Word Accentuation Prediction Using a Neural Net Classifier *

Taniya Mishra, Emily Tucker Prud'hommeaux, Jan van Santen

Center for Spoken Language Understanding
OGI School of Science & Engineering at OHSU
20000 NW Walker Road, Beaverton, OR 97006, USA
{mishra, emtucker, vansanten}@cslu.ogi.edu

Abstract

Automatic prediction of pitch accent assignment is an important but challenging task in text-to-speech synthesis (TTS). Early work in accent prediction relied on simple word-class distinctions, but recently more sophisticated inductive learning models using multiple features have been applied to the problem. For our neural network accent classifier, we developed a corpus that was labeled according to judgments of accent assignment appropriateness in synthesized speech rather than the usual ToBI annotation guidelines. Because the resulting training set was imbalanced, the baseline neural network we developed for this task had a very high accuracy rate (84%) but performed only slightly better than chance according to our ROC analysis. Balancing our training data using downsizing, oversampling, and cost-based post-processing yielded significant improvement in this informative measure. We anticipate that balance adjustments and the inclusion of more complex features will lead to further improvement.

1. Introduction

Human speech is characterized by modulations in pitch, energy, segment duration, and spectral properties that cause certain words in an utterance to be perceived as more prominent, or accented. Part of being able to speak a language is knowing where to place accents in an utterance; native speakers assign accent with little thought or difficulty. Automatic word accentuation prediction, however, remains a difficult task akin, in the words of one researcher, to mind-reading [1].

Because speech with no pitch accents or incorrectly placed pitch accents can sound unnatural and even confusing, the ability to automatically predict which words in an utterance should be accented is an important task for text-to-speech synthesis (TTS). Word accentuation can be viewed as a classification task: a word in an utterance can be classified as either accented or unaccented. Early accent classification relied on only simple features such as function/content distinctions or part of speech [2]. Recent work, however, has made use of more complex semantic and syntactic features as well as techniques from machine learning, including hidden Markov models, rule-learning algorithms [3], decision trees [4], memory-based learning [5], and neural networks [6].

In this paper, we describe a neural network [7, 8, 9] approach to pitch accent prediction. We have chosen to use neural networks to model word accentuation for a number of reasons. In neural networks, mappings are learned from examples rather than from a priori rules, which require human supervision, or probabilities, which would require a large labeled corpus. Because we plan to use our accentuation model to predict pitch accent placement in novel utterances, we also need a system whose mappings extend effectively to new input data. Neural networks are also able to model nonlinear and complex relationships, which we suspect will be important for this particular task.

2. Description of the corpus

The corpus consists of 16263 words that were obtained from 1030 sentences. The sentences were extracted from the AP newswire using a greedy search algorithm in order to maximally cover the feature space generated by three features: part of speech tag of the previous word, part of speech tag of the current word, and part of speech tag of the next word. There were 72 possible types of part of speech tags. To label the words as accented or unaccented, an innovative iterative perceptual procedure was used in which a labeler used markup tags to indicate accented words in each sentence, synthesized the marked-up sentence using a synthesizer and listened to the resulting synthesized utterance. If the accentuation of any of the words was perceived as incorrect, the markup tags were changed and the rest of the process was repeated until the labeler was satisfied with the word accentuation. The labeler also adjusted the punctuation. For our project, the labeler was a female adult native speaker of American English.

This iterative perceptual labeling procedure that we have outlined has enormous advantages over the commonly used paper and pencil accent labeling procedure. In case of the paper and pencil labeling procedure, the labeler has to *imagine* (or mumble to him/herself) what it would sound like if different words were accented or unaccented. Whereas in case of the iterative perceptual labeling procedure, the labeler will be able to *listen and perceive clearly* the result of accenting or unaccenting different sets of words in a given sentence, which certainly makes the labeling task easier for the labeler, but more importantly, it makes the obtained accent labels more suitable for building a word accent predictor/classifier that will be used for TTS. It is an unavoidable fact that there exists an interaction between accentuation and the TTS-specific acoustic realization of pitch accents. For example, if a TTS system creates ugly pitch accents then it is better to accent fewer words - a reasonable heuristic that the labeler can use *and test* by employing the iterative perceptual labeling procedure, thus making the obtained accent labels more suitable for building a word accent predictor/classifier for text-to-speech synthesis.

