A SURVEY OF MACHINE LEARNING METHODS FOR PREDICTING PROSODY IN RADIO SPEECH

BY

AARON SCOTT COHEN

B.S., University of Illinois at Urbana-Champaign, 2002

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2004

Urbana, Illinois

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Manual prosodic transcription of large corpora is very expensive. Therefore it is desirable to have methods to automatically transcribe prosodic information with reasonable accuracy. Since large corpora exist that only have text transcriptions, such as Switchboard [1], if prosodic transcriptions could be made from text transcriptions then prosody dependent speech recognition could be extended to larger domains. To eliminate unintentional bias caused by human transcribers and create a system that can be implemented on corpora without punctuation marks, it is desirable to have a system that is highly automatic. As breaths can be marked reliably using machine methods, as shown in Price et al. [2], a system using breaths to divide text into "sentences" could be used in most environments.

This thesis compares the efficacy in prosody prediction of different machine learning algorithms on the Boston University Radio News Corpus [3]. These algorithms yield up to 83.1% accuracy on predicting accent (compared to 57.4% baseline) and up to 89.8% accuracy in prediction of boundaries with phrasal and intermediate boundaries folded together (compared to 80.9% baseline).

Chapter 2 will provide a general overview of both the linkage between prosody and syntax in linguistic theory and past efforts to model both prosodic boundaries and stress.

Chapter 3 will describe the prosodic model used in this thesis and a discussion of each learning algorithm used. These methods include tree based learners, rule generators, neural networks, and naive Bayes systems.

Chapter 4 will present the formulation of the data, how to run the learners, and how to read the output of the learners.

Chapter 5 will present a discussion of the results. This is split in two main sections, one discussing prediction of accent, the other discussing prediction of prosodic boundaries. At the end of the chapter there are suggestions for future work.

Chapter 6 will be a summary of the thesis.

Following the chapters will be appendices detailing the location of the implementations of algorithms used, the full results, the labels for part of speech and syntax information in each of the data sets, and the frequencies of the labels appearing.

CHAPTER 2

BACKGROUND ON PROSODY

2.1 Theoretical Linkage of Prosody and Syntax

The theory that prosody of speech is related to the syntactic structure of the utterance is not a recent development. In 1970 Laver noted that "tone-groups" existed with one very prominent syllable in a string of about seven syllables. The boundaries of these units often matched those of the syntactic clause. He even noted that slips of the tongue provided evidence since they generally involved the prominent syllable and only rarely was there interference between two tone-groups [4]. More recently, Chafe [5] reports that in writing, people write while thinking about the way the text would be spoken with the assumption that the reader would speak from the text the same way and interpret the meaning from the internal dialogue. Thus, the reader and writer are able to experience auditory information even where the writing does not show these features well. A possible way to detect this relation is to have people read text aloud, with the caveat that this does not exactly correspond to natural speech. Adding to the difficulty is that different readers often assign prosodic information differently to the same written texts.

Much recent research in prosody has been for the purpose of improving the intelligibility and naturalness of text-to-speech systems. In this research, accent has been found in many studies to be related to part of speech information, or a restriction of these such as content versus function words [6]-[9]. The difference between content and function words, as well as a definition,

are found in Lee [10]. In this paper, Lee found that important words in speech tended to be stressed. The stressed words tended to be verbs, adjectives, adverbs, and nouns while the more predictable words were reduced. These grammatically predictable words included articles and prepositions. The first category contains the "content" words while the second contain the "function" words.

Swerts and Geluykens [11] found that prosody features appear to be used to separate information units. This allows speakers to mark the start and end of units through the manner in which they speak. This view is shared by Shattuck-Hufnagel [12] who notes that evidence of direct influence on phonology by syntax has been observed in a number of ways. For example, prosody's use in differentiating syntactically ambiguous phrases, the deviation of intonational phrases from their usual position based on surrounding words, and the final lengthening at syntactic boundaries all indicate some degree of connection between prosody and syntax.

Conjunctions (e.g., *and*, *but*, *next*) are used lexically to link together segments of text while indicating their connection. Conjunctions often have pitch boundaries at their end and generally are either immediately followed or preceded by a pause [13]. Because of this, conjunctions tend to occur with prosodic events, which intuitively makes sense since both are providing extra information. Breaks between nouns are atypical, but in certain structures, such as a lot of nouns in sequence, prosodic breaks are more likely to exist [14].

Major prosodic breaks are also correlated to pauses for breathing [2]. In a limited study on radio news, 85% of sentence boundaries occurred during periods where the speaker took a breath. Likewise, 53% of intonational phrase boundaries were marked by a breath. Despite this high correlation between breaths and boundaries, the presence of any pause does not necessarily indicate a prosodic break [11], [15] but instead might just be caused by hesitation. Pauses are useful because they occur more often during prosodic breaks and last longer in breaks [15].

In a test comparing human performance marking boundaries between people provided with both text and the corresponding speech versus text alone, the feasibility of marking boundaries with just the textual information can be seen. In this experiment on Dutch speech, all punctuation was removed and subjects were told to mark the paragraphs, without being given a definition

of a paragraph. It was found that both groups of untrained transcribers had the paragraph markers in similar positions, although the agreement among the subjects is higher in the group of subjects provided with the audio input. It was assumed that more people agreeing on a boundary indicated a stronger boundary, and excluding the markers of no boundary (which dominate the results) there is a high Pearson correlation between the two transcriber groups of 0.82. Most of the data that did not show exact correlation between the two transcriber groups differed on the side of greater agreement among the subjects given access to the audio input [16]. These results suggest that it is possible to get fairly high accuracy in automatic boundary prediction using purely text based methods, although the expected performance would increase with the inclusion of audio data.

2.2 Prior Efforts to Predict Prosody

One early automatic parser done by Price, Ostendorf, and Wightman [2] was based on duration and breath modeling. Using a very small database, they were able to get 90% correct lexical stress on a 25-sentence set. The study also found breath detection to be a good criterion for predicting boundaries with 53% of intonational phrase boundaries and 85% of sentence boundaries marked by breaths. As 93% of the breaths were correctly detected, this indicator can be found robustly. In addition to breath detection, their system also used duration to predict breaks. Work on using duration of aligned phones with additional acoustic features for boundary prediction continued with Wightman and Ostendorf [17], [18]. More advanced work was done by Wang and Hirschberg [19] with classification and regression tree (CART) techniques, using part of speech information, presence of accent immediately before or after the potential boundary, and acoustic information, such as utterance and phrase duration. When adding hesitations and disfluencies to the definition of intonational phrase boundaries, this system achieved 83% accuracy using predicted accents in place of the manually transcribed accents as one of the inputs to the tree.

Ross [20] looked at the problem of accent prediction in more depth, even trying to determine which syllable of a word is accented. This is done in a second pass using the predicted accented words. The decision tree for pre-

dicting whether or not a word was accented included: eight categories for part of speech, the part of speech of the previous and following words, more detailed information about the word class (e.g., content word, proper noun), prosodic phrase structure, paragraph structure, new/given information status, and labels of surrounding units. The limit of part of speech categories to eight groups is due to a limitation in CART that allows it to handle a maximum of eight categories.

Similar work was done by Hirschberg and Rambow [21], who used 12 classes for part of speech, the length of a sentence, position of the word, and an enhanced tag, called a supertag in a five-word window. The supertags are names of trees from [22] and each supertag is essentially a part of speech tag with information about things such as whether a verb is in active or passive voice. Coupled with this is information obtained from a syntactic feature dependency tree, e.g., the size of the subtree headed by the current word. Using all of these features together, an error rate of 10.2% was found in predicting intonational phrases compared to a baseline error of 20% on a corpus of read text from the *Wall Street Journal*. In a later experiment [23], Hirschberg trained a decision tree for predicting pitch accent. Of all of the syntactic variables, the tree only used a word's part of speech, which she concluded meant that although adding more training data might increase performance, adding more variables will not. The study also investigated predicting phrase boundaries, and for this, the work had minimal syntactic parse information but used extensive information on distances from boundaries and position within utterances. For the algorithm, the presence of stressed words, one of the features used, was obtained using only features that could be identified automatically. Including this, the features used were a window of four words for part of speech, the predicted accent immediately before and after the potential boundary, the mutual information of the words in the four-word window, position of the word in the utterance, relationship with the nearest noun phrase, and constituency information. Using this feature set, an accuracy of 88.4% was achieved on the DARPA Air Travel Information Service corpus with intonational and intermediate phrase boundaries combined as a single unit. In the radio news domain, Hirschberg [8] used CART methods and distinctions between new and given information, contrast, cue_phrase and five additional categories on a small sample on the Radio News Corpus correctly predicting 82.4% of accents.

Simon Arnfield [6] used part of speech information but did not use parse trees. At the time of this work, no high accuracy syntactic parsers were available, so the study was limited to using word class and bigram frequencies. The study included several speaking styles, but on news broadcasts it correctly marked 92% of the words in the training set, with 52.9% of the overall words in the corpus stressed.

2.3 Conclusion

Based on this evidence, we expect to find a high correlation between stress and part of speech information. Deeper syntactic information, such as whether or not the word is at the start or end of an utterance and its position within clauses is expected to influence stress, although at a much lower level of significance than part of speech.

For intonational phrase boundary, it is expected that there will be a high correlation between prosodic boundaries and syntactical boundaries. For example, intonational boundaries are likely present between the closing of one syntactic clause or major subclause, and the opening of the next. Part of speech information may play a contributing role, but should not be the primary determiner.

The relationship between syntactic and prosodic information is investigated using multiple algorithms from the machine learning literature. The relative importance of syntactic information is determined by examining the decision tree structure, the decision rules, and the boosted decision rules to see which features are invoked and how important they are to the final decision.

CHAPTER 3

ALGORITHMS USED

This thesis seeks to answer a scientific question and a technological question. The scientific question is: What kind of relations exist between syntactic parsing data and prosodic information? The technological question is: How do we find a system that can best be used to prosodically transcribe unseen speech data? For this problem it is important not only to have reasonable results, but also to be able to adapt to new environments where the rules for determining the prosody might be slightly different but not enough data exists to create the starting environment.

For determining relationships between prosody and syntax, the most appropriate recognizer for this problem is likely a decision tree based learner. The problem fits the criteria that a decision tree is generally best suited for [24]. Namely, the target function is a yes/no question for both accent and boundary. The training data will contain errors and the instances in the training data have a fixed number of attribute values. Even the numeric cases (number of openings and closings of clauses) is discrete and over a very limited domain. This compares favorably with an artificial neural network, since there is no good way to define the attribute values for the parts of speech, which have numerous options each of unknown influence on the outcome (e.g., NN vs. NNP vs. VBD). A tree-based learner also has the benefit of being more human legible than the output of most learning algorithms. In cases where the output is not human legible, it is possible to convert the tree into a series of if-then clauses which are human legible, at the cost of a penalty in performance.

For the second task, we do not care about being able to understand the results, so the set of usable algorithms can be expanded. If understanding

the meaning of the output is not important then neural network or Bayesian methods can be used. For a neural network learner, it would be necessary to allow one input for each of the parts of speech, but this would result in adding 151 more features to the feature set. The computational complexity of this approach is too high for too many rounds of training, but it has the benefit that trained weights can be used as a starting point for retraining the learner on any new environment, such as telephone conversations.

Similarly, Bayesian learning results can be incrementally improved to adapt to new training sets. This can be done by setting the a priori values of the new target system to the output of the original. It also will need fewer features than the neural network because of the lack of multiple levels, and more importantly it will give a probabilistic measure of the presence of prosodic features. Therefore, if it gives results comparable to those of the other learning algorithms, Bayesian learning will be good for extending the learner to new prosodic domains.

If everything within an utterance affected the probability of a prosodic event e_i at word i, then the probability of an event would be $p(e_i|o_1, \ldots, o_n)$, where o_j contains all of the observations that can be made textually at word j and there are n words in the utterance. Because of evidence that humans do not plan speech far in advance, it is assumed in this thesis that $p(e_i|o_{i-2}, o_{i-1}, o_i, o_{i+1}) \approx p(e_i|o_1, \ldots, o_n)$, where stress occurs on the word corresponding to o_i and prosodic boundary occurs between the words corresponding to o_{i-1} and o_i. Further, it is assumed that a syntactic parser summarizes long-term context in the label for each word, so o_{i-2} and o_{i+1} only contain information about part of speech while o_{i-1} and o_i contain full syntactic parse information and not just the part of speech.

In the following sections, the algorithms tested in this thesis will be described. All were tested using freely available distributions of the algorithms with the addresses for the versions used listed in Appendix C. The algorithms used and described were C4.5, QUEST, SLIPPER, an implementation of an Artificial Neural Network, and a Naive Bayes Learner.

3.1 Information Theory Basics

When analyzing different machine learning algorithms, it is useful to have some background in information theory. This section is meant as a refresher and very brief introduction to information theory.

The first thing to understand is the measure of uncertainty in the variable. This is defined as the entropy of a system. The simplest case involves a binary split of examples. For this case, in a set S, there are p positive examples and n negative examples. The entropy of S with respect to this classifier is

$$Entropy(S) \equiv -\frac{p}{p+n}\log_2\frac{p}{p+n} - \frac{n}{p+n}\log_2\frac{n}{p+n} \tag{3.1}$$

In the more general sense, if there is a set of objects S which can be seperated into n different subsets S_i with no overlap, the entropy is

$$Entropy(S) \equiv -\sum_{i=1}^{n}\frac{|S_i|}{|S|}\log_2\frac{|S_i|}{|S|} \tag{3.2}$$

Entropy is measured in bits and signifies the average number of bits of information needed to distinguish the class of an object within a set. This means that if a set contains only examples from one group, it has an entropy of 0 since it is perfectly separated. If there are only two possible classes but there is an even distribution, the entropy is 1 since you would always need to be given an extra bit of information to determine the class. Entropy cannot be less than 0, but the general goal of machine learning is to lower the entropy based on observable events to get a reasonable guess at the actual class an object is in.

3.2 C4.5

This section describes the internal workings of release 8 of the C4.5 algorithm. Most material in this section is based on the presentation in [25] with any updated algorithms and implementation details substituted for those from the original book version (release 5).

C4.5 works off of the divide and conquer principle. The simplest description of how it works is as follows: It starts with all of the cases joined together at the top and splits the data up into parts based on a splitting rule. Each of

these parts represents a smaller problem which is easier to work with. These problems are again split until eventually the tree is formed.

As with other decision trees, C4.5 uses a greedy algorithm that is biased towards shorter trees with high information gain splits close to the root. Although no learner can be perfect, a noteworthy limitation of all of the tree learner algorithms is that they do not perform well when the tested relationship involves a linear combination of attributes. Since decision trees cannot divide the data set by any attributes that are not known, quantized in some manner, and included in the tests, a decision tree divides the results into hyperplanes othogonal to the various attributes available. If the actual relation is a linear combination between two of the attributes, then it is impossible to truly map this relation. Instead, the decision tree will create smaller and smaller rectangles that more finely track this line as more data is available. If this condition occurs, it is expected that the decision tree will grow rapidly with data. This problem is not present in all of the learners tested, for example the neural network learner which performs very well on relationships involving linear combinations of attributes.

3.2.1 Tree generator

At the start, C4.5 groups some of the words in the training set together to select a subset, called a window, of the data. This initially was for memory reasons, but it had the unintended benefit of generally creating more accurate trees. This window is chosen in a way that tries to make the distribution of classes within fairly uniform, which tends to give better starting points even in highly skewed data sets. After training on the window, the resultant tree is run on every example in the training set, and the examples it got wrong are evaluated. At least half of these are added to the window and a new tree is built on this expanded window. The cycle repeats until it reaches a stopping criterion or until the trees stop increasing in accuracy. The use of windowing also allows multiple trees for the same data to be constructed, which allows more trees to choose from for best results and more trees to base the rule generator on.

The trees themselves are constructed by picking the "best" attribute to split on, and dividing the data into two or more groups. This attribute has to be determined by some criterion, and in C4.5 the criterion is a variant on

the information gain. Information gain is the drop in entropy caused by the split. This is determined by taking the difference between the entropy of the system before the split and the weighted entropy after the split. For a set S over an attribute with subsets S_v, the gain is determined by this relation:

$$Gain(S, Attribute) \equiv Entropy(S) - \sum_{v \in Attribute} \left\{ \frac{|S_v|}{|S|} \times Entropy(S_v) \right\} \quad (3.3)$$

This test was strongly biased towards splits that have many outcomes, basically spreading out the data into as many small buckets as possible.

To reduce this bias, a new criterion for picking the "best" split was created, the gain ratio. Its purpose is to estimate the information gain that is useful. To avoid trivial splits, the information gain from (3.3) must also be at least the average over all of the tests looked at. The reference used to compare all of these gains is the amount of information that is gained by just splitting the data into those classes. For example, if there are 8 cases in a 4-way split, simply splitting it into 4 parts gives 2 bits worth of information since there is now only 1 bit needed to determine which case it is. In a 2-way split, you would need 2 bits to have the same thing. Therefore, any test that would split it into 4 groups would have to have a lot more gain than a test splitting it into 2 groups to ensure the information gain is not caused simply by the extra bit of information given in the encoding. The exact formula used by C4.5 to figure out the gain ratio, with n being the number of partitions from splitting on the chosen attribute, is

$$Gain\ Ratio(S) = \frac{Gain(S, Attribute)}{-\sum_{i=1}^{n} \left\{ \frac{|S_i|}{|S|} \times \log_2 \frac{|S_i|}{|S|} \right\}} \quad (3.4)$$

Using the above gain ratio has the added benefit of generally giving smaller trees, but it has a tendency to favor unbalanced trees with one of the subsets much smaller than the rest.

To fix an initial bias in continuous attributes, a different cost function based on the Minimum Description Length (MDL) principle is used [26]. This theory basically evaluates the cost of transmitting the partitioning theory plus how much it costs to indicate the cases that do not follow the theory, known as exceptions. Under a normal split, the reduction in exceptions cost is given by $|S| \times Gain(S, T)$, where S is the set of data and T is the test. For

a continuous case, this has to be modified due to the increased number of bits to send the information compared to other tests. As there are *N-1* possible thresholds for a $A \leq thresh$ test, this test would take an extra $\log_2(N-1)$ bits. To compensate for these additional bits, $Gain(S,T)$ can be reduced by $\frac{log_2(N-1)}{|S|}$, where there are *N-1* possible thresholds for the continuous value. There is also a modification to the gain ratio for continuous cases to prevent it from heavily favoring splitting the data in half. Instead, for continuous variables it just chooses a threshold for maximizing gain. The choice of whether or not to include the test in the tree is made from the gain ratio with the only change from normal operation being the use of the adjusted gain.

Instead of deciding when to stop while building the tree, C4.5 overbuilds the tree and then prunes it after it is finished running. This is done to prevent overfitting the data and to improve performance on unseen cases with the added benefit of being simpler. Pruning will also cause leaves to contain examples from multiple cases, since there is no longer a perfect assignment of an output to a leaf. This pruning is done by collapsing subtrees into single leaves and assigning each leaf the most common class. In C4.5, this pruning is done using only the training data the original tree was built on using pessimistic pruning. This is done using a confidence measure and acting as if the errors are observed events of a random process and the total number of samples in the leaf is the number of trials. The cases are treated as if they are part of a binomial distribution. Based on the single run of N trials, it is possible to put a confidence bound on the percentage of trials where the event (error) occurs. The upper confidence bound on this is used as if it is the actual error. The predicted number of errors of a subtree is simply the sum of the predicted errors in each of its branches.

3.2.2 Rule generator

Once trees have been constructed, it is possible to create decision rules. These rules are expected to have worse performance than the decision trees, but in exchange for this sacrifice the results can be made much simpler for humans to understand. On certain trees, it is also possible to reduce fragmentation by creating decision rules. Simply writing a rule that covers each leaf would not substantially simplify the tree, so it is necessary to look for ways to merge

rules to cover multiple cases (or leaves of the initial tree) [25].

Initially, the simplistic method of writing a rule per leaf is used. These rules need to be generalized, which is done by first creating a contingency table for the first rule chosen. For each rule, the contingency table is 2×2, and is divided in one direction by whether or not the examples satisfy the deleted condition, or in other words whether or not it satisfies the initial rule. In the other direction, it has a distinction between being part of the target class or not being part of it. Every example that fulfills the rule after the change is used to find the number of cases that belong in each location in the table. Using the same upper confidence bound test from the tree building, the rule (missing one condition) with the lowest pessimistic error is compared to that of the original rule, and if lower replaces it. This is repeated until the rule cannot be made more general without increasing the estimated error rate. Rule generalization is done for each path of the original tree, and when this is finished the final rule set needs to be picked.

Once the final rules are all created, they are divided by class. A subset of these rules is chosen randomly. Each round, the percentage of the overall possible rules contained in the subset increases and greedy searches for new rules are performed. Rules are chosen using the MDL principle to try to reduce the combined theory and exception cost. The subset with the lowest combined cost is used. To prevent some poor costs using the standard MDL schemes, a modified scheme is used that is biased towards having roughly equal numbers of false positives as false negatives while keeping the overall number of errors low. With D being the dataset, C and U being the cases covered and uncovered by the rule, respectively, e being the total errors, and f_p f_n being the false positives and negatives, the total exceptions cost is

$$\begin{aligned} cost &= log_2(|D|+1) + f_p \times (-log_2(\frac{e}{2C})) \\ &\quad +(C-f_p) \times (-log_2(1-\frac{e}{2C})) + f_n \times (-log_2(\frac{f_n}{U})) \\ &\quad +(U-f_n) \times (-log_2(1-\frac{f_n}{U})) \end{aligned} \tag{3.5}$$

By picking candidate rules with the lowest cost, the unimportant rules can be weeded out, leaving only those that increase the accuracy of the full set of rules [27].

3.3 Slipper Learner

Slipper works by using a simple rule builder repeatedly and combining the outcomes as a weighted sum of the rules. If this sum exceeds a threshold, the test is positive. This process is called boosting, and the concept of boosting is to use many weak rules and combine them together to create a better rule. If a rule has its criteria met, a counter with an initial bias has a fixed value added to it. This continues down the set of rules, with each rule that has its conditions met adding to the counter and the other rules doing nothing. Since it uses an initial rule builder similar to that of RIPPER, the rule generation of RIPPER is discussed here.

