Learning the Logic of Simple Phonotactics

Erik F. Tjong Kim Sang[1] and John Nerbonne[2]

[1] CNTS - Language Technology Group, University of Antwerp, Belgium
erikt@uia.ua.ac.be,
WWW: http://lcg-www.uia.ac.be/~erikt/
[2] Alfa-informatica, BCN, University of Groningen, The Netherlands
nerbonne@let.rug.nl
WWW: http://www.let.rug.nl/~nerbonne

Abstract. We report on experiments which demonstrate that by abductive inference it is possible to learn enough simple phonotactics to distinguish words from non-words for a simplified set of Dutch, the monosyllables. The monosyllables are distinguished in input so that segmentation is not problematic. Frequency information is withheld as is negative data. The methods are all tested using ten-fold cross-validation as well as a fixed number of randomly generated strings. Orthographic and phonetic representations are compared. This paper is part of a larger project comparing different machine learning techniques on linguistic data.

1 Introduction

This paper describes an application of abduction to recognising the structure of monosyllabic words. It is part of a larger project comparing various learning methods on language-learning tasks. The learning methods compared in the larger project are taken from three paradigms: stochastic learning (Hidden Markov Models), connectionist learning (simple recurrent nets using back propagation), and symbolic learning (abductive inference) In each case we use the methods to build an acceptor for the monosyllables from positive data only, and we 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 similar for other related languages (in current experiments).

The study focuses on three questions. First, we ask whether by abductive inference it is possible to learn the structure of monosyllabic words. Second, we seek to learn the influence of data representation on the performance of the learning algorithm and the models it produces. Third, we would like to see whether the learning process is able to create better models when it is equipped with basic initial knowledge, so-called innate knowledge.

The phonotactic models that will be produced can be used for suggesting non-dictionary correction alternatives for words generated by Optical Character Recognition (OCR) software. This study on phonotactics is also important

[0] This paper is a revised compilation of part of the work described in [Tjong Kim Sang, 1998] and [Tjong Kim Sang and Nerbonne, 1999].

for the Groningen research group because it is our first application of machine learning techniques to natural language processing. The problem chosen is deliberately simple in order to make possible a good understanding of the machine learning techniques. The results of this study will be the basis of future research in even more challenging applications of machine learning to natural language processing.

2 Theoretical background

2.1 Problem description

Why is 'pand' a possible English word and why not 'padn'? Why is 'mloda' a possible Polish word but not a possible Dutch word? For answers to these questions one has to know the syllable structures which are allowed in English, Polish and Dutch. Native speakers of English can tell you that 'pand' is a possible English word and that 'padn' is not. For this judgement they do not need to have seen the two words before. They use their knowledge of the structure of English words to make his decision. How did they get this knowledge?

Which words are used depends on the PHONOTACTIC structure of the language—which sequences basic sound elements may occur in. Certain languages, such as Polish, allow *ml* onsets of words 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. Two possibilities exist 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 to develop a model capable of learning phonotactics, we can use it to acquire the phonotactic structure of many languages.

Both artificial intelligence and psychology offer a wide variety of learning methods: rote learning, induction, learning by making analogies, explanation based learning, statistical learning, genetic learning and connectionist learning. We are not committed to one of these learning methods but we are interested in finding the one that performs best on the problem we are trying to tackle: acquiring phonotactic structure. For the experiments in the project reported on here we have chosen methods from three machine learning paradigms, Hidden Markov Models (from stochastic learning), abductive inference (from symbolic learning), and simple recurrent networks (from connectionist learning). This paper focuses on abductive inference.

The problem of distinguishing words from nonwords is not particularly difficult, as linguistic problems range. One perspective on the complexity of the task is given by comparison to an appropriate baseline. The simplest learning algorithm we could think of accepted all and only strings that consist of character pairs (bigrams) which appear in the training data. This algorithm accepted $99.3 \pm 0.3\%$ of the positive orthographic data and rejected $55.7 \pm 0.9\%$ of the negative orthographic data. For the phonetic data (see below) these scores were $99.0 \pm 0.5\%$ and $76.8 \pm 0.5\%$ respectively — indicating an easier task. This baseline algorithm was good in accepting positive data but performed less well in rejecting negative data.

