Effects of Developmental Heuristics for Natural Language Learning

Steven A. Engels

April 1, 2003

A thesis presented to the University of Waterloo

in fulfillment of the thesis requirement

for the degree of

Master of Mathematics

School of Computer Science

University of Waterloo

Waterloo, Ontario

Abstract

Machine learning in natural language has been a widely pursued area of research. However, few learning techniques model themselves after human learning, despite the nature of the task being closely connected to human cognition. In particular, the idea of learning language in stages is a common approach for human learning, as can be seen in practice in the education system and in research on language acquisition. However, staged learning for natural language is an area largely overlooked by machine learning researchers.

This thesis proposes a developmental learning heuristic for natural language models, to evaluate its performance on natural language tasks. The heuristic simulates human learning stages by training on child, teenage and adult text, provided by the British National Corpus. The three staged learning techniques that are proposed take advantage of these stages to create a single developed Hidden Markov Model. This model is then applied to the task of part-of-speech tagging to observe the effects of development on language learning.

Acknowledgements

Seeing my name alone on the title page seems wrong somehow. I may have put in some work and some writing here and there, but I would never have completed this thesis without the support and guidance of the people around me over these past three years.

I would first like to thank my colleagues in the LPAIG/Statistical Inferencing and Learning Group. I'd like to thank Relu Patrascu for his words of wisdom, Fuchun Peng for his NLP help, Alan Angold for his years of experience and Alfred Renaud for his formatting help and his limitless *joie de vivre*. In particular, I'd like to thank Paul Nijjar, for his brutally honest editing skills, his encouraging words, and for taking as long to finish as I did. Paul is a living reminder that even really smart people sometimes take three years to finish.

I would also like to thank the friends who encouraged me to keep going in this academic marathon of mine. Whether it was to cheer me on at the start of the race, to give me words of encouragement when I couldn't see the finish line, or to push me along at the end, I am grateful to all of you. Many thanks to Martha Roberts for her psychology expertise, to the friends who were patient when I disappeared for months, and especially to those friends who saved my life by giving me food and shelter over those last few homeless months at Waterloo.

I wish to thank my family as well, having faith in me, for learning eventually not to ask me how my thesis is going, and for supplying me with support, both financially and spiritually. They say that you can choose your friends but not your family, but I've been lucky in both respects.

Thanks also go to the University of Waterloo for being my home for the past eight years, and the University of Toronto for offering me a new home for the years to come. I would also like to thank my readers Charlie Clarke and Chrysanne DiMarco. Chrysanne in particular has been a friend and mentor in more ways than I would have expected, which has been a comfort during stressful times.

Finally, I would like to thank my supervisor, Dale Schuurmans. I only made it through my graduate experience though Dale's infinite patience, his boundless energy, his sense of humour and his inexplicable interest in my well-being. My graduate school experience and my life are both greatly enriched because of Dale, and I would never have reached this point without the things he did for me.

Contents

List of Tables

List of Figures

1 Introduction

The study of natural language has been an essential branch of artificial intelligence, in part due to the importance of language to how humans function and interact in their daily lives. Much research into natural language attempts to simulate language tasks that humans perform easily, such as checking grammar and spelling, question answering, document summarization and extracting information from text, to name a few[5, 24]. Having machines analyze, understand and generate human language has been a key goal in the quest to create realistic, artificially intelligent software.

However, despite the interest in natural language research and the advances made over the years, machines have not been able to produce a comprehensive model of human grammar. Humans are still able to perform natural language tasks easily, but have trouble instilling this language knowledge in the machines that they design and program. Few natural language researchers would consider this fact surprising, and most would readily admit that the techniques currently employed could not reproduce the complex cognitive mechanisms used by humans.

As machines are attempting to achieve human performance in language tasks, it seems natural to suppose that human cognition could inspire advancement in the current state of the art. Machine learning in particular could benefit from human heuristics. Machines generally use brute-force techniques to learn a language model from data, a process that bears little resemblance to the gradual, incremental way humans learn language. It is reasonable to hypothesize that if humans are taught language knowledge such as grammar and vocabulary in stages, the performance of computational language systems might also improve through the integration of staged learning techniques with their current learning algorithms.

In this thesis, I propose a heuristic that attempts to incorporate one of the ideas behind human language development into a learning algorithm for automatic text tagging. The premise of the heuristic is that a machine should train initially on basic text, and then build on its knowledge with training from texts that are more and more advanced. By mimicking how humans learn in stages by studying texts with increasingly sophisticated reading levels, I investigate possible implementations of developmental learning, and what effects they have on language learning tasks.

2 Historical Background

The work in this thesis is an extension to research conducted in two areas: cognitive language development and machine learning. As both areas have been explored quite thoroughly, the hope is that a combined approach will integrate the strengths of these two well-known areas of research.

The following sections describe the research that has been done in the areas of human language development and machine learning of natural language, in particular in how these areas would apply to developmental machine learning techniques.

2.1 Human Language Acquisition

The study of human language has always stimulated great interest among psychologists, due to the difficulty in describing this unique, creative and intelligent behaviour with any concise theory. Over the course of time, the properties of language have been attributed to various theories, such as innate cognitive knowledge[29], constructivist theories[29], connectionist models[7, 14, 34] or other cognitive learning techniques.

Despite this wealth of psychological research, only a small subset is applicable to the focus of this thesis. Much of the research on language learning discusses reading skills such as word recognition and semantic understanding, whereas the research in this thesis is mostly concerned with grammar and vocabulary learning. These are areas that are more interested in the structure of language, instead of the mapping of symbols to words, and words to meaning.

Human learning studies have found that humans learn in stages, with the greatest portion of this learning taking place during childhood[17]. Children learn grammar quickly and unconsciously at a very young age, while vocabulary knowledge is built up continuously over time, with the greatest accumulation of words in the preschool and primary school years[27].

The fact that humans learn language so early in life makes it difficult to determine whether teaching language in stages is a proper technique, or whether it is dependent on cranial or cognitive development. Chomsky and Piaget have argued that the learning is based on innate grammar ability and cognitive construction, respectively[29]. In both cases, the implication is that learning stages exist as a result of the overall development of the human. However, staged learning is used in instructing adults as well, in particular for adults learning the grammar and vocabulary of a second language[17, 27]. People who are learning English as a second language (ESL) tend to begin

by learning phrases that are shorter and less complex, and then working up to more difficult structure and content. In addition, ESL learners are forced to learn more structured grammar and vocabulary rules, since the language knowledge was not assimilated unconsciously during their childhood. This gives credence to the idea that staged learning is beneficial for all language learners, not just those in early stages of mental development[16].

2.2 Machine Learning Techniques

Several techniques exist at the moment for learning language models for human grammar and vocabulary. Historically, this is still a recent phenomenon. Until the 1980s, computational linguistics research was involved in manually generating grammar and knowledge bases for natural language processing (NLP). However, over approximately the past ten years there has been a shift from this "hand-crafted" grammar knowledge to more automated, corpus-based learning[5]. This has been motivated by the success of automated techniques for natural language tasks, and the improved power of machines, making it possible to perform these computationally demanding techniques.

Despite the fact that the automated language learning field is still developing, there are many techniques that have been well-explored over the past decade. Each of these techniques features a learning component to a different extent, depending on the degree to which the learning element is considered the enhancement or the core concept of the technique. On the one hand, some research applies learning techniques to existing parsing or rule-based systems[8, 28], while on the other hand, others apply stochastic models to capturing the elements of language knowledge[14, 34].

2.2.1 Applying Learning to Manually-Generated Systems

The benefits of applying learning techniques to manually-generated computational linguistic systems can be seen in work done by researchers such as Brill, Charniak, Pereira and Schabes, to name a few[4, 8, 28]. These researchers constructed systems with a significant component of hand-crafted rules or structure. These systems then learned to weight the importance of these hand-crafted components, based on natural language training data.

Brill invented the transformation-based tagger, which labels all of the words in a sentence with the most likely part-of-speech(POS) tag, and then uses transformation rules to decide whether to change a word's tag based on context[4]. These rules are manually generated, but the order in which they

are applied is learned, until the tagging error rate is minimized (more information about POS tagging is discussed in Section 3.2.1). This is similar to the decision tree technique, except that decision trees split the number of applicable transformation rules in half with each decision, making transformation-based taggers strictly more powerful[4, 24].