3.3.1 RIPPER

Ripper was created as a rule learner that scales well with the sample size on noisy data. Many algorithms try to overfit a decision tree and then prune the data using reduced error pruning, but this has the drawback of being very inefficient computationally. One attempt to overcome this problem is by direct construction of rules using an algorithm called *incremental reduced error pruning (IREP).* This performed reasonably well compared to other learners using significantly less resources, and with modification became a program called RIPPER *(Repeated Incremental Pruning to Produce Error Reduction).*

RIPPER is a divide and conquer algorithm that does a greedy search for the best rule and immediately removes all data that fulfills the conditions of the rule. The available data is split into a growing set and a training set, with the growing set containing 2/3 of the data. The rule is found using FOIL *(First Order Inductive Logic).* At every stage, FOIL considers adding conditions of either matching an attribute or satisfying a greater than/less than comparison on a continuous attribute. The condition that maximizes the information gained by FOIL is used [28].

At every step, FOIL looks at all positive cases that do not satisfy any existing rules and all negative cases. It initializes the possible rule so every positive example that is left satisfies it. As long as negative cases fulfill the conditions of the rule or the rule becomes too complex, it finds an attribute to add to the testing and uses it to form a new rule. This process is repeated

until the stopping conditions are found, and then it begins with what is left of the positive clauses and all of the negative ones. To determine the new attribute most likely to make the prediction better, a variant of the information gain is used. With a set of all examples fulfilling the conditions of the rule S, having S_+ positive examples, a subset $S\prime$ being the set of examples that fulfill the new version of the rule with the added attribute A, and $s = |S_+ \bigcap S'|$, the gain can be determined by

$$\begin{aligned} info_{FOIL}(S) &= -log_2(\frac{S_+}{|S|}) \\ gain_{FOIL}(A) &= s \times (info_{FOIL}(S) - info_{FOIL}(S')) \end{aligned} \tag{3.6}$$

[29] Here, the information is the knowledge conveyed by knowing the examples fulfilling the conditions of the rule should be classified as positive eamples, with the gain partially dependent on the number of positive examples that fulfill the conditions of the new rule to favor learning rules that cover many examples.

Once FOIL has given its possible rules, RIPPER immediately prunes them on the pruning set P. With $P_{+(-)}$ being the set of positive (negative) samples in the pruning set and $p_{+(-)}$ being the number of positive (negative) pruning set samples covered by the tested rule, RIPPER removes a single attribute in order to create the rule "$rule'$" that maximizes

$$g(rule', P_+, P_-) \equiv \frac{p_+ - p_-}{p_+ + p_-}. \tag{3.7}$$

If the value of g for rule$'$ is greater than the value of g on the original rule, the attribute is eliminated and the process repeats until no attribute deletion can improve g. The overall stopping criterion for adding rules is based on MDL principles to keep the rules from overfitting the data. There are also some more extensions to the pruning process for the actual implementation of RIPPER which are contained in [28] that are not relevant for the present discussion but are available for the interested reader.

3.3.2 Slipper algorithm

SLIPPER trades off some comprehensibility of its results for performance gains. This is true of all boosting algorithms, and it is caused by the un-

even weighting of the various conditions in determining the final outcome. SLIPPER *(Simple Learner with Iterative Pruning to Produce Error Reduction)* produces rules as in RIPPER above, but instead of removing covered examples from the training set it reduces the weight of examples that fulfill conditions of existing rules to make it more beneficial for the learner to find rules that cover new examples, as in AdaBoost (see [30], [31] for original algorithm). This allows rules that do not apply to a particular example not to alter the decision of the final output while the applicable rules add their weight to the vote. Additionally, there are no rules with negative confidence rating to maintain some amount of intelligibility to the ruleset. Instead, there is an initial default rule that is negative so it could take multiple active rules for the sum to exceed 0 to classify the example as a positive.

At the start of training, every example has equal weight, and the sum of all of the weights is 1. As each rule is chosen it is assigned a confidence value, and this value is used to update the weighting of all of the examples that fulfill the conditions of the rule. New rules are made and the weights updated until the stopping criterion is met. At every step, the sum of the weights of all of the examples stays 1. The final hypothesis is determined by the sign of the sum of all fulfilled rules' confidence values, including the default rule.

The general AdaBoost update rule over a data set S with weights $S(i)$, value y_i (-1 if example is negative, 1 if positive), value of t'th rule on example x_i $r_t(x_i)$, and a scalar Z_t which normalizes the weights to sum to 1 for rule t is $S_{t+1}(i) = \frac{S_t(i) \times e^{-y_i r_t(S_i)}}{Z_t}$, where Z_t is as small as possible.

For the specific implementation used by SLIPPER, denoting the set of examples fulfilling the conditions of the rule R_t and the other examples R'_t, the normalizing factor is

$$Z_t = \sum_{x_i \in R'_t} S_t(i) + \sum_{x_i \in R_t} S_t(i) e^{-y_i r_t(x_i)}$$

or

$$Z_t = \sum_{x_i \in R'_t} S_t(i) + \sum_{x_i \in R_t : y_i = -1} S_t(i) e^{r_t(x_i)} + \sum_{x_i \in R_t : y_i = 1} S_t(i) e^{-r_t(x_i)} \quad (3.8)$$

After including a smoothing factor of $\frac{1}{2n}$ to prevent infinite confidence

values, Z_t is minimized by setting

$$r_t(x_i) = \begin{cases} \frac{1}{2} \ln \frac{\sum_{x_i \in R_t : y_i = 1} S_t(i) + \frac{1}{2n}}{\sum_{x_i \in R_t : y_i = -1} S_t(i) + \frac{1}{2n}} & \text{if} \quad x_i \in R_t \\ 0 & \text{if} \quad x_i \in R'_t \end{cases} \tag{3.9}$$

After simplification, Equation (3.8) becomes

$$Z_t = 1 - \left(\sqrt{\sum_{x_i \in R_t : y_i = 1} S_t(i)} - \sqrt{\sum_{x_i \in R_t : y_i = -1} S_t(i)} \right)^2 \tag{3.10}$$

which is minimized by trying to maximize

$$\hat{Z}_t = \sqrt{\sum_{x_i \in R_t : y_i = 1} S_t(i)} - \sqrt{\sum_{x_i \in R_t : y_i = -1} S_t(i)} \tag{3.11}$$

With the equation to maximize by the rules determined, it is possible to try building the rule. SLIPPER first starts with a blank rule that covers every example, and then iterates adding a new attribute that maximizes $\hat{Z}_t$. This continues until there are no negative examples in the rule or $\hat{Z}_t$ is no longer improved. After the rule has been found, it will be necessary to prune the rule. Working with the data split for the pruning set, let $P_{+(-)}$ denote the number of positive (negative) examples in the pruning set covered by the rule divided by the total number of examples in the pruning set. Then, to prune, r_t is found using the full rule on the test set, with the value of r_t always equal to $\frac{1}{2} \ln \frac{\sum_{x_i \in R_t : y_i = 1} S_t(i) + \frac{1}{2n}}{\sum_{x_i \in R_t : y_i = -1} S_t(i) + \frac{1}{2n}}$ here. Then find the single attribute to remove from the rule to minimize

$$(1 - P_+ - P_-) + P_+ e^{-r_t(x_i)} + P_- e^{r_t(x_i)} \tag{3.12}$$

This is repeated as often as possible, and then compared against the default rule. Whichever has the lowest value for $\hat{Z}_t$ will be returned to the booster. There is a five-way cross-validation run on the training set to determine the number of rounds to train on by setting the number of rounds equal to the number of rounds that produces the lowest average error on the holdout data. To keep this from running forever, an initial value is set which is the maximimum number of rounds the cross-validation sets will run.

3.4 QUEST

QUEST (*Quick, Unbiased, Efficient, Statistical Tree*) was designed to develop trees with reduced bias. This is because exhaustive searches (like those performed by C4.5 and most other tree learners) tend to prefer selections that have many splits over ones that do not. This can also lead to development of theories that have little relation to any pattern that may actually be present. QUEST uses initial analysis to determine which variable to split on instead of blindly splitting on every variable to test the gain. Using analysis to determine which splits will be considered has the added advantage of decreasing computational complexity since fewer splits will be tested. Unless otherwise noted, everything in this section is based on [32].

3.4.1 Finding discriminant coordinates

For both linear and univariate versions of QUEST it is necessary to convert multivalued discrete categorical variables into variables for which a distance metric can be defined. This is done by transforming each discrete variable to a linear discriminant coordinate (CRIMCOORD) value. By mapping discrete variables into 0-1 dummy vectors, the CRIMCOORDS can be made. One problem with this method is it can lead to singular covariance matrices if care is not taken in the transformation. To avoid difficulties with singular matrices without potentially splitting the node into more than two subnodes, the attribute values not present are eliminated before the mathematical transformation. For example, let X be a discrete variable represented in the subset being investigated, where $x_i \in \{a_1, a_2, \ldots, a_m\}$ and where all m possible values are used by at least one example in the training data. This can then be converted into a binary vector by making a vector that is 0 at every component except for the one corresponding to the attribute present. Thus, $X \Rightarrow \mathbf{v} =< v_1, v_2, \ldots, v_m >'$ where $X = a_i \Rightarrow v_i = 1, v_j = 0 \; \forall \; j \neq i$.

In order to use this information, it is necessary to find the average value of the vector $\mathbf{v}$, $\overline{\mathbf{v}}^{(k)}$, in each output class k. This is done by adding the vector for all examples $\mathbf{v}_i$ in the set being investigated corresponding to the output class k and then dividing it by the number of examples used. Using this method, $\overline{\mathbf{v}}^{(k)} = \frac{1}{N^{(k)}} \sum_{i=1}^{N^{(k)}} \mathbf{v}_i$, where $N^{(k)}$ is the number of examples contained in output class k. This is expanded to cover all output classes by

combining all of the $\overline{\mathbf{v}}^{(k)}$ together as follows:

$$\overline{\mathbf{v}} = \frac{1}{N} \sum_{k=1}^{K} \left(N^{(k)} \times \overline{\mathbf{v}}^{(k)} \right)$$

where N is the total number of examples and K is the total number of output classes.

Now that the discrete variable has been converted into vector form, it is possible to transform it into CRIMCOORD values using standard discriminant analysis. This is done by making $M \times M$ matrices

$$\mathbf{B} = \sum_{k=1}^{K} \left(N^{(k)} (\overline{\mathbf{v}}^{(k)} - \overline{\mathbf{v}})(\overline{\mathbf{v}}^{(k)} - \overline{\mathbf{v}})' \right) \tag{3.13}$$

$$\mathbf{W} = \sum_{k=1}^{K} \sum_{i=1}^{N^{(k)}} (\mathbf{v}_i^{(k)} - \overline{\mathbf{v}}^{(k)})(\mathbf{v}_i^{(k)} - \overline{\mathbf{v}}^{(k)})' \tag{3.14}$$

$$\mathbf{T} = \mathbf{B} + \mathbf{W} \tag{3.15}$$

The CRIMCOORD is determined by first finding the vector $\mathbf{a}'$ that maximizes the ratio $\frac{\mathbf{a}'\mathbf{B}\mathbf{a}}{\mathbf{a}'\mathbf{W}\mathbf{a}}$. If $\mathbf{W}^{-1}$ exists, then $\mathbf{a}$ is the eigenvector with the largest eigenvalue of $\mathbf{W}^{-1}\mathbf{B}$.

After finding the vector $\mathbf{a}$, the CRIMCOORD is made by projecting $\mathbf{v}$ along $\mathbf{a}$ as $\mathbf{a}'\mathbf{v}$.

3.4.2 Linear

In QUEST, a "linear" tree is one with each node in the tree computing a linear combination of attributes. The path from the node is determined by comparing this linear combination against some threshold, with these splits continuing until eventually a final output is chosen.

To perform linear combination splits, first create an observation vector $\mathbf{z}$ with the numerical attributes and the numerical projection of the discrete category from Section 3.4.1. The vector $\mathbf{z}$ is $\mathbf{z} =< z_1, z_2, \ldots, z_m >'$ with z_j equaling the numerical value if the attribute has one or $\mathbf{a}_j'\mathbf{v}_j$ where j is an indicator of which discrete attribute in the sequence is being transformed into continuous form as shown in Section 3.4.1, $1 \leq j \leq m$. There will be one value of $\mathbf{z}$ for every example in the training set, and these will be combined to form the matrix $\mathbf{Z}$ with the $\mathbf{z}$ for each example being its own column. Next,

perform singular value decomposition. With N training examples and $\mathbf{H}$ a square matrix with every value $\frac{-1}{N}$ except along the diagonal which has value $1-\frac{1}{N}$ and $\mathbf{D}$ being a diagonal matrix with all values greater or equal to 0 listed by descending values (i.e., $\mathbf{d}_1 \geq \mathbf{d}_2 \geq \ldots \geq \mathbf{d}_m \geq 0$), solve $\mathbf{H}\,\mathbf{Z} = \mathbf{P}\,\mathbf{D}\,\mathbf{Q}'$ where $\mathbf{P}$ and $\mathbf{Q}$ are unitary matrices. Define ϵ as the machine precision (the smallest value that can be added to 1 that will make the outcome greater than 1 for the computer). Assuming there are more training examples than attributes, every eigenvalue d_i is set to 0 if $d_i \leq N d_1 \epsilon$, and $\mathbf{D}$ is reformed using these values. Let $\mathbf{F}$ be the matrix that consists of the columns in $\mathbf{Q}$ that have nonzero corresponding values in $\mathbf{D}$, and let $\mathbf{U}$ be a diagonal matrix that consists of the inverses of all nonzero values in $\mathbf{D}$.

For each output class k, define a matrix $\mathbf{L}_k = \left[\overline{\mathbf{z}}^{(k)} - \overline{\mathbf{z}}, \ldots, \overline{\mathbf{z}}^{(k)} - \overline{\mathbf{z}}\right]$ with N_k columns. Then fill a matrix $\mathbf{G}$ with an entry for each output class by setting $\mathbf{G} = [L_1, L_2, \ldots, L_K]'$ with K being the total number of output classes. This means that $\mathbf{B} = \mathbf{G}'\mathbf{G}$ where $\mathbf{B}$ is from Equation (3.13). After finding the eigenvector $\mathbf{a}$ with the largest eigenvalue of the matrix $\mathbf{GFU}$, this is used to find ξ. Transform every example $\mathbf{z}$ to a value of ξ by setting $\xi = \mathbf{a}'\mathbf{U}\mathbf{F}'\mathbf{z}$.

Once ξ has been found, the 2-means algorithm of [33] is used. Since the cases we are interested in only have two output classes each, for discussion the method used can be simplified to just two output classes. Normally all of the data has to be split into two superclasses that can contain multiple classes using the same 2-means algorithm. In our case it is possible to just find the mean and variance of each class k. These will be denoted μ_k and σ_k^2 for the mean and variance of class k, respectively. The prior probabilities p_k for each class can also be found by dividing the number of examples in class k by the total number of examples.

Using $\phi(x) = \frac{e^{-\frac{x^2}{2}}}{\sqrt{2\pi}}$ (the usual normal distribution), the boundary between classes is the set of ξ that satisfy

$$p_1\sigma_1^{-1}\phi\{\frac{\xi-\mu_1}{\sigma_1}\} = p_2\sigma_2^{-1}\phi\{\frac{\xi-\mu_2}{\sigma_2}\} \tag{3.16}$$

where 1 and 2 indicate the two output classes k, with 1 being the class with more examples. Taking the logarithm of both sides, the equation

$$((\sigma_1)^2 - (\sigma_2)^2)\xi^2 + 2(\mu_1(\sigma_2)^2 - \mu_2(\sigma_1)^2)\xi + c = 0$$

$$(\mu_2\sigma_1)^2 - (\mu_1\sigma_2)^2 + 2(\sigma_1)^2(\sigma_2)^2 \log\left[\frac{p_1\sigma_2}{p_2\sigma_1}\right] \overset{\text{where}}{=} c \qquad (3.17)$$

can be solved with the roots being the split.

If the two variances are the same, and the two means are different, there is only one root with

$$\xi_{split} = \frac{\mu_1 + \mu_2}{2} - (\mu_1 - \mu_2)^{-1}(\sigma_1)^2 \log\left[\frac{p_1}{p_2}\right]$$

If the two variances and two means are the same, there is no split since there is no way way to seperate the two distributions, so the algorithm just sets $\xi_{split} = \mu_1$.

If the variances are different, there are three possible cases for determining ξ_{split}. If $\left(2(\mu_1\sigma_2^2 - \mu_2\sigma_1^2)\right)^2 - 4(\sigma_1^2 - \sigma_2^2)c < 0$, then set $\xi_{split} = \frac{\mu_1+\mu_2}{2}$. Otherwise, if it is possible to get two nonempty nodes by setting ξ_{split} to whichever of the two roots $\frac{-2(\mu_1\sigma_2^2-\mu_2\sigma_1^2) \pm \sqrt{\left(2(\mu_1\sigma_2^2-\mu_2\sigma_1^2)\right)^2 - 4(\sigma_1^2-\sigma_2^2)c}}{2(\sigma_1^2-\sigma_2^2)}$ is closer to μ_1, then ξ_{split} is set to be that root. If both of the prior cases fail to set ξ_{split}, then revert to defining $\xi_{split} = \frac{\mu_1+\mu_2}{2}$

The stopping rule is given in [34]. The rule says that the algorithm above continues until additional splits no longer cause the error rate to decrease, or only one class in the node has more examples than a set minimum. The error rate is found not to decrease if

$$\sum_{i=1}^{N} C(l(t)|i)p(i|t) \leq \sum_{j=1}^{N}\{\sum_{i=1}^{N} C(l(t_j)|i)p(i|t_j)\}$$

where N is the number of output classes, each t_j corresponds to a child node, $l(t)$ is the label assigned to the node, and $C(l(t)|i)$ is the misclassification cost.

3.4.3 Univariate

A second type of tree can be built by QUEST. This tree is the "univariate" type where each node can only have its immediate descendants determined by a binary split on a single attribute. When splitting on an attribute with a discrete set of values, the values are partitioned into two distinct subsets

through use of discriminant coordinates.

To avoid bias that can easily exist with a combination of discrete and numerical variables, it is necessary to use care in determining how to rank variables for split selection. This is done using the Pearson χ^2 test for categorical variables and using ANOVA on the continuous variables.

First it is necessary to know how to use ANOVA (analysis of variance) for testing ordered statistics. If there are J possible output classes, we use ANOVA on an attribute with distribution X in the following manner. Define $\overline{X}$ as the average value of X and $\overline{X}_{(j)}$ as the average value of X in class j. Then find the overall sum of squares SS_X by setting it equal to $\sum_{X_i \in X}(X_i - \overline{X})^2$. In the same fashion it is possible to find the sum of squares for each class and use it to define the sum of squares for all outputs, $SS_{X_{(J)}} = \sum_{j=1}^{J} \sum_{X_i \in X_{(j)}} (X_i - \overline{X}_{(j)})^2$. Once this has been found, the F-value is found to be

$$F_X = \frac{SS_X - SS_{X_{(J)}}}{SS_{X_{(J)}}} \times \frac{N - J}{J - 1} \tag{3.18}$$

If it is necessary to find the Levene [35] F-statistic, it is done as follows. Compute the averages as in ANOVA. Then, instead of using the sum of squares, use the absolute deviations (L^1 instead of L^2 norm). Thus, $\|X - \overline{X}\|_1 = \sum_{X_i \in X} \left|X_i - \overline{X}\right|$ and $\|X_{(J)} - \overline{X}_{(J)}\|_1 = \sum_{j=1}^{J} \sum_{X_i \in X_{(j)}} |X_i - \overline{X}_{(j)}|$. With this definition,

$$F_{(LEVENE)_X} = \frac{\|X - \overline{X}\|_1 - \|X_{(J)} - \overline{X}_{(J)}\|_1}{\|X_{(J)} - \overline{X}_{(J)}\|_1} \times \frac{N - J}{J - 1} \tag{3.19}$$

In either case, it is possible to find the P-value for the test. The P-value is $\Pr(Z \geq F)$. A lower P-value is desirable, as it indicates with greater certainty that this split is significant and did not happen purely by chance.

For discrete variables there is only one test done, but it is used in both the initial decision round and the second chance decision round. If a discrete variable takes N values, and there are C classes in the node, then the chi-square with $(C - 1)(N - 1)$ degrees of freedom is used. If the smallest P value resulting from division of any attribute is under a threshold, then the split it corresponds to is used. If this fails, Levene's F-test is computed for the continuous variables, and the P values for the new continuous F-test and the original discrete tests are compared to a secondary threshold. If this also fails, the attribute with the lowest P value from the first test is used.

Continuous attribute split point selection

The algorithm to determine the split point for continuous variables is less involved. Here you can just do a direct application of Equation (3.16). Again, p_1 corresponds to the output class with the most examples. The split happens in exactly the same fashion as in the linear case starting immediately after Equation (3.16) occurs in that algorithm.

Discrete attribute split point selection

If a discrete attribute is picked, the split point is determined in a very similar fashion as in the linear version of QUEST. To alter the algorithm, first replace $\mathbf{z}$ with $\mathbf{z} =< z_1, \ldots, z_N >'$, where there are N categories for the discrete variable and $z_i = 1$ if category i is active, 0 otherwise. $\mathbf{Z}$ is the matrix created by finding the value of $\mathbf{z}$ for each example and making them all their own column, as in the algorithm for linear combination splits. Letting M equal the number of examples and assuming more examples than attributes, the eigenvalues that are less than $Md_1\epsilon$ are now defined as zero. There are no other changes that need to be made to the algorithm used for linear splits, except the dimensions of matrices will change, and after the value of ξ is found to make the split on, it will have to be converted to a set of values for the attribute that are in each node.

