Chapter 1

Introduction

This thesis contains a study of the application of machine learning methods to natural language. We will use the learning methods Hidden Markov Models, Simple Recurrent Networks and Inductive Logic Programming for automatically building models for the structure of monosyllabic words. These three learning algorithms have been chosen as representatives for three main machine learning paradigms: statistical learning, connectionist learning and rule-based learning. The language data to which they will be applied has been limited to monosyllabic words in order to keep down the complexity of the learning problem. We will work with Dutch language data but we expect that the results of this study would have been the same if it had been done with another related language.

The study will focus on three questions. First, we want to know which of the three learning methods generates the best model for monosyllabic words. Second, we are interested in finding out what the influence of data representation is on the performance of the learning algorithms and the models they produce. Third, we would like to see if the learning processes are able to create better models when they are equipped with basic initial knowledge, so-called innate knowledge.

This book contains five chapters. The first chapter will describe the problem. In the second chapter we will introduce the statistical learning method Hidden Markov Models and present the results of the experiments we have performed with this method. In the third and fourth chapters we will do the same for the connectionist method Simple Recurrent Networks and the rule-based method Inductive Logic Programming respectively. The final chapter contains a comparison of the results of all experiments and some concluding remarks.

1 Theoretical background

In this section we will give a description of the learning problem we want to tackle: building phonotactic models for monosyllabic words. We will also introduce some theoretical issues related to this problem: the importance of a good representation of data, the influence of negative training examples and the influence of innate knowledge.

1.1 Problem description

Why is *pand* a possible English word while *padn* is impossible in English? Why is *mloda* a possible Polish word but not a possible Dutch word? For giving the 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. Judging the words does not require that the native speakers have seen them before. They use their knowledge of the structure of English words to make their decision. How did they obtain this knowledge?

In the example we have presented we showed that the possibility that a word exists depends on the structure of the language the word appears in. Certain languages, like Polish, allow *ml* onsets of words but others, like English and Dutch, do not. The structure of words in a language is called the phonotactic structure of a language. Different languages may have different phonotactic structures.

There are two possibilities for entering a language dependent phonotactic structure into a computer program. The first is by making humans examine the language and make them create a list of rules defining the phonotactic structure. This requires a lot of labor which has to be done for all languages. The second possibility is making the program *learn* the phonotactic structure of a language by providing it with language data. People manage to learn phonotactic rules which restrict the phoneme sequences in their language so it might be possible to construct an algorithm that can do the same. If we are able to develop a model for the phonotactic structure learning process we can use the model to analyze 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 our experiments we have chosen three machine learning paradigms: statistical learning, connectionist learning and rule-based learning. We will use learning methods from these three paradigms and compare their performance on our learning problem.

A possible application of these phonotactic models lies in the field of Optical Character Recognition (OCR). OCR software frequently has to make a choice between two or more possible interpretations of a written or printed word, for example between *ball* and *baII*. The most easy way of solving this problem is by looking up the words in a dictionary and choosing the one which appears in the dictionary. This approach fails when neither one of the words is present in the dictionary. In that case the software should be able to determine the probability that the words exist in the language which is being read. A phonotactic model for the language can be used for this.

This study on phonotactics is also important for our 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.

1.2 Data representation

Every artificial intelligence text book emphasizes the importance of knowledge representation. The way one represents the input data of a problem solving process can make the difference between the process finding a good result or finding no result. A nice example of the usefulness of knowledge representation is the Mutilated Checkerboard Problem presented in (Rich et al. 1991). Here a checkerboard from which two opposite corner squares have been removed, needs to be covered completely with dominoes which occupy two squares each. This problem is unsolvable. This fact can be proven by trying out all possible domino configurations but that will require a lot of work.

The solution of the Mutilated Checkerboard Problem can be found more quickly by changing the representation of the problem and representing the board as a collection of black and white squares. An inspection of the board reveals that it contains 30 white squares and 32 black squares. Each domino must cover exactly one white square and exactly one black square so this problem is unsolvable. One could ask if this more suitable problem representation could have been foreseen. Unfortunately the best way for representing data is dependent on the problem that one wants to solve. There is no algorithmic method for deciding what data representation is the best for what type of problem.