Applying learning methods to context-free grammars also yields successful techniques. Context-free grammars take advantage of the hierarchical nature of language, to recursively decompose a sentence into smaller grammatical units, down to the word level. The final representation for this hierarchical structure is a parse tree. For example, Charniak created a natural language parser that learns a probabilistic context-free grammar (PCFG) by training on the Penn Treebank, a collection of text that provides parse trees for each sentence in the corpus[8, 9]. This technique collects frequency statistics on a sentence's parse tree components, based on each component's head (most important lexical item), its type (grammatical tag) and the type of its parent. These statistics are then used to find the best parse for a sentence by calculating the cumulative likelihood of each possible parse, and selecting the highest likelihood as the correct parse. Although his approach uses explicit grammar rules that reduce ambiguity, this causes sparse data problems, which Charniak attempts to improve by using word "clusters" (general grouping of similar words) in the algorithm instead of the word itself. Michael Collins is also known for his work on creating an accurate PCFG parser[12], which is similar to Charniak's, without the adherence to using strict grammar rules or considering the parent in the calculation of statistics[8]. Collins' technique finds sets of base phrases (NP fragments) and phrase dependencies, and uses the probabilities attached to each of these to calculate the most likely parse, correcting for phrase attachment distance and sparse data problems. This technique achieves performance of 88.1% recall and 88.6% precision on parsing tasks[12]. Finally, Pereira and Shabes have an approach that takes a grammar of all possible decomposition rules in Chomsky Normal Form, and learns which rules are most compatible with tagged training data that has some constituents bracketed within it[28]. Once learning is complete, the model consists of weighted decomposition rules that are used to produce the most likely parses of a sentence.

The challenges inherent in each of these techniques are those of overcoming sparse data problems and incorporating local context. Sparse data problems occur not only because the task of creating parse tree data is more taxing on human taggers, thus producing smaller quantities of training data, but also because the lexical decomposition rules used by these approaches

involve several component parts. As a result, these complex decomposition rules can occur infrequently in training data, especially after smoothing. These techniques also lose local context information, since decomposing a sentence fragment into parts isolates the analysis of each part, preventing the information contained in on part to improve the analysis of the other. There are enhancements to each technique to improve this, but these problems are less prevalent in the techniques of the following section.

2.2.2 Stochastic Models for Natural Language

An alternate approach to language learning has been to use stochastic models for sequences, and apply them to natural language problems. This contrasts with the approaches described above in that the core idea is the stochastic model, instead of the traditional computational linguistic concepts. The main examples of such research involve Markov models or neural networks for tasks such as grammar learning and part-of-speech tagging.

Neural networks have been applied to grammar tasks for just over a decade. They are composed of a layer of input nodes, a layer of output nodes, and an optional hidden layer of nodes in between. In the case where a hidden layer exists, each input node has a forward connection to each hidden node, and each hidden node is in turn connected to each output node. Observations are fed into the input nodes, and based on the weights on each of the connections, these combined inputs activate certain hidden nodes, which in turn activate certain output nodes that denote grammatical information (part-of-speech for particular words, text category, for example). This method works copes well sparse training data, performing language tasks as well as other models[14, 34].

The other stochastic model that is prevalent in the natural language community is the Markov model. Andrei Markov invented Markov models in 1913, for modeling linear sequences of data, where the value of each element in the sequence depends on the ones before it[25]. They appear in several forms in language research, such as the n-gram model, for example.

2.3 Developmental Learning and Natural Language

Some work has been done in the past, relating the developmental learning approaches of Section 2.1 with Section 2.2. One example is through the use of recurrent neural networks, which feed the output of a neural network back into the inputs, to retain information about the network's previous state. Elman[14] showed that a recurrent neural network that is more successful at tagging when trained first on sentences with certain English complexities removed, and then gradually introducing on the full sentences into the training. However, contrary experimental evidence has been produced by Rohde and Plaut[32], and Chalup suggests that this improvement is caused by the developing neural architecture instead of the staged learning algorithm[7].

There are also developmental learning ideas that have been proposed in areas outside of natural language. One is suggested by Prince[30] in his paper on developing theories in artificial intelligence systems. In it, he justifies the use of a development approach in building theories, suggesting that development in natural language is a likely extension of this concept. Drescher[13] also proposes a developmental schema that is based on Piaget's theories of staged learning, which is referenced often by developmental research.

In general, research does exist that relates to developmental algorithms for natural language learning. However, this research is sparse and weakly connected to the problem addressed in this thesis, which indicates the possibility for further discoveries in this area of study.

2.3.1 Scientific Motivation

In practice, stochastic language models such as n-gram models are widely used in language and speech systems for their simplicity and strength. However, despite the general acceptance of these models in the natural language community, researchers often attempt to incorporate elements of human cognition to the task as well, in an attempt to improve their performance. As Brill[4] puts it:

> There is a continuing push among members of the speech recognition community to remedy the weaknesses of linguistically-impoverished n-gram language models. It is widely believed that incorporating linguistic concepts can lead to more accurate language models and more accurate speech recognizers[4].

This motivation for increased linguistic influence has led researchers to produce models that trigger on salient features of words in text[2, 26], hand-

crafted grammar rules[8, 9], and other techniques based on cognition and psycholinguistics. As stated by Allen[1]:

> ...the technological goal cannot be realized without using sophisticated underlying theories on the level of those being developed by linguists, psycholinguists, and philosophers. On the other hand, the present state of knowledge about natural language processing is so preliminary that attempting to build a cognitively correct model is not feasible. Rather, we are still attempting to construct any model that appears to work.

This push for techniques that employ linguistic theories to produce 'systems that work' is the motivation behind many heuristics and new improvements to stochastic models for language learning. For example, heuristics that trigger on linguistic features reflect the human tendency to train on verb endings, capitalization and other linguistic features when learning[16]. Similarly, the hand-crafted grammar rules reflect the intuitions put forward by Chomsky that the understanding and generation of human language can be modeled by a generative rule structure[29].

The motivation behind this thesis is the exploration of another aspect of human language learning, namely that of incremental language acquisition. This concept is also based on human learning theory, as described in Section 2.1. However, little modeling has been done in this area to date, save for that of Elman[14]. Given the importance that humans place on language learning in the education system, the lack of research into this area for natural language systems either suggests that there is potential for new discoveries in the application of incremental learning, or that the area is being ignored as having no research value. This mail goal of this thesis is to investigate this issue, to determine whether developmental heuristics can imbue a language model with a more human-like understanding of language[30], or whether developmental heuristics are unable to capture any meaningful human intuition[32]

3 Technical Background

The research in this thesis attempts to discover whether the performance of learned language models can be improved through the use of human learning stages. In order to do this, experiments were designed to illustrate the effects of staged learning techniques on a particular language task. By scoring the performance of staged models against a model that does not use a developmental heuristic, the experiments will confirm or refute the hypothesis that these heuristics have a positive effect on existing learning algorithms.

The creation of these experiments involved four design decisions:

1. The staging of the data
2. The natural language task
3. The language model
4. The learning mechanism

The following sections describe the specific implementation of these four components, the factors that influenced the design decisions, and the impact that each of these decisions will have on the experiments.

3.1 Staging the Data

It is impossible to attempt to train a machine to learn language with exactly the same incremental stages that humans use. Not only is the task of dividing human learning accurately into stages entirely subjective, but also the divisions would be either be too small to implement practically, or a crude estimate.

In order to model the learning stages for these experiments, a more approximate approach was used. A language model was chosen that could be trained incrementally on a series of data sources. The learning stages were achieved by obtaining data from a large corpus of text, where individual pieces of text within the corpus are tagged with a human reading level. These pieces of text are then grouped according to reading level, where each grouping represents the training data for a stage in the incremental learning model.

The model is then trained on the individual stages, started from the lowest reading level and building on that with training on increasingly advanced reading levels. The result is a learning model whose training data becomes increasingly advanced at each stage, much in the same way a human's reading level would increase over time[17, 27].

The text was obtained from the British National Corpus (BNC), a tagged text corpus of written works from various English sources. Each text is segmented into sentences, within which each word is automatically assigned one of 61 grammatical class (part-of-speech) tags. These grammatical tags distinguish the main categories found in most English grammars, such as nouns, verbs, adjectives, etc., and are similar to those used in the LOB Corpus[22, 23]. Certain ambiguous words are assigned combined tags and certain multi-word phrases are assigned a single tag, but in general each word has a single corresponding tag.

The reason for choosing the British National Corpus is because each of the 4,214 texts in the British National Corpus is marked with one of three reading levels: child, teenage and adult. This makes it ideal for staged learning. By separating the corpus according to these reading-level labels, three training sets were extracted from the corpus, each containing about one-two million words.

3.2 Language Task

Language tasks are practical applications of a language model, usually used for testing purposes to evaluate the quality of the model. For this experiment, as implied above, the language task is part-of-speech tagging. The reasons for this are threefold:

1. The task is data-dependent. For example, tasks such as sentence parsing require parsed sentences to train on to develop a language model. As the main focus of this research is the impact of staged training data on learning algorithms and the BNC provides staged datasets, the experiments are constrained to tasks that operate on part-of-speech data.