Alternate splitting criteria

While running the program, there are four alternate cost functions that can be used for splitting discrete categories in place of the χ^2 function. These four additional options are the likelihood statistic (also used by C4.5), Gini index, mean posterior improvement (MPI), and a criteria that is altered by an initial user setting. In this thesis, the likelihood ratio was the only additional algorithm found to perform the best on any distribution. For a discussion of the three criteria not used as well as the two that are used, see [36]. The two class generalized likelihood statistic G^2 with p_L and p_R being vectors containing the relative proportion of each output class in the left and right subnodes, p being the relative proportion of each output class in the root node, N_L and N_R being the number of examples in the subnodes, and the

superscript indicating the output class, is

$$G^2 = 2 \times \left(N_L \sum_{i=1}^{2} (p_L^i \log_2(\frac{p_L^i}{p^i})) + N_R \sum_{i=1}^{2} (p_R^i \log_2(\frac{p_R^i}{p^i})) \right) \tag{3.20}$$

3.5 Neural Network

Multilayer neural networks have a series of inputs to first layer sigmoids, and the output of these sigmoids are the inputs of other sigmoids until the final layer where the output is formed. A sigmoid basically sums all of its inputs together with an initial bias and passes the result through a nonlinearity. In order for an update algorithm to work cleanly, this has to be a differentiable function that behaves similarly to a unit step response in that once a threshold is crossed, the amplitude becomes full. The sigmoid function is defined as

$$\sigma(y) \equiv \frac{1}{1 + e^{-y}} \tag{3.21}$$

which is differentiable everywhere as $\frac{d\sigma(y)}{dy} = \sigma(y) \times (1 - \sigma(y))$. Alternatively, the *tanh* function can be used.

Assuming there is at least one hidden layer, the Backpropagation algorithm is generally used as the update rule. For a more general treatment of the algorithm and a thorough derivation of the rule see Mitchell [24]. For this type of network, let x_i be the input to unit i, and w_{ji} be the weight from i to j. After creating the initial network and initializing all weights to random small numbers, send each training example through the network. For the final network output, find its error term. With o_f being the output of the system, t being the correct output, and η being a user assigned learning rate,

$$E \equiv \frac{1}{2} \sum_{Training\ data} (t - o_f)^2 \tag{3.22}$$

$$\frac{\partial E}{\partial x_f} = o_f(1 - o_f)(t - o_f) \tag{3.23}$$

$\forall$ hidden units h connected to the output, calculate the error, e_h

$$e_h = \frac{\partial E}{\partial x_h} = o_h(1 - o_h)\, w_{fh} \frac{\partial E}{\partial x_f} \tag{3.24}$$

$\forall$ hidden units a connected to hidden units $b_1, b_2, \ldots, b_n$

calculate the error, e_a

$$e_a = \frac{\partial E}{\partial x_a} = o_a(1 - o_a) \sum_{b_i \in b_1, b_2, \ldots, b_n} w_{ab_i} e_{b_i} \tag{3.25}$$

Update every weight within the network using:

$$w_{ji} = w_{ji} - \eta \frac{\partial E}{\partial w_{ji}} \tag{3.26}$$

This is carried out for every example in the training set and then repeated until some stopping criterion is met [24].

3.6 Bayesian Learning

The naive Bayes classifier can be used to get a learner that outputs a probability measure that indicates the likelihood of the prosodic event under consideration [24]. With an attribute value set $< a_1, a_2, \ldots, a_n >$ associated with an instance with target value y_i which is 1 in the presence of the target event, otherwise 0, the goal is to find the most probable target value:

$$\begin{aligned} y_{MAP} &= \arg\max_{y_i \in 0,1} P(y_i | a_1, a_2, \ldots, a_n) \\ &= \arg\max_{y_i \in 0,1} \frac{P(a_1, a_2, \ldots, a_n | y_i) P(y_i)}{P(a_1, a_2, \ldots, a_n)} \\ &= \arg\max_{y_i \in 0,1} P(a_1, a_2, \ldots, a_n | y_i) P(y_i) \end{aligned} \tag{3.27}$$

For simplification, naive Bayes assumes that the various attributes are conditionally independent from each other but not the target value. Thus $P(a_1, a_2 | y_i) = P(a_1 | y_i) \times P(a_2 | y_i)$ to use a simple example with only two attributes. Thus, the naive Bayes classifier finds

$$y_{NB} = \arg\max_{y_i \in 0,1} P(y_i) \prod_{j}^{n} P(a_j | y_i) \tag{3.28}$$

by substituting into Equation (3.27). This requires us to only estimate the distinct $P(a_j | y_i)$ terms, which is much smaller than the full set from Equation (3.27). For discrete attributes with more than two options, it is necessary to split the set into separate attributes each indicating the presence or absence of a single option of the attribute.

For continuous attributes, WEKA follows the procedure outlined in [37]. Although traditionally a single Gaussian is used to model a continuous variable, in this updated version a kernel density estimation is used instead. This

is advantageous since a Gaussian estimator has only two degrees of freedom (mean and variance).

If the continuous attribute is X and the class label is again y_i, the kernel estimator function is

$$p(X = x|Y = y_i) = \frac{1}{n} \sum_j \frac{1}{\sqrt{2\pi}\ \sigma_i} e^{-\frac{(x-\mu_j)^2}{2(\sigma_i)^2}} \tag{3.29}$$

Here, j goes across every example of class y_i in the training data, n is the number of training points, and $\mu_j = x_j$ so the estimator is a sum of a large number of Gaussian distributions each centered at a data point. There is no certain way to guess the best value of σ_i, but it should shrink to zero as there are more examples. A rule that works fairly well over most distributions is to set $\sigma_i \ \alpha \ \frac{1}{\sqrt{n_i}}$, where n_i is the number of examples with class i.

CHAPTER 4

METHOD

To train and test the dependence of prosody on textual information, a large enough corpus is required. For this the Boston University Radio News Corpus [3] was used. This corpus used read speech, but the speech was read several times, which would make any required locations for prosodic information clear and provide insight into locations where boundaries and accent are optional but not required. Although it has part of speech information following the Penn Treebank [38] standards included, to test the feasibility of prosodically marking other corpora, this information is discarded and replaced by part of speech information automatically generated. Likewise, punctuation markers were eliminated from the transcription before syntactic parsing, since punctuation marking is not always available and may bias the system towards human placed boundary markers such as periods and commas. To replace the punctuation marks, transcribed breaths were used to determine the starting and ending of "sentences" as input to the parsers. As breath detection can be done with high accuracy, this seemed a reasonable way to break long chunks of text into sizes usable by the parsers without introducing significant human transcription bias. The prosodic information was transcribed according to the ToBI [39] system.

Although the artificial style may not accurately reflect the prosody of normal speech, it may be more instructive for finding prosodic patterns since newscasters have been found to use clearer and more consistant prosody than normal speech. Speech from seven speakers (three female, four male) are in the corpus with two of the female speakers accustomed to reading news live and the four male speakers, and the other female speaker accustomed to prerecording news stories. For this study, a subset of the corpus containing the

three female speakers and two of the male speakers was used. The corpus had an agreement between transcribers of 91% in presence versus absence of accent, and 95% for prosodic boundaries.

4.1 Preprocessing

Before machine learning methods can attempt to discover relationships between accent, prosodic breaks, and syntactic features it is necessary to label the syntactic features. To make the predictions more applicable for data sets without any marked punctuation, all punctuation marks, including apostrophes, were stripped from a prosodically transcribed corpus, the Boston University Radio News Corpus [3]. Then the data was run through Roth's shallow parser [40], [41] for part of speech information and Charniak's parser [42] for deeper syntactic structure and part of speech information.

4.2 Formulation of the Data

For the first data set, the part of speech information came from the Roth parser. For prediction of accent, the part of speech was all the information available to the learner, but there were a variety of windows. For prosodic boundaries, a small subset of syntactic information from the Charniak parser was used. Here, using the names the Charniak parser uses, only the openings of a SENT, NP, VP, PP, ADJP, ADVP, and SBAR were used. This limited set was only used to mark the syntactic information on the word immediately after potential boundaries. This simplified form was used to rapidly test out the theory of prosodic dependence on syntax and to allow larger numbers of runs from C4.5, which is a very time consuming algorithm on large data sets.

For the second data set, all of the syntactic information from the Charniak parser is used, and the part of speech information from Roth's parser is discarded in favor of the part of speech information from the Charniak parser. In addition, the number of openings and closings of clauses given by the parser is used as a variable. The full data set consists of a four-word window of part of speech information and a two-word window of syntactic information. This is arranged so there is part of speech information for two words before and after prosodic breaks and the full syntactic information

set for the word immediately before and after the prosodic break. The full syntactic information is the number of openings and closing of clauses, and which clauses are opening and closing (e.g., NP, CONJP, ADJP, ...). For a full listing of the attributes used by the Charniak parser and how often they appear in the test and training sets, see Appendix A. There are other possible tags that the Charniak parser can give, but these were never seen in the Radio News Corpus.

For the third data set, syntactic information from the Charniak parser is combined into categories and the information from Roth's parser is discarded in favor of the part of speech information from Charniak. With the exception of this combination, every other aspect of this data set is the same as that used in the second run. For a list of the parts of speech used and how the full set was combined, see Appendix B.

For all runs, the data are matched with the accent and the boundary markers made for [43]. For training of the trees, the dataset was divided with 90% of the transcriptions used for training and 10% for testing. This resulted in 19 344 words in the training set and 2068 words for the test set. This also resulted in 3767 prosodic breaks (19.5%) in the training set and 395 (19.1%) in the test set. For accent, there were 10 840 (56%) accented words in the training set and 1188 (57.4%) in the test set.

4.3 Testing for Prosody

To prevent unnecessary splitting of the data, a separate run for each learner is made for accent information and boundary information. The boundary information was also presented in two forms, preboundary and postboundary, to see if either marker of the boundary performs better. In the detection of prosodic breaks, detection of the word immediately following a boundary had superior performance, in this thesis only the post-boundary results are considered. The data is transformed into comma delimited sets using a Perl script with a field for every used separation criterion. After the data set is created, the test variable is pasted at the end. These test variables are created by using a combination of a Perl script and hand alignment, with the hand alignment simply a check to ensure the tested attribute matches the word by making sure the word matches (performed every couple of hundred

words).

This data was then fed to the various recognizers. For the C4.5 recognizer, the option to allow multiple tests on the same attribute was used, so at a node there could be a test on whether or not the part of speech was any combination of the parts of speech available. For SLIPPER, the default options were used. For QUEST, 10 runs per data set were made, 5 each for multivariate and univariate splits. The default options were used for each splitting criterion were used. The neural network was trained with 10 iterations through the data, a decaying learning rate, and with the option of building the structure automatically. The final structure was the same for both the accent and prosodic boundary prediction tasks, with one hidden layer containing 92 nodes for data set 2 and 134 nodes for data set 3.

4.4 Interpreting the Results

Once the data has been trained, it is necessary to read the trees to implement them. The exact method is slightly different for each algorithm, but they will be summarized here. For C4.5, the trees grown were large, and so detailed that it looks like the true nature of the problem is not being modeled. The first thing to do is to go to the end of the log file and find the tree with the best performance. This is the tree marked with a "<<" in the evaluation segment. Then start from the beginning and look for the first "Simplified Decision Tree" after the trial number listed. This is the pruned decision tree. Then, each line represents the question asked at the node. If there is a decision made, immediately after the colon the output is listed. The numbers in parentheses indicate the number of examples likely to fall under that node, and of those, how many are errors, respectively. If there is no decision made at that node, the lines immediately below the node that are indented one level more than the parent node are the child nodes. This continues until all of the decisions have been made. To get an estimate of the accuracy of the resulting tree, it is necessary to go back to the end of the file where the evaluation section is. Here is indicated the number of errors on the test data after pruning, and the predicted error rate. If those numbers are close together, then the classifier generalized roughly as well as predicted on unseen results. The log file also contains the confusion matrix for the best

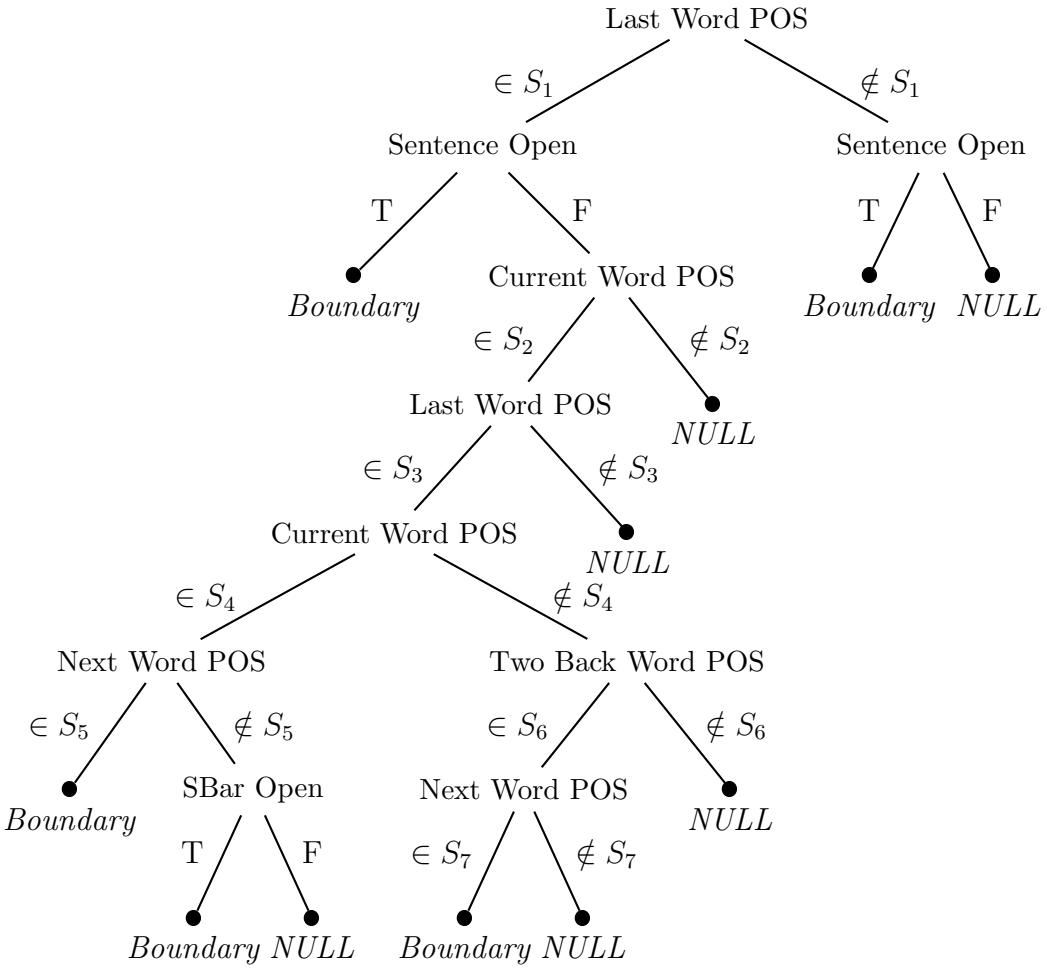
tree. The columns indicate the output of the classifier, the row indicates the actual class. This is the convention that will be adopted for this thesis.

More reliable results can often be achieved with the rule form of the C4.5 classifier. These rules have the added benefit of being easier to read. To implement the rules, first find the composite ruleset. Then set the output variable to the "default" class listed at the bottom. After this is done, the various rules are tested in order, with the output assigned by the first rule that applies to the test case. The order of implementation within an output class does not matter, and all of the possible output classes have their rules separated. The percentage indicator at the end of the rule is the predicted accuracy of the individual rule. The higher the percentage (and earlier the rule) the more important it is to the system. The overall error of the ruleset can be seen at the end of the logfile with the error rate on the "**" ruleset, with the confusion matrix for the set listed directly above it.

For SLIPPER, finding the output requires the use of a dummy value. The rules can be tested in any order, and if the test case fulfills a rule, add to the dummy variable the amount indicated in the parentheses. For all cases except for the "default" these values are positive, which means that every rule can only increase the chance of the desired output. Likewise, rules with larger weighting are more important, so it is possible to understand which rules are more important than the others. Once every rule has been evaluated, if the dummy variable is positive, then the output is the nondefault value. Otherwise it is the default value, which is generally the output that occurs the majority of times. The error rates for SLIPPER are listed at the end of the log file, with the predicted range of the error on unseen cases given in the parenthesis.

Unlike the other algorithms, QUEST gives the option of creating an output that can readily be visualized when in the univariate mode as shown in this example (see Figure 4.1). For the subsets that are shown in the tree, it is necessary to look at the log file which will have the full decision tree. Also in the log file will be the confusion matrix for the tree on the training set. The linear combination version of QUEST returns results that cannot readily be analyzed by humans but can be implemented. The results given are the coefficients for creating the equivalent of a multilayer perceptron with the discriminate coordinates and linear coefficients given with the node they

are used in. As in the univariate version, a confusion matrix for performance on the training set is given.



Set	Contains	Does not contain
S1	JJR, JJS, NN, NNP, NNS, UH, VBN, VBP, VBZ	CC, CD, DT, EX, FW, IN, JJ, MD, NULL, PDT, PP$, PRP, RB, RBR, RBS, RP, TO, VB, VBD, VBG, WDT, WP, WP$, WRB
S2	CC, DT, EX, IN, JJR, JJS, MD, PP$, PRP, RBR, TO, VBD, VBG, VBP, VBZ, WDT, WP, WP$, WRB	CD, JJ, NN, NNP, NNS, PDT, RB, RBS, RP, VB, VBN
S3	JJS, NN, NNP, NNS, UH	JJR, VBN, VBP, VBZ
S4	CC, DT, PP$, PRP, TO, WDT, WP, WRB	EX, IN, JJR, JJS, MD, RBR, VBD, VBG, VBP, VBZ, WP$
S5	CD, IN, JJ, MD, NNP, PRP, RBS, TO, VBD, VBG, VBP, VBZ, WP, WRB	CC, DT, JJR, NN, NNS, PP$, RB, VB, VBN
S6	CD, FW, JJ, NN, PP$, TO, VBN, VBZ, WDT, WP, WRB	CC, DT, IN, JJR, JJS, NNS, PDT, PRP, RB, RP, VB, VBD, VBG, VBP, WP$
S7	CD, JJR, PP$, RB, RBR, TO, VB, VBD, VBG, VBN, WP	CC, DT, EX, IN, JJ, MD, NN, NNS, PDT, PRP, RP, VBP, VBZ, WDT, WRB

Figure 4.1: Example decision tree from QUEST. The tree is built on the first data set, testing for prosodic boundary, modified to be easier to read.

CHAPTER 5

RESULTS

In this chapter, the results from the various configurations in predicting prosodic information will be presented. It will be separated by configuration and will include diagrams of successful learners. Every presented result uses the same utterances for their training, and all of the test data comes from the same set of utterances. The category "S1" indicates the start of a "sentence" that the syntactic parser received, and since these are marked by breaths, S1 is replaced with "breath" wherever it appears.

5.1 First Data Set

As stated earlier, this data set contained only part of speech information for accent prediction with the addition of limited syntactic information for prosodic boundary information.

For accent prediction, each run of C4.5 had 25 trials from which to pick the best tree. Four different windows were used, with the current word being the one tested for accent:

1. Current word POS
2. Current word POS and word immediately prior
3. Current word POS, word immediately prior POS, and word immediately after POS
4. Current word POS, word immediately prior POS, and two words back POS

The error rates are shown in Table 5.1 below, with the configuration number refering to the list above. As the error rate does not significantly drop between configuration 2 and either 3 or 4, the confusion matrix for configuration 2 is shown in Table 5.2.

Table 5.1: C4.5 performance on Accent prediction using Roth POS information

Configuration	Training Errors	Testing Errors
1	3922 (20.6%)	417 (20.2%)
2	3529 (18.2%)	378 (18.3%)
3	3424 (17.7%)	376 (18.2%)
4	3369 (17.4%)	379 (18.3%)

Table 5.2: Confusion matrix on test set for configuration 2

	Marked Accented	Marked Unaccented
Actually Accented	1062	126
Actually Unaccented	252	628

Table 5.3: C4.5 rules performance on Accent prediction using Roth POS information

Configuration	Training Errors	Testing Errors
1	3922 (20.3%)	417 (20.2%)
2	3590 (18.6%)	382 (18.5%)
3	3570 (18.5%)	375 (18.1%)
4	3539 (18.3%)	391 (18.9%)

The same configurations were run through C4.5's rule generator. This caused error rates similar to the tree form of C4.5, as is shown in Table 5.3. The composite ruleset where the performance stops dropping, 3, has a simple ruleset and a confusion matrix as shown in Table 5.4. The ruleset has the output accented unless it falls under any of the following rules:

- $CurrentWordPOS \in \{WP\$, TO, CC, DT, WP, PP\$, EX, IN, PRP, WDT, MD, WRB, RBS\}$
- $(CurrentWordPOS \in \{RBS, VBP, FW, VBZ, VBD, VB, RP\}) \wedge (PriorWordPOS \in \{FW, RP, CD, JJR, VBN, RB, VBG\})$

- $(CurrentWordPOS \in \{RBS, VBP, FW, VBZ, VBD, VB\}) \wedge$ $(PriorWordPOS \in \{FW, VBZ, VBD, RP, CD, JJR, RB, JJ, VBG\})$ $\wedge(NextWordPOS \in \{CC, WP, EX, WDT, MD, VBP, VB, NS, NN\})$
- $(CurrentWordPOS \in \{VBP, VBD, VB\}) \wedge (PriorWordPOS \in$ $\{DT, NN\}) \wedge (NextWordPOS \in \{CC, WP, EX, WDT, MD, VBP,$ $VB, NNS, NN\})$

Table 5.4: Confusion matrix on test set for C45 Rules, configuration 3

	Marked Accented	Marked Unaccented
Actually Accented	1042	146
Actually Unaccented	229	652

For SLIPPER, it is expected the output in configuration 1 will be the same, but the rest subtly different. Slipper achieves accuracy on par with C4.5 as the values in Table 5.5 match closely with those of Table 5.1. For comparison purposes, a modified version of the rules is listed below. Listed below are the boosted rules after transformation into individual rules by hand, which was done for legibility purposes. This was only possible because of the simplicity of the final rules. The current word was marked as accented (default class) unless it matched any of these rules:

- $CurrentWordPOS \in \{TO, CC, WP\$, DT, EX, IN, WP, PP\$, PRP,$ $WDT, FW, WRB, MD\}$
- $(CurrentWordPOS =$ "VBP'') $\wedge (PriorWordPOS \in \{VBG, RB,$ $CD, JJ\})$
- $(CurrentWordPOS \in \{VBP, VBD, VBZ, VB\}) \wedge (PriorWordPOS$ $\in \{VBN, JJR\})$

Table 5.5: SLIPPER performance on Accent prediction using Roth POS information

Configuration	Training Errors	Testing Errors
1	3922 (20.6%)	417 (20.2%)
2	3650 (18.8%)	384 (18.6%)
3	3614 (18.7%)	382 (18.5%)
4	3632 (18.8%)	381 (18.4%)

Due to the simplicity of the attribute set and the trend for a two-word window outperforming a one-word window and being on par with the other setups, QUEST was not run on the data set for accent prediction in the first data run.