* This material is based on work supported by the National Science Foundation under Grants No. 0205731 (ITR: Prosody Generation for Child Oriented Speech Synthesis, Jan van Santen, PI)

Of the 16263 words in the data, 2580 were labeled as accented while 13683 were marked as unaccented. The sentences were passed through the OGI version of the Festival Speech Synthesis System [10] to obtain the following set of eight features per word:

1. word position in the sentence (ranges from 1 to n);
2. type of left phrase boundary (0 or B, boundary type);
3. type of right phrase boundary (0 or B, boundary type);
4. distance from the left phrase boundary (0, 1, or 2);
5. distance from the right phrase boundary (0, 1, or 2);
6. part of speech of the previous word;
7. part of speech of the word; and
8. part of speech of the next word.

The part of speech tags were mapped to seven categories: "noun", "verb", "adjective", "adverb", "number", "pronoun", and "other". This mapping procedure was performed because using all the part-of-speech tags initially produced (18) would drastically increase the amount of training data required. In addition, the tags mapped to the category, "others", consist of function words of which approximately 93% are unaccented.

The accentuation labels are the targets in the classifier, while the low-level syntactic and prosodic features obtained from Festival are the training features of the classifier. A binary encoded training vector is obtained from the accentuation labels such that the class value for each element in the vector is 0 or 1, indicating unaccented and accented, respectively.

The training features, which have a variety of scales, are also normalized so that their values range from 0 to 1. The word position in the sentence is scaled to be between 0 and 1 using the simple scaling formula: $X_{scaled} = (X - X_{min})/(X_{max} - X_{min})$. The remaining features extracted from Festival are encoded as n binary inputs, where n is the number of values that each feature can take. For example, the feature relating to distance from the left phrase boundary can assume three values: 0, 1 or 2. This feature in its normalized form is represented as 3 binary input units, $\{(1\ 0\ 0), (0\ 1\ 0), (0\ 0\ 1)\}$. Normalizing the data in this manner yields a total of 32 scaled feature vectors that will be used as inputs to the classifier.

¿From this corpus, 20% of the data was randomly selected as a test set, and 20% was randomly selected as the validation set. The remaining 60% (= 9757 words) was used as the training set to train the neural net classifier.

3. Measuring classifier performance

The corpus is highly imbalanced. In this two-class corpus, 85% of the corpus is the class of unaccented words, while only 15% of the corpus is accented words, the class of interest in this prediction task. Since the training, test, and validation sets were selected randomly, it can be assumed that each of these sets contains a bias similar to the bias of the whole corpus.

Most classifiers trained on such a biased training set can predict instances of the majority class with a high degree of accuracy but have very low predictive accuracy on instances of the under-represented class. The measure, *classification accuracy* (defined as the ratio of the number of correctly predicted instances to the total number of instances in the test set) is not a good measure for assessing the performance of a classifier trained and tested on such biased sets. Classification accuracy assumes a test set in which both classes are equally represented.

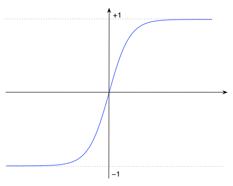

Figure 1: *The tangent-sigmoid (tansig) function*

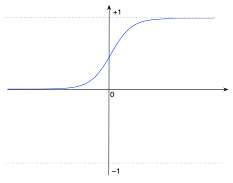

Figure 2: *The log-sigmoid (logsig) function*

However, as in our case, if the two classes are distributed in the ratio 85:15, and the classifier predicts the unaccented class with 100% accuracy but misclassifies the instances of the accented class completely, the classifier accuracy would still be 85%. This metric would not reflect the fact that the class of accented words, which is the class of interest, is completely misclassified.

The metric better suited to such imbalanced datasets is the *area under the ROC curve*. ROC refers to the Receiver Operator Characteristics of a classifier. It is obtained by plotting the *true positive rate* against the *false positive rate*, thus illustrating the tradeoff between these two quantities. The *false positive rate* is the rate at which negative instances were misclassified as positive, while the *true positive rate* is the rate at which positive instances were classified as positive. The ROC curve has a 0-to-1 scale on both axes.