2.2 Data representation

The importance of knowledge representation is widely acknowledged. The representation of input to a problem solving process can make the difference between the process finding a good result or finding no result. The input for our learning methods can be represented in two ways. The first one is orthographic: 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" will be represented as [ðə sʌn ɪz ʃaynɪŋ]. The second representation uses the *International Phonetic Alphabet* (IPA), which enjoys general acceptance among scholars of phonotatics [Ladefoged, 1993].

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 is most suitable for the learning methods. Therefore we perform two sets of experiments: one with data in the orthographic representation and one with the same data in the phonetic representation.

We work with Dutch and since Dutch orthography is fairly transparent, it turned out that there is less need to distinguish the two problems in Dutch. The similarity of the orthographic and phonetic problems is also reflected in the similar baseline performances. A more detailed phonetic representation, in which common features are directly reflected, might still yield significantly different results.

Our learning algorithms 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 algorithms are forced to take the context of a character into account when they consider extensions. Without this context they would make embarrassing errors.

2.3 Innate knowledge

A recurrent issue in modeling language acquisition is the amount of innate knowledge available or required. Linguists have emphasised that important aspects of

language learning require some innate knowledge [Pinker, 1994]. Debates in the language acquisition literature have led to a general acceptance of the assumption that children use innate linguistic constraints when they acquire their native language. [Finch, 1993] describes approaches to language acquisition which assume no innate knowledge. We are interested in what artificial language learning systems can gain from equipping them with initial linguistic constraints. Therefore we will perform two versions of our experiments: one version without specific linguistic constraints and another in which the learning algorithm starts from general phonotactic knowledge. In the case of other methods we have studied in the larger project (notably connectionist methods), this required some creativity, but we believe that reasonable operationalizations were 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 characterise perfectly [Gold, 1967], but we will be satisfied with good performance on the task of distinguishing words and nonwords. 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. This presents a problem to us: 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 will approach the acquisition of models for monosyllabic words from the research perspective of child language acquisition. We will 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 will 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 Experiment setup

3.1 Evaluating syllable models

We vary experiments by using two initialisation types and two data representation types. In order to be able to compare the results of the experiments (especially within the larger project), we perform all experiments with the same training and test data and use only one linguistic model for all linguistic initialisation. We use ten-fold cross-validation which means that we randomly 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. The results presented here are the average performances over the ten test sets.

A further question is how to evaluate the monosyllabic word models. After a learning phase the word models are tested with two sets of data. One set contains unseen positive language data, that is words which occur in the language but have not been present in the training data. The other data set will contain negative data: strings which do not occur as words in the language. The algorithm can make two errors. Firstly, it can classify positive test data as incorrect (false negatives). Secondly, it can classify negative test data as correct (false positives). Our goal will be to find a model which makes as few errors as possible.

3.2 The training and test data

The learning algorithms receive training data as input and use this set for building models of the structure of Dutch monosyllabic words. The models are able to evaluate arbitrary strings. They can either accept a string as a possible monosyllabic Dutch word or reject it. A good phonotactic model will accept almost all strings of the positive test data and reject almost all strings of the negative test data.

The data sets were derived from the CELEX cd-rom [Baayen et.al., 1993]. From the Dutch Phonology Wordforms directory (DPW) we have extracted 6218 monosyllabic word representation pairs. The 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. The list contained 6190 unique orthographic strings and 5695 unique phonetic strings. We used the character frequencies of the two data sets for generating two sets of 1000 unique random strings which do not appear in the related data set. We use these lists of random strings as negative data in the main experiments.