Prosodic boundary prediction

For prosodic boundary detection, a number of different configurations were experimented with for C4.5, although not as many were used for any of the other algorithms. The information available was described in Section 4.2, and here *POS* will indicate use of part of speech information and *SYN* will indicate use of Syntactic data. Following the convention that the prosodic break the test is looking at occurs between the current and the previous word, for C4.5, the data sets used were the following:

1. POS from two words before and two words after
2. POS from four words before and word after
3. POS from five words before and word after
4. POS from four words before, one word after, and SYN from word after
5. POS from two words before, two words after, and SYN from word after

The error rates for all five cases in C4.5 are given in Table 5.6, but only case 5 was used in any other learners. The confusion matrix for case 5 follows in Table 5.7.

Table 5.6: C4.5 performance on prosodic boundary prediction using Roth POS and partial Charniak SYN information

Configuration	Training Errors	Testing Errors
1	2437 (12.6%)	299 (14.5%)
2	2014 (10.4%)	398 (19.2%)
3	1643 (8.5%)	323 (15.6%)
4	1481 (7.7%)	248 (12.0%)
5	1732 (9.0%)	229 (11.1%)

Table 5.7: Confusion matrix on prosodic boundary prediction with C4.5, configuration 5

	Marked Boundary	Marked Nonboundary
Actually Boundary	214	181
Actually Nonboundary	48	1625

The ruleset for C4.5 rules was derived from the tree used in configuration 5 in C4.5 rules. SLIPPER and QUEST were also run off of this configuration, with all five splitting criteria for both linear and univariate splits used. The combined results (with only the best QUEST results) are shown in Table 5.8.

Table 5.8: Multiple algorithm performance on prosodic boundary prediction using Roth POS and partial Charniak SYN information

Learner	Training Errors	Testing Errors
C4.5 Rules	2193 (11.3%)	251 (12.1%)
SLIPPER	2212 (11.4%)	237 (11.5%)
QUEST Univariate (G^2)	2098 (10.8%)	246 (11.9%)
QUEST Univariate (χ^2)	2086 (10.8%)	247 (11.9%)
QUEST Linear (G^2)	2065 (10.7%)	243 (11.7%)
QUEST Linear (χ^2)	2165 (11.2%)	244 (11.7%)

The rulesets for SLIPPER and the univariate QUEST are both fairly simple. For SLIPPER, there is no prosodic boundary unless:

- Start Sentence $\in$ Current Word SYN
- (Current Word POS = "CC") $\wedge$ (Start PP $\notin$ Current Word SYN) $\wedge$ ((Start SBar $\in$ Current Word SYN) $\vee$ (Last Word POS $\in$ <NNS, NN>))

- ((Last Word POS ∈ <NNS, NN>) ∨(Start PP ∈ Current Word SYN)) ∧ (Start SBar ∈ Current Word SYN)
- (Last Word POS = "NNS") ∧(Start PP ∈ Current Word SYN)

While in the univariate QUEST, it follows the tree in Figure 5.1.

The confusion matrix for the QUEST linear G^2 test, QUEST univariate χ^2 test, and SLIPPER test are given in Tables 5.9-5.11. These correspond to Table 5.8 from above, with the decision models for SLIPPER and univariate QUEST above.

Table 5.9: Confusion matrix for QUEST Linear G^2 test

	Marked Boundary	Marked Nonboundary
Actually Boundary	176	219
Actually Nonboundary	24	1649

Table 5.10: Confusion matrix for QUEST Univariate G^2 test

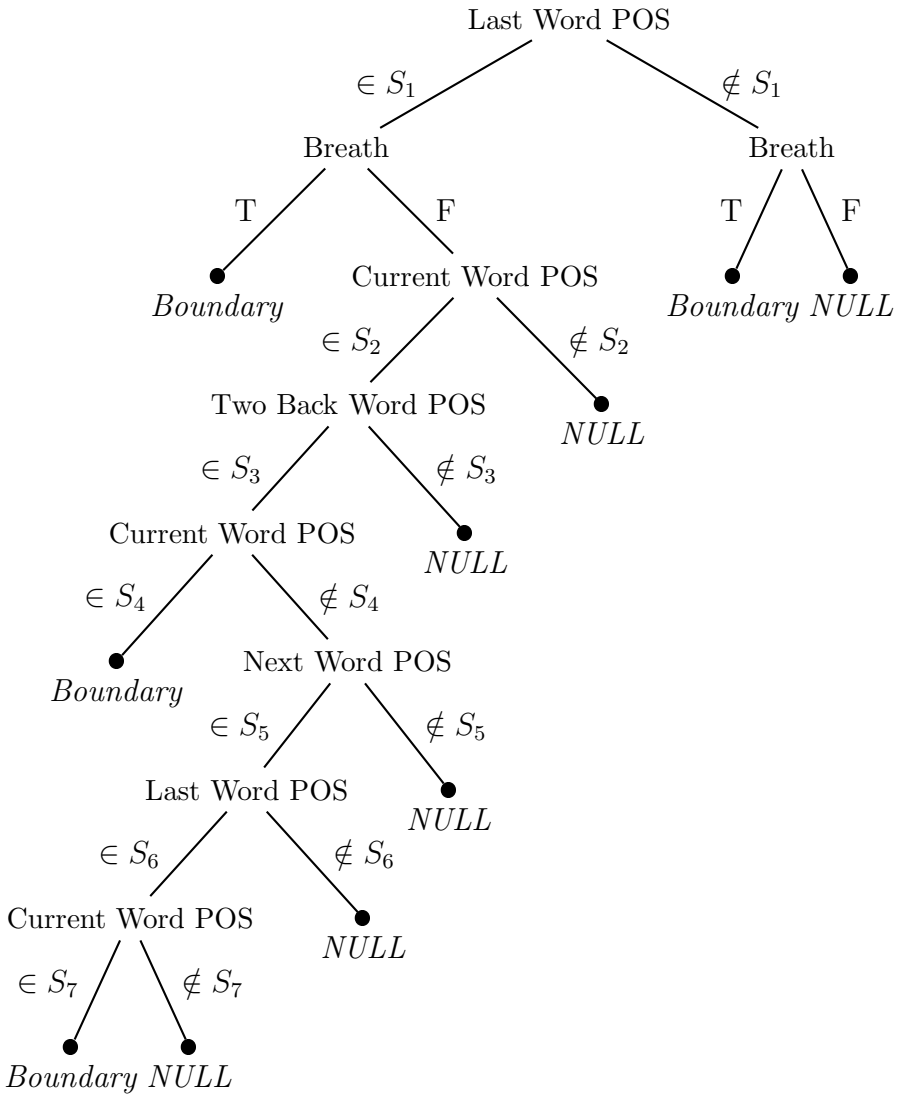
	Marked Boundary	Marked Nonboundary
Actually Boundary	288	207
Actually Nonboundary	39	1634

Table 5.11: Confusion matrix for SLIPPER

	Marked Boundary	Marked Nonboundary
Actually Boundary	202	193
Actually Nonboundary	44	1629

5.2 Second Data Set

For this data set, all syntactic and part of speech information came from the Charniak parser. To reduce the number of inputs, various syntactic categories were folded together and combined as is indicated in Appendix A. For consistency, the same data set was used for both accent prediction as well as prosodic boundary prediction. With the convention of the current word being the word tested for accent (or for being immediately following a prosodic boundary), prosodic information exists for two words back, one word back, current word, and next word. Syntactic information exists on



Set	Contains	Does not contain
S1	NN, NNP, NNS, UH, VBN	CC, CD, DT, EX, FW, IN, JJ, JJR, JJS, MD, NULL, PDT, PP$, PRP, RB, RBR, RBS, RP, TO, VB, VBD, VBG, VBP, VBZ, WDT, WP, WP$, WRB
S2	CC, DT, IN, JJR, JJS, MD, PP$, PRP, RB, TO, VBD, VBG, VBN, VBP, VBZ, WDT, WP, WP$, WRB	CD, EX, JJ, NN, NNP, NNS, PDT, RBR, RBS, RP, VB
S3	CD, FW, JJ, JJR, NN, PP$, TO, VBN, WDT	CC, DT, IN, JJS, NNS, PDT, PRP, RB, RBR, RP, VB, VBD, VBG, VBP, VBZ, WP, WP$, WRB
S4	JJS, PP$, PRP, WDT, WP, WRB	CC, DT, IN, JJR, MD, RB, TO, VBD, VBG, VBN, VBP, VBZ, WP$
S5	CD, JJ, JJR, PP$, PRP, RB, RBR, TO, VB, VBG, VBN, WP, WRB	CC, DT, EX, IN, MD, NN, NNS, PDT, VBD, VBP, VBZ, WDT
S6	NN, NNS, UH	VBN
S7	CC, IN, MD, RB, TO, VBG, VBN, VBP	DT, VBD, VBZ

Figure 5.1: Decision tree for predicting prosodic boundaries on the first data set using the univariate χ^2 QUEST tree builder.

the current word and one word back, with open designating the start of a syntactic unit and close designating the closing of a syntactic unit. The configurations for each recognizer were standard, with the neural network recognizer limited to 10 retrainings with decay (so each time it retrains it is worth less) due to computation time.

The results for every classifier are listed in Table 5.12. It is instructive to look at the confusion matrices for the univariate χ^2 QUEST, C4.5 Rules, and neural network learners, since they are a combination of the best human legible learners and the best overall learner. These matrices can be found in Tables 5.13-5.15.

Table 5.12: Multiple algorithm performance on accent prediction using combined Charniak parser data

Learner	Training Errors	Testing Errors
C4.5	2843 (14.7%)	356 (17.2%)
C4.5 Rules	3336 (17.2%)	365 (17.6%)
SLIPPER	3358 (17.4%)	365 (17.6%)
QUEST Univariate (G^2)	3370 (17.4%)	362 (17.5%)
QUEST Univariate (χ^2)	3370 (17.4%)	362 (17.5%)
QUEST Linear (G^2)	3395 (17.6%)	359 (17.4%)
QUEST Linear (χ^2)	3395 (17.6%)	359 (17.4%)
Neural Network	3333 (16.9%)	350 (16.9%)
Naive Bayes	4241 (21.9%)	442 (21.4%)

Table 5.13: Confusion matrix for univariate χ^2 QUEST test on accent prediction, data set 2

	Marked Accented	Marked Unaccented
Actually Accented	1073	115
Actually Unaccented	247	633

Table 5.14: Confusion matrix for C4.5 Rules test on accent prediction, data set 2

	Marked Accented	Marked Unaccented
Actually Accented	1062	126
Actually Unaccented	239	641

From C4.5 rules (and converting folded classes to their full forms), the current word is accented unless

Table 5.15: Confusion matrix for neural network learner on accent prediction, data set 2

	Marked Accented	Marked Unaccented
Actually Accented	1070	118
Actually Unaccented	232	648

- (Current Word POS $\in$ <AUX, CC, DT, EX, IN, MD, PRP, PRP$, TO, WDT, WP, WP$, WRB>) $\wedge$ (Last Word POS $\notin$ <AUXG, FW, RBS, WP$> $\wedge$ (Next Word POS $\notin$ <CC, RP, UH, WDT, WP$>) $\wedge$ ($\neg$ Current Close <PP, S, SBAR, SINV, SQ, FRAG>)

- (Current Open Number ≥ 3) $\wedge$ (Current Open <WHADJP, WHADVP, WHNP>)

- (Next Word Pos $\in$ <CD, EX>) $\wedge$ ($\neg$ Last Close VP) $\wedge$ (Last Word POS $\in$ <CD, FW, JJ, JJR, JJS, MD, NN, NNS>) $\wedge$ ($\neg$ Current Close <ADJP, ADVP, CONJP, INTJ>)

- (Current Word POS $\in$ <AUXG, FW, RBR, RBS, RP, VBP>) $\wedge$ (Last Word POS $\in$ <AUX, CC, DT, FW, IN, JJ, NN, NNS, PRP, RB, VB, VBD, VBG, VBN, VBP, VBZ, WDT, WP, WRB>) $\wedge$ (Next Word POS $\in$ <FW, JJ, JJR, MD, NN, NNP, NNS, VB, VBN, VBP, VBZ>) $\wedge$ ($\neg$ Current Close PP) $\wedge$ ($\neg$ Last Close <WHADJP, WHADVP, WHNP>)

The tree made by the univariate G^2 QUEST can easily be converted to rule form, with the default being unaccented unless it follows either of the following rules:

- Current Word POS $\in$ <AUXG, CD, FW, JJ, JJR, JJS, NN, NNP, NNS, RB, RBR, VB, VBD, VBG, VBN, VBP, VBZ>

- (Current Word POS = "PDT") $\wedge$(Next Word POS $\notin$ <AUX, CC, EX, IN, NULL, RP, TO, VBP, VBZ, WDT, WP, WP$, WRB >)

- (Current Word POS $\in$ <DT, RBS, RP, UH>) $\wedge$ (Next Word POS $\in$ <AUX, CC, EX, IN, NULL, PRP, RP, TO, VBP, VBZ, WDT, WP, WP$, WRB>)

Prosodic boundary prediction

The same feature set is used to predict prosodic boundaries, with the potential boundary falling between the last word and the current word. The configurations used were also the same, and the results of these test are in Table 5.16.

Table 5.16: Multiple algorithm performance on prosodic boundary prediction using combined Charniak parser data

Learner	Training Errors	Testing Errors
C4.5	1439 (7.4%)	226 (10.9%)
C4.5 Rules	1945 (10.1%)	231 (11.2%)
SLIPPER	1950 (10.1%)	224 (10.8%)
QUEST Univariate (G^2)	2039 (10.5%)	220 (10.6%)
QUEST Univariate (χ^2)	1937 (10.0%)	229 (11.1%)
QUEST Linear (G^2)	1846 (9.6%)	219 (10.6%)
QUEST Linear (χ^2)	1911 (9.9%)	225 (10.9%)
Neural Network	1945 (10.0%)	216 (10.4%)
Naive Bayes	2316 (12.0%)	237 (11.5%)

Here, the best performance is achieved by the neural network followed by the G^2 test for univariate QUEST. The confusion matrices for these tests as well as for linear G^2 QUEST, C4.5, and SLIPPER are given in Tables 5.17-5.21 since linear QUEST, C4.5, and Slipper are not far behind in performance.

Table 5.17: Confusion matrix for Neural Network on prosodic boundary prediction, data set 2

	Marked Boundary	Marked Nonboundary
Actually Boundary	211	184
Actually Nonboundary	32	1641

Table 5.18: Confusion matrix for QUEST Univariate G^2 test on prosodic boundary prediction, data set 2

	Marked Boundary	Marked Nonboundary
Actually Boundary	207	188
Actually Nonboundary	32	1641

Table 5.19: Confusion matrix for QUEST linear G^2 test on prosodic boundary prediction, data set 2

	Marked Boundary	Marked Nonboundary
Actually Boundary	222	173
Actually Nonboundary	46	1627

Table 5.20: Confusion matrix for C4.5 on prosodic boundary prediction, data set 2

	Marked Boundary	Marked Nonboundary
Actually Boundary	246	149
Actually Nonboundary	77	1596

Table 5.21: Confusion matrix for SLIPPER on prosodic boundary prediction, data set 2

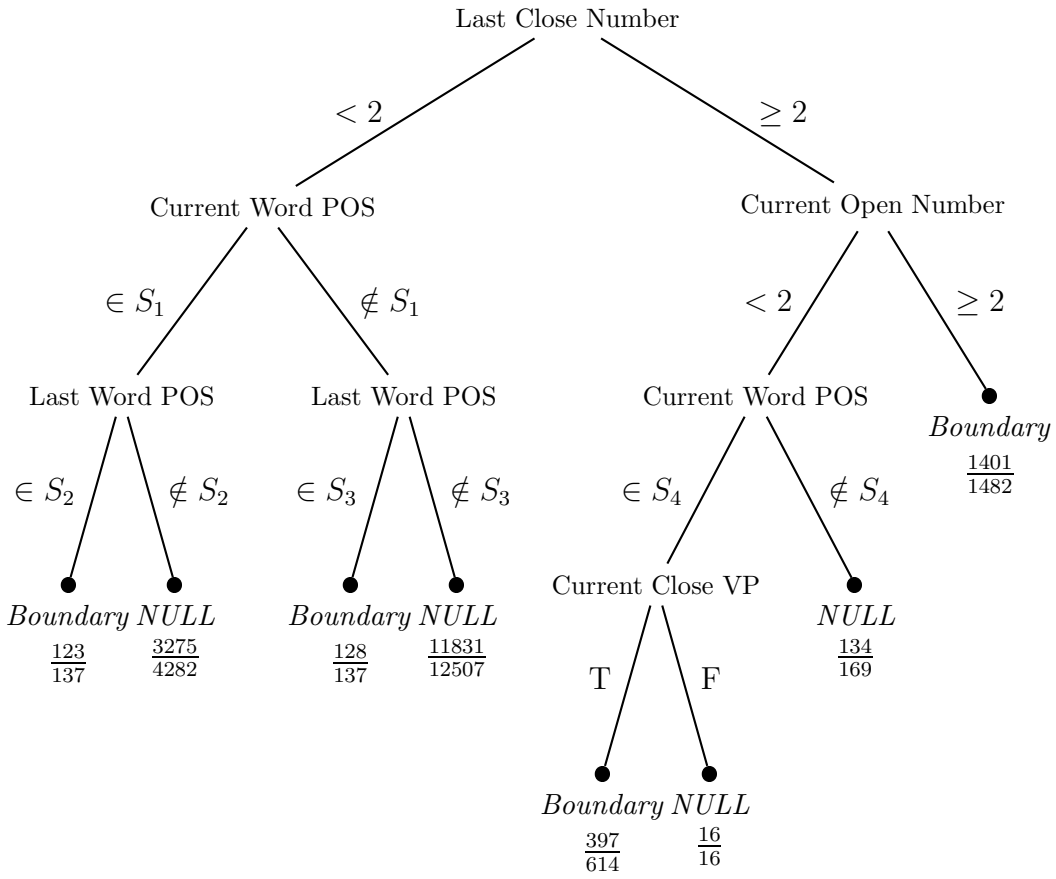
	Marked Boundary	Marked Nonboundary
Actually Boundary	226	169
Actually Nonboundary	55	1618

The legible C4.5 rule generator provides significantly poorer performance than the other learners, but the tree generated by univariate G^2 QUEST gave high accuracy results. This tree is shown in Figure 5.2 with the clustered syntactic information expanded to their original values.

5.3 Third Data Set

This data set was the same as that in data set 2, except here none of the syntactic information was grouped. All of the program settings were the same as in data set 2 to allow direct comparison of results.

The performance of every algorithm on predicting accent follows in Table 5.22.



Set	Contains	Does not contain
S1	AUX, CC, EX, IN, MD, NNP, TO, WDT, WP, WP$, WRB	AUXG, CD, DT, FW, JJ, JJR, JJS, NN, NNS, PDT, PRP, PRP$, RB, RBR, RBS, RP, UH, VB, VBD, VBG, VBN, VBP, VBZ
S2	NULL, RBS	AUX, AUXG, CC, CD, DT, EX, IN, JJ, JJR, JJS, MD, NN, NNP, NNS, PRP, PRP$, RB, RBR, RP, TO, VB, VB, VBD, VBG, VBN, VBP, VBZ, WDT, WP, WRB
S3	NULL	AUX, AUXG, CC, CD, DT, EX, FW, IN, JJ, JJR, JJS, MD, NN, NNP, NNS, PDT, PRP, PRP$, RB, RBR, RBS, RP, TO, VB, VBD, VBG, VBN, VBP, VBZ WDT, WP, WP$, WRB
S4	AUX, CC, DT, IN, MD, PRP, PRP$, RB, RP, TO, VBD, VBG, VBN, VBP, VBZ	CD, EX, JJ, JJR, NN, NNP, NNS, RBR, VB

Figure 5.2: Prosodic boundary prediction tree on dataset 2. The numerator indicates the number of examples in the training set correctly labeled; the denominator indicates overall number of examples.

Table 5.22: Multiple algorithm performance on accent prediction using full Charniak parser data

Learner	Training Errors	Testing Errors
C4.5	2828 (14.6%)	369 (17.8%)
C4.5 Rules	3284 (17.0%)	357 (17.3%)
SLIPPER	3343 (17.3%)	367 (17.7%)
QUEST Univariate (G^2)	3384 (17.5%)	362 (17.5%)
QUEST Univariate (χ^2)	3370 (17.4%)	362 (17.5%)
QUEST Linear (G^2)	3381 (17.5%)	360 (17.4%)
QUEST Linear (χ^2)	3381 (17.5%)	360 (17.4%)
Neural Network	3331 (17.2%)	354 (17.1%)
Naive Bayes	4278 (22.1%)	448 (21.7%)

Here, the performance leader was the neural network, closely followed by the C4.5 Rules algorithm and the two univariate QUEST trees. Univariate QUEST (χ^2) performed slightly better on the training data than the G^2 version. The confusion matrices for C4.5 Rules, univariate QUEST χ^2, and artificial neural network are below in Tables 5.23-5.25.