The area under the ROC curve is a good metric for comparing the performance of different classifiers. The larger the area, the better the classifier. A perfect classifier has an area of 1, indicating a 100% true positive rate and 0% false positive rate. A classifier that randomly guesses has a ROC curve that lies on the diagonal line connecting (0, 0) and (1, 1) and has an area of 0.5. For our project, we will be considering the area under the ROC curve as the metric for measuring the performance of the classifiers that predict word accentuation.

4. Baseline classifier

The neural network classifier was created as a three-layer feedforward backpropagation neural network, using the Matlab command, `newff` [11]. This command takes as input the maximum and minimum values of the input nodes, the number of hidden neurons, the number of output neurons, the transfer functions of each layer, and the network training function that updates weight and bias values.

We selected `tansig` [12], the tangent-sigmoid function, depicted in Figure 1 as the transfer function at the hidden layer. The `tansig` function used in the Matlab Neural Networks Toolbox, is mathematically equivalent to the hyperbolic tangent function, whose output values range from -1 to +1. The input data was accordingly normalized to lie on the -1 to +1

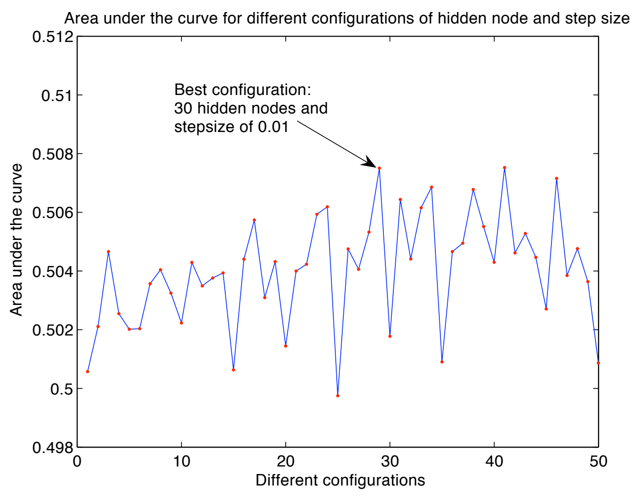

Figure 3: *The change in area under the ROC curve for different configurations of hidden nodes and step size and no momentum values.*

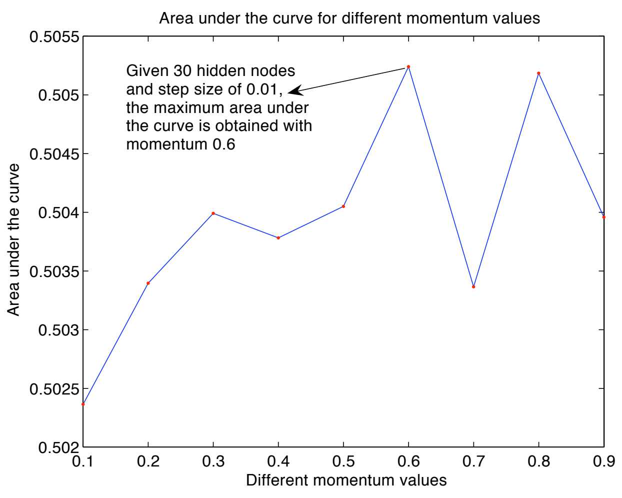

Figure 4: *The change in area under the ROC curve for different momentum values for 30 hidden nodes and step size of 0.01*

scale. The transfer function we selected for the output layer was `logsig` [13], the log-sigmoid transfer function shown in Figure 2, whose output values range from 0 to 1, the desired output range of our classifier. We selected `traingdx` [14] as the network training function. This function updates weight and bias values according to gradient descent momentum and an adaptive step size.

We specified a single output neuron for this classifier. Since the output transfer function is the log-sigmoid function, the output values range from 0 to 1. The output value is treated as the probability that the word is accented, and the word is classified in the following manner: if the output value is less than 0.5, the word is unaccented; if the output value greater than or equal to 0.5, the word is accented.

We also specified 30 hidden layer neurons, a learning rate (or step size) of 0.01, and a momentum value of 0.6. The last three values were obtained from a calibration process that is described in Section 5. The weights and biases were randomly specified by Matlab. Note that we used early stopping when building our neural nets in order to avoid overfitting.