3.3 The linguistic initialisation model

We perform two versions of the learning experiments: one without initial knowledge and one which is equipped with some linguistic constraints. As an initialisation model we have chosen the syllable model which is presented in [Gilbers, 1992] (see Figure 1). This model is a mixture of the syllable models by [Cairns and Feinstein, 1982] and [Van Zonneveld, 1988]. Hence it will be called the Cairns and Feinstein model.

The Cairns and Feinstein model is a hierarchical syllable model consisting of a tree which contains seven leaves. 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 slanted line between two nodes indicates that the daughter

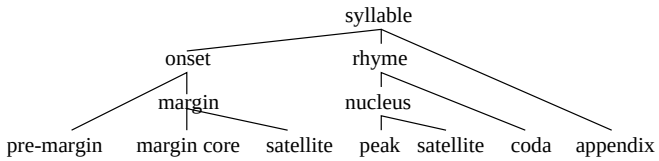

Fig. 1. The syllable model of Cairns and Feinstein

node is dependent on the sister node that is a main constituent. A dependent constituent can only be filled if this main constituent is filled. For example, the margin satellite can only contain a phoneme if the margin core contains a phoneme.

This syllable model can be used to explain consonant deletion in child language. For example, the Dutch word *stop* fits in the model as *s*:pre-margin, *t*:margin-core, *o*:peak and *p*:coda (the *t* cannot occur in the margin satellite in Dutch). The model predicts that a child that has difficulty producing consonant clusters will delete the dependent part in the onset cluster and produce *top*. Another example is the word *gram* which fits in the model as *g*:margin-core, *r*:margin satellite, *a*:peak and *m*:coda (the *g* cannot occur in the pre-margin in Dutch). In this case the model will predict that the child will produce *gam*. Both predictions are correct.

In our experiments with initial knowledge we will supply the learning algorithms with the syllable structure presented in Figure 1. Two extra constraints will be provided to the algorithms: the peak can only contain vowels while the other leaves are restricted to consonants. Finally the division of the phonemes in vowels and consonants will be made explicit for the learning algorithms. Their task will be 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 will convert the general Cairns and Feinstein model to a language-specific syllable model.

4 Abductive Reasoning

4.1 Introduction

We use a version of abductive inference which is related to Inductive Logic Programming (ILP). This is a logic programming approach to machine learning [Muggleton, 1992]. The term induction denotes a reasoning technique which can be seen as the inverse of deduction. In ILP, one makes a distinction between three types of knowledge namely background knowledge, observations and the hypothesis [Muggleton, 1992]. Background knowledge is the 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 hypothesis contains the model of the domain that ILP should build. The relation between these three knowledge types can be described with two rules:

DR $B \wedge H \vdash O$
IR $B \wedge O \longmapsto H$

These rules contain three symbols: $\wedge$ stands for "and", $\vdash$ stands for "leads deductively to" and $\longmapsto$ stands for "leads inductively to". DR represents the deductive rule which states that the observations (O) are derivable from the background knowledge (B) and the hypothesis (H) [Muggleton, 1992]. The inductive rule IR represents the inductive step that we want to make: derive a hypothesis from the background knowledge and the observations.

In ILP, the hypothesis that is derived consists of rules which contain variables. We will specify the rule formats in advance and restrict the derivation to variable-free instances of these rules. Hence we perform abduction rather than induction. We will regard word production as a process of adding prefix and suffix characters to words[1]. The possibility of adding a prefix (suffix) character to a word will depend on the first (last) character of the word. Our models will consist of prefix clauses PC(A,B) and suffix clauses SC(A,B) which state in which context certain characters can be added and of basic word clauses BWC(A) which state which basic words are correct[2].

These clauses are made more concrete in the following three rules:

SUFFIX RULE
Suppose there exists a suffix clause SC(P,F) and a word W such that F is the final character of W and P is the penultimate character of W and word W less F (W - F) is W without its final character F.
In that case the fact that W is a valid word will imply that W - F is a valid word and vice versa.