Table 5.23: Confusion matrix for C4.5 Rules on accent prediction, data set 3

	Marked Accented	Marked Unaccented
Actually Accented	1078	110
Actually Unaccented	247	633

Table 5.24: Confusion matrix for QUEST Univariate χ^2 on accent prediction, data set 3

	Marked Accented	Marked Unaccented
Actually Accented	1073	115
Actually Unaccented	247	633

Table 5.25: Confusion matrix for neural network learner on accent prediction, data set 3

	Marked Accented	Marked Unaccented
Actually Accented	1066	122
Actually Unaccented	232	648

The ruleset given by C4.5 is slightly different from the others, since by its

composition it really is using "else if" clauses. It follows the rules and skips to the very end once a rule has been activated. If no rule has been activated, by default the word is unaccented. The rules are shown below:

- If((¬ Current Open UCP) ∧ (¬Last Open WHADJP)∧(¬Last Close NX) ∧ (Current Word POS ∈ <AUX, CC, DT, EX, IN, MD, PRP, PRP$, RBS, TO, UH, WDT, WP, WP$, WRB>) ∧ (¬Current Close <VP, PP, SBAR>) ∧ (Last Word POS ∈ <NULL, AUX, CC, CD, DT, EX, IN, JJ, JJR, JJS, MD, NN, NNP, NNS, PDT, PRP, PRP$, RB, RBR, RP, TO, VB, VBD, VBG, VBN, VBP, VBZ, WDT, WP, WRB>) ∧ (Next Word POS ∈ <NULL, AUX, AUXG, CD, DT, FW, IN, JJ, JJR, JJS, MD, NN, NNP, NNS, PDT, PRP, PRP$, RB, RBR, RBS, TO, VB, VBD, VBG, VBN, VBP, VBZ, WRB>)) ⇒ Unaccented
- Else If((¬Current Open S) ∧ (Two Back Word POS ∈ <CC,DT,IN,JJ, NN,NNS,RB>) ∧(Next Word POS ∈ <CD,VBN>) ∧ (Last Word POS ∈ <CD,FW,MD,NN>)) ⇒ Unaccented
- Else If((¬Current Open ADVP)∧(¬Last Close SBAR)∧ (Last Close WHADVP)) ⇒ Unaccented
- Else If((Current Open PRT)∧(¬Current Close NP) ∧ (¬Last Open SINV) ∧ (Two Back POS ∈ <AUX,MD,NN,PRP,VB>)) ⇒ Unaccented
- Else If(Current Open SQ) ⇒ Unaccented
- Else If(Current Open WHADJP) ⇒ Unaccented
- Else If((¬Current Close PP)∧(Current Close SBAR)) ⇒ Accented
- Else If((Last Breath)∧(Next Word POS ∈ <AUX, MD>)) ⇒ Accented
- Else If((Next Word POS ∈ <EX,IN,PRP,TO,VBP,VBZ,WP>)∧ (Current Word POS ∈ <DT,MD>)∧ (Last Word POS ∈ <NULL,AUX, CC,CD,IN,JJ,NN,NNS,RB,TO,VB,VBG,VBZ,WDT>)) ⇒ Accented
- Else If((¬Current Open <CONJP,WHADVP,SQ,WHADJP>)∧ (¬Last Open WHADJP)∧ (Current Word POS ∈ <AUXG,CD,FW,JJ, JJR,JJS,NN,NNP,NNS,PDT,RB,RBR,RBS,RP,VB,VBD,VBG,VBN, VBP,VBZ>)) ⇒ Accented

- Else If(Current Close UCP) $\Rightarrow$ Accented
- Else If((Next Word POS = "CC")$\wedge$($\neg$Current Close PP)) $\Rightarrow$ Accented
- Else Unaccented

The tree generated by the univariate QUEST (χ^2) can be simplified to a series of decision rules. Here, the tested word is marked unaccented unless it matches any of the following conditions:

- Current Word POS $\in$ <AUXG, CD, FW, JJ, JJR, JJS, NN, NNP, NNS, RB, RBR, VB, VBD, VBG, VBN, VBP, VBZ>
- (Current Word POS = "PDT")$\wedge$(Next Word POS $\in$AUXG, CD, DT, FW, JJ, JJR, JJS, MD, NN, NNP, NNS, PDT, PRP$, RB, RBR, RBS, UH, VB, VBD, VBG, VBN >)
- (Current Word POS $\in$ <DT, RBS, RP, UH >)$\wedge$(Next Word POS $\in$ <AUX, CC, EX, IN, NULL, PRP, RP, TO, VBP, VBZ, WDT, WP, WP$, WRB>)

Prosodic boundary prediction

The performance of every algorithm on predicting accent follows in Table 5.26.

Table 5.26: Multiple algorithm performance on prosodic boundary prediction using full Charniak parser data

Learner	Training Errors	Testing Errors
C4.5	1467 (7.6%)	231 (11.2%)
C4.5 Rules	1872 (9.7%)	231 (11.2%)
SLIPPER	1904 (9.8%)	210 (10.2%)
QUEST Univariate (G^2)	2010 (10.4%)	220 (10.6%)
QUEST Univariate (χ^2)	2023 (10.5%)	231 (11.2%)
QUEST Linear (G^2)	1704 (8.8%)	227 (11.0%)
QUEST Linear (χ^2)	1872 (9.7%)	229 (11.1%)
Neural Network	1957 (10.1%)	223 (10.8%)
Naive Bayes	2192 (11.3%)	229 (11.1%)

Here, the best performer is SLIPPER, followed closely by the univariate G^2 QUEST learner and the Neural Network learner. The performance of C4.5's rule learner is also fairly good, and has the advantage of being relatively legible. The confusion matrices for all of these learners is shown in Tables 5.27-5.30.

Table 5.27: Confusion matrix for SLIPPER on prosodic boundary prediction, data set 3

	Marked Boundary	Marked Nonboundary
Actually Boundary	219	176
Actually Nonboundary	55	1618

Table 5.28: Confusion matrix for Neural Network on prosodic boundary prediction, data set 3

	Marked Boundary	Marked Nonboundary
Actually Boundary	205	190
Actually Nonboundary	33	1640

Table 5.29: Confusion matrix for QUEST Univariate G^2 test on prosodic boundary prediction, data set 3

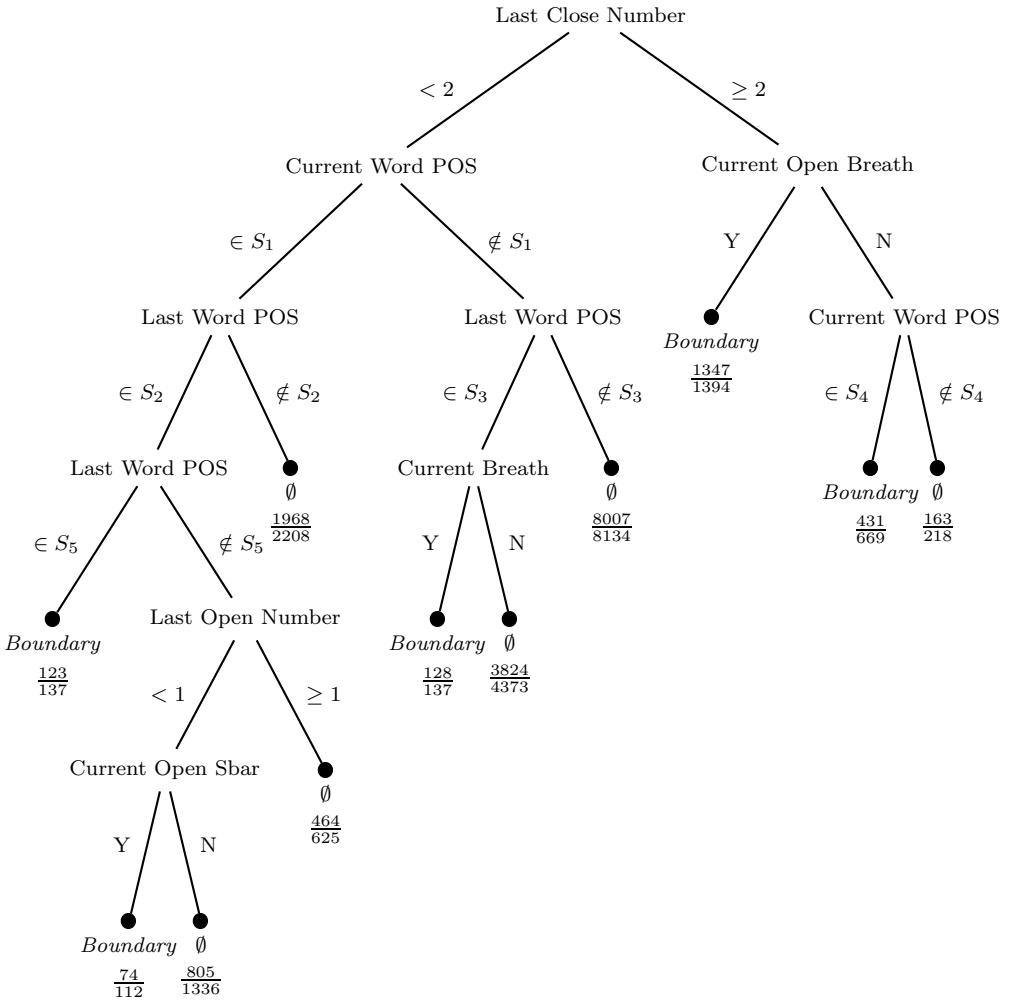
	Marked Boundary	Marked Nonboundary
Actually Boundary	211	184
Actually Nonboundary	36	1637

Table 5.30: Confusion matrix for C45 rules on prosodic boundary prediction, data set 3

	Marked Boundary	Marked Nonboundary
Actually Boundary	219	176
Actually Nonboundary	55	1618

The boosted ruleset provided by SLIPPER is too complicated to be really understood, but the decision tree created by the univariate G^2 QUEST (Figure 5.3) has reasonable performance and is easier to study.

The C4.5 rule generator also has legible output that might provide insight into the relation between syntax and prosodic boundaries. Here, by default a test case is classified as not being a boundary unless it satisfies any of the following conditions, listed in descending order of accuracy:



Set	Contains	Does not contain
S1	AUX, CC, EX, IN, MD, NNP, TO, WDT, WP, WP$, WRB	AUXG, CD, DT, FW, JJ, JJR, JJS, NN, NNS, PDT, PRP, PRP$, RB, RBR, RBS, RP, UH, VB, VBD, VBG, VBN, VBP, VBZ
S2	JJ, JJS, NN, NNS, NULL, RBR, RBS, VB	AUX, AUXG, CC, CD, DT, EX, IN, JJR, MD, NNP, PRP, PRP$, RB, RP, TO, VBD, VBG, VBN, VBP, VBZ, WDT, WP, WRB
S3	NN, NNP, NNS, NULL, PDT, RB, RBR, RP, VB, VBD, VBN, VBP, VBZ	AUX, AUXG, CC, CD, DT, EX, FW, IN, JJ, JJR, JJS, MD, PRP, PRP$, RBS, TO, VBG, WDT, WP, WP$, WRB
S4	AUX, CC, DT, IN, MD, PRP, PRP$, RB, RP, TO, VBD, VBG, VBN, VBZ, WDT, WP, WRB	CD, EX, JJ, JJR, NN, NNP, NNS, RBR, VB, VBP
S5	NULL, RBS	JJ, JJS, NN, NNS, RBR, VB

Figure 5.3: Prosodic boundary prediction tree on dataset 3. The numerator indicates the number of examples in the training set correctly labeled; the denominator indicates overall number of examples.

- Current Breath
- (Next Word POS ∈ <NNP,VBP>)∧ (Last Word POS = “JJ”)∧ (Current Close NP) ∧(¬Current Close PP)∧(¬Current Close VP)
- (Last Close PP)∧(¬Last Close NP)
- (¬Current Open ADJP)∧(Current Word POS ∈ <CC, JJR, PRP$>) ∧(Last Close Number > 0)∧(Last Word POS ∈ <CD, JJ, JJR, JJS, NN, NNP, NNS, PDT, RB, RBR, RBS, RP, VB, VBD, VBG, VBN, VBP, VBZ>)
- (¬Current Open ADVP)∧(¬Current Close NP)∧(Last Close NP) ∧(Current Word POS ∈ <AUX, CC, CD, DT, IN, MD, NNS, PRP, RB, TO, VBD, VBG, VBN, VBP, VBZ, WDT, WRB>)
- (Current Open WHADVP)∧(Last Word POS ∈ <CD, JJ, JJR, JJS, NN, NNP, NNS, VB, VBG, VBN, VBP>)
- (¬Last Breath)∧(Current Open WHNP) ∧ (Next Word POS ∈ <AUX, MD, NN, PRP, PRP$, RB, VBD, VBP, VBZ, WRB>) ∧ (Last Word POS ∈ <CD, JJ, JJR, JJS, NN, NNP, NNS, PDT, RB, RBR, RBS, RP, VB, VBD, VBG, VBN, VBP, VBZ>)
- Current Open CONJP
- (Two Back Word ∈ <CD, JJ, JJR, NN, NNP, NNS, PRP$, RB, VBN>) ∧ (¬Current Open ADJP) ∧ (¬Current Close NP) ∧ (¬Last Open NP) ∧ (¬Current Open WHNP) ∧ (¬Current Close PP) ∧ (¬Last Open PP) ∧ (¬Last Open VP) ∧ (¬Last Breath) ∧ (Current Word POS ∈ <AUX, CC, DT, EX, IN, JJS, MD, PRP, PRP$, RB, TO, VBD, VBG, VBN, VBZ, WRB>) ∧ (Last Word POS ∈ <JJ, JJS, NN, NNP, NNS, RBR>)
- (Last Open Numer = 0)∧(Next Word POS ∈ <CC, NNP, VB>)∧ (Current Word POS ∈ <DT, JJ, JJR, NNP, TO>) ∧ (Last Word POS = “NN”) ∧ (¬Current Open SBAR) ∧ (¬Current Close VP)
- (Two Back Word POS ∈ <DT, RB>)∧(Current Open SBAR) ∧ (¬Current Open WHNP)

CHAPTER 6

ANALYSIS

This chapter will include a summary and discussion of the results and the human legible learning structures. This chapter will be divided into a section discussing the prediction of accent and a section discussing the prediction of prosodic boundaries. In general, prosodic boundary prediction performs better than accent prediction with around 90% accuracy compared to around 82.5%. When compared to baseline performance, the accent prediction appears to be superior with 57.4% baseline for accent prediction compared to 80.9% for prosodic boundary prediction.

6.1 Accent Prediction

For accent prediction, the most important factor was always the part of speech of the word. Significant improvement could be achieved by allowing the part of speech from the word immediately before to influence the decision on certain part of speech tags that were fairly ambiguous about indicating accent. This could be seen on data set 1 where the data sets that C4.5 and SLIPPER had access to were specifically designed to highlight relevant information by manipulating which part of speech tags were seen. As reported in Table 6.1, the part of speech of the word can accurately predict the presence of accent about 80% of the time. Including the part of speech information from the word immediately before increases the accuracy to a little under 82% by removing uncertainty with added contextual information.

Investigating the results from C4.5 with a window of the present part of speech and that of the word before, certain relationships can be observed. The decision tree cannot be put in a very compact visual form, and even a

Table 6.1: C4.5 performance on Stress prediction using Roth POS information

Configuration	Training Errors	Testing Errors
(w_i)	3922 (20.6%)	417 (20.2%)
(w_{i-1}, w_i)	3529 (18.2%)	378 (18.3%)
(w_{i-1}, w_i, w_{i+1})	3424 (17.7%)	376 (18.2%)
(w_{i-2}, w_{i-1}, w_i)	3369 (17.4%)	379 (18.3%)

plain text condensed version takes well over a page, so it is not reproduced in this thesis. Observations could still be made from the full tree, and are given below.

For example, a modal verb is not normally accented, but if one occurs right after a conjuction, the word *to*, or an existential *there*, it is accented. Thus, the *was* is predicted to be accented in the utterance "There was a party."

It can also be noted that the majority of words following an existential *there* are predicted to be accented. This makes sense intuitively since the existential *there* is defined as an unaccented *there*, triggering inversion. The two exceptions to this rule are the participles and possesive form of wh- pronouns. Likewise, any word other than a possesive wh- pronoun is predicted to be accented following a conjunction.

The various forms of nouns and adjectives, including gerunds which are considered nouns, are nearly always accented. Verbs and adverbs have the presence of accent determined by the context, although in general more are accented than unaccented.

Another observation that can be made about the C4.5 tree learner is the disparity between training and testing performance for the cases including part of speech information other than that in the target word and the one before. This indicates overtraining, which should be reduced by the conversion into rule based form. This was in fact seen, since the rule output had a more consistent error rate between the train and test sets for the various configurations. Here, the best performance on the test set was achieved using a three-word window with the part of speech information on the target word, the next word, and the prior word.

As before, words that are not nouns, adjectives, verbs, and adverbs (in the general sense) tend to be unaccented. This matches previously published observations since the other words are connecting words and therefore less important for sentence comprehension. The various noun forms and most adjectives are always accented, with the verb and adverbs depending on context. Verbs appear to be accented if licensed by the structure of the verb phrase they are in.

The output of SLIPPER is simpler than that of C4.5, with the common thread that words that are not nouns, adjectives, verbs, or adverbs tend not to be accented. This agrees with the content/function distinction with the words that are not in those four categories nearly always being function words. The output of SLIPPER also more clearly illustrates that, of those four categories, verbs have a higher tendency to be unaccented, with verbs following past participles and comparative adjectives generally unaccented.

Expanding the window as in data sets 2 and 3, it is possible to see if full syntactic information (or a combined subset of syntactic information) increases performance. The main difference between the two data sets was whether or not syntactic categories were combined. There should also be a performance change due to the use of a different syntactic parser, which will cause the part of speech information to be different. By including all of the syntactic information, including syntactic phrase closings, it will be possible to test if Hirschberg was right about syntactic information not affecting performance or if their study was too limited.

Using the smaller attribute set, the performance for all learners hovers around 82.5%. The exceptions to this are the naive Bayesian learner which performs significantly worse than the others at under 79% and the neural network learner which works a little better at 83%.

The tree produced by C4.5 significantly overfits the data, since its performance on the training data is significantly different than its performance on the test set. Ignoring the linear QUEST models (which perform no better than the univariate models despite their reduced legibility), it can be seen that the C4.5 Rule learner, the univariate χ^2 QUEST, and neural network learner have similar confusion matrices with around 1070 of the accented cases marked correctly (about 90%) with differences in the number of unaccented words incorrectly marked.

Here, the usage of syntactic information varies between the Univariate χ^2

QUEST tree learner and the C4.5 Rule learner. The tree built by QUEST only uses part of speech information from the actual word and the next word. Here, most noun, adjective, verb, and adverb forms are always accented. The part of speech classes that have their accent determined by the next word are the determiners, predeterminers, participle, interjection, and superlative adverbs. This means that these classes of words are sometimes accented for emphasis, probably indicating the importance of the coming word.

The C4.5 rule learner was fairly similar by performance, but it used syntactic cues other than the part of speech. For example, the word starting "wh-" phrases was unaccented if the phrase was prominent (as indicated by more than three syntactic phrase openings) but accented if the word did not match any of the other rules. Another example where syntax made significant changes is where the current word is a pronoun, "wh-" word, or a small handful of connecting words. These are generally unaccented unless they are at the close of a sentence (or sentence fragment) or a prepositional phrase. This implies that these parts of speech tend not to be accented unless they conclude a major thought unit.

Expanding the search to include every possible syntactic clause, as in data set 3, has the potential to cause mixed results. This is because it can refine the discovered relationships by finding more specific patterns, but it can also hide them by taking similar syntactic clauses and reducing the apparent effect on accentuation by reducing the number of observations. Looking at the results, it appears that the expanded syntax has very little effect on the accent prediction. This suggests that the syntactic information other than part of speech has very little relevance for prediction of accent, as the literature suggested.

On this data set, once again the univariate χ^2 QUEST tree ignores everything but the part of speech of the current word and the next word. The tree does not contain a single change from the one generated with data set 2. This suggests that any impact of the other aspects of syntax is minor in predicting accent, and even allowing the data to be as specific as possible there are no places where QUEST finds an information gain.

The results from C4.5 Rules are a little trickier to look at, since they are set up in the following structure: Let U denote the set of all rules indicating the word is unaccented and A indicate the rules indicating the word is accented. If an example fulfills the conditions of any rule in A without

fulfilling the conditions of any rule contained in U, it is accented. Otherwise, it is unaccented. It is clear, however, that it follows the trend of having nouns, adjectives, verbs, and adverbs accented with some of the connecting words accented in the proper context. The rules from the composite ruleset for data set 3 are listed in Appendix E with statistics on how often each rule is used and how often it is correct.

6.2 Prosodic Boundary Prediction

As in accent prediction, the first task was to find the effect of the window of words examined on prosodic boundary prediction. For this limited task, the syntactic information was pared down to a small number of categories and only the start of syntactic clauses immediately after the prosodic boundary were investigated. Here, it was found that on POS information alone, the best performance would be a window of the two words after and the two words before. Expanding the window to include more words from before the boundary resulted in worse performance on the test data yet much better performance on the training set. This is a sign of overfitting of the data, which was caused here by too many available categories. Adding in the small syntactic information caused significant gains for both part of speech windows tested, with the best performance achieved by adding the syntactic information to the two word before, two word after window. This raised the accuracy on test data to 88.9%, though it is likely that the tree would not perform that well on unseen cases out of this domain since there was a fairly large disparity in performance on the train and testing data.