2. Tagging is a useful problem. It can be a useful intermediate step for parsing and understanding a sentence, or can be used directly in information extraction applications[3, 2, 26]. By knowing a word's part-of-speech, one can disambiguate its meaning, and thus identify the role that the word plays in the sentence.

3. Part-of-speech tagging has a relatively limited scope. Although it is not as complete as parsing when describing syntactic structure, tagging is a much easier task to solve than parsing (success rates are close to 97% for the most successful approaches[24]). As a result, more experiments may be conducted, since experiment designs for this task are less complex and time-consuming than for more elaborate tasks.

3.2.1 Background on Part-of-Speech Tagging

Part-of-speech (POS) tags are labels assigned to words, denoting that word's syntactic or semantic category. Examples of tags are noun, verb, adjective, third person singular present verb, and so on. The task of part-of-speech tagging is thus the task of assigning a part-of-speech tag to each word in a sentence. For instance, the sentence fragment, *Mary had a little lamb* could be tagged either as:

Mary-NP0 had-VHD a-AT0 little-AJ0 lamb-NN1

or as:

Mary-NP0 had-VHD a-AT0 little-DT0 lamb-NN1

Despite being different, both of these taggings are valid. This is because the word "little" has more than one syntactic category, and can be thought

of either as an adjective (AJ0) or a determiner (DT0). The first tagging is more likely, and corresponds to the nursery-rhyme meaning of Mary owning a small baby sheep. The second tagging has a very different meaning, and would be appropriate if Mary was a restaurant critic and chose to sample a small portion of lamb chop. As one can see, the tags associated with each word convey a great deal about the usage of the word, and can alter the meaning of the sentence as a whole.

(A list of part-of-speech tags and their associated meanings can be found in Appendix A.)

3.3 Language Models

A language model is basically a computational description of a language. The model stores the representation of the language, and how the language is to be interpreted, understood, and generated. As a result, the language model for a natural language experiment has a large bearing on the results of that experiment.

The choice of a model will determine the kind of structures that can be captured within human language. For example, probabilistic context-free grammars (PCFGs) are good at creating a hierarchical parse of a sentence, consequently producing several valid interpretations of a sentence on the fly and providing an intuitive representation of the structure of human language[21]. However, they have trouble in other areas, such as incorporating local lexical context when tagging a word, coverage given sparse training data and the cost in creating, maintaining, and running a PCFG tagger.

For this experiment, a Hidden Markov Model (HMM) was chosen as the language model for the part-of-speech tagger. This is because it is a powerful model, with a high success rate and strong statistical grounding[24, 31]. The theory behind Hidden Markov Models is also easy to implement and understand, which helps in developing, manipulating, and analyzing various experimental models. Finally, whereas other models require both positive and negative training examples, HMMs benefit from being trained on large amounts of positive tagged data in their learning algorithms, which is what the British National Corpus provides[22, 23].

It should be noted that another key reason for choosing this language model was because of its ability to integrate a staged learning algorithm. Since HMMs have a learning algorithm that is elegant yet straightforward to understand, it was easier to incorporate the staged training data into its learning algorithm, thus making it a superior choice.

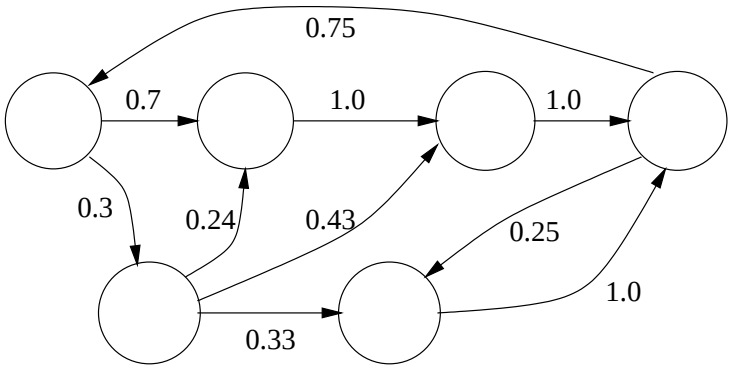

Figure 1: Example of Markov Model Structure

3.3.1 Background on Hidden Markov Models

A Markov model is a probabilistic tool, used to model the probabilities of linear sequences of events[24, 26, 31]. These sequences are often taken from real-life sources, such as words from a text[20, 26], protein sequences in a DNA strand[19], or even a sequence of coin tosses[24]. The models are often illustrated as non-deterministic finite state automata, with probabilities attached to the arcs, as demonstrated in Figure 1.

The Hidden Markov Model is a special variation of Markov models that is used in this research. HMMs are different from the Markov Models described above in that the observed events are considered to be emissions from a sequence of unobserved underlying states. The HMM describes the transitions within the set of underlying states, and the observations emitted from each underlying state[31].

The result of using this model is that given an observed sequence of words from a sentence, an HMM can provide the possible underlying state sequences that could generate the observed words. Since each state has some syntactic or semantic meaning, by analyzing and deciding on the most likely sequence, one can extract information about each word or about the sentence as a whole.

For the purpose of this research, the states are associated with the 91 parts-of-speech in the BNC. Applying the HMM to a sentence will then label the words with the most likely POS tags, as determined by the model.

3.3.2 Formal Definition of HMM

As stated above, the HMM is a stochastic model that describes the probabilities of the transitions between states and the observed emissions from these states. The sequence of underlying states is represented mathematically as a linear sequence of random variables $X = (X_1, X_2, \ldots, X_T)$, where the value of the random variables at each instant in time is one of the N discrete state values $S = (s_1, s_2, \ldots, s_N)$. These states in turn emit one of the M possible observations $K = (k_1, k_2, \ldots, k_M)$ [2, 10, 24].

There are three sets of probabilities needed to describe an HMM:

1. A transition matrix A describes the probability of transitions between any two states, where $a_{ij} = P(X_{t+1} = s_j | X_t = s_i)$
2. Emission probabilities B describe the probability of observations from each state, where $b_i(k) = P(O_t = k_j | X_t = s_i)$
3. Initial state probabilities Π describe the probability of the sequence beginning with a particular state, where $\pi_i = P(X_1 = s_i)$

This model is thus specified by the five-tuple (S, K, Π, A, B), whose notation is described below:

$S = \{s_1, \ldots, s_N\}$ (finite set of possible states)

$V = \{v_1, \ldots, v_M\}$ (finite set of possible observations)

$\Pi = \{\pi_i\}$, $s_i \in S$ (initial state probabilities)

$A = \{a_{ij}\}$, $s_i, s_j \in S$ (state transition probabilities)

$B = \{b_i(k)\}$, $s_i \in S, v_k \in V$ (observed emission probabilities)

As changes in a model's parameters are usually reflected in the latter three probability measures, the model may also be expressed with the compact notations $\lambda = (A, B, \Pi)$. These probability descriptions assume the following notation:

$\mathbf{T}$ = length of the sequence of observations

$\mathbf{N}$ = number of states (from training set)

$\mathbf{M}$ = number of possible observations (from training set)

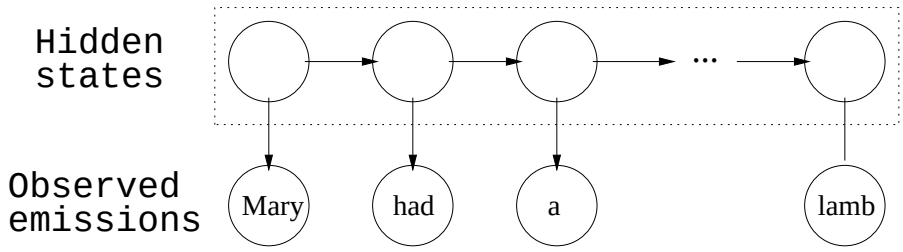

Figure 2: Example of HMM Structure for a Simple Sentence

X_t = random variable denoting the state at time t (state variable)

O_t = random variable denoting the observation at time t (output variable)

S_t = value of state variable at time t

V_t = value of observed emission at time t

As stated earlier, the Hidden Markov Model assumes that one only needs to know the value of the current random value to predict future random variables. Stated more formally, we are assuming that the model is based on the two Markov properties[24]:

Limited Horizon: Each state in a sequence only depends on the previous state, and not on any of the states previous to that.

$$P(X_{t+1} = s_k|X_1, \ldots, X_t) = P(X_{t+1} = s_k|X_t) \tag{1}$$

Time Invariance: The transition probabilities between states do not depend on their position in the sequence.

$$P(X_{t+1} = s_k|X_t) = P(X_2 = s_k|X_1) \tag{2}$$

These properties establish the basis for using the state transition and emission probabilities to calculate the probability of a sequence of states, as described above.