PREFIX RULE
Suppose there exists a prefix clause PC(I,S) and a word W such that I is the initial character of W and S is the second character of W and word W - I is W less its initial character I (W - I).
In that case the fact that W is a valid word implies that W - I is a valid word and vice versa.

BASIC WORD RULE
The existence of a basic word clause BWC(W) implies that word W is a valid word and vice versa.

The suffix and the prefix rules can be written in Prolog as

```
bwc(WminF):-bwc(W),sc(P,F),append(R,[P,F],W),append(R,[P],WminF).
bwc(W):-bwc(WminF),sc(P,F),append(R,[P,F],W),append(R,[P],WminF).
bwc([S|R]):-bwc([I,S|R]),pc(I,S).
bwc([I,S|R]):-bwc([I|R]),pc(I,S).
```

where `bwc(W)`, `pc(I,S)` and `sc(P,F)` are the three clauses that can be derived by the abduction process. The words are represented as lists of characters. Two

[1] See [Kazakov and Manandhar, 1998] for a related approach to word segmentation.
[2] The three clauses can be proven to be equivalent to regular grammars.

standard Prolog functions are used to put characters in front of a list and behind a list.

Another important issue that we should take a look at is deciding how we are going to generate monosyllabic word models in practice. We will use a derivation scheme which is based on the three clauses used in our word models. The derivation scheme consists of three rules: one for prefix clauses, one for suffix clauses and one for basic word clauses:

> BASIC WORD CLAUSE INFERENCE RULE
> If word W is a valid word then we will derive the basic word clause BWC(W). All observed words are valid words.

> PREFIX CLAUSE INFERENCE RULE
> If W with initial character I and second character S is a valid word and W - I, which is W less its initial character I, is a valid word as well then we will derive the prefix clause PC(I,S).

> SUFFIX CLAUSE INFERENCE RULE
> If W with final character F and penultimate character P is a valid word and W - F, which is W less its final character F, is a valid word as well then we will derive the suffix clause SC(P,F).

An example: suppose we are looking for a model describing the three words *clan*, *clans* and *lans*. We can use the basic word clause inference rule for deriving three basic word clauses for our model: BWC(*clan*), BWC(*clans*) and BWC(*lans*). The first two in combination with the suffix clause inference rule enable us to derive suffix clause SC(*n*,*s*). The prefix clause PC(*c*,*l*) can be derived by using BWC(*clans*) and BWC(*lans*) in combination with the prefix clause rule. This new prefix clause can be used in combination with BWC(*clan*) and the prefix clause rule to derive BWC(*lan*). This new basic word clause makes the other basic word clauses superfluous. Our final model consists of the clauses BWC(*lan*), PC(*c*,*l*) and SC(*n*,*s*).

4.2 Algorithm

The abductive clause inference algorithm will process the data in the following way:

1. Convert all observations to basic word clauses.
2. Make a pass through all basic words and process one at a time. We will use the symbol W for the word being processed and assume that W has initial character I, second character S, penultimate character P and final character F. We will perform the following actions:
   (a) If W without I (W - I)is a valid word as well then derive the prefix clause PC(I,S) and remove the basic word clause for W.
   (b) If W without F (W - F) is a valid word as well then derive the suffix clause SC(P,F) and remove the basic word clause for W.

(c) If the prefix clause PC(I,S) exists then derive the basic word clause BWC(W - I) and remove the basic word clause for W.
(d) If the suffix clause SC(P,F) exists then derive the basic word clause BWC(W - F) and remove the basic word clause for W.