The input data for our learning problem can be represented in several ways. We will take a look at two representation methods. 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 the phonetic way. If we use the phonetic representation then the sentence *the sun is shining* will be represented as [ðəsʌnɪʃainɪŋ]. We do not know which of the two representations will enable the learning processes 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 method is most suitable for our learning problem. Therefore we will perform two variants of our learning experiments: one with data in the orthographic representation and one with the same data in the phonetic representation.

1.3 Positive and negative learning examples

A learning algorithm can receive two types of learning input: positive examples and negative examples. A positive example is an example of something that is correct in the language that has to be acquired and a negative example is an example of something that is incorrect. Together with the examples the algorithm will receive classifications (correct/incorrect).

Gold's landmark paper (Gold 1967) has shown that it is not possible to build a perfect model for a general language that contains an infinite number of strings by only looking at positive examples of the language. The reason for this is that for any set of positive example strings there will be an infinite number of models that can produce these examples. Without negative examples it is not possible to decide which of these models is the correct one. The research result of Gold has consequences for natural language learning. Natural languages are infinite because they contain an infinite number of sentences. This means that according to language learning theory it is not possible to build a perfect model for a natural language by only looking at correct sentences of that language.

With Gold's research results in mind one would predict that children use negative language examples for acquiring natural language. However, research in child language acquisition has found no evidence of children using negative examples while learning their first language (Wexler et al. 1980). Even when children are corrected they will pay little attention to the corrections. Here we have a problem: according to computational learning theory, children need negative examples for learning if they want to be able to learn a natural language. Children do not seem to make use of negative examples and yet they manage to acquire good models for natural languages.

We will approach the acquisition of models for monosyllabic words from the research results in child language acquisition. We will supply our learning methods with positive information only. However mathematical language learning theory predicts that negative examples are required for obtaining good language models. We will assume that negative information can be supplied implicitly. In the next section we will deal with a possible solution for the absence of negative examples in our learning experiments.

1.4 Innate knowledge

There have been a number of attempts to explain the gap between what learning theory states about the necessity of negative examples and what child language acquisition reports about the absence of these negative examples. One proposed explanation assumes that the learners use available semantic information for putting constraints on natural language utterances (Finch 1993). Another explanation suggests that learners acquire reasonably good language models rather than perfect models (probably approximately correct (PAC) learning (Adriaans 1992)). A third explanation restricts the languages that human learners can acquire to a small subset of the languages which are possible theoretically (Finch 1993).

All three explanations have some cognitive plausibility and could be applicable to our learning problem. The usage of extra semantic information in computational learning experiments has a practical problem: this information is unavailable (Finch 1993). The ideas behind probably approximately correct (PAC) learning are interesting and we will use some of them in our experiments. As in PAC learning we will accept models that perform as well as possible rather than restricting ourselves to perfect models. However unlike some PAC learning algorithms we will rely on the fact that all of our learning examples are correct and we will not use the ORACLE concept mentioned in (Adriaans 1992) because that would imply using negative examples.

We will perform learning experiments in which the set of languages that can be acquired will be restricted. This can be done in practice by enabling the learning algorithm to choose from a small set of models instead of all possible models. This simplifies the task of the learning algorithm. Restricting the set of languages is an approach which is also suggested in human language acquisition theory (Wexler et al. 1980) (Chomsky 1965). Humans are not regarded as being capable of learning all mathematically constructible languages. They can only acquire languages of a smaller set: the natural languages. The restriction to this smaller set is imposed by innate cognitive constraints. Human language learning can be modeled by a system which sets parameters in a general language device in order to change it to a language-specific device.

While it might be necessary to assume extra initial knowledge for the acquisition of a complete language model, one could also try to generate some reasonably good language models without using initial knowledge. The thesis of Steven Finch (Finch 1993) gives some examples of extracting lexical information from positive data without assuming innate language knowledge. We do not know whether such an approach would be successful for our learning problem. We are interested in what gain artificial language learning systems can get from equipping them with initial linguistic knowledge. Therefore we will perform two versions of our experiments: one version without initial knowledge and another in which the learning algorithm starts from basic phonotactic knowledge. The linguistic model that we will use as initial knowledge will be explained in section 2.4.