As a result, a sentence that has been fully tagged by an HMM is often represented as a chain of states, as shown in Figure 2.

3.3.3 Training the HMM

In order to create an HMM that can accurately model language, it must first be trained on a large corpus of tagged sentences. By providing the model with examples of tagged sentences, it is able to capture the nature of the language being modeled by learning the frequency of state transitions and emissions.

The main probability components of the HMM are the transition matrix, the emission probabilities, and the initial state probabilities[31]. Therefore, in order to extract a model from tagged training data, the training algorithm should use the frequency counts of the various tags to estimate these probabilities.

The HMM probabilities can be estimated using the following calculations, where $C(\ldots)$ is the count, or number of times the event in parentheses occurs:

$$\begin{aligned} a_{ij} &= P(X_{t+1} = s_j | X_t = S_i) \qquad 1 \leq i, j \leq N \\ &= \frac{C(X_t = s_i, X_{t+1} = s_j)}{C(X_t = s_i)} \end{aligned} \tag{3}$$

$$\begin{aligned} b_i(k) &= P(O_t = v_k | X_t = s_i) \qquad 1 \leq i \leq N, 1 \leq k \leq M \\ &= \frac{C(X_t = s_i, O_t = v_k)}{C(X_t = s_i)} \end{aligned} \tag{4}$$

$$\begin{aligned} \pi_i &= P(X_t = s_i) \qquad 1 \leq i \leq N \\ &= C(X_t = s_i) \end{aligned} \tag{5}$$

One of the implications of this training procedure is that the model that the training produces might overfit the data, especially if the data originates from a homogeneous source[24]. For instance, a model that is trained on text from the Wall Street Journal would perform well in tagging an article from the same source, but would not perform as well if presented with a chapter from a Harry Potter book. For this reason, the training data has been extracted from several different sources, so that the only common element for all the text in a training set is the reading level associated with it.

3.3.4 Smoothing the Model

After training, certain transition and emission probabilities will have a zero probability value, since they have never been observed in the data. Smoothing a model eliminates these zero probabilities by giving them a small value,

and adjusting the non-zero probabilities to maintain the same probability weights, relative to each other. An 'unknown' entry is also added to the list of observed emissions, with each part-of-speech tag having an equally likely probability of being assigned to the 'unknown' emission.

Although it may seem foolish to assign a probability to certain ungrammatical transitions or unobserved emissions, the intuition behind smoothing is illustrated in the following passage from *Through the Looking-Glass*, by Lewis Carroll:

> 'Twas brillig, and the slithy toves
> Did gyre and gimble in the wabe:
> All mimsy were the borogroves,
> And the mome raths outgrabe.
>
> *Jabberwocky*, [6]

This might appear to be an ungrammatical piece of text in the eyes of a language model, but most humans would be able to understand the structure of the passage, and be able to decipher some meaning as well. As Alice puts it, "...somebody killed *something*, that's clear at any rate"[6]. However, this meaning will be lost to the language model, if it fails to allow for unobserved emissions such as "gyre" and "gimble", or unseen transitions such as "mimsy were" (since adjective/be-verb transitions might only appear in poetic text).

These experiments use Katz smoothing[18]. In layman's terms, this smoothing technique steals from the rich and gives to the poor, where the 'poor' are the transitions and emissions with zero probability, and the 'rich' are the non-zero transitions and emissions. This technique was chosen because it was easy to implement, but didn't skew the probability distribution significantly. Alternate smoothing techniques that were considered are:

Additive Smoothing : The simplest smoothing technique, which adds one to the number of counts for each transition and emission[11].

Good-Turing Smoothing : Taking transition probabilities as an example, this smoothing method adjusts every transition that occurs r times by pretending that it occurs r^* times, where $r^* = (r+1) \cdot \frac{n_{r+1}}{n_r}$ and n_r is the number of transitions that occur r times. This smoothing method has to be smoothed to adjust for values of $n_r = 0$, but it is still considered to be the best smoothing technique available for natural language[11].

Jelinek-Mercer Smoothing : Instead of boosting all zero probabilities equally, the Jelinek-Mercer method boosts a transition more if the start state for that transition is generally more likely than other states. This is based on the idea that if a state such as AT0 (the, at, a) occurs more often than the state ORD (first, sixth, 23rd), then unseen transitions starting with AT0 (the borogroves) are more likely than unseen transitions starting with ORD (seventh borogroves)[11].

3.3.5 Extracting the State Sequence

Once the HMM has been fully trained, there is still the task of inferring the most likely state sequence, given the model and the observations in the sentence. To determine the most likely underlying state sequence $S_{1,T}$, we find the states that would maximize the maximum *a posteriori* probability of the observations, as shown in the following equation:

$$
\begin{aligned}
S_{1,T} &= \underset{S_{1,T}}{\operatorname{argmax}} \; [\, P(X = S_{1,T} | O = V_{1,T}) \,] && (6)\\
&= \underset{S_{1,T}}{\operatorname{argmax}} \; \left[\frac{P(O = V_{1,T} | X = S_{1,T}) P(X = S_{1,T})}{P(O = V_{1,T})} \right] && (7)\\
&= \underset{S_{1,T}}{\operatorname{argmax}} \; [\, P(O = V_{1,T} | X = S_{1,T}) P(X = S_{1,T}) \,] && (8)\\
&= \underset{S_{1,T}}{\operatorname{argmax}} \; [\, P(X_1 = S_1) \cdot P(O_1 = V_1 | X_1 = S_1) \\
&\qquad \cdot P(X_2 = S_2 | X_1 = S_1) \cdot P(O_2 = V_2 | X_2 = S_2) \cdot \\
&\qquad \ldots && (9)\\
&\qquad \cdot P(X_T = S_T | X_{T-1} = S_{T-1}) \cdot P(O_T = V_T | X_T = S_T) \,] && (10)\\
&= \underset{S_{1,T}}{\operatorname{argmax}} \; [\, \pi_{S_1} \cdot b_1(k_{V_1}) \cdot a_{S_1 S_2} \cdot b_{S_2}(k_{S_2}) \cdot \\
&\qquad \ldots \cdot a_{S_{T-1} S_T} \cdot b_{S_T}(k_{S_T}) \,] && (11)
\end{aligned}
$$

Bayes' Rule is used to restate the equation in Step (7), and the Markov Properties are used to separate the massive joint probability in Step (8) into the chain probability in Step (10). The final step involves substituting the HMM parameters for the probability terms in Step (10).

Since there are T parameters to evaluate, and N values for each parameter, the Markov Properties must be used again to avoid having to evaluate N^T possible state combinations[24]. The calculation of the best state sequence $S_{1,T}$ is done through the use of the Viterbi algorithm, which uses dynamic programming to reduce the complexity of the calculation to $T \cdot N^2$.

The Viterbi algorithm uses induction to find the most probable path of states ending in state X_{t+1}, based on the best paths that lead up to state X_t. This is done by defining a new quantity α for a model λ:

$$\alpha_t(i) = \max_{S_1,S_2,\ldots,S_{t-1}} P(S_1 \cdot S_2 \cdots S_t = i, O_1 \cdot O_2 \cdots O_t|\lambda) \tag{12}$$

This equation expresses how the Viterbi algorithm determines the most likely sequence of states for a given observation sequence. It initializes the procedure with $\pi_i b_i$ probabilities for the initial states, then repeatedly increments the time step t, inductively calculating the α_i probabilities at each time step. The algorithm halts when it reaches $t = T$ (the end of the sequence), and the state sequence that corresponds to the highest $\alpha_T(i)$ value is considered the 'best' state sequence.

4 Experiment Design and Implementation

Once the language task, language model, and training data are chosen, one can begin designing experiments that test the hypothesis that incremental learning will improve tagging performance. This is not a simple task, for the nature of development in humans is still unclear. Do humans employ developmental learning because it is the best way to build language knowledge, or because the human brain has learning limits when growing up? If development is the correct approach, then what is the nature of that development, and can it be represented in a computational model?

4.1 Developmental Hypothesis

Despite the uncertainties that surround the use of stages in language learning, there is evidence that developmental approaches in machine learning might work, from how they correspond to human learning (see Section 2.1. For example, staged learning is used for language learners of all ages, not just for those in early stages of cognitive development. This implies that developmental learning is not specific to children, and can therefore be used on adult humans and machines of fairly fixed architecture.

If one assumes that developmental learning of some kind would be useful, then there is still the issue of what aspect of the language model should be developed. In the case of these experiments, the effects of development are being tested through the creation of a Hidden Markov Model whose parameters are a product of training on different reading levels. By using various staged training methods and testing these methods by applying the resulting model to a language task such as POS tagging, the experiments will hopefully show the effects of stages on learning syntactic structure.