Using this configuration in the other learners, it is possible to compare results, reproduced in Table 6.2. The best performing algorithm was SLIPPER (88.5% accuracy) with univariate χ^2 QUEST performing slightly poorer and still being human legible. Looking at the rules for SLIPPER, relations can be explored. As intuition suggests, if the current word is at the start of a syntactic sentence, then it immediately follows a prosodic break. Also, a prepositional phrase starting after a plural noun indicates a prosodic break in between the words. The other two rules caused by SLIPPER are highly interrelated. Combining them, a prosodic break is indicated if the current word is the start of a clause introduced by a subordinating conjunction and

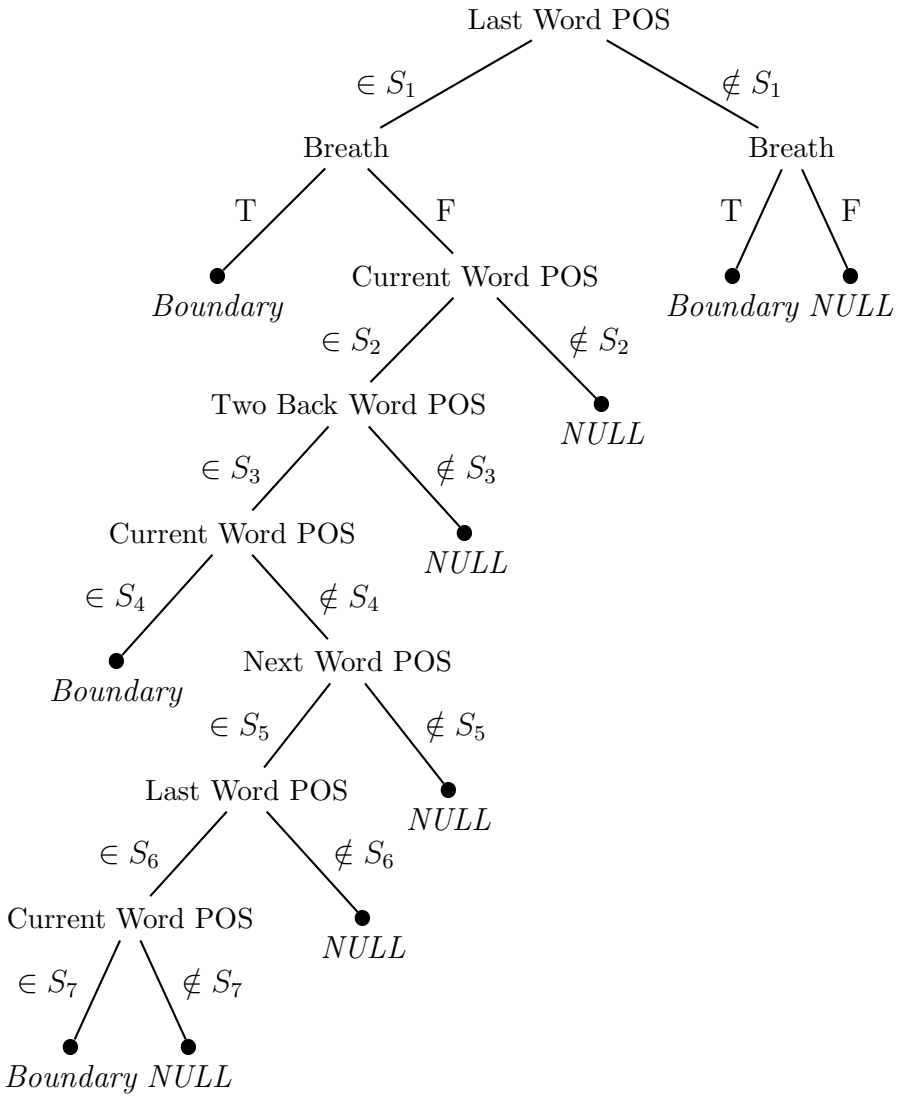
is either the start of a prepositional phrase, the word is a coordinating conjunction, or the word before was a noun (plural or singular). Lastly, it was possible for the boundary to exist if the current word is a conjunction that is after a noun and not the start of a prepositional phrase. Looking at this more carefully, it looks like a clause that was introduced by a subordinating conjunction almost always has a prosodic boundary in between the clauses. The high correlation between the start of either a declarative clause or a clause introduced by a conjunction and prosodic boundary shows a strong relation between prosodic boundaries and syntactic boundaries.

Table 6.2: Mutltiple algorithm performance on prosodic boundary prediction using Roth POS and partial Charniak SYN information

Learner	Training Errors	Testing Errors
C4.5 Rules	2193 (11.3%)	251 (12.1%)
SLIPPER	2212 (11.4%)	237 (11.5%)
QUEST Univariate (G^2)	2098 (10.8%)	246 (12.1%)
QUEST Univariate (χ^2)	2086 (10.8%)	247 (11.9%)
QUEST Linear (G^2)	2065 (10.7%)	243 (11.7%)
QUEST Linear (χ^2)	2165 (11.2%)	244 (11.7%)

The output of the QUEST tree, reproduced below in Figure 6.1, gives indication of order of importance of attributes. The top two nodes are basically interchangeable, since no matter what happens, being immediately after a breath indicates it is just after a prosodic boundary. After this, the next most important attribute is the part of speech of the word before the potential boundary. If the part of speech is a noun, possesive noun, plural noun, interjection, or past participle verb, it is possible for it to be immmediately before a prosodic boundary. If the part of speech immediately after the potential boundary is a pronoun, a "wh-" word, or a superlative adjective, then the part of speech of the word two before the potential boundary is the only other thing that counts. Otherwise, there are more restrictions based on the part of speech of all four words in the window.

Switching to the part of speech tags from the Charniak parser and using the combined syntactic clause level information provides a slight gain in overall performance. The neural network learner has the best overall performance which, although useful for expanding the results to future corpora



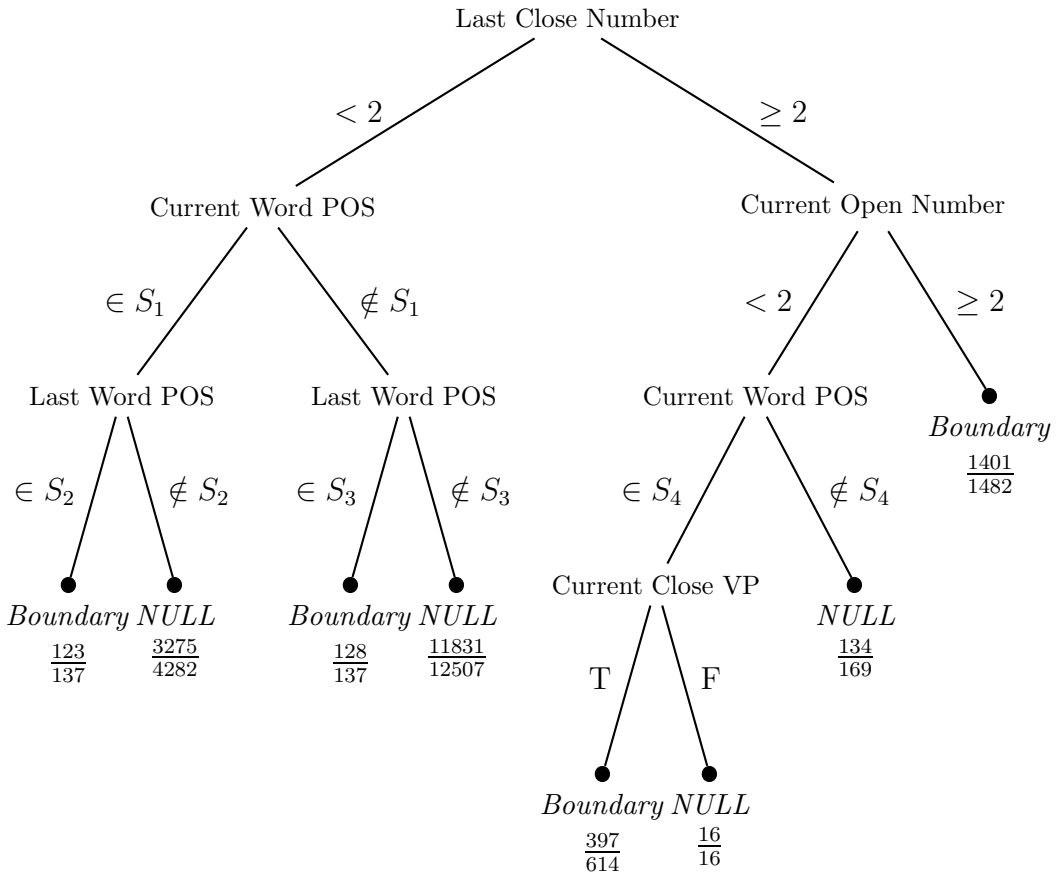
Set	Contains	Does not contain
S1	NN, NNP, NNS, UH, VBN	CC, CD, DT, EX, FW, IN, JJ, JJR, JJS, MD, NULL, PDT, PP$, PRP, RB, RBR, RBS, RP, TO, VB, VBD, VBG, VBP, VBZ, WDT, WP, WP$, WRB
S2	CC, DT, IN, JJR, JJS, MD, PP$, PRP, RB, TO, VBD, VBG, VBN, VBP, VBZ, WDT, WP, WP$, WRB	CD, EX, JJ, NN, NNP, NNS, PDT, RBR, RBS, RP, VB
S3	CD, FW, JJ, JJR, NN, PP$, TO, VBN, WDT	CC, DT, IN, JJS, NNS, PDT, PRP, RB, RBR, RP, VB, VBD, VBG, VBP, VBZ, WP, WP$, WRB
S4	JJS, PP$, PRP, WDT, WP, WRB	CC, DT, IN, JJR, MD, RB, TO, VBD, VBG, VBN, VBP, VBZ, WP$
S5	CD, JJ, JJR, PP$, PRP, RB, RBR, TO, VB, VBG, VBN, WP, WRB	CC, DT, EX, IN, MD, NN, NNS, PDT, VBD, VBP, VBZ, WDT
S6	NN, NNS, UH	VBN
S7	CC, IN, MD, RB, TO, VBG, VBN, VBP	DT, VBD, VBZ

Figure 6.1: Decision tree for predicting prosodic boundaries on the first data set using the univariate χ^2 QUEST tree builder.

and for incorporating auditory information into the prediction, is not useful for understanding purposes. In this case, the output generated by SLIPPER is not convertible to an easily understood form, but fortunately the best performing learner other than the neural network was the univariate G^2 QUEST with 89.4% correctness for QUEST and 89.6% for the neural network. It is interesting to note that the confusion matrices between the various tests are very similar, suggesting that they all make the same mistakes.

For the output tree of univariate G^2 QUEST reproduced below in Figure 6.2, the most important attribute is the number of syntactic clauses that are terminated immediately before the potential boundary. If there are two or more, and there are at least two levels of syntactic clauses starting immediately after the potential boundary, it is a prosodic boundary 95% of the time based on 1482 cases in the training set, representing 37% of the prosodic boundaries in the training set. If the number of syntactic phrases starting is less than two, the next important distinction is the part of speech of the word following the boundary. About half of the potential parts of speech for the word following the boundary were found in the training set, but it is clear that if the part of speech is not a noun or adjective then it is possible to be after a boundary. If the part of speech is not a noun or adjective, and the word does not mark the closing of a verb phrase, then roughly 66% of the time it is immediately after a prosodic boundary. Altogether, just under half of the prosodic breaks are associated with two or more syntactic clauses terminating immediately before.

If the number of syntactic phrase closings is less than two, the next split made is on the part of speech of the current word. This split does not lead to significant change further down the tree, since the next node in both cases tests for the part of speech of the last word and does not have much difference between the two decision sets. In both cases, if the word immediately before was "NULL" (which happens if it is the start of a transcription) it signaled a prosodic boundary. Otherwise, if the last word was a superlative adverb, the part of speech of the current word matters. In this case, it is necessary that the part of speech on the current word be a "wh-" word or a connecting word such as a conjunction. The relative lack of prosodic boundaries that are not nestled between the end of one set of syntactic clauses and the start of the next compared to how many of them occur between clauses suggests that this is the most important attribute attainable through textual analysis.



Set	Contains	Does not contain
S1	AUX, CC, EX, IN, MD, NNP, TO, WDT, WP, WP$, WRB	AUXG, CD, DT, FW, JJ, JJR, JJS, NN, NNS, PDT, PRP, PRP$, RB, RBR, RBS, RP, UH, VB, VBD, VBG, VBN, VBP, VBZ
S2	NULL, RBS	AUX, AUXG, CC, CD, DT, EX, IN, JJ, JJR, JJS, MD, NN, NNP, NNS, PRP, PRP$, RB, RBR, RP, TO, VB, VB, VBD, VBG, VBN, VBP, VBZ, WDT, WP, WRB
S3	NULL	AUX, AUXG, CC, CD, DT, EX, FW, IN, JJ, JJR, JJS, MD, NN, NNP, NNS, PDT, PRP, PRP$, RB, RBR, RBS, RP, TO, VB, VBD, VBG, VBN, VBP, VBZ WDT, WP, WP$, WRB
S4	AUX, CC, DT, IN, MD, PRP, PRP$, RB, RP, TO, VBD, VBG, VBN, VBP, VBZ	CD, EX, JJ, JJR, NN, NNP, NNS, RBR, VB

Figure 6.2: Prosodic boundary prediction tree on dataset 2. The numerator indicates the number of examples in the training set correctly labeled; the denominator indicates overall number of examples.

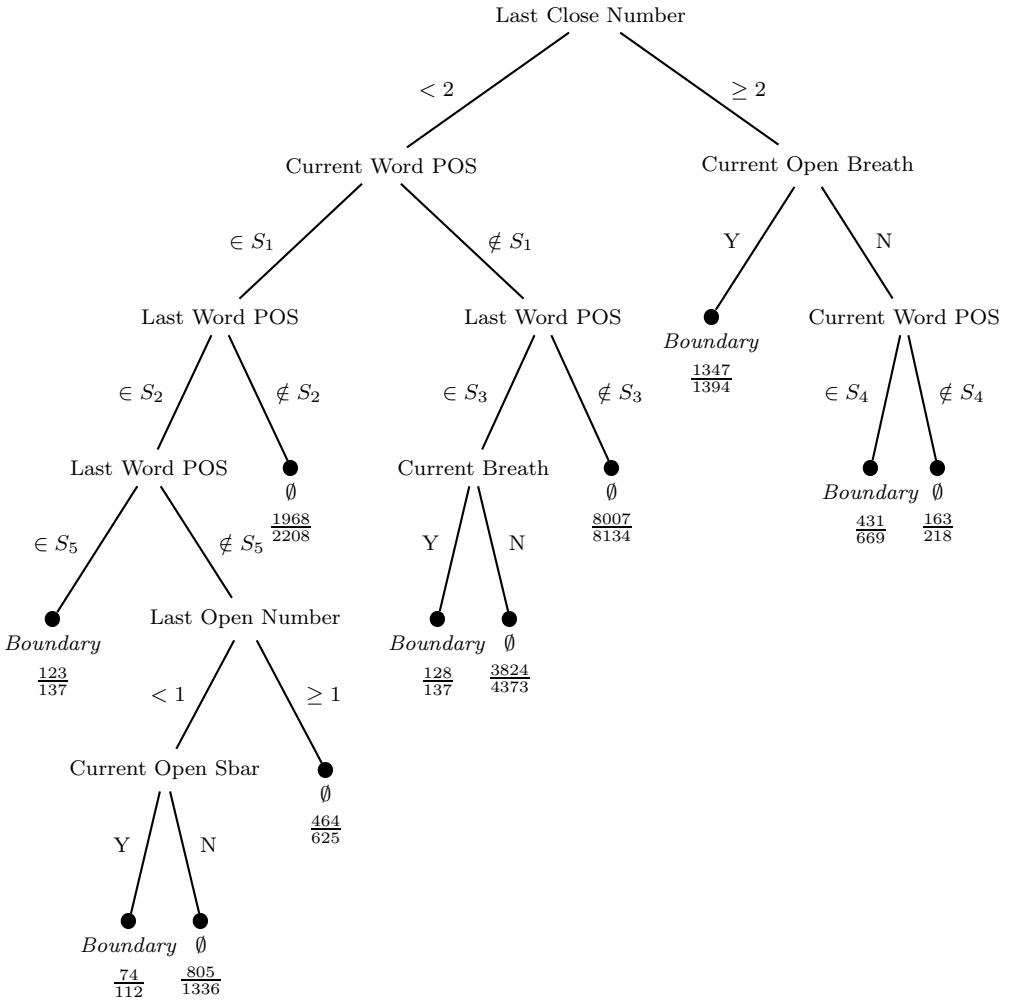
This is an important find, since it indicates a direct dependence of prosodic boundaries on syntactic boundaries.

Making distinctions between every member of the syntactic set produced by the Charniak parser produced very similar results with gains made by some learners (most notably SLIPPER) and some with losses (most notably linear G^2 QUEST). Since the performance basically stayed the same, these might provide more details into subtleties of the labeling, but indicate that some amount of intelligent combination of variables is needed to optimize performance.

Looking at the univariate G^2 QUEST tree, reproduced in Figure 6.3 (which has the same performance on the test set as the one formed on data set 2), the top level is the same with a branch depending on how many syntactic clauses terminated just before the boundary. If there are at least two syntactic clauses ending immediately before the start of a sentence, there is a prosodic boundary 97% of the time on about 1400 of the words in the training set, which acounts for about 35% of the total prosodic boundaries. If there are at least two syntactic clauses ending and the next word does not start a new sentence, it is marked immediately postboundary if it is not a noun, adjective, or base form of a verb.

If there are fewer than two syntactic phrase closings immediately before the potential boundary, the part of speech of the word immediately after the boundary has the highest bearing. Next in importance is the part of speech of the previous word. Depending on the split from the current word, the potential boundary can be labeled at this level alone. If it cannot, then the label is determined either by whether it is at the start of a sentence or if it is the start of a clause introduced by a conjunction that is not immediately after another syntactic clause's start.

Most of the rules generated by C4.5 rules are too detailed to readily comprehend, but some general observations can be made. For example, the start of a sentence again marks a prosodic boundary. The start of a multiword coordinating conjunction is a marker, as is following the end of a prepositional phrase that is not simultaneously the end of a noun phrase. The rest of the rules start being complex, but a general trend is for syntactic clauses ending on the word before or starting on the word after a potential boundary.



Set	Contains	Does not contain
S1	AUX, CC, EX, IN, MD, NNP, TO, WDT, WP, WP$, WRB	AUXG, CD, DT, FW, JJ, JJR, JJS, NN, NNS, PDT, PRP, PRP$, RB, RBR, RBS, RP, UH, VB, VBD, VBG, VBN, VBP, VBZ
S2	JJ, JJS, NN, NNS, NULL, RBR, RBS, VB	AUX, AUXG, CC, CD, DT, EX, IN, JJR, MD, NNP, PRP, PRP$, RB, RP, TO, VBD, VBG, VBN, VBP, VBZ, WDT, WP, WRB
S3	NN, NNP, NNS, NULL, PDT, RB, RBR, RP, VB, VBD, VBN, VBP, VBZ	AUX, AUXG, CC, CD, DT, EX, FW, IN, JJ, JJR, JJS, MD, PRP, PRP$, RBS, TO, VBG, WDT, WP, WP$, WRB
S4	AUX, CC, DT, IN, MD, PRP, PRP$, RB, RP, TO, VBD, VBG, VBN, VBZ, WDT, WP, WRB	CD, EX, JJ, JJR, NN, NNP, NNS, RBR, VB, VBP
S5	NULL, RBS	JJ, JJS, NN, NNS, RBR, VB

Figure 6.3: Prosodic boundary prediction tree on dataset 3. The numerator indicates the number of examples in the training set correctly labeled; the denominator indicates overall number of examples.

6.3 Summary

Overall, use of part of speech information alone is sufficient to achieve fairly high accuracy on predicting word accentuation. It appears that no additional syntactic information plays a significant role in accent prediction, suggesting that any further improvements in prediction would need acoustic information. Strangely, it seems that despite the fact that part of speech is by far the most influential attribute in determining the accent from text sources, it performs better when the part of speech information is obtained as part of a full syntactic parse of a sentence. Prosodic boundary prediction clearly needs syntactic information, but it is unclear at what level of distinction the syntactic information gets fractured too much to be useful. Including information of how many levels of syntactic clause start and end on words seems to be fairly effective at solving this dilemna, but more investigation is needed to see if syntactic clauses that indicate similar phenomena syntactically can be grouped together for better performance on prosodic problems.

These results indicate a direct connection between prosodic boundaries and syntactic boundaries. These results also indicate that a neural network learner can achieve results similar to if not better than those given by tree learners. Neural networks have the benefit of being easier to retrain on new domains with less available training information, and can also incorporate auditory information without much modification to the learner.

Tags were made without the benefit of punctuation marks, indicating the parsing can be done without the benefit of significant human intervention on corpora that have word level transcription but no indicated punctuation.

6.4 Suggestions for Future Work

This work showed high accuracy could be achieved through the use of neural networks and suggests that careful combinations of syntactic tags may increase the accuracy or at least cause minimal loss while reducing computation and training costs. The high accuracy for neural networks is important since neural networks can be adapted to include acoustic information as well as the syntactic information. This can also cause increased performance by including a new set of features that are not as interdependent as the syntactic features. The exact gain caused by inclusion of acoustic information as

well as careful clustering of syntactic tags have yet to be examined. Also, a comparison of various syntactic clustering information to determine which one is optimal has yet to be investigated.

CHAPTER 7

CONCLUSION

This thesis explored the relation between syntax and prosody. This was done using machine learning methods as described in the text; a description of how the learning algorithms work was included. Sources of the implementations used for the algorithms are given in the appendices along with more detailed information about the syntactic information.

It was found that the presence of stress on a word was determined in large part by its part of speech, and prosodic boundaries tended to occur at the same places where there were large syntactic boundaries. It was found that stress prediction appears to have a ceiling at around 83% accuracy that cannot be overcome without adding features outside of the domain of syntax. For the task of predicting stress on transcribed speech to increase the number of corpora available for prosodic research, acoustic information would be the first additional feature to study since it appears that methods based exclusively on text analysis cannot improve the results.

Similar results were found for all tested methods except for naive Bayesian learning which performed significantly worse in all cases. This was true for both prosodic boundary and accent prediction.

The human legible learners suggested a strong correspondence between prosodic boundaries and syntactic boundaries. The neural network learner, although slower to train, performed slightly better on prosodic boundary tasks and should be useful in any future work that incorporates direct auditory information into the learner because of the ease of inclusion of this data and its suitability to adapting to new data using starting positions based on larger domains.

It was also confirmed that syntactic information other than part of speech

plays little role in determining presence of accent in a word, suggesting that any further improvement in accuracy would require accent prediction to include acoustic features. This agrees with Hirschberg's earlier finding that part of speech is the most important attribute for prediction of accent. Hirschberg's finding is modified here by the observation that syntactic information played a role in some exceptional cases, and therefore increased accent prediction accuracy slightly. Examples include the start of a "wh-" phrase coinciding with several syntactic phrases starting, words at the end of major syntactic phrases, and some of the conditions learned by C4.5 Rules from the third data set. The slight outperformance of the neural network learner over the other methods, and the incorporability of acoustic information to this model, suggest that further investigation of using a neural network for accent prediction is warranted.