2 Experiment setup

This section contains the practical issues concerning approaching our phonotactic learning problem. We will start with describing the goals of the experiments we want to perform. After that we will take a look at the format and the complexity of our training and test data. We will continue with examining the linguistic model we will use in our experiments which start with basic phonotactic knowledge. The section will be concluded with a paragraph containing the statistical theory which will be used for interpreting the experiment results.

2.1 Goals

We will perform experiments with monosyllabic phonotactic data and attempt to derive a phonotactic model from positive examples of the data. The model should be able to decide whether strings are members of the language from which the training data has been taken or not. It can be considered as a black box which takes strings as input and returns *yes* if the string is a possible member of the training language and *no* if it is not. The model might assign more than two evaluation scores to the strings. If that is the case then we will assume that the scores can be ordered and that the comparison of their values with a threshold value will determine whether they should be counted as *yes* or *no*.

Our training algorithms will not receive the complete set of monosyllabic data during the training phase. The consequence of this is that memorizing the training data is insufficient for obtaining a good model. The phonotactic models will have to be able to give a reasonable evaluation of unseen words which might be correct despite the fact that they were not present in the training data. In other words: the models have to be able to generalize.

In order to test the generalization capabilities of the models we will test them with unseen positive data. We will require that after training the models accept all training data so we can skip testing their performance on this data. We will also test the models with incorrect data to make sure that they do not accept a lot of incorrect strings. The two tests will result in two scores: the probability of accepting a string of the unseen positive data and the probability of rejecting a string of the negative data.

While performing the phonotactic model acquisition experiments we will look for the answers to the following questions:

1. What learning algorithm produces the best phonotactic model?
2. What data format results in the best phonotactic model?
3. Does starting from initial knowledge produce better phonotactic models?

We will perform the same experiment with algorithms representing three different machine learning paradigms: Hidden Markov Models (statistical learning), Simple Recurrent Networks (connectionist learning) and Inductive Logic Programming (rule-based learning). All experiments will be done with the same training and test data and under the same conditions to make a comparison of the results fair. It is not possible for us to test every possible learning algorithm so the final comparison might not point to the best algorithm. However it will give an indication to which machine learning paradigm performs best on this problem.

We will compare two different data formats: the orthographic format and phonetic format (see section 1.2). For each experiment we will perform two variants: one with training and test data in orthographic format and one with the same data in phonetic format. We are interested in finding out which data format will enable the learning algorithms to produce the best phonotactic models.

We would also like to find out whether learning algorithms that are equipped with initial phonotactic knowledge will generate better phonotactic models than algorithms without this knowledge. Therefore we will perform two variants of each learning experiment: one in which the learning algorithms start without any knowledge and one in which they have some initial phonotactic knowledge. The initial model will be derived from a phonological model which will be described in section 2.4.

Thus we will perform twelve variants of a phonotactic acquisition experiment with three learning techniques, two data formats and two initialization types. Care will be taken to perform the experiments under the same conditions so that a final comparison between the results will be fair.

2.2 The training and test data

The learning algorithms will receive a training data set as input and they will use this set for building models for the structure of Dutch monosyllabic words. The models will be able to compute acceptance values for 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 correct unseen words (positive test data) and reject almost all impossible words (negative test data, also called random data).

The positive data sets have been derived from the CELEX cd-rom (Baayen et al. 1993). From the Dutch Phonology Wordforms directory (dutch/dpw/dpw.cd) we have extracted 6218 monosyllabic word representation pairs.[1] 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). We have removed 10 words of the list because they were not mentioned in the standard Van Dale dictionary for Dutch (Geerts et al. 1992) (*flute*, *flutes*, *frite*, *Joosts*, *move*, *moves*, *rocks*, *straight*, *switch* and *switcht*). Another three words have been removed because they had been incorrectly classified as monosyllabic words (*racend*, *fakend* and *shakend*). We obtained 6205 unique pairs. The list contained 6177 unique orthographic strings and 5684 unique phonetic strings.

After this we randomly chose 600 pairs from the list. We made sure that these pairs contained neither duplicate orthographic strings nor duplicate phonetic strings. The 600 orthographic strings and the corresponding 600 phonetic strings will be used as test data. The remaining 5577 orthographic words and 5084 phonetic words will be used as training data. The orthographic data contained the twenty six characters of the alphabet plus the quote character ' in order to allow for words as *ski's*. The phonetic data contained 41 different characters.