4.2 General Methodology

To test the hypothesis that incremental learning may be applied to natural language learning, the experiments evaluate the performance of part-of-speech tagging with regular brute-force training against a series of proposed incremental training techniques. These incremental techniques, described in more detail later, simulate language development by developing an increasingly-complex language model from the staged datasets of the BNC. If the developmental heuristics are able to improve the performance of the tagging task, then it would support the hypothesis that developmental heuristics would be useful to pursue as an enhancement to language

learning systems. If the results show that these heuristics have a negative or marginal effect on the performance, then the experiments will indicate that this developmental intuition does not transfer over to this particular language model and task.

The general method for each experiment is as follows. The text data is read in from the training file and stored as a series of sentences. An untrained HMM is then created, and the probability transitions $\lambda = (A, B, \Pi)$ are populated with the observed transitions in the training text. Depending on the nature of the heuristic, either three models are trained on each of the child, teenage and adult texts before being combined at the end, or a model is trained on the child text, the teenage model, and the adult text successively to create a fully learned model.

Not only are the various developed models compared with a regularly-trained model, they are also compared internally by varying the proportions of child, teenage and adult training data used to create each model, to see what ratio of the three training sources produces the best overall tagging performance.

4.3 HMM Design

The Hidden Markov Model implements the theory described in Section 3.3. It is a basic bigram model, meaning that every state or emission is only dependent on the previous state.

The training of the model is performed on the following training sets:

Dataset		**Corpus Size**
Child Data	:	1,086,316 words in 42 texts
Teen Data	:	2,084,157 words in 77 texts
Adult Data	:	2,924,389 words in 100 texts

These sets are composed of several full texts, each of which has been categorized by hand by members of the BNC organization, based on the level of complexity of the text. Observations on the nature of the each text show that the children's text is composed of shorter, less complex sentences. This gives rise to a sparser transition matrix and smaller observed vocabulary than the teen, which in turn is less complex than the adult model.

Individual child, teenage and adult models are trained on 90% of the total data, with 10% held out to test performance later. When combining models, the counts are scaled to make the proportions of child, teen and

adult training approximately equal. In addition to the various observed words, an 'unknown' word category is created to anticipate words that occur in the test set but not in the training set. Finally, the parameters of the trained models are smoothed to ensure that all transitions and emissions have a minimal non-zero probability.

The performance measure for the experiments is the accuracy with which the HMM tags unlabelled sentence data with POS tags.

4.4 Software Design

The software is programmed in C, on a Linux platform. The design of the software is divided into four modules:

Tokenizer : Extracts sentences from file, and tokenizes them into sequences of words and POS tags.

Training : Using sentence sequence, train A, B and Π parameters for HMM, as outlined in Section 3.3.3. Also has methods for combining various models for developmental algorithms.

Test : Uses trained models and held-out test sets to test model's accuracy at tagging unlabelled data. Applies Viterbi algorithm (Section 3.3.5) to determine the best state sequence for a sentence of unlabelled words, and compares them with the correct tag sequence provided by the BNC.

HMM Controller : Supervises sentence extraction, HMM training and testing. Chooses developmental algorithm to test, and tests various parameters for each algorithm.

4.5 Staged Learning Methods

When integrating child, teen and adult training into a full, developed model, one has to consider the way these training stages come together in the human brain. For instance, the manner in which one combines different stages would be different if the effects of learning in humans are additive over time, or if the lessons we learn earlier on reinforce the learning that takes place at later stages.

In testing the effects of development, this research investigates a few approaches, including the cumulative and reinforced combinations mentioned above. Not all produced viable results, but the more promising techniques are presented below.

4.5.1 Mixture Model Approach

The mixture model approach trains models for the child, teenage and adult stages individually, and then performs a weighted linear combination of the three to produce a final model. The weights are all non-negative, and the resulting model is scaled down to have the same mean as the adult model. This approach is based on the idea that language data is learned additively at each of the child, teenage and adult stages, where the weights signify the "importance" of that stage in development.

The prediction for this approach was that mixtures of the three models would perform better than any model alone, and that mixtures would perform better if weighted more heavily on earlier reading levels. The reason for this second intuition is that language knowledge that is learned as a child consists of more fundamental language rules and vocabulary, and would therefore be a stronger basis for a final language model than either the teenage or adult stages. The major drawback of this technique is the fact that mixture model approaches imply that learning stages are order-independent, which goes against the intuition of human learning.

To test the performance of this method and the hypothesis mentioned above, the parameters of the three models are combined with various weights, and the result is tested on the held-out data to measure their tagging accuracy.

4.5.2 Balanced Mixture Approach

The balanced mixture approach is similar to the mixture model approach in that it is based on an additive combination. However, it differs in how it handles zero transitions and emissions in combining stages, and how it incorporates the ordered nature of human learning.

This method combines the child and teen stages first by making the resulting model equal to the teen model wherever the child model has zero values. In all other cases this technique performs the mixture model's weighted combination of the two stages. The combined portions of the model are scaled to the same mean value of the teen portion before combination. Once this is complete, the same operation is performed using the child-teen model and the adult model, to produce a final model.

The intuition behind this approach is that the final model is similar to a model trained entirely on adult data, which is the norm for most HMM training. However, the model also incorporates the teenage and child data in smaller amounts wherever these previous models and the adult model

coincide, thus adjusting the adult model to compensate for earlier data stages.

Testing for this technique is done in the same way as the testing for the mixture model approach.

4.5.3 Reinforced Learning Approach

The reinforced learning approach abandons the additive combinations for a multiplicative combination. The intuition here is that earlier learning reinforces the importance of common transitions and emissions by enhancing the effect of future training on these parameters. As a result, any language lessons learned as a teenager are reinforced by the "important" lessons learned as a child.

This development method first builds a model based on the child data, which is smoothed to eliminate non-zero transition and emission probabilities. A model is then built from the teen data, and the parameters of the child model are used as scaling factors for the parameters of the teen model. The parameters of this child-teen model are similarly used as scaling factors when building the adult model. The result is a model that is based on adult data, but reinforced by data found in the teenage and child models.

Weights are also used in these tests, but instead of scalar weights, the weights here are exponential, with the value of these exponential weights being between 0 and 1. The reason for these limits is because the lower limit causes a stage to have no effect on the final model, and exponents higher than one cause the parameters of the final model to diverge wildly. Tests for accuracy are the same as those in the mixture model and balanced mixture approach.

4.6 Implementation

As stated in Section 4.4, the implementation is divided into the tokenizer, trainer, tester and controller. The controller is trivial, since it simply executes the other modules in the order in which they are listed. The three main modules are described in more detail here.

4.6.1 Tokenizer Module

The tokenizer is composed of two parts: the preprocessor and the sentence reader. The preprocessor is actually a Perl script that reads in a list of the texts within the BNC that are labeled with child, teenage and adult reading levels. It then finds the texts for a particular reading level in the BNC directory structure, and reads in the raw BNC corpus data one line at a time. Each line corresponds to a sentence in the text. As the preprocessor reads in a sentence, it removes all the tags that do not correspond to the part-of-speech or language level of the text. The resulting sentences are then stored to be processed by the sentence reader.

The sentence reader takes in the processed sentences, and stores them in a linked data structure, with each node in the link storing the word, the actual part-of-speech and a space for the estimated part-of-speech. Once the sentence structure is complete, it is passed to the trainer module to train the HMM.

4.6.2 Trainer Module

The trainer is responsible for taking in tagged sentences from the tokenizer, and producing the model parameters $\lambda = (A, B, \Pi)$ for the HMM. The transition probabilities A are stored in an array that records the number of transitions from one part-of-speech to another, with each of the array dimensions equaling the number of part-of-speech tags found in the corpus. The emission probabilities B are stored in a dictionary list, where each entry has a word from the text and the frequency of that word's different part-of-speech tags. In order to make the $P(O_t = k_j | X_t = s_i)$ calculations, an array of the total occurrences of each part-of-speech must also be stored along with this dictionary list. Finally, the initial probabilities Π are stored in an array that records the number of occurrences of each part-of-speech in the start position of a text's sentences.

As each a sentence is passed into the module, the initial part-of-speech tag is added to the Π array, and the module iterates through the sequence of nodes, storing transition count information into A and word emission

information into B at each step. Once complete, the sentence structure is freed and the module awaits the next sentence. Once the last sentence has been processed, the HMM parameters are stored in a file for debugging purposes, and the completed model is passed to the testing phase.

4.6.3 Tester Module

Once the language model is complete, the tester takes in sentences from the test dataset through the tokenizer, and creates an NxT matrix, where N is the number of possible part-of-speech tags, and T is the number of words in the current sentence. Each position in the matrix stores a reference to the part-of-speech state that had the highest probability of leading to the current state, and the probability of being in the current state, given that that previous state. The mathematics behind this are described in Section 3.3.5.