3. Repeat step 2 until no new clauses can be derived.

Steps 1, 2(a) and 2(b) are straightforward applications of the inference rules for basic words, prefix clauses and suffix clauses which were defined in section 4.1. The steps 2(c) and 2(d) are less intuitive applications of the basic word clause inference rule in combination with the rules noted above. In the suffix rule we have defined that W - F will be a valid word whenever W is a valid word and a suffix clause SC(P,F) exists. This is exactly the case handled by step 2(d) and because of the fact that W - F is a valid word we may derive BWC(W - F) by using the basic word clause inference rule. Step 2(c) can be explained in a similar way.

The steps 2(c) and 2(d) will be used to make the basic words as short as possible. This is necessary to enable the algorithm to derive all possible prefix and suffix clauses. Consider for example the following intermediate configuration clauses set:

BWC(yz)
BWC(yx)
SC(y,z)

By applying step 2(d) we can use SC(y,z) and BWC(yz) to add the basic word clause BWC(y) and remove BWC(yz). On its turn this new basic word clause in combination with BWC(yx) can be used for deriving the suffix clause SC(y,x). In this example shortening a basic word has helped to derive an extra suffix clause. The abductive clause inference algorithm will repeat step 2 until no more new clauses can be derived.

4.3 Experiments

In this section we will describe our abduction experiments. The experiments were performed on the data described above. We have constructed an algorithm that performed several passes over the data while deriving prefix clauses, suffix clauses and basic word clauses while removing redundant basic word clauses whenever possible.

Experiments without linguistic constraints We used abductive inference to derive a rule-based model for the orthographic training data. The tree rules were used for generating clauses from the training words, the observations, without using any background knowledge. We used a ten-fold cross-validation set-up which means that we ramdomly divided the training data in ten parts and used each of them as positive test data while using the other nine as training data. The same negative data set was used in all ten experiments.

data type	method	number of clauses			% accepted	% rejected
		basic word	prefix	suffix	positive data	negative data
orthographic	abduction	27±0	377±2	377±2	99.3±0.3	55.7±0.9
orthographic	baseline				99.3±0.3	55.7±0.9
phonetic	abduction	41±0	577±2	577±2	99.1±0.4	74.8±0.2
phonetic	baseline				99.0±0.5	76.8±0.5

Fig. 2. Average performance and size of the models generated by abductive inference without using background knowledge. The algorithm converts the training strings into models that on average contain 800 orthographic rules and 1200 phonetic rules. The models perform well on the positive test data (more than 99% were accepted) but poorly on negative data (rejection rates of 56% and 75%)—roughly the baseline recognition rate for bigrams.

The performance of the model was similar to the baseline performance (see Figure 2). For orthographic data it performed well in accepting positive data (99.3%) but bad in rejecting negative data (55.7%). The false negatives included loan words like *bye*, *fjord*, *hadzj* and *kreml* and short words containing an apostrophe like *'m* and *q's*. The false positives contained reasonable strings like *eep* but also peculiar ones like *bsklt* and *zwsjn*.

We discovered that the learning algorithm had difficulty dealing with single characters that appeared in the training data. These characters were identified as basic words. Because of this most strings could be analysed with a basic word clause combined with either only prefix clauses or suffix clauses. For example, *man* can be accepted with with the clauses PC(m,a), PC(a,n) and BWC(n).

Because of the absence of single characters in the phonetic data, it is more difficult to find out what the origin was of the problems of the models for phonectic data. However, the result was the same as for the orthographic models: after training almost all symbols were identified as basic words. Again many strings could be accepted by the models in an incorrect way. The acceptance rate for the positive data was high (99.1%) but the reject rate for negative strings was not so good (74.8%). Neither for orthographic data nor for phonetic data did we manage to improve the baseline performance (see table 2).

Adding extra linguistic constraints As noted above, we look to the phonetic model by [Cairns and Feinstein, 1982] for suggestions about possibly innate linguistic constraints, and we will use this model in our experiments as well. One problem we have to solve is the conversion of the Cairns and Feinstein model to the clause structure. For this purpose we will make use of a model developed in our stochastic experiments [Tjong Kim Sang, 1998]. This model is derived from Figure 1 of Section 3.3. It consists of a set of linked states and the production capabilities of each state have been limited to a subset of all characters. We will use this model for deriving some extra learning constraints.