[1] The monosyllabic words have been selected by removing all lines with hyphenation marks and all lines with empty phonetic representations. After that the fields Head and PhonolCPA were extracted, the upper case characters were converted to lower case and the duplicate pairs were removed.

We have used the character frequencies of the orthographic data for generating 700 random orthographic strings. The generation process has assumed that characters occurred independently of each other. We have transcribed the 700 random strings and thus we have obtained a list of 700 random phonetic strings. From the lists we have removed 60 strings that resembled Dutch strings, 2 strings that had a phonetic representation that occurred earlier in the list and 38 strings from the end of the list. We obtained a list of 600 unique implausible orthographic strings and a list of 600 corresponding unique phonetic strings. We will use these lists as negative test data files for our experiments.

The final operation we performed on the data sets was appending to each string an end-of-word character.[2] This was necessary because in the statistical and the connectionist learning methods the words will be presented to the learning algorithms in a sequence and the algorithms will need some way for determining where the current string ends and a new string begins.

2.3 Data complexity

The performance of the learning algorithm is dependent on how difficult the data is. Acquiring a phonotactic model for a language that consists of the strings *a*, *aa* and *aaa* is much easier than acquiring a model for monosyllabic Dutch words. There are different measures which formalize this intuitive notion of data complexity. In this section we will look at two of these. We will also apply these complexity measures to our data.

The first complexity measure we will examine is called ENTROPY.[3] This is a number which indicates how uncertain the result is of drawing elements from a set. For example we have a set containing two *x*'s, one *y* and one *z* and the probability of drawing an *x* is 50% and the probability of drawing a *y* or a *z* is 25%. The entropy of this experiment is 1.5 (the computation will be performed below). A small entropy means that it is easy to predict the results of the draws and a large entropy means that the prediction is difficult.

The results of draws from a set can be represented with the concept STOCHASTIC VARIABLE. Our example is equivalent to a stochastic variable with the values *x*, *y* and *z* in which the probabilities of the values are 50%, 25% and 25% respectively. We give this stochastic variable the name *W* and call the three values c_1, c_2 and c_3. Then we can compute the entropy H(W) of this variable by using the probabilities P(c_i) of each value and the formula (Charniak 1993):

$$H(W) = - \sum_{c_i} (P(c_i) * log_2(P(c_i)))$$

So H(W) is the negation of the sum of the products of the probability P(c_i) and its $\log_2$ value for all values c_i. If we apply this formula to our example with the probabilities

[2] In chapter 2 we will give a motivation for using a start-of-word character as well.

[3] The complexity measure PERPLEXITY is related to entropy. It will not be not discussed here.

P(x)=0.5, P(y)=0.25 and P(z)=0.25 then we obtain the following computation: H(W) = -(0.5*$\log_2$(0.5) + 0.25*$\log_2$(0.25) + 0.25*$\log_2$(0.25)) = -(-0.5+-0.5+-0.5) = 1.5.

In order to be able to compute the entropy of our data we will regard the words in the positive data as sequences of draws from a set of characters. The probability of each character can be estimated by computing the frequency of the character in the positive data and dividing it by the total number of characters in the data. After having acquired the character probabilities we can compute the entropy of the data. Our positive orthographic data with 6177 words and 41005 characters of which 29 are unique has an entropy of 4.157. Our positive phonetic data with 5684 words and 34430 characters of which 43 are unique has an entropy of 4.294.[4]

A language model that estimates the validity of a word by looking at isolated characters is called a UNIGRAM MODEL. Such a model uses probabilities of characters without looking at their contexts. We will also use models which compute the probability of a character given information about previous characters. The probabilities in these models will say something about the occurrence of pairs of characters and therefore they are called BIGRAM MODELS.

Words in natural languages are not arbitrary sequences of characters. The context of a character has an influence on its probability. For example, in our positive orthographic data the probability that an arbitrary character is a *u* is 3.8% but the probability that a character following a *q* is a *u* is 92%. The task of predicting a character in word is easier when one knows the preceding character. Thus the earlier defined concept of entropy is not very useful for bigram models.