APPENDIX A

COMPOSITION OF DATA SET 2

Listed in this appendix are the syntactic phrase and part of speech tags generated by the Charniak parser, their frequency, which syntactic tags are combined, and the frequency of the combined set on both the training and testing set. To conserve space, the frequency values are only given for the current word. There will be a disparity between syntactic clause opening and closings, since the start or end of multiple syntactic clauses of the same type are considered to occur only once in this thesis.

Table A.1 shows all of the phrase level syntactic clauses used grouped by the category used to combine them together. The frequency of each of these syntactic groups is shown in Table A.2. The frequency of all of the part of speech tags on the target word is shown in Table A.3.

Table A.1: Syntactic label merging

Label	Category number
ADJP	1
ADVP	1
CONJP	1
INTJ	1
NP	2
NX	2
PP	3
S	4
SBAR	4
SINV	4
SQ	4
FRAG	4
VP	5
WHADJP	6
WHADVP	6
WHNP	6

Table A.2: Frequency of syntactic groups on Current Word

	Training set		Testing set	
Label	Open	Close	Open	Close
1	774	549	91	63
2	5247	2423	561	242
3	2039	1368	214	145
4	3200	848	339	90
5	3713	1476	407	151
6	213	139	24	16

Table A.3: Frequency of part of speech tags on Current Word

Label	Frequency on training set	Frequency on test set
AUX	783	88
AUXG	12	1
CC	583	61
CD	485	53
DT	1870	176
EX	18	3
FW	27	0
IN	2148	227
JJ	1683	208
JJR	96	11
JJS	49	4
MD	270	37
NN	3909	385
NNP	1104	116
NNS	1607	183
PDT	12	0
PRP	401	44
PRP$	276	25
RB	676	82
RBR	20	4
RBS	16	6
RP	98	12
TO	551	60
UH	2	0
VB	571	74
VBD	290	24
VBG	311	24
VBN	436	60
VBP	240	20
VBZ	585	56
WDT	66	9
WP	81	11
WP$	5	0
WRB	63	4

APPENDIX B

COMPOSITION OF DATA SET 3

Listed in this appendix are the syntactic phrase and part of speech tags generated by the Charniak parser, their frequency, which syntactic tags are present, and the frequency of the tags on both the training and testing set. To conserve space, the frequency values are only given for the current word. There will be a disparity between syntactic clause opening and closings, since the start or end of multiple syntactic clauses of the same type are considered to occur only once in this thesis.

The frequency of the syntactic clauses is shown in Table B.1. The frequency of all of the part of speech tags on the target word is shown in Table B.2.

Table B.1: Frequency of syntactic groups on Current Word

	Training set		Testing set	
Label	Open	Close	Open	Close
ADJP	350	305	38	37
ADVP	414	273	52	32
CONJP	10	9	1	1
FRAG	148	21	14	3
INTJ	1	1	0	0
NP	5245	2422	561	242
NX	2	2	0	0
PP	2039	1368	214	145
PRT	94	78	10	8
QP	106	101	13	11
S	2609	823	279	87
S1 = Breath	1667	0	172	0
SBAR	823	443	91	50
SINV	21	3	0	0
SQ	12	1	2	1
UCP	18	12	1	1
VP	3713	1476	407	151
WHADJP	5	3	1	1
WHADVP	56	31	3	1
WHNP	154	107	20	14
X	3	1	0	0

Table B.2: Frequency of part of speech tags on Current Word

Label	Frequency on training set	Frequency on test set
AUX	783	88
AUXG	12	1
CC	583	61
CD	485	53
DT	1870	176
EX	18	3
FW	27	0
IN	2148	227
JJ	1683	208
JJR	96	11
JJS	49	4
MD	270	37
NN	3909	385
NNP	1104	116
NNS	1607	183
PDT	12	0
PRP	401	44
PRP$	276	25
RB	676	82
RBR	20	4
RBS	16	6
RP	98	12
TO	551	60
UH	2	0
VB	571	74
VBD	290	24
VBG	311	24
VBN	436	60
VBP	240	20
VBZ	585	56
WDT	66	9
WP	81	11
WP$	5	0
WRB	63	4

APPENDIX C

SOFTWARE SOURCES

The parsers and learning algorithms used in this thesis were all obtained through freely available implementations. These are listed below.

Roth parser
http://l2r.cs.uiuc.edu/~cogcomp/software/posreg.html

Charniak parser
ftp://ftp.cs.brown.edu/pub/nlparser/parser.tar.gz

C4.5/C4.5 Rules
http://www.cse.unsw.edu.au/~quinlan/c4.5r8.tar.gz

Slipper
http://software.cs.rutgers.edu/slipper/slipper_license.html

QUEST
http://www.stat.wisc.edu/p/stat/ftp/pub/loh/treeprogs/quest/

WEKA (neural network, naive Bayes)
http://prdownloads.sourceforge.net/weka/weka-3-4.exe

APPENDIX D

RAW DATA FROM SEVERAL LEARNERS

In this appendix, the raw data that was output by some of the better learners is given for free use to anyone who wishes to build on the results on this thesis. The best predictors for accent are contained in the body of this thesis, but are reproduced here.

SLIPPER on prediction of prosodic boundaries, compressed label set. The categories used are from Table A.1, with the first label for each category (alphabetically) used to designate the category.

Final hypothesis is:
BOUNDARY :- nextPOS=NNP, CurrentPOS=NNS, TwoBackPOS=IN (+2.43679 : 0.0219308/0.00014203).
BOUNDARY :- TwoBackPOS=NULL, LastPOS=VBD (+2.30277 : 0.00255988/0).
BOUNDARY :- LastCloseNumber≥2, CurrentOpenNumber≥2, CurrentOpenADJP=NULL (+2.26538 : 0.181128/0.00192564).
BOUNDARY :- LastPOS=NULL, CurrentCloseNumber≤1 (+1.90984 : 0.0388489/0.000826863).
BOUNDARY :- TwoBackPOS=PRP$, nextPOS=VBN (+1.90416 : 0.00113922/0).
BOUNDARY :- LastOpenADJP=ADJPopen, LastOpenS=Sopen, CurrentOpenNP=NPopen, CurrentCloseNumber≤1 (+1.8844 : 0.0110439/0.000229641).
BOUNDARY :- LastOpenNumber≥6 (+1.54799 : 0.00123602/3.12269e-05).
BOUNDARY :- LastPOS=NNP, CurrentPOS=DT (+1.34599 : 0.00713138/0.000459029).
BOUNDARY :- LastPOS=NN, CurrentPOS=NNP, nextPOS=NNP, LastOpenNumber≤1, CurrentOpenNP=NULL (+1.25211 : 0.00917543/0.000726264).
BOUNDARY :- LastPOS=VB, nextPOS=NNS, CurrentOpenNP=NPopen, LastOpenNumber≤1 (+0.875357 : 0.00803262/0.0013735).
BOUNDARY :- CurrentPOS=TO, CurrentOpenS=NULL, LastOpenNumber≤1, LastOpenPP=NULL, LastOpenVP=NULL, CurrentOpenNumber≥1, LastCloseADJP=NULL, LastOpenS=NULL (+0.868893 : 0.0163393/0.00285293).
BOUNDARY :- CurrentPOS=CC (+0.855906 : 0.0368855/0.00663807).
BOUNDARY :- LastPOS=NULL, CurrentOpenVP=NULL (+0.79746 : 0.0158084/0.00318732).
BOUNDARY :- LastCloseNumber≥3, CurrentOpenNumber≤1, CurrentOpenPP=NULL, CurrentCloseNumber≤1 (+0.79345 : 0.0369846/0.00754496).
BOUNDARY :- CurrentOpenS=Sopen, LastOpenS=NULL, LastCloseNP=NULL, LastOpenPP=NULL, CurrentOpenADJP=NULL, CurrentOpenNumber≤4, CurrentCloseNumber≤3, LastOpenNumber≤2, CurrentOpenNumber≤4 (+0.744234 : 0.0819472/0.018477).
BOUNDARY :- LastPOS=VBN, TwoBackPOS=AUX (+0.739446 : 0.0180613/0.00409603).
BOUNDARY :- LastPOS=VB, CurrentOpenS=NULL, CurrentCloseNumber≤1, CurrentOpenADJP=NULL, LastOpenNumber≤2 (+0.692571 : 0.0297924/0.0074373).

BOUNDARY :- CurrentOpenADJP=ADJPopen, CurrentCloseADJP=NULL, LastOpenNumber≤1, CurrentPOS=RB, CurrentOpenVP=NULL, CurrentOpenNP=NULL, LastCloseWHADJP=NULL, LastOpenPP=NULL, LastOpenVP=NULL, LastOpenNP=NULL (+0.680954 : 0.0126604/0.00322401).
BOUNDARY :- LastCloseNumber≥1, LastOpenS=NULL, CurrentCloseNumber≤1, LastOpenPP=NULL, LastOpenNumber≤2 (+0.671253 : 0.341101/0.0890732).
BOUNDARY :- CurrentOpenWHADJP=WHADJPopen, LastCloseNP=NULL, LastOpenNumber≤3, LastOpenPP=NULL (+0.60235 : 0.0130701/0.00390009).
BOUNDARY :- LastPOS=VBP (+0.518402 : 0.0164361/0.00581133).
BOUNDARY :- CurrentPOS=IN, LastOpenNumber≤1, CurrentCloseNumber≤0, LastOpenPP=NULL, LastOpenS=NULL (+0.487145 : 0.10895/0.0411081).
BOUNDARY :- TwoBackPOS=RB, LastOpenNP=NULL, CurrentOpenNumber≥1, LastOpenNumber≤1, CurrentOpenADJP=NULL, CurrentCloseNumber≤3, CurrentOpenWHADJP=NULL, LastOpenS=NULL (+0.469902 : 0.0210256/0.00819906).
BOUNDARY :- CurrentPOS=AUX, LastCloseNumber≥1, CurrentOpenNumber≤1, LastOpenADJP=NULL, LastCloseNumber≤8, LastOpenVP=NULL, CurrentOpenVP=VPopen, LastOpenNumber≤3, CurrentCloseNP=NULL, LastOpenNumber≤3 (+0.467684 : 0.0357708/0.0140223).
BOUNDARY :- TwoBackPOS=NNP, LastOpenNumber≤1, CurrentCloseNumber≤1, LastOpenADJP=NULL, LastOpenNP=NULL, LastOpenPP=NULL (+0.440029 : 0.0435421/0.0180443).
BOUNDARY :- CurrentOpenPP=PPopen, LastCloseNumber≤0, CurrentOpenS=NULL, LastOpenPP=NULL, CurrentCloseNumber≤1 (+0.431904 : 0.0360772/0.0151935).
BOUNDARY :- LastCloseNumber≥6 (+0.428116 : 0.0259447/0.0110054).
BOUNDARY :- LastPOS=VBZ, LastCloseNumber≤0 (+0.399829 : 0.0309206/0.013884).
BOUNDARY :- LastPOS=VBG (+0.39241 : 0.0140969/0.00641696).
BOUNDARY :- LastPOS=NNS, CurrentOpenNumber≥1, CurrentCloseNumber≤2, LastOpenNumber≤4 (+0.382046 : 0.0708189/0.0329706).
BOUNDARY :- LastOpenVP=VPopen, CurrentOpenNP=NPopen, CurrentOpenS=NULL, LastOpenNumber≤2 (+0.375905 : 0.050281/0.0236945).
BOUNDARY :- LastOpenADJP=ADJPopen, CurrentOpenVP=NULL (+0.360565 : 0.0265872/0.0129135).
BOUNDARY :- LastPOS=NNP, CurrentCloseNumber≤1 (+0.336799 : 0.0436956/0.0222664).
BOUNDARY :- LastPOS=JJ, CurrentCloseNumber≤2, LastOpenNumber≤2, LastOpenPP=NULL, CurrentCloseADJP=NULL (+0.33188 : 0.0412484/0.0212267).
BOUNDARY :- LastPOS=NN, CurrentCloseNumber≤0 (+0.330754 : 0.146595/0.0756411).
BOUNDARY :- TwoBackPOS=NULL (+0.31801 : 0.0211763/0.0111985).
BOUNDARY :- LastPOS=NNS, CurrentCloseNumber≤2 (+0.235313 : 0.0789482/0.0493023).
BOUNDARY :- LastPOS=NN, CurrentCloseNumber≤1, CurrentOpenPP=NULL (+0.209895 : 0.123245/0.0809859).
BOUNDARY :- CurrentOpenNumber≥3 (+0.186287 : 0.0638136/0.0439569).
BOUNDARY :- LastCloseNumber≥1, CurrentCloseNumber≤1, LastOpenWHADJP=NULL, LastClosePP=NULL (+0.152419 : 0.23961/0.176644).
BOUNDARY :- LastPOS=NN, CurrentOpenNumber≥1 (+0.129358 : 0.131233/0.101311).
BOUNDARY :- LastCloseNumber≥1 (+0.0818133 : 0.28295/0.240237).
BOUNDARY (+7.25076e-05 : 0.500021/0.499948).
default NULL (-2.15387 : 0.0108156/0.805197).
resolution=best.
================================ summary ================================
Train error rate: 10.08% +/- 0.22% (19344 datapoints) <<
Test error rate: 10.83% +/- 0.68% (2068 datapoints) <<
Hypothesis size: 43 rules, 193 conditions
Learning time: 218.14 sec

Univariate G^2 QUEST on accent prediction, compressed label set.

Unaccented unless follows any of the following rules:

- Current Word POS ∈ <AUXG, CD, FW, JJ, JJR, JJS, NN, NNP, NNS, RB, RBR, VB, VBD, VBG, VBN, VBP, VBZ>
- (Current Word POS = "PDT") ∧(Next Word POS ∉ <AUX, CC, EX, IN, NULL, RP, TO, VBP, VBZ, WDT, WP, WP$, WRB >)
- (Current Word POS ∈ <DT, RBS, RP, UH>) ∧ (Next Word POS ∈ <AUX, CC, EX, IN, NULL, PRP, RP, TO, VBP, VBZ, WDT, WP, WP$, WRB>)

SLIPPER on prediction of prosodic boundaries, full label set.

Final hypothesis is:

BOUNDARY :- nextPOS=NNP, LastPOS=JJ, TwoBackPOS=IN, CurrentCloseNumber≤2, CurrentCloseNumber≥2 (+3.08665 : 0.0223039/2.06951e-05).

BOUNDARY :- LastCloseADJP=ADJPclose, CurrentOpenS=Sopen (+2.45827 : 0.00350317/0).

BOUNDARY :- CurrentOpenS1=S1open (+2.26773 : 0.211914/0.00224658).

BOUNDARY :- TwoBackPOS=PRP$, nextPOS=VBN (+1.97967 : 0.00132917/0).

BOUNDARY :- CurrentOpenCONJP=CONJPopen (+1.7418 : 0.0014672/1.99834e-05).

BOUNDARY :- LastOpenNumber≥6 (+1.58154 : 0.00129679/3.00932e-05).

BOUNDARY :- LastOpenADVP=ADVPopen, CurrentOpenVP=NULL, LastCloseNumber≥1, LastCloseWHNP=NULL, CurrentOpenNumber≥1, CurrentCloseNumber≤1 (+1.19126 : 0.017867/0.00162597).

BOUNDARY :- LastPOS=VBP, CurrentOpenSBAR=NULL, LastOpenNumber≤1, CurrentOpenVP=NULL, CurrentOpenS=NULL, CurrentOpenPP=NULL, CurrentOpenPRT=NULL, CurrentOpenUCP=NULL, CurrentCloseNumber≤1 (+1.07445 : 0.00889104/0.00101398).

BOUNDARY :- CurrentOpenWHADVP=WHADVPopen, LastOpenFRAG=NULL, LastClosePP=NULL, LastOpenPP=NULL, LastCloseNumber≤11 (+1.00689 : 0.00396935/0.000507444).

BOUNDARY :- LastOpenADJP=ADJPopen, CurrentOpenNumber≥1, CurrentOpenSBAR=NULL, CurrentOpenVP=NULL, LastOpenUCP=NULL, CurrentOpenADVP=NULL, CurrentOpenQP=NULL, LastOpenS1=NULL (+0.915599 : 0.00925234/0.00146072).

BOUNDARY :- CurrentOpenADVP=ADVPopen, LastCloseNumber≤4, LastOpenNumber≤1, LastOpenVP=NULL, CurrentCloseNumber≤4, LastOpenADJP=NULL (+0.864433 : 0.0127487/0.00224146).

BOUNDARY :- CurrentOpenSBAR=SBARopen, LastOpenS1=NULL, LastCloseNumber≤13, LastOpenPP=NULL, CurrentOpenNumber≤4, CurrentOpenADVP=NULL (+0.858787 : 0.0667558/0.0119615).

BOUNDARY :- CurrentPOS=CC (+0.843376 : 0.0376657/0.00695161).

BOUNDARY :- LastPOS=VBN, TwoBackPOS=AUX, CurrentOpenPRT=NULL, CurrentCloseNP=NULL (+0.794997 : 0.0184637/0.00374466).

BOUNDARY :- LastCloseNumber≥1, LastOpenNumber≤1, CurrentCloseNumber≤1, CurrentOpenADVP=NULL, LastOpenADVP=NULL, LastCloseNumber≤13, LastOpenPRT=NULL, CurrentOpenNumber≤5, LastOpenPP=NULL (+0.790325 : 0.311143/0.0640254).

BOUNDARY :- CurrentPOS=TO, nextPOS=VB, CurrentOpenSBAR=NULL, LastOpenS=NULL (+0.783134 : 0.0193065/0.0040112).

BOUNDARY :- LastPOS=VB, CurrentCloseNumber≤1, LastOpenNumber≤1, CurrentOpenPRT=NULL, CurrentOpenADJP=NULL, CurrentOpenS=NULL (+0.6797 : 0.029474/0.00755015).

BOUNDARY :- LastCloseNumber≥1, CurrentPOS=PRP (+0.673281 : 0.00931924/0.00240512).

BOUNDARY :- CurrentPOS=NNP, CurrentCloseNumber≤0, LastCloseNumber≤1, LastOpenNP=NULL, CurrentOpenS=NULL, CurrentOpenVP=NULL (+0.640749 : 0.0177131/0.00489886).

BOUNDARY :- LastPOS=RP (+0.611947 : 0.00654625/0.00190689).

BOUNDARY :- LastCloseNumber≥2, CurrentCloseNumber≤0 (+0.540832 : 0.079309/0.0268711).

BOUNDARY :- LastPOS=NULL (+0.511557 : 0.0138594/0.00496553).

BOUNDARY :- LastCloseNumber≥1, LastOpenNumber≥2, CurrentOpenS=NULL, LastOpenS1=NULL, LastOpenNumber≤3, CurrentOpenADVP=NULL, CurrentCloseS=NULL (+0.491849 : 0.0435559/0.0162705).

BOUNDARY :- TwoBackPOS=RB, LastOpenNP=NULL, LastOpenNumber≤1, CurrentOpenNumber≥1, LastCloseNumber≤0, CurrentCloseNumber≤3, CurrentOpenADJP=NULL, CurrentOpenPRT=NULL, CurrentOpenADVP=NULL, CurrentOpenSQ=NULL, CurrentOpenWHADJP=NULL, LastOpenSBAR=NULL (+0.490792 : 0.0159438/0.00595825).

BOUNDARY :- TwoBackPOS=NNP, LastOpenNumber≤1, CurrentCloseNumber≤1, LastOpenNP=NULL, LastOpenPP=NULL, CurrentOpenADJP=NULL, LastOpenADVP=NULL, CurrentOpenNumber≤4, CurrentOpenPRT=NULL, CurrentOpenQP=NULL (+0.487957 : 0.0451664/0.0170047).

BOUNDARY :- nextPOS=VBG, LastOpenNumber≤2, CurrentOpenS=NULL, CurrentCloseNumber≤2 (+0.477839 : 0.0142409/0.00546044).

BOUNDARY :- LastPOS=VBZ, LastCloseVP=NULL (+0.447843 : 0.0332252/0.0135515).

BOUNDARY :- CurrentOpenPP=PPopen, LastOpenNumber≤2, LastOpenPP=NULL, LastOpenS1=NULL, CurrentOpenSBAR=NULL, LastOpenSBAR=NULL, CurrentOpenUCP=NULL, CurrentCloseNumber≤9, LastClosePRT=NULL, CurrentOpenNumber≤4 (+0.419213 : 0.115449/0.0499046).

BOUNDARY :- LastPOS=NN, CurrentCloseNumber≤0, CurrentOpenPP=NULL (+0.407616 : 0.104533/0.0462453).

BOUNDARY :- LastPOS=RB, CurrentCloseNumber≤0 (+0.381382 : 0.0224029/0.0104344).

BOUNDARY :- LastPOS=NN, CurrentCloseNumber≤1, LastCloseS=NULL, LastCloseQP=NULL, CurrentOpenVP=NULL, CurrentOpenPP=NULL, LastCloseNumber≤5, CurrentOpenNumber≤3, LastOpenSBAR=NULL, LastOpenUCP=NULL, LastOpenADVP=NULL, CurrentOpenADJP=NULL (+0.357525 : 0.0709678/0.0347019).

BOUNDARY :- LastOpenVP=VPopen, CurrentOpenVP=NULL, CurrentOpenADVP=NULL, CurrentOpenNumber≥1, CurrentOpenADJP=NULL, LastCloseS=NULL, CurrentCloseNumber≤5, LastOpenNumber≤3, CurrentOpenS=NULL, LastCloseNumber≤0, CurrentOpenNumber≤3, CurrentOpenPRT=NULL, CurrentOpenUCP=NULL, CurrentClosePP=NULL, CurrentCloseADVP=NULL, CurrentCloseSBAR=NULL (+0.348908 : 0.0830403/0.0413138).

BOUNDARY :- LastPOS=NNS, CurrentOpenNumber≥1 (+0.344791 : 0.0690535/0.0346372).

BOUNDARY :- LastPOS=NNP, CurrentCloseNumber≤1 (+0.334412 : 0.043776/0.0224143).

BOUNDARY :- LastPOS=VBG (+0.32918 : 0.013751/0.00710641).

BOUNDARY :- LastPOS=NNS, LastCloseNP=NULL, CurrentCloseNumber≤2, CurrentOpenPP=NULL (+0.308465 : 0.0461072/0.0248675).