The automaton in Figure 3 contains two separate parts with nine states in total. Each of the states s_1 to s_7 corresponds with one of the seven leaves

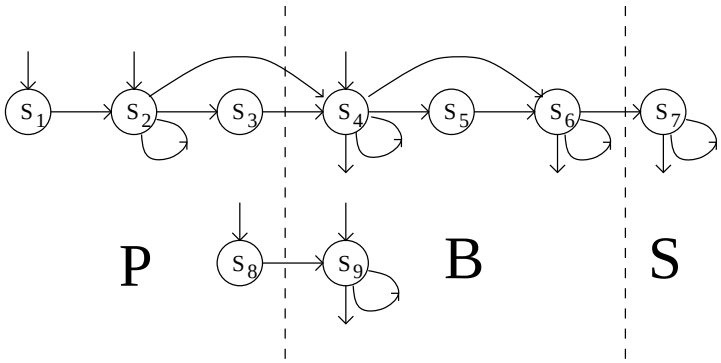

Fig. 3. An initial model for orthographic data, derived from the Cairns and Feinstein model. States produce characters and arcs denote possible transitions. The model has been divided in three parts: a part P in which the prefix rules operate, a part B generated by the basic word clauses and a part S in which the suffix clauses work. Constraints within each part of the model will be converted to constraints on the format of the clauses.

of the three in Figure 1 (the left-to-right order is the same). States s_8 and s_9 are duplicates of s_2 which will be used for generating non-vowel strings. States produce characters and arcs denote the possibility of moving from one state to another. Arcs without a start state mark initial states and arcs without an end state mark final states.

We want to divide the model in three parts: one part in which only prefix clauses operate, one part in which only suffix clauses work and one part that is generated by basic word clauses. This division is shown in Figure 3: part P is the part for the prefix clauses, part B is for the basic word clauses and part S is for the suffix clauses. Each character production by states in the parts P and S corresponds to a set of prefix or suffix clauses. The states s_5 and s_6 have been put in the basic word clause part because s_6 is able to produce vowels and we want to produce all vowels in the basic word clauses. This extension is necessary to allow the model to cover loan words like *creme* and *shares*.

The model of Figure 3 is equivalent to the initial Hidden Markov Model used in our stochastic experiments with linguistic constraints [Tjong Kim Sang, 1998]. However, for learning purposes there are differences between the two. Each character produced by a state in the P and the S parts can be modeled with a prefix or a suffix clause. But we cannot produce every single character in the B part with a separate basic word clause because basic word clauses contain character sequences and, analogous with abduction models, we will not combine basic word clauses in the model. Instead of making each basic word state produce only one character, we will make them behave as a group of states and allow them to produce character sequences. This may cause problems when there are internal parts which repeat themselves an arbitrary number of times.

The behaviour invoked by the self-links from the states s_4, s_6 and s_9 cannot be modeled with the basic word clauses. The learning process will not generate models with extendible basic word clauses and this means that the generalisation capabilities of the resulting rule models will be weaker than those of the HMMs developed in Tjong Kim Sang's thesis [Tjong Kim Sang, 1998].

Now that we have changed the Cairns and Feinstein initialisation model to the structure that we are using in this section, we can attempt to derive usable constraints from this model. The Cairns and Feinstein model imposes constraints on the characters that can be generated by a particular state. We have defined the following constraints for our orthographic data: the vowels *a*, *e*, *i*, *o*, *u* and the quote character *'* can only be generated by the states s_4 and s_6, the ambiguous vowel/consonant *y* can be generated by any state and all other characters are consonants which can be generated by any state except s_4. The new states s_8 and s_9 are consonant states: they can generate any character except the six characters *a*, *e*, *i*, *o*, *u* and *'* (we regard the quote character here as a vowel).

