the phonotactic structure of Dutch monosyllabic words. HMMs perform better for phonetic data but initial knowledge does not help them. ILP performs better for phonetic data and initial knowledge aids its performance. In the knowledge-aided experiments both HMMs and ILP outperform the baseline method with respect to negative data rejection. SRNs tend to accept nearly all positive data and most of the negative data (see [Tjo95] and [Tjo98]).

extending the task to multisyllable words and to related tasks in phonological learning.

References

[BPvR93] R.H. Baayen, R. Piepenbrock, and H. van Rijn. *The Celex Lexical Database (CD-ROM).* Linguistic Data Consortium, University of Pennsylvania, Philadelphia, PA, 1993.

[CF82] Charles E. Cairns and Mark H. Feinstein. Markedness and the theory of syllable structure. *Linguistic Inquiry*, 13(2), 1982.

[Cle93] Axel Cleeremans. *Mechanisms of Implicit Learning.* The MIT Press, 1993. ISBN 0-262-03205-8.

[CSSM89] A. Cleeremans, D. Servan-Schreiber, and J.L. McClelland. Finite state automata and simple recurrent networks. *Neural Computation*, pages 372–381, 1989.

[Elm90] Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14, 1990.

[Fin93] Steven P. Finch. *Finding Structure in Language.* PhD thesis, University of Edinburgh, 1993.

[Gil92] D.G. Gilbers. *Phonological Networks.* PhD thesis, University of Groningen, 1992. ISSN 0928-0030.

[Gol67] E. Mark Gold. Language identification in the limit. *Information and Control*, 10:447–474, 1967.

[Mug92] Stephen Muggleton. Inductive logic programming. In Stephen Muggleton, editor, *Inductive Logic Programming*, pages 3–27. Academic Press, 1992.

[Pin94] Steven Pinker. *The Language Instinct.* W. Morrow and Co., New York, 1994.

[Rab90] Lawrence Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In Alex Waibel and Kai-Fu Lee, editors, *Readings in speech recognition*, pages 267–296. Morgan Kaufmann, San Mateo, 1990. originally IEEE tutorial (1989).

[Reb76] A.S. Reber. Implicit learning of synthetic languages: The role of the instructional set. *Journal of Experimental Psychology: Human Learning and Memory*, 2, 1976.

[SSCM91] D. Servan-Schreiber, A. Cleeremans, and J.L. McClelland. Graded state machines: The representation of temporal continguincies in simple recurrent networks. *Machine Learning*, pages 161–193, 1991.

[Tjo95] Erik F. Tjong Kim Sang. The limitations of modeling finite state grammars with simple recurrent networks. In Toine Andernach and Anton Nijholt, editors, *Computational Linguistics in The Netherlands*, pages 133–44, University of Twente, 1995.

[Tjo98] Erik F. Tjong Kim Sang. *Machine Learning of Phonotactic Structure.* PhD thesis, University of Groningen, 1998.

[vA92] Paul van Alphen. *HMM-Based Continuous Speech Recognition: Systematic Evaluation of Various System Components.* PTT Research, Amsterdam, 1992.