BOUNDARY :- CurrentOpenS=Sopen, LastCloseNumber≤0, LastOpenSBAR=NULL, LastOpenPP=NULL (+0.302713 : 0.0450152/0.0245594).
BOUNDARY :- LastOpenADJP=ADJPopen (+0.293643 : 0.0137243/0.00761696).
BOUNDARY :- CurrentPOS=AUX, LastCloseNumber≥1 (+0.248136 : 0.034486/0.0209848).
BOUNDARY :- LastOpenNumber≤0, TwoBackPOS=NN (+0.234298 : 0.0581414/0.0363799).
BOUNDARY :- LastCloseNumber≥1, CurrentOpenS=NULL, CurrentOpenNumber≥1, CurrentOpenS1=NULL, CurrentCloseNumber≤2 (+0.198282 : 0.218088/0.146684).
BOUNDARY :- LastOpenNumber≤0, CurrentOpenNumber≥1 (+0.160547 : 0.189195/0.137226).
BOUNDARY :- TwoBackPOS=JJ (+0.130675 : 0.0597886/0.046032).
BOUNDARY :- CurrentOpenPP=PPopen (+0.103014 : 0.105398/0.0857696).
BOUNDARY (+0.000191299 : 0.500153/0.499962).
default NULL (-2.14296 : 0.0110547/0.805197).
resolution=best.
================================ summary ================================
Train error rate: 9.84% +/- 0.21% (19344 datapoints) <<
Test error rate: 10.15% +/- 0.66% (2068 datapoints) <<
Hypothesis size: 45 rules, 238 conditions
Learning time: 567.99 sec

C4.5 Rules accent prediction, full label set:

- If((¬ Current Open UCP) ∧ (¬Last Open WHADJP)∧(¬Last Close NX) ∧ (Current Word POS ∈ <AUX, CC, DT, EX, IN, MD, PRP, PRP$, RBS, TO, UH, WDT, WP, WP$, WRB>) ∧ (¬Current Close <VP, PP, SBAR>) ∧ (Last Word POS ∈ <NULL, AUX, CC, CD, DT, EX, IN, JJ, JJR, JJS, MD, NN, NNP, NNS, PDT, PRP, PRP$, RB, RBR, RP, TO, VB, VBD, VBG, VBN, VBP, VBZ, WDT, WP, WRB>) ∧ (Next Word POS ∈ <NULL, AUX, AUXG, CD, DT, FW, IN, JJ, JJR, JJS, MD, NN, NNP, NNS, PDT, PRP, PRP$, RB, RBR, RBS, TO, VB, VBD, VBG, VBN, VBP, VBZ, WRB>)) ⇒ Unaccented
- Else If((¬Current Open S) ∧ (Two Back Word POS ∈ <CC,DT,IN,JJ, NN,NNS,RB>) ∧(Next Word POS ∈ <CD,VBN>) ∧ (Last Word POS ∈ <CD,FW,MD,NN>)) ⇒ Unaccented
- Else If((¬Current Open ADVP)∧(¬Last Close SBAR)∧ (Last Close WHADVP)) ⇒ Unaccented
- Else If((Current Open PRT)∧(¬Current Close NP) ∧ (¬Last Open SINV) ∧ (Two Back POS ∈ <AUX,MD,NN,PRP,VB>)) ⇒ Unaccented
- Else If(Current Open SQ) ⇒ Unaccented
- Else If(Current Open WHADJP) ⇒ Unaccented
- Else If((¬Current Close PP)∧(Current Close SBAR)) ⇒ Accented
- Else If((Last Breath)∧(Next Word POS ∈ <AUX, MD>)) ⇒ Accented

- Else If((Next Word POS ∈ <EX,IN,PRP,TO,VBP,VBZ,WP>)∧ (Current Word POS ∈ <DT,MD>)∧ (Last Word POS ∈ <NULL,AUX, CC,CD,IN,JJ,NN,NNS,RB,TO,VB,VBG,VBZ,WDT>)) ⇒ Accented
- Else If((¬Current Open <CONJP,WHADVP,SQ,WHADJP>)∧ (¬Last Open WHADJP)∧ (Current Word POS ∈ <AUXG,CD,FW,JJ, JJR,JJS,NN,NNP,NNS,PDT,RB,RBR,RBS,RP,VB,VBD,VBG,VBN, VBP,VBZ>)) ⇒ Accented
- Else If(Current Close UCP) ⇒ Accented
- Else If((Next Word POS = "CC")∧(¬Current Close PP)) ⇒ Accented
- Else Unaccented

APPENDIX E

C4.5 RULES ACCENT PREDICTION

In this appendix, the full set of rules generated by C4.5 Rules on data set 3 for accent performance and their usage on the test set are shown. Rules are listed in order of importance by output class, and the first rule that has its conditions fulfilled dominates the other rules.

Composite ruleset:

Rule 8:
CurrentOpenUCP = NULL
CurrentPOS in {AUX, CC, DT, EX, IN, MD, PRP, PRP$, RBS, TO, UH, WDT, WP, WP$, WRB}
LastPOS in {NULL, AUX, CC, CD, DT, EX, IN, JJ, JJR, JJS, MD, NN, NNP, NNS, PDT, PRP, PRP$, RB, RBR, RP, TO, VB, VBD, VBG, VBN, VBP, VBZ, WDT, WP, WRB}
CurrentCloseVP = NULL
LastOpenWHADJP = NULL
LastCloseNX = NULL
CurrentClosePP = NULL
CurrentCloseSBAR = NULL
nextPOS in {NULL, AUX, AUXG, CD, DT, FW, IN, JJ, JJR, JJS, MD, NN, NNP, NNS, PDT, PRP, PRP$, RB, RBR, RBS, TO, VB, VBD, VBG, VBN, VBP, VBZ, WRB}
⇒ class NULL [86.2%]

Rule 23:
CurrentOpenS = NULL
nextPOS in {CD, VBN}
LastPOS in {CD, FW, MD, NN}
TwoBackPOS in {CC, DT, IN, JJ, NN, NNS, RB}
⇒ class NULL [83.5%]

Rule 3:
CurrentOpenADVP = NULL
LastCloseSBAR = NULL
LastCloseWHADVP = WHADVPclose
⇒ class NULL [70.7%]

Rule 10:
TwoBackPOS in {AUX, MD, NN, PRP, VB}
CurrentOpenPRT = PRTopen
CurrentCloseNP = NULL
LastOpenSINV = NULL
⇒ class NULL [62.3%]

Rule 5:
CurrentOpenSQ = SQopen
⇒ class NULL [61.1%]

Rule 11:
CurrentOpenWHADJP = WHADJPopen
⇒ class NULL [54.6%]

Rule 12:
CurrentClosePP = NULL
CurrentCloseSBAR = SBARclose
⇒ class Accented [89.3%]

Rule 20:
LastOpenS1 = S1open
nextPOS in {AUX, MD}
⇒ class Accented [84.7%]

Rule 31:
nextPOS in {EX, IN, PRP, TO, VBP, VBZ, WP}
CurrentPOS in {DT, MD}
LastPOS in {NULL, AUX, CC, CD, IN, JJ, NN, NNS, RB, TO, VB, VBG, VBZ, WDT}
⇒ class Accented [80.3%]

Rule 21:
CurrentOpenCONJP = NULL
CurrentOpenWHADVP = NULL
CurrentOpenSQ = NULL
CurrentOpenWHADJP = NULL
LastOpenWHADJP = NULL
CurrentPOS in {AUXG, CD, FW, JJ, JJR, JJS, NN, NNP, NNS, PDT, RB, RBR, RBS, RP, VB, VBD, VBG, VBN, VBP, VBZ}
⇒ class Accented [80.2%]

Rule 14:
CurrentCloseUCP = UCPclose
⇒ class Accented [79.4%]

Rule 7:
nextPOS = CC
CurrentClosePP = NULL
⇒ class Accented [61.0%]

Default class: NULL

Rule	Size	Error	Used	Wrong	Advantage
23	4	16.5%	25	4 (16.0%)	1 (4|3) NULL
10	4	37.7%	4	2 (50.0%)	0 (2|2) NULL
5	1	38.9%	2	0 (0.0%)	0 (0|0) NULL
11	1	45.4%	1	0 (0.0%)	0 (0|0) NULL
12	2	10.7%	22	5 (22.7%)	0 (0|0) Accented
20	2	15.3%	16	5 (31.2%)	-2 (2|4) Accented
31	3	19.7%	6	2 (33.3%)	2 (4|2) Accented
21	6	19.8%	1278	235 (18.4%)	782 (1013|231) Accented
7	2	39.0%	3	0 (0.0%)	3 (3|0) Accented

Tested 2068, errors 357 (17.3%) <<

(a)	(b)	<-classified as
1078	110	(a): class Accented
247	633	(b): class NULL

APPENDIX F

SYNTACTIC LABELS

All the contents of this Appendix are copied with minor modification from the part-of-speech tagging and treebank II bracketing style manuals from http://www.cis.upenn.edu/~treebank/home.html [44], [45].

F.1 Bracket Labels

F.1.1 Clause level

S — Simple declarative clause, i.e., one that is not introduced by a (possibly empty) subordinating conjunction or *wh*-word and that does not exhibit subject-verb inversion.

SBAR — Clause introduced by a (possibly empty) subordinating conjunction.

SBARQ — Direct question introduced by a *wh*-word or *wh*-phrase. Indirect questions and relative clauses should be bracketed as SBAR, not SBARQ.

SINV — Inverted declarative sentence, i.e., one in which the subject follows the tensed verb or modal.

SQ — Inverted yes/no question, or main clause of a *wh*-question, following the *wh*-phrase in SBARQ.

F.1.2 Phrase level

ADJP — Adjective Phrase. Phrasal category headed by an adjective (including comparative and superlative adjectives). Example: *outrageously*

expensive.

ADVP — Adverb Phrase. Phrasal category headed by an adverb (including comparative and superlative adverbs). Examples: *rather timidly*, *very well indeed*, *rapidly*.

CONJP — Conjunction Phrase. Used to mark certain "multi-word" conjunctions, such as *as well as, instead of*.

FRAG — Fragment.

INTJ — Interjection. Corresponds approximately to the part-of-speech tag UH.

LST — List marker. Includes surrounding punctuation.

NAC — Not A Constituent; used to show the scope of certain prenominal modifiers within a noun phrase.

NP — Noun Phrase. Phrasal category that includes all constituents that depend on a head noun.

NX — Used within certain complex noun phrases to mark the head of the noun phrase. Corresponds very roughly to N-bar level but used quite differently.

PP — Prepositional Phrase. Phrasal category headed by a preposition.

PRN — Parenthetical.

PRT — Particle. Category for words that should be tagged RP, as described below

QP — Quantifier Phrase (i.e., complex measure/amount phrase); used within NP.

RRC — Reduced Relative Clause.

UCP — Unlike Coordinated Phrase.

VP — Verb Phrase. Phrasal category headed a verb.

WHADJP — *Wh*-adjective Phrase. Adjectival phrase containing a *wh*-adverb, as in *how hot*.

WHADVP — *Wh*-adverb Phrase. Introduces a clause with an ADVP gap. May be null (containing the 0 complementizer) or lexical, containing a *wh*-adverb such as *how* or *why*.

WHNP — *Wh*-noun Phrase. Introduces a clause with an NP gap. May be null (containing the 0 complementizer) or lexical, containing some *wh*-word, e.g., *who*, *which book*, *whose daughter*, *none of which*, or *how many leopards*.

WHPP — *Wh*-prepositional Phrase. Prepositional phrase containing a *wh*-noun phrase (such as *of which* or *by whose authority*) that either introduces a PP gap or is contained by a WHNP.

X — Unknown, uncertain, or unbracketable. X is often used for bracketing typos and in bracketing *the...the*-constructions.

F.2 Part of Speech Tags

CC Coordinating conjunction

CD Cardinal number

DT Determiner

EX Existential *there*

FW Foreign word

IN Preposition or subordinating conjunction

JJ Adjective

JJR Adjective, comparative

JJS Adjective, superlative

LS List item marker

MD Modal

NN Noun, singular or mass

NNS Noun, plural

NNP Proper noun, singular

NNPS Proper noun, plural

PDT Predeterminer

POS Possessive ending

PRP Personal pronoun

PRP$ Possesive pronoun

RB Adverb

RBR Adverb, comparative

RBS Adverb, superlative

RP Participle

SYM Symbol

TO *to*

UH Interjection

VB Verb, base form

VBD Verb, past tense

VBG Verb, gerund or present participle

VBN Verb, past participle

VBP Verb, non-3rd person singular present

VBZ Verb, 3rd person singular present

WDT Wh-determiner

WP Wh-pronoun

WP$ Possesive wh-pronoun

WRB Wh-adverb

REFERENCES

[1] J. J. Godfrey, E. C. Holliman, and J. McDaniel, "Switchboard telephone speech corpus for research and development," in *Preceedings of IEEE Conference on Acoustics, Speech and Signal Processing*, vol. 1, San Francisco, March 1992, pp. 517–520.

[2] P. J. Price, M. Ostendorf, and C. W. Wightman, "Prosody and parsing," in *Proceedings: Speech and Natural Language Workshop*. Morgan Kaufman, Publishers, Inc., October 1989, pp. 5–11.

[3] M. Ostendorf, P. J. Price, and S. Shattuck-Hufnagel, "The Boston University radio news corpus," ECS Department, Boston University, Boston, MA, Tech. Report ECS-95-001, February 1995.

[4] J. Laver, "The production of speech," in *New Horizons in Linguistics*, J. Lyons, Ed. Harmondsworth: Penguin Books, 1970, pp. 53–75.

[5] W. L. Chafe, *Discourse, Consciousness, and Time*. Chicago: University of Chicago Press, 1994, ch. 22, pp. 278–295.

[6] S. C. Arnfield, "Prosody and syntax in corpus based analysis of spoken English," Ph.D. dissertation, The University of Leeds School of Computer Studies, 1994, http://www.linguistics.rdg.ac.uk/staff/Simon.Arnfield/papers/thesis.ps.gz.

[7] J. Hirschberg, "Using discourse context to guide pitch accent decisions in synthetic speech," in *Talking Machines*, G. Baily and C. Benoit, Eds. North-Holland: North-Holland Elsevier Science Publishers B.V., 1992, pp. 367–376.

[8] J. Hirschberg, "Pitch accent in context: Predicting intonational prominence from text," *Artificial Intelligence*, vol. 63, no. 1-2, pp. 305–340, 1993, http://www1.cs.columbia.edu/~julia/papers/aij93.ps.

[9] A. Black and P. Taylor, "Assigning intonation elements and prosodic phrasing for English speech synthesis from high level linguistic input," in *Proceedings ICSLP*, 1994, pp. 715–718.

[10] W. A. Lee, "Prosodic aids to speech recognition," in *Trends in Speech Recognition*, W. A. Lee, Ed. Englewood Cliffs, N.J.: Prentice-Hall, Inc., 1980, pp. 166–205.

[11] M. Swerts and R. Geluykens, "Prosody as a marker of informational flow in spoken discourse," *Language and Speech*, vol. 37, no. 1, pp. 21–43, 1994.

[12] S. Shattuck-Hufnagel, "Phrase-level phonology in speech production planning: Evidence for the role of prosodic structure," in *Prosody: Theory and Experiment*, M. Horne, Ed. Boston: Kluwer Academic Publishers, 2000, pp. 201–230.

[13] A. Wennerstrom, *The Music of Everyday Speech*. Oxford: Oxford University Press, 2001, ch. 4, pp. 67–95.

[14] A. W. Black and P. Taylor, "Assigning phrase breaks from part-of-speech sequences," in *Proc. Eurospeech '97*, Rhodes, Greece, 1997, pp. 995–998.

[15] C. W. Wightman and M. Ostendorf, "Automatic labeling of prosodic patterns," *IEEE Transactions on Speech and Audio Processing*, vol. 2, no. 4, pp. 469–481, October 1994.

[16] M. Swerts, "Prosodic features at discourse boundaries of different strength," *Journal of the Accoustical Society of America*, vol. 101, no. 1, pp. 514–521, January 1997.

[17] C. W. Wightman and M. Ostendorf, "Automatic recognition of prosodic phrases," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 1, April 1991, pp. 321–324.

[18] C. W. Wightman and M. Ostendorf, "Automatic recognition of intonational features," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 1, March 1992, pp. 221–224.

[19] M. Q. Wang and J. Hirschberg, "Automatic classification of intonational phrase boundaries," *Computer Speech and Language*, vol. 6, pp. 175–196, 1992, http://www1.cs.columbia.edu/~julia/papers/csl92.ps.

[20] K. N. Ross, "Modeling of intonation for speech synthesis," Ph.D. dissertation, Boston University College of Engineering, 1995, http://ssli.ee.washington.edu/papers/grad/kr_phd.html.

[21] J. Hirschberg and O. Rambow, "Learning prosodic features using a tree representation," in *Proceedings of Eurospeech*, 2001, pp. 1175–1180, http://www1.cs.columbia.edu/~julia/papers/euro01-phr.ps.

[22] The XTAG Group, "A lexicalized tree adjoining grammar for English," Institute for Research in Cognitive Science, University of Pennsylvania," Tech. Rep. IRCS-98-18, 1998.

[23] J. Hirschberg, "Training accent and phrasing assignment on large corpora," in *Data-Driven Techniques in Speech Synthesis*, R. I. Damper, Ed. Boston: Kluwer Academic Publishers, 2001, pp. 239–273.

[24] T. M. Mitchell, *Machine Learning.* Burr Ridge, IL.: WCB Mcraw-Hill, 1997.

[25] J. R. Quinlan, *C4.5: Programs for Machine Learning.* Boston: Morgan Kaufmann Publishers, 1993.

[26] J. R. Quinlan, "Improved use of continuous attributes in C4.5," *Journal of Artificial Intelligence Research*, vol. 4, pp. 77–90, 1996, ftp://ftp.cs.cmu.edu/project/jair/volume4/quinlan96a.ps.

[27] J. R. Quinlan, "MDL and categorical theories (continued)," in *Proceedings of the 12th International Conference on Machine Learning*, 1995, pp. 464–470, http://www.cse.unsw.edu.au/~quinlan/q.ml95.ps.

[28] W. W. Cohen, "Fast effective rule induction," in *Proceedings of the Twelfth International Conference on Machine Learning*, 1995, pp. 115–123, http://www-2.cs.cmu.edu/~wcohen/postscript/ml-95-ripper.ps.

[29] J. R. Quinlan and R. M. Cameron-Jones, "FOIL: A midterm report," in *Machine Learning: ECML-93*, 1993, pp. 3–20, http://www.cse.unsw.edu.au/~quinlan/q+cj.ecml93.ps.

[30] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[31] R. E. Schapire and Y. Singer, "Improved boosting algorithms using confidence-rated predictions," in *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, 1998, pp. 80–91.

[32] W.-Y. Loh and Y.-S. Shih, "Split selection methods for classification trees," *Statistica Sinica*, vol. 7, pp. 815–840, 1997, http://www.stat.wisc.edu/p/stat/ftp/pub/loh/treeprogs/quest1.7/sinica.pdf.

[33] J. Hartigan and M. A. Wong, "Algorithm as 136: A k-means clustering algorithm," *Applied Statistics*, vol. 28, no. 1, pp. 100–108, 1979, http://links.jstor.org/sici?sici=0035-9254\%281979\%2928\%3A1\%3C199\%3AAA1ACA\%3E2.0.CO\%3B2-V.

[34] W.-Y. Loh and N. Vanichsetakul, "Tree-structured classification via generalized discriminant analysis," *Journal of the American Statistical Association*, vol. 83, pp. 715–728, 1988, http://links.jstor.org/sici?sici=0162-1459\%28198809%2983\%3A403\%3C715\%3ATCVGDA\%3E2.0.CO\%3B2-U.

[35] H. Levene, "Robust tests for equality of variances," in *Contributions to Probability and Statistics*, I. Olkin, S. G. Ghurye, W. Hoeffding, W. G. Madow, and H. B. Mann, Eds. Stanford: Stanford University Press, 1960, pp. 278–292.

[36] Y.-S. Shih, "Families of splitting criteria for classification trees," *Statistics and Computing*, vol. 9, pp. 309–315, 1999, http://lilac.math.ccu.edu.tw/~yshih/papers/sinica.pdf.

[37] G. H. John and P. Langley, "Estimating continuous distributions in Bayesian classifiers," in *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*. San Mateo, 1995, pp. 338–345, http://citeseer.nj.nec.com/john95estimating.html.

[38] M. P. Marcus and B. Santorini, "Building a very large natural language corpora: The penn treebank," *Computational Linguistics*, vol. 19, no. 2, pp. 313–330, 1993.

[39] K. S. et al., "TOBI: A standard for labeling prosody," in *Proceedings of the International Conference on Spoken Language Processing*, 1992, pp. 867–870.

[40] D. Roth and D. Zelenko, "Part of speech tagging using a network of linear separators," in *COLING-ACL '98*, August 1998, pp. 1136–1142, http://l2r.cs.uiuc.edu/~danr/Papers/pos.pdf.

[41] Y. Even-Zohar and D. Roth, "A sequential model for multi-class classification," in *EMNLP'01: The Joint SIGDAT Conference on Empirical Methods in Natural Language Processing*, June 2001, pp. 44–53, http://l2r.cs.uiuc.edu/~danr/Papers/emnlp01.pdf.

[42] E. Charniak, "A maximum-entropy-inspired parser," Brown, Tech. Rep. CS99-12, 1999.

[43] S. Borys, "Recognition of prosodic factors and detection of landmarks for improvements to continuous speech recognition systems," B.S. thesis, Electrical and Computer Engineering Department at the University of Illinois at Urbana-Champaign, 2003, http://www.ifp.uiuc.edu/speech/pubs/2003/borys_thesis.pdf.

[44] A. Bies, M. Ferguson, K. Katz, and R. MacIntyre, "Bracketing guidelines for Treebank II style Penn Treebank project," University of Pennsylvania, Tech. Rep., 1995, ftp://ftp.cis.upenn.edu/pub/treebank/doc/manual/root.ps.gz.

[45] B. Santorini, "Part-of-speech tagging guidelines for the Penn Treebank Project (3rd revision)," Department of Computer and Information Science, University of Pennsylvania, Philadelphia, Tech. Rep. MS-CIS-90-47, Line Lab 178, 1990, ftp://ftp.cis.upenn.edu/pub/treebank/doc/manual/root.ps.gz.