In order to describe the data of a bigram model we need to use a kind of conditional entropy which we will call BIGRAM ENTROPY. The value is computed in a similar way to standard entropy. However instead of character probabilities this computation uses conditional character probabilities: the probability of a certain character c_j given that a specific character c_i precedes it. We have used an adapted version of the formula presented in (Van Alphen 1992) page 96:

$$\mathrm{H(W)} = - \sum_{c_i} P(c_i) \sum_{c_j} (P(c_j|c_i) * log_2(P(c_j|c_i))$$

We have applied this formula to our orthographic and phonetic data and obtained the bigram entropy values of 3.298 and 3.370 respectively.[5] From these values we can conclude that the data is less complex when it is considered as a sequence of character bigrams rather than a sequence of isolated characters. In other words, it is easier to predict a character of a word when one knows the preceding character.

Entropy and bigram entropy give an indication about the complexity of the data. It is unclear how useful these concepts are for predicting the degree of learnability of the data. The general expectation is that data with a large entropy is more difficult to learn

[4] In both orthographic and phonetic data each word contains a start-of-word and an end-of-word character. Without these characters the orthographic data has an entropy of 4.255 while the phonetic data has an entropy of 4.551.

[5] The bigram entropy values without the end-of-word characters were 3.463 for the orthographic data and 3.617 for the phonetic data.

than data with a small entropy. However there are counter-examples. The entropy of a language A consisting of strings that contain an arbitrary number of *a*'s is 0 while a language B with strings that contain any combination of *a*'s and *b*'s has an entropy of 1. If we restrict A to strings with a length *n* where *n* must be a prime number then A would probably be more difficult to learn than B. Yet the entropy of A is still 0 which predicts that A is easier to learn than B.

An alternative measure which can be used for determining the complexity of our data is the CHOMSKY HIERARCHY. This is a hierarchy of classes of grammar which can be used for modeling artificial and natural languages. The hierarchy distinguishes one grammar class for finite languages and four grammar classes for infinite languages. The complexity of a language is determined by the place in the hierarchy of the least complex grammar that is capable of producing it.

Our phonotactic data is finite. The longest strings in our data contain nine characters without begin-of-word and en-of-word character. Monosyllabic words with a few more characters might be possible but the existence of an English or Dutch monosyllabic word of twenty or more characters should be ruled out. Both our orthographic and our phonetic data should be placed in the lowest spot in the Chomsky hierarchy: the group of finite languages. This indicates that a model for the data can be acquired by looking at positive data only (Gold 1967).[6]

A problem of using the Chomsky hierarchy for determining the complexity of data is that the differences within each language/grammar class are large. The empty language and the language consisting of prime numbers with less than a million digits are both finite languages. Yet the first is much easier to learn than the second. The Chomsky hierarchy and the related language learning classes put forward by (Gold 1967) give only a rough indication of the learnability of languages (see also (Adriaans 1992) section 2.3.4).

We conclude that the available techniques for determining data complexity do not have an exact answer on the question of how difficult the learning process will be.

2.4 The linguistic initialization model

We will perform two versions of our learning experiments: one that starts with initial phonotactic knowledge and one that start without any initial knowledge. As an initialization model we have chosen the syllable model which is presented in (Gilbers 1992) (see Figure 1.1). This model is a mixture of 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 with seven leaves. Each leaf can either be empty or contain one phoneme. The appendix leaf may contain two phonemes. Each leaf is restricted to a class of phonemes: in models for Dutch and many other languages the peak may only contain vowels and

[6]Derivation of a perfect model is only guaranteed if one is able to evaluate the complete data set. However we will not provide the complete data set to our learning algorithms.

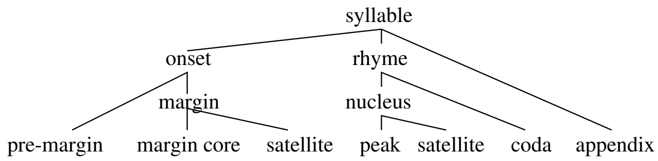

Figure 1.1: The syllable model of Cairns and Feinstein

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.