Once the Viterbi algorithm has been used to determine the most likely state sequence for the input sentence, this is compared with the actual state sequence to evaluate the accuracy of the language model's estimated tags. The cumulative statistics are the ones presented in the results in Section 5.

4.6.4 Combining Training

As stated earlier, there are three proposed development heuristics for this research: the mixture model, the balanced mixture model and the reinforced learning model. To implement these combinations, the following approaches were used.

Mixture Model: For this combination, separate models are trained on the child, teenage and adult text data, and then the parameters of the three models are combined by multiplying each by a scaling factor, such that when the three scaled portions are added together, the linear combination results in a model of similar size to the original three. The scaling factor for each of the child, teenage and adult components is thus proportional to the weight being given to that component for that test.

The way this is accomplished is to declare a constant integer value k and variable integers k_c, k_t and k_a for a particular set of tests, such that $k_c + k_t + k_a = k$. The components are then multiplied by $\frac{k_c}{k}$ (for the child component), $\frac{k_t}{k}$ (for the teen component) and $\frac{k_a}{k}$ (for the adult component).

Balanced Mixture: This combination is similar to that of the mixture model, except that where one of the components has a zero value for a particular parameter, that component is neglected for the calculation of the new combined parameter. The linear combination takes place as described above, but with only two components instead of three. The same adjustment takes place if two components have zero values, and with two zero components the result for that parameter will have the same value as the parameter for the remaining component.

Reinforced Learning: For this technique, the components are not summed together, but are instead multiplied by each other to create a combined model. The integer variables k_c, k_t and k_a continue to add to a constant integer k, but for the reinforced model, the fractions $\frac{k_c}{k}$, $\frac{k_t}{k}$ and $\frac{k_a}{k}$ are exponent for the parameters in each of the child, teenage and adult components, respectively. Once each component has been raised to its respective exponent, the parameters are multiplied together, and the result is scaled so that the parameters of the combined model have the same overall magnitude as the parameters of the original datasets.

5 Experimental Results

The following tables describe the results of various tests on the held-out corpus data. The results tables are illustrated in Table 5 and can be read as follows:

- Columns : increasing portions of child data, from leftmost column to rightmost column.
- Rows : increasing portions of teen data, from top row to bottom row.
- Diagonals : increasing portions of adult data, from bottom right to top left.

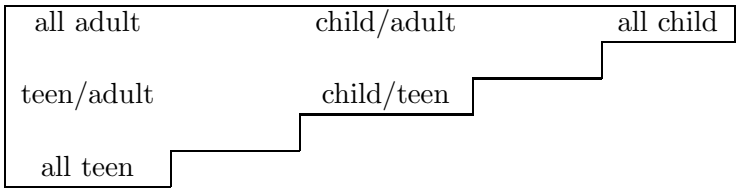

Table 1: Illustration of Relative Portions of Training Data in Result Tables

There are three tables for each learning method, denoting child, teen and adult test data. These tests show how appropriate each resulting model was on the various stages of data by showing the percentage tagging accuracy that the different models had on the test data. The tables are best read as height levels on a contour map, where higher values closer to the top-right corner indicate that the child-weighted model performed best with that input, higher values closer to the bottom-left corner indicate better performance for the teen-weighted model, and higher values in the top-left corner indicate better performance for the adult-weighted model.

The table for adult test data is the most meaningful indicator, since the ideal target for the completed model is adult-level text, although the child and teen test sets illustrate some interesting trends as well.

As a reference, the baseline comparison is against a regular adult-trained model, which scored 93.12% on child text, 92.44% on teenage text, and 94.06% on adult text.

5.1 Mixture Model Results

The following tables illustrate the results of testing the mixture model heuristic on the three test sets.

93.12%	95.48%	95.69%	95.76%	95.83%	95.55%
93.74%	95.59%	95.78%	95.84%	95.81%	
93.83%	95.64%	95.80%	95.60%		
93.85%	95.67%	95.70%			
93.86%	95.58%				
92.72%					

Table 2: Results of Mixture Model Testing on Child Data

92.44%	92.74%	92.78%	92.74%	92.58%	88.37%
94.50%	94.59%	94.59%	94.50%	93.93%	
94.55%	94.63%	94.64%	93.48%		
94.59%	94.69%	94.18%			
94.61%	94.25%				
94.07%					

Table 3: Results of Mixture Model Testing on Teenage Data

94.06%	94.14%	94.10%	93.95%	93.76%	84.82%
94.30%	94.32%	94.24%	94.11%	90.81%	
94.30%	94.30%	94.22%	90.00%		
94.24%	94.25%	91.15%			
94.17%	91.17%				
90.81%					

Table 4: Results of Mixture Model Testing on Adult Data

From these tables, it seems that the models that perform the best on a given set of test data are the ones which are more heavily-weighted with training data from the same source. Models that are trained solely on that source of training data do not appear to have improved performance however, which indicates that certain portions of other training data are needed for improved performance.

5.2 Balanced Mixture Results

The following tables illustrate the results of the balanced mixture tests.

80.98%	95.38%	95.42%	95.47%	95.33%	73.82%
93.72%	95.52%	95.62%	95.62%	73.03%	
93.79%	95.56%	95.66%	72.56%		
93.78%	95.61%	71.96%			
93.68%	71.79%				
69.78%					

Table 5: Results of Balanced Mixture Testing on Child Data

80.04%	92.62%	92.57%	92.42%	92.15%	68.27%
94.50%	94.55%	94.46%	94.30%	71.34%	
94.52%	94.54%	94.44%	71.07%		
94.52%	94.55%	70.86%			
94.48%	70.78%				
70.97%					

Table 6: Results of Balanced Mixture Testing on Teenage Data

80.89%	94.08%	93.99%	93.86%	93.59%	66.42%
94.28%	94.24%	94.13%	93.89%	69.72%	
94.26%	94.19%	93.89%	69.50%		
94.18%	94.09%	69.32%			
94.01%	69.30%				
69.23%					

Table 7: Results of Balanced Mixture Testing on Adult Data

From these tables, it seems that the balanced mixture model has similar results as the mixture model, in that the regions with the best performance in the result tables occur where the training data is weighted more heavily towards the same source as the test data. However, the peaks in this table are not as distinct as in the mixture model tables, so the region of high performance are more diffused than the high-performance regions in the mixture model tables.

5.3 Reinforced Learning Approach

The following tables illustrate the results of testing the reinforced learning model on the three test sets.

95.16%	95.56%	95.79%	95.85%	95.86%
95.18%	95.62%	95.80%	95.86%	
95.24%	95.62%	95.78%		
95.24%	95.58%			
95.37%				

Table 8: Results of Reinforced Learning Testing on Child Data

94.21%	94.24%	94.23%	94.08%	93.87%
94.27%	94.42%	94.32%	94.24%	
94.37%	94.48%	94.40%		
94.48%	94.53%			
94.61%				

Table 9: Results of Reinforced Learning Testing on Teenage Data

94.30%	91.01%	94.06%	93.73%	93.23%
94.19%	94.15%	93.95%	93.54%	
94.12%	94.10%	93.74%		
93.98%	93.86%			
93.91%				

Table 10: Results of Reinforced Learning Testing on Adult Data

The performance illustrated in the reinforced approach indicates that the models heavily favour models whose training data is taken completely from the same source as the test data. In this case the models do not favour the addition of other training data like the previous two models.

6 Discussion

From the results, it can be seen that the combination of models has a positive effect on the performance of the language model for the POS tagging task. This is encouraging for the developmental hypothesis. It shows that at the very least, staging the training data slightly enhances the performance of the model.

6.1 Observations

There are a few other observations about the data to note:

1. The performance of the data falls into a few regions of statistical significance. The data in the middle of the table is one region, for the results in that region have the highest tagging accuracy, but the difference between adjacent results in this region is not statistically significant. The regions of lowest accuracy are in the corners, where the model is made up entirely of one component. The third region is on the edges of the table, where the performance is between that of the center and corner regions.

2. There is a trend for each model to perform better on certain test sets than others. In general, given a test set with a particular reading level, the model that performs best is the model trained mostly on that level. In the case of the mixture models, the best case also has small portions of training on the other two reading levels, whereas the reinforced models perform best when trained entirely on the target reading level.

3. Each method manages to perform around 0.30% better than the baseline, when examining performance on the adult test data. When comparing the baseline to performance on the other test sets, the performance is understandably better for the combined models, for the reasons stated in the preceding point.