When we inspect the model with these character production constraints in mind we can make two interesting observations. First of all, the prefix clause states cannot produce the characters *a*, *e*, *i*, *o*, *u* and *'*. We will call these characters PURE VOWELS. Since the characters produced by these states are put before a word by a prefix clauses, this means that prefix clauses cannot add a pure vowel prefix to a word. Second, the suffix clause state cannot produce a pure vowel. A character produced by this state is a character appended to a word by a suffix clause. This means that suffix clauses cannot append a pure vowel to a word. We can summarise these two observations in the following two rules:

PREFIX CLAUSE CONSTRAINT
In a prefix clause PC(I,S) the character I that is prepended to a word cannot be a pure vowel.

SUFFIX CLAUSE CONSTRAINT
In a suffix clause SC(P,F) the character F that is appended to a word cannot be a pure vowel.

It is not possible to derive a similar constraint for the basic word clauses because these can contain both vowels and consonants. The derivation presented here applies only to orthographic data. In a similar fashion one can encode alternative initial phonetic models, including similar constraints for prefix and suffix clauses[3]. Our phonetic data contains 18 vowels.

We repeated the experiments for deriving rule-based models for our orthographic and our phonetic data by using the prefix and the suffix clause constraints presented in this section. Apart from these extra constraints the experiment setup was the same as described in the previous section. The resulting models were evaluated in the same way as in the previous experiments. The results of these tests can be found in Figure 4.

[3] An additional constraint can be derived for phonetic data: the basic word clauses cannot consist of a mixture of vowels and consonants. We did not use this constraint because we expected it would cause practical problems in the learning phase.

data type	method	number of clauses			% accepted	% rejected
		basic word	prefix	suffix	positive data	negative data
orthographic	abduction	197±4	376±3	194±1	98.6±0.3	84.9±0.3
phonetic	abduction	38±1	674±4	456±2	99.0±0.5	91.9±0.3

Fig. 4. The models generated by abductive inference and their performance after training with the extra prefix and suffix clause constraints. They perform well in accepting valid test data (maximally 1.4% error) and reasonable in rejecting negative data (error rates of 8% and 15%). These models eliminate about two-thirds of the baseline error rates for negative data.

The added constraints during learning make the abduction process generate better models. The orthographic model performs worse in accepting positive data (98.6% versus 99.3%) but remarkedly better in rejecting negative data (84.9% versus 55.7%). The phonetic model performs about equally well in accepting positive data (99.1% versus 99.0%) and much better in rejecting negative data (91.9% versus 74.8%).

4.4 Abductive Reasoning in Language Learning

In this paper we have described experiments considering building models for monosyllabic words with abductive inference. We have designed an abduction process which is capable of handling orthographic and phonetic data. We performed two variants of the learning experiments: one that started without background knowledge and one that was supplied with constraints which were extracted from the syllable model by Cairns and Feinstein [Cairns and Feinstein, 1982]. The process without background knwoledge produced models which performed well for recognising positive data but they performed poorly in rejecting negative data (see Figure 2). The linguistically initialised process generated models that performed well in both cases (see Figure 4).

From the results of the experiments described in this section we may conclude that abductive inference is a good learning method for building mono-syllabic orthographic and phonetic models. Abduction in combination with linguistic constraints generates models that perform much better than the models that were generated without background knowledge.

It is natural to note that abductive inference not only performs reasonably well on this learning task, but that it also produces models which are subject to direct examination and analysis by area specialists, in this case phonologists.

5 Conclusions and Future Directions

The problem of phonotactics as it has been tackled here is basically the problem of sequencing. The results show that abductive inference perform credibly, if not perfectly on this task. [Tjong Kim Sang, 1998] and [Tjong Kim Sang and

Nerbonne, 1999] compare this work to learning by stochastic automata (Hidden Markov Models) and biologically and cognitively inspired Neural Networks (so-called "Simple Recurrent Networks"). These results indicate that rule abduction performs about as well on the sequence learning task as the popular HMM's and also that further advancements in the application of neural networks to sequencing are needed[4].