This syllable model can be used to explain consonant deletion in child language. For example, the word *stop* fits in the model like *s*:pre-margin, *t*:margin core, *o*:peak and *p*:coda. An alternative structure with *t* in the margin core satellite is prohibited by a constraint for Dutch which states that satellite positions only may contain sonorant consonants (like *l*, *m*, *n* and *r*). The model predicts that a child which has difficulty producing consonant clusters will delete the dependent part in the onset cluster and produce *top*. The alternative *sop* would violate the constraint that dependent constituents, in this case pre-margin, cannot be filled if the corresponding main constituent, here margin, is not filled. Another example is the word *glad* which fits in the model like *g*:margin core, *l*:margin satellite, *a*:peak and *d*:coda (the *g* is prohibited in the premargin in Dutch). In this case the model will predict that the child that has problems with producing consonant clusters will delete the *l* rather than the *g*. Both predictions are correct.

In our experiments with initial knowledge we will supply the learning algorithms with the syllable structure presented in Figure 1.1. Two extra constraints will be provided to the algorithms: the fact that the peak may only contain vowels while the other leaves are restricted to consonants. Furthermore the division of phonemes in vowels and consonants will be made available 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.

2.5 Elementary statistics

In this thesis we will need to compare the performances of different learning algorithms on the acquisition of phonotactic models. The performance of the models is defined by two scores: the percentage of accepted correct strings and the percentage of rejected incorrect strings. Both scores have an average and a standard deviation. The question is how we can compare these scores and determine whether one is significantly better than the other.

Two numbers with an average and a standard deviation can be compared with the t-test (Freedman et al. 1991). By using this test we can determine whether the difference between the numbers is caused by a real difference or by a chance error. The test computes a number *t* by dividing the difference of the averages of two numbers we want to compare by the standard error of this difference. The standard error of the difference is equal to the square root of the sum of the squares of the standard errors of the two input numbers. Here are all necessary formulas for comparing two numbers which have averages AVG_x and AVG_y and standard deviations SD_x and SD_y based on measuring them *n* times:

$\mathrm{SE}_x = \mathrm{SD}_x/\sqrt{n-1}$ (standard error of *x*)
$\mathrm{SE}_y = \mathrm{SD}_y/\sqrt{n-1}$ (standard error of *y*)
$\mathrm{SE}_d = \sqrt{SE_x^2 + SE_y^2}$ (standard error of *x*-*y*)
$\mathrm{t} = \frac{AVG_x - AVG_y}{SE_d}$ (t-score)

The result of these computations will be a number *t(n-1)*. This number can be looked up in a *t*-table to find out what the probability is that the difference was caused by a chance error. If the probability is smaller than 5.0% then we will assume that there is a real significant difference between the two numbers.

Example: We have performed two sets of 5 experiments with Hidden Markov Models that process orthographic data: one set was randomly initialized and the other one was initialized with a general version of the Cairns and Feinstein model. In the first set of experiments the HMM needed 108, 87, 81, 48 and 65 training rounds to become stable. In the second experiment set the learning algorithm needed 43, 17, 63, 22 and 47 training rounds. The models with phonotactic initialization seem to train faster and we want to know if the difference is real. Therefore we compute the averages (77.8 and 38.4), the standard deviations (20.3 and 16.9), the standard errors (10.1 and 8.4) the standard error of the difference (13.3) and *t(4)* (3.0). We look up this *t(4)* value in a *t*-table and find out that the probability *p* that the difference between the two number lists was caused by a chance error is between 1.0% and 2.5% ($<$5.0%). We will adopt the notation used in (Van den Bosch 1997) and express this result as *t(4)*=3.0, $p<0.025$. So the difference cannot be explained by a chance error: HMMs with phonotactic initialization train faster than randomly initialized HMMs for this problem.

3 Related work

In this section we will examine literature on the problem of machine learning of phonotactics and related areas. We will start with the work of T. Mark Ellison who has published a thesis and some papers about applying machine learning to phonology. After that we will take a look at the work of Walter Daelemans and his group on using machine learning for building models for hyphenation, syllabification, stress placement and grapheme-to-phoneme conversion. We will conclude with an overview of related work by other authors.

3.1 The work by Ellison

In the thesis (Ellison 1992) Mark Ellison investigates the possibility of using machine learning algorithms for learning phonological features of natural languages. He imposes five constraints on the algorithms:

1. they should work in isolation, that is without help of a human teacher,
2. they should be independent of the way the (phonetic) data has been encoded,
3. they should be language independent,
4. they should generate models which are accessible and
5. they should generate models which are linguistically meaningful.