4. When compared to each other, the mixture model approach seems to have a slightly better performance of the three, but drops off a little when training and test data are mismatched. The reinforced method has nearly the same best performance as the mixture model, and seems to exhibit the fewest performance differences across mixtures. The balanced method had the poorest performance of the three, barely matching the best cases for the adult tests, and having irregular and

severe drops in performance for bad mixtures, especially those with little adult component.

5. Before the models were smoothed to eliminate zero transition and emission probabilities, there were several "missed" words, caused by these zero probabilities. The number of missed words was highest in the models trained on child data, second-lowest for the models trained on teen data, and third-lowest on the models trained on adult data. The mixed models tended to have the optimal (lowest) missed word count.

6.2 Insight

The results of these developmental experiments are hard to interpret, since there are few sharp trends to illustrate the effects of staged learning techniques. At the very least, the developmental heuristic provides a minor enhancement to the performance of the part-of-speech tagger implementation. A more optimistic interpretation is that the heuristic shows the potential to be a worthwhile addition to POS tagging, and natural language techniques in general. After all, when results are in the range of 95% tagging accuracy, a difference of a percentage point can make a noticeable difference in the number of whole sentences correctly tagged[24]. Not only this, but this developmental heuristic might improve the performance of other language tasks more dramatically, which is something that could not be explored here.

The difficulty with reducing the 6% error for the baseline adult case is that one must determine whether a technique would be capable of correcting those tags being mislabeled by the baseline approach, without compromising the performance on those tags that are already correct. The developmental approach did not address this particular challenge directly by focusing on the incorrect cases, but since it attempted to introduce a human concept to a human problem, there was the hope that it would provide some of what past algorithms were lacking.

In order to test this theory, it has to be shown somehow that the POS categories that are being mislabeled have a certain correspondence to the POS coverage of the teen and/or child training sets. Once there is evidence that the overlap value has surpassed a certain threshold, implementing the development algorithm should allow the learning stages to produce more accurate tagging results. Without this sort of measure, the results will be as in this experiment: Marginally helpful, but not enough to justify widespread application to general POS taggers.

6.3 Future Direction

The idea of training language models with developmental techniques is still a promising one, and further possibilities have opened up over the course of this research:

The first is the idea to continue researching various model combinations, in the hope of finding one that produces higher accuracy through the emulation of human learning. One such possibility is to implement a voting algorithm, that produces state sequences using the three models listed above, and chooses the result of one as the true result. This choice would be based on a confidence measure that such as the relative probability of the best state sequence for each model. Although computationally expensive, this extension would model the cognitive decision-making that takes place when reasoning between two interpretations of the same sentence. A similar approach would be to have an agent or other result analyzer that selected a model to use based on the perceived type of test data. This takes advantage of the fact that models trained on certain data perform better on data of the same type.

The second research possibility is to ignore the training issues, and instead focus on evolving the model itself over various development stages. Since this research assumes a fixed model, it is possible that better results could be obtained if the language model and its connections could develop, by spontaneously growing and extinguishing connections between nodes the way a human brain develops new connections while growing and learning. This could be done by having the model adapt its weights based on occurrences of certain words in the data, or by generating additional nodes to represent other variables that affect the model.

A final research direction would examine the effects of developmental approaches on different tasks, such as deducing rules for parse trees, or estimating tag classes from unlabelled data. The former would be more complicated, since staged data for parsing tasks isn't as readily available, but the second could be pursued with the data at hand.

7 Conclusion

The developmental learning algorithm is able to improve the accuracy results of a part-of-speech tagger with a basic stochastic model. These taggers generally perform better with combinations of models where the largest component is from the same source as the test data, and the degree to which this effect is felt depends on whether it is an additive combination (slight effect) or a multiplicative combination (significant effect). Further studies are warranted to determine whether this minor improvement can be enhanced either through a better measure of how applicable development techniques would be for this problem, or through newer extensions of development ideas to this or other language models.

References

[1] James Allan, *Introduction to Natural Language Understanding, 2nd ed.,*. Addison-Wesley Publishing Co., 1995.

[2] Shumeet Baluja, Vibhu Mittal and Rahul Sukthankar, *Applying Machine Learning for High Performance Named-Entity Extraction*, Computational Intelligence, Vol.16, No.4, November 2000.

[3] Daniel M. Bikel, Richard L. Schwartz and Ralph M. Weischedel, *An Algorithm that Learns What's in a Name*, Machine Learning, Vol.34, No.1-3, pp.211–231, 1999.

[4] Eric Brill, Radu Florian, John C. Henderson and Lidia Mangu, *Beyond N-Grams: Can Linguistic Sophistication Improve Language Modeling?*, Proceedings of COLING-ACL'98, pp.186–190, 1998.

[5] Claire Cardie and Raymond Mooney, *Guest Editors' Introduction: Machine Learning and Natural Language*, Machine Learning, Vol.34, Nos.1–3, pp.5–9, 1999.

[6] Lewis Carroll, *Through the Looking-Glass, and What Alice Found There*, Macmillan and Co., London, 1872.

[7] Stephan K. Chalup, *Issues of Neurodevelopment in Biological and Artificial Neural Networks*, Proceedings, International New Zealand Conference on Artificial Neural Networks and Expert Systems, New Zealand, 2001.

[8] Eugene Charniak, *Statistical Parsing with a Context-Free Grammar and Word Statistics*, Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI 96), pp.1031–1036, 1996.

[9] Eugene Charniak, *Statistical Techniques for Natural Language Processing*, AI Magazine, pp.33–43, 1997.

[10] Eugene Charniak, Curtis Hendrickson, Neil Jacobson and Mike Perkowitz, *Equations for Part-of-Speech Tagging*, National Conference on Artificial Intelligence, pp.784–789, 1993.

[11] Stanley F. Chen, *Building Probabilistic Models for Natural Language*, Ph.D. Thesis, Harvard University, 1996.

[12] Michael Collins, *A New Statistical Parser Based on Bigram Lexical Dependencies*, Proceedings of the 34th Annual Meeting of the ACL, pp.184–191, California, 1996.

[13] Gary L. Drescher, *Made-up Minds*, MIT Press, 1991.

[14] Jeffrey L. Elman, *Finding Structure in Time*, Cognitive Science, vol.14, No.2, pp.179–211, 1990.

[15] Ton van der Geest, *Some Aspects of Communicative Competence and Their Implications for Language Acquisition*, Van Gorcum, Assen/Amsterdam, 1975.

[16] Jean Berko-Gleason, editor, *The Development of Language*, Allyn and Bacon, Needham Heights, MA, 1997.

[17] Evelyn Hatch, *Psycholinguistics: A Second Language Perspective*, Rowley, Mass : Newbury House, 1983.

[18] S. M. Katz, *Estimation of Probabilities from Sparse Data for the Language Model Component of a Speech Recognizer*, IEEE Transactions on Acoustics, Speech and Signal Processing, ASSP-35(3):400–401, March 1987.

[19] Anders Krogh, Michael Brown, Saira Mian, Kimmen Sjolander, and David Haussler, *Hidden Markov Models in Computational Biology: Applications to Protein Modeling*, Mol. Biology, Vol.235, pp.1501–1531, 1994.

[20] Julian Kupiec *Robust Part-of-speech Tagging Using a Hidden Markov Model*, Computer Speech and Language, Vol.6, pp.225–242, 1992.

[21] Lillian Lee, *Learning of Context-Free Languages: A Survey of the Literature*, Harvard University Technical Report TR-12–96, 1994.

[22] Geoffrey Leech, *A Brief Users' Guide to the Grammatical Tagging of the British National Corpus*, http://www.hcu.ox.ac.uk/BNC/what/gramtag.html, 1997.

[23] Geoffrey Leech, Roger Garside, and Michael Bryant, *CLAWS4: The tagging of the British National Corpus*, Proceedings of the 15th International Conference on Computational Linguistics (COLING 94), pp. 622–628, 1994.

[24] Christopher D. Manning and Hinrich Schütze, *Foundations of Statistical Natural Language Processing*, The MIT Press, Cambridge, Massachusetts. 1999.

[25] Andrej A. Markov, *An Example of Statistical Investigation in the Text of 'Eugene Onyegin' Illustrating Coupling of 'Tests' in Chains.*, Proceedings of the Academy of Sciences, St. Petersburg, Vol 7, pp.153–162, 1913.

[26] Kristie Seymore, Andrew McCallum and Ronald Rosenfeld, *Learning Hidden Markov Model Structure for Information Extraction*, AAAI-99 Workshop on Machine Learning for Information Extraction, pp.37–42, 1999.

[27] Marilyn A. Nippold, *Later Language Development: The School-Age and Adolescent Years, 2nd Edition*, Pro Ed, Austin, Texas, 1998.