The results further indicate that using linguistic constraints helps the learning algorithms since this results in improvements in speed and accuracy. Abductive inference is particularly well-suited to accommodating background knowledge which linguists have developed, typically in the form of rule systems.

Finally, results show that learning from written symbols is only slightly more difficult than learning from phonetic representation, but this may have to do with the fairly phonic Dutch orthography.

The models derived in this paper satisfy four of the five properties Mark Ellison outlined in his thesis [Ellison, 1992]. They are cipher-independent (independent of the symbols chosen for the phonemes); language-independent (they make no assumptions specific for a certain language); accessible (in symbolic form); and linguistically meaningful. They fail to satisfy Ellison's first property (operation in isolation) because they receive specific language input: monosyllabic words. This was deliberate, naturally, since we wished to focus on the single problem of sequencing. Moreover, the techniques can help a linguist to get a rough impression of the syllable structure of a language.

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 (following Ellison's criterion) coding input for phonetic properties, and naturally extending the task to multisyllable words and to related tasks in phonological learning.

Acknowledgements

The authors want to thank two reviewers for valuable comments on earlier versions of this paper. This work was sponsored by the former Dutch Graduate Network for Language, Logic and Information (currently Dutch Graduate School in Logic, OZSL).

[4] Without background knowledge, the HMMs accepted 99.1% of the positive orthographic data and rejected 82.2% of the negative orthographic data. Using linguistic constraints changed these figures to 99.2% and 77.4% respectively. For phonectic data these numbers were 98.7%, 91.6%, 98.9% and 92.9%. The neural networks performed poorly. They were not able to reject more than 10% of the negative data in any experiment [Tjong Kim Sang and Nerbonne, 1999].

References

1. R.H. Baayen, R. Piepenbrock, and H. van Rijn. *The Celex Lexical Database (CD-ROM)*. Linguistic Data Consortium, University of Pennsylvania, Philadelphia, PA, 1993.
2. Charles E. Cairns and Mark H. Feinstein. Markedness and the theory of syllable structure. *Linguistic Inquiry*, 13(2), 1982.
3. T. Mark Ellison. *The Machine Learning of Phonological Structure*. PhD thesis, University of Western Australia, 1992.
4. Steven P. Finch. *Finding Structure in Language*. PhD thesis, University of Edinburgh, 1993.
5. D.G. Gilbers. *Phonological Networks*. PhD thesis, University of Groningen, 1992. ISSN 0928-0030.
6. E.M. Gold. Language identification in the limit. *Information and Control*, 16:447–474, 1967.
7. Dimitar Kazakov and Suresh Manandhar. A hybrid approach to word segmentation. In David Page, editor, *Proceedings of the ILP-98*. Springer, 1998. Lectures Notes in Computer Science, vol. 1446.
8. Peter Ladefoged. *A Course in Linguistic Phonetics*. Philadelphia, 3 edition, 1993.
9. Stephen Muggleton. Inductive logic programming. In Stephen Muggleton, editor, *Inductive Logic Programming*, pages 3–27. 1992.
10. Steven Pinker. *The Language Instinct*. W. Morrow and Co., New York, 1994.
11. Erik F. Tjong Kim Sang. *Machine Learning of Phonotactic Structure*. PhD thesis, University of Groningen, 1998.
12. Erik F. Tjong Kim Sang and John Nerbonne. Learning simple phonotactics. In *Neural, Symbolic, and Reinforcement Methods for Sequence Learning*, pages 41–46, 1999. Proc. IJCAI workshop.
13. R.M. van Zonneveld. Two level phonology: Structural stability and segmental variation in dutch child language. In F. van Besien, editor, *First Language Acquisition*. ABLA papers no. 12, University of Antwerpen, 1988.