The goal of Ellison in his thesis is to show that it is possible to create successful machine learning applications which satisfy all five constraints for phonological problems. He evaluates a number of existing learning applications in related domains and concludes that none of them fulfills all five requirements. Many learning systems violate the first constraint because they are supplied with extra knowledge like a vowel-consonant distinction or monosyllabic data. Other systems like statistical and connectionist learning algorithms fail to generate models which are accessible and linguistically meaningful.

Ellison investigates machines learning techniques on three different phonological tasks with positive input data. The first task is the derivation of vowel-consonant distinction models. For this derivation Ellison used an inductive learning technique combined with a model evaluation measure which favored a simple model over more complex models. The learning algorithm represented phonological models as sets of parameters and searched the parameter space for the best model by using the searching technique simulated annealing. Ellison's learning algorithm was applied to data from thirty languages. In all but four languages it divided the phonemes in two groups: the vowels and the consonants. The problems in the four languages were caused by the fact that the program erratically had divided either the consonants or the vowels in two groups based on their positions in the words.

Ellison applied the same learning algorithm to the second and the third learning task. The second task consisted of deriving the sonority hierarchy of natural languages where sonority means the loudness of phonemes. As in the first task the learning algorithm was applied to data from thirty languages. It generated on average three sonority classes per languages and put the vowels in different classes than the consonants in all except one language. The program performed well in separating consonants from vowels but performed less well in building vowel hierarchies and consonant hierarchies.

In the third learning task the algorithm had to derive harmony models. These models contain context constraints on phoneme sequences. The learning algorithm was applied to data from five languages. It discovered correct constraints on vowel sequences for all five languages. Ellison wanted to study vowel harmony and therefore he supplied the learning algorithm with vowel sequences. He argued that this does not mean that the learning algorithm fails his first isolation constraint because it should be considered as part of a larger program in which the vowel-consonant distinction was discovered by the untutored first learning program.

The learning algorithm used by Ellison in his three experiments satisfies his five learning constraints. Its is interesting to compare the experiments we want to perform with these constraints. Only one of our algorithms will satisfy the fourth and the fifth constraint about generating accessible and meaningful results: the rule-based learning method. Neither the statistical nor the connectionist method will generate accessible and meaningful phonotactic models. All our algorithms will be language-independent and use an arbitrary data representation. However they will not satisfy the first untutored constraint because they will process monosyllabic data and will be supplied with linguistic knowledge in the initialized experiments.

3.2 The work by Daelemans et al.

The group of Walter Daelemans has done a lot of work on applying machine learning in natural language processing areas such as hyphenation, syllabification, placement of stress, grapheme-to-phoneme conversion and morphological segmentation. In this work different machine learning algorithms have been applied to several linguistic problems. The algorithms used are inductive memory-based techniques and connectionist techniques. The learning methods that performed best were memory-based techniques which simply stored learning examples rather than building an abstract representation for them.

(Daelemans et al. 1993) describes the application of three learning methods for acquiring the stress pattern in Dutch. Their data contained 4868 monomorphemes for Dutch words. In their first experiment they have applied backpropagation (BP), Analogical Modeling (ANA) and Instance-Based Learning (IBL) for predicting the stress patterns of the words. IBL and BP performed approximately equally well on unseen data while ANA performed slightly less well. IBL and ANA were used in a second experiment with two versions of the data set (one phonetic version). Their learning methods performed reasonable even for stress patterns which are difficult to predict for state-of-the-art theory. The authors conclude that computational learning methods such as ANA, BP and IBL are an alternative to Principles and Parameters based learning theories.

(Daelemans et al. 1995) presents a study in which machine learning techniques are used for building a models for determining the diminutive suffix of Dutch nouns. The learning method used is the decision tree learning method C4.5 (Quinlan 1993). The model generated by C4.5 performed well: it obtained an error rate of 1.6% on this task and outperformed the theoretical model presented in (Trommelen 1983) which paid special attention to the formation of diminutives. By comparing the results of training C4.5 with different parts of the data the authors have been able to falsify Trommelen's claim that rhyme information of the last syllable of a Dutch noun is sufficient for predicting its diminutive suffix.