[28] Fernando Pereira and Yves Schabes, *Inside-outside reestimation from partially bracketed corpora*, 30th Annual Meeting of the ACL, pp.128–135, Delaware, 1992.

[29] Massimo Piatelli-Palmarini, *Language and Learning. The debate between Jean Piaget and Noam Chomsky*, Cambridge Ma.: Harvard University Press, 1980.

[30] Christopher G. Prince, *Theory grounding in embodied artificially intelligent systems*, First International Workshop on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems, Lund, Sweden, 2001.

[31] L. R. Rabiner, *A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition*, Proc. of the IEEE, Vol.77, No.2, pp.257–286, 1989.

[32] D. Rohde and D. Plaut, *Language acquisition in the absence of explicit negative evidence: How important is starting small*, Cognition, Vol.72, pp.67–109, 1999.

[33] Thomas J. Santner and Diane E. Duffy, *Statistical Analysis of Discrete Data*, Springer-Verlag, New York, 1989.

[34] Helmut Schmid, *Part-of-Speech Tagging with Neural Networks*, Proceeding of COLING-94, pp.172–176, 1994.

A The British National Corpus Tagset

(Data cited herein has been extracted from the British National Corpus Online service, managed by Oxford University Computing Services on behalf of the BNC Consortium. All rights in the texts cited are reserved)

Each tag consists of three characters. Generally, the first two characters indicate the general part of speech, and the third character is used to indicate a subcategory. When the most general, unmarked category of a part of speech is indicated, in general the third character is 0. (For example, AJ0 is the tag for the most general class of adjectives.)

A.1 The Basic Tagset

AJ0 Adjective (general or positive) (e.g. good, old, beautiful)

AJC Comparative adjective (e.g. better, older)

AJS Superlative adjective (e.g. best, oldest)

AT0 Article (e.g. the, a, an, no) [N.B. no is included among articles, which are defined here as determiner words which typically begin a noun phrase, but which cannot occur as the head of a noun phrase.]

AV0 General adverb: an adverb not subclassified as AVP or AVQ (see below) (e.g. often, well, longer (adv.), furthest. [Note that adverbs, unlike adjectives, are not tagged as positive, comparative, or superlative.This is because of the relative rarity of comparative and superlative adverbs.]

AVP Adverb particle (e.g. up, off, out) [N.B. AVP is used for such "prepositional adverbs", whether or not they are used idiomatically in a phrasal verb: e.g. in 'Come out here' and 'I can't hold out any longer', the same AVP tag is used for out.

AVQ Wh-adverb (e.g. when, where, how, why, wherever) [The same tag is used, whether the word occurs in interrogative or relative use.]

CJC Coordinating conjunction (e.g. and, or, but)

CJS Subordinating conjunction (e.g. although, when)

CJT The subordinating conjunction that [N.B. that is tagged CJT when it introduces not only a nominal clause, but also a relative clause, as

in 'the day that follows Christmas'. Some theories treat that here as a relative pronoun, whereas others treat it as a conjunction.We have adopted the latter analysis.]

CRD Cardinal number (e.g. one, 3, fifty-five, 3609)

DPS Possessive determiner (e.g. your, their, his)

DT0 General determiner: i.e. a determiner which is not a DTQ. [Here a determiner is defined as a word which typically occurs either as the first word in a noun phrase, or as the head of a noun phrase. E.g. This is tagged DT0 both in 'This is my house' and in 'This house is mine'.]

DTQ Wh-determiner (e.g. which, what, whose, whichever) [The category of determiner here is defined as for DT0 above. These words are tagged as wh-determiners whether they occur in interrogative use or in relative use.]

EX0 Existential there, i.e. there occurring in the there is ... or there are ... construction

ITJ Interjection or other isolate (e.g. oh, yes, mhm, wow)

NN0 Common noun, neutral for number (e.g. aircraft, data, committee) [N.B. Singular collective nouns such as committee and team are tagged NN0, on the grounds that they are capable of taking singular or plural agreement with the following verb: e.g. 'The committee disagrees/disagree'.]

NN1 Singular common noun (e.g. pencil, goose, time, revelation)

NN2 Plural common noun (e.g. pencils, geese, times, revelations)

NP0 Proper noun (e.g. London, Michael, Mars, IBM) [N.B. the distinction between singular and plural proper nouns is not indicated in the tagset, plural proper nouns being a comparative rarity.]

ORD Ordinal numeral (e.g. first, sixth, 77th, last) . [N.B. The ORD tag is used whether these words are used in a nominal or in an adverbial role. Next and last, as "general ordinals", are also assigned to this category.]

PNI Indefinite pronoun (e.g. none, everything, one [as pronoun], nobody) [N.B. This tag applies to words which always function as [heads of]

noun phrases. Words like some and these, which can also occur before a noun head in an article-like function, are tagged as determiners (see DT0 and AT0 above).]

PNP Personal pronoun (e.g. I, you, them, ours) [Note that possessive pronouns like ours and theirs are tagged as personal pronouns.]

PNQ Wh-pronoun (e.g. who, whoever, whom) [N.B. These words are tagged as wh-pronouns whether they occur in interrogative or in relative use.]

PNX Reflexive pronoun (e.g. myself, yourself, itself, ourselves)

POS The possessive or genitive marker 's or ' (e.g. for 'Peter's or somebody else's', the sequence of tags is: NP0 POS CJC PNI AV0 POS)

PRF The preposition of. Because of its frequency and its almost exclusively postnominal function, of is assigned a special tag of its own.

PRP Preposition (except for of) (e.g. about, at, in, on, on behalf of, with)

TO0 Infinitive marker to

UNC Unclassified items which are not appropriately classified as items of the English lexicon. [Items tagged UNC include foreign (non-English) words, special typographical symbols, formulae, and (in spoken language) hesitation fillers such as er and erm.]

VBB The present tense forms of the verb BE, except for is, 's: i.e. am, are, 'm, 're and be [subjunctive or imperative]

VBD The past tense forms of the verb BE: was and were

VBG The -ing form of the verb BE: being

VBI The infinitive form of the verb BE: be

VBN The past participle form of the verb BE: been

VBZ The -s form of the verb BE: is, 's

VDB The finite base form of the verb BE: do

VDD The past tense form of the verb DO: did

VDG The -ing form of the verb DO: doing

VDI The infinitive form of the verb DO: do

VDN The past participle form of the verb DO: done

VDZ The -s form of the verb DO: does, 's

VHB The finite base form of the verb HAVE: have, 've

VHD The past tense form of the verb HAVE: had, 'd

VHG The -ing form of the verb HAVE: having

VHI The infinitive form of the verb HAVE: have

VHN The past participle form of the verb HAVE: had

VHZ The -s form of the verb HAVE: has, 's

VM0 Modal auxiliary verb (e.g. will, would, can, could, 'll, 'd)

VVB The finite base form of lexical verbs (e.g. forget, send, live, return) [Including the imperative and present subjunctive]

VVD The past tense form of lexical verbs (e.g. forgot, sent, lived, returned)

VVG The -ing form of lexical verbs (e.g. forgetting, sending, living, returning)

VVI The infinitive form of lexical verbs (e.g. forget, send, live, return)

VVN The past participle form of lexical verbs (e.g. forgotten, sent, lived, returned)

VVZ The -s form of lexical verbs (e.g. forgets, sends, lives, returns)

XX0 The negative particle not or n't

ZZ0 Alphabetical symbols (e.g. A, a, B, b, c, d)

A.2 Punctuation Tags

PUL Punctuation: left bracket - i.e. (or [

PUN Punctuation: general separating mark - i.e. . , ! , : ; - or ?

PUQ Punctuation: quotation mark - i.e. ' or "

PUR Punctuation: right bracket - i.e.) or]

A.3 Ambiguity Tags

(tags for words with an ambiguous or unclear part-of-speech category)

AJ0-AV0 Ambiguous adjective/adverb

AJ0-VVN Ambiguous adjective/past participle

AJ0-VVD Ambiguous adjective/past tense verb

AJ0-NN1 Ambiguous adjective/common noun

AJ0-VVG Ambiguous adjective/gerundive verb

AVP-PRP Ambiguous adverb particle/preposition

AVQ-CJS Ambiguous Wh- adverb/subordinating conjunction

CJS-PRP Ambiguous subordinating conjunction/preposition

CJT-DT0 Ambiguous subordinating conjunction *that*/general determiner

CRD-PNI Ambiguous cardinal number/indefinite pronoun

NN1-NP0 Ambiguous common noun/proper noun

NN1-VVB Ambiguous common noun/verb base form

NN1-VVG Ambiguous common noun/gerundive verb

NN2-VVZ Ambiguous plural common noun/third person verb

VVD-VVN Ambiguous past verb tense/past participle