(Daelemans et al. 1996) discusses grapheme to phoneme conversion based on data-oriented methods. The authors have used the machine learning technique IG-Tree for building grapheme to phoneme conversion for the languages Dutch, English and French. IG-Tree performed significantly better than a connectionist state-of-the-art solution (NETtalk, Sejnowski et al. 1987) and a theoretical model (Heemskerk et al. 1993). Daelemans and Van den Bosch conclude that IG-Tree has three advantages: it does not require gathering rules, it is reusable and it is accurate.

(Van den Bosch et al. 1996) describes the application of four inductive learning algorithms and one connectionist method on three variants of the problem of dividing words in morphemes. The algorithm that generated the best model for all three tasks was IB1-IG, an inductive algorithm that stores all learning examples and compares unseen data with the stored data while taking into account that some data features are more important than others. The algorithm uses information gain, a concept from information theory, for computing the importance of each data feature for the task. It uses the current character and six characters in its context to decide whether to insert a morpheme boundary or not.

(Van den Bosch et al. 1997) presents a motivation for the good results of lazy learning techniques in the domain of natural language learning. A categorization of language data will contain many small clusters of related elements. Since lazy learning techniques store the complete training data they will have less chance of missing small variations than learning methods which summarize data. Equipped with an information-theoretic-based weight of the data features lazy learning techniques will be even more successful in categorizing the data. The authors present an empirical study with a word pronunciation task to support their claims.

3.3 Other work

In (Wexler et al. 1980) Kenneth Wexler and Peter W. Culicover describe a method for learning transformational grammars describing natural language syntax. Transformational grammars are capable of generating arbitrary type 0 languages (Partee et al. 1993) and therefore they are neither learnable from positive examples only nor from both positive and negative information (Gold 1967). The learning method of Wexler and Culicover is based on putting restrictions on the transformational grammars that are to be learned. The input of the method consists of positive example pairs *(b,s)* where *b* is a phrase-marker and *s* is the corresponding surface string.

The five transformational grammar restrictions suggested by Wexler and Culicover are: the Binary Principle, the Freezing Principle, the Raising Principle, the Principle of No Bottom Context and the Principle of the Transparency of Untransformable Base Structures (see (Wexler et al. 1980) section 4.2). The authors have proven that the restrictions are necessary for making the grammars learnable. They have also discussed the linguistic plausibility of the restrictions and they have concluded that first one probably is linguistically plausible, the second one is plausible and the third one probably not. The plausibility of the fourth and the fifth restriction have not been discussed.

(Adriaans 1992) describes a rule-based learning method for learning categorial grammar. The learning method has access to an example sentence generator and an oracle which evaluates arbitrary sentences. Adriaans has put restrictions on the generator and the grammatical rules that should be learned. The generator produces sentences in an order based on their complexity and the complexity of the rules is restricted by the assumed complexity of the complete grammar.

Adriaans's algorithm contains four steps. In the first step *s* sentences are generated where *s* is dependent on the expected complexity of the grammar. After that all possible rules explaining the sentences will be extracted. In the third step these rules will be combined and simplified. Finally the validity of the rules will be tested by using them for generating sentences and supplying the sentences to the oracle. Adriaans has been able to prove that his learning system can effectively learn context-free languages when the restrictions on the generator processing and target rule complexity are satisfied.

(Gildea et al. 1996) describes an interesting learning experiment in which an induction algorithm is applied to problem with and without extra domain constraints. The problem was deriving phonological rules for phenomena as flapping, r-deletion and word-final stop devoicing. The rules were represented as deterministic transducers. The learning algorithm without the constraints failed to learn the rules. However equipped with the three constraints Faithfulness (usually an underlying segment will be the same as the corresponding surface segment), Community (segments with similar features will act the same) and Context (rules need context information) the algorithm was able to learn the target rules from positive information only. With this experiment the authors have shown that computational learning methods applied to natural language can benefit from being equipped with basic linguistic knowledge.

The three studies described in this section have in common that they attempt to tackle the learnability problems of complex domains by adding restrictions to the domain. By showing that these restrictions are linguistically plausible they have contributed to reduce the gap between mathematical learning theory and observations in child language acquisition.