5. Calibration of the neural net classifier

The calibration process was a two-step process. The first step involved using no momentum values and systematically varying the number of hidden layer neurons and the step size (implemented as a double for-loop in Matlab). The number of hidden layer neurons were varied in this way: (5, 10, 15, 20, 25, 30, 33, 40, 45, 50). The step size was varied as follows: (0.00001, 0.0001, 0.001, 0.01, and 0.1). For each choice of hidden layer neurons and step size, the neural net was initialized 10 times with random weight and bias values.

For each initialization, the obtained neural net classifier was used for accent prediction of the word examples in the validation set. Using the predicted values and the known true values, the area under the ROC curve (AUC) was computed. For each configuration of n hidden layer neurons, and step size m, the mean AUC of the 10 initializations was plotted on a graph in Figure 3. This was done for all 50 configurations that emerge from the possible values of step size and number of hidden layer neurons. From the graph, the configuration with 30 hidden layer neurons and a step size of 0.01 was selected for the baseline neural net, because it yielded the maximum mean AUC.

The next calibration step involved finding the best momentum value. Using 30 hidden layer neurons and the step-size of 0.01, the momentum values were systematically varied from 0.1 to 0.9. For each momentum value, the neural net was initialized 10 times with random weight and bias values. Again, the resulting neural net was tested on the validation set. and the mean area under the ROC curve for each momentum value was plotted on a graph, as shown in Figure 4. From this figure, we found that the maximum mean AUC was obtained for momentum value 0.6, which we selected as the momentum value for the baseline.

6. Imbalanced training set

As shown in Figure 5, the baseline classifier has a false positive rate (= 100-true negative rate) of 0.8364% on the validation set and 0.9158% on the test set. The low false positive rate indicates that the baseline classifier predicts unaccented words with high accuracy (99.17% for the validation set and 99.08% for the test set). However, it does not predict accented words well, as indicated by the low true positive rate (2.3857% on the validation set and 3.2505% on the test set). It misclassifies most of them as unaccented. This state of the classifier performance is reflected in the metric, the area under the ROC curve, which value is only a little over 0.5 for both the validation set and the test set, indicating that the baseline classifier performs barely over chance. (The classification accuracies of 84.1992% and 83.6766% on the validation set and test sets, respectively, are a reflection of the high proportion of unaccented examples in the test set, which the classifier learned to predict well, as discussed in Section 3.)

To improve the performance of the baseline classifier which was trained on a highly imbalanced training set, we employ three techniques that have been demonstrated to improve the classifier's ability to predict the minority class: 1) Downsizing, 2) Oversampling, and 3) Cost-based thresholding. Each of these

	Baseline NN		Downsizing NN		Oversampling NN		Cost-based NN	
	Valset	Testset	Valset	Testset	Valset	Testset	Valset	Testset
AUC	0.5077	0.5117	0.6759	0.6419	0.6808	0.6457	0.6597	0.6483
TPR	2.3857	3.2505	62.8231	56.2141	70.7753	62.9063	68.1909	64.2447
FPR	0.8364	0.9158	27.6364	27.8388	34.6182	33.7729	36.2545	34.5788
Acc	84.1992	83.6766	70.8884	69.5973	66.2158	65.6932	64.4328	65.2321

Figure 5: *Results of the performance of the four different types of classifiers on the validation set and the test set. AUC = area under curve, TPR = true positive rate, FPR = false positive rate, Acc = overall accuracy*

methods will be described in the next sections.

7. Solution 1: Downsizing

The first solution that we used to improve the performance of the neural net classifier is called downsizing. Downsizing involves removing random examples of the over-represented class (in our case, unaccented words) from the training set to match the number of examples in the under-represented class (in our case, accented words) [15]. This method is called downsizing because the size of the balanced training set is smaller than the overall training set. This method hinges on the concept that the under-represented class is the class of interest, and all examples of that class need to be retained. In [16], it was found that downsizing is effective in improving the performance of the neural net classifier.

For our project, we downsized the training set in this manner 100 times. For each of the 100 repetitions, the neural net was initialized with random weight and bias values; however, in all repetitions, the 30 hidden layer neurons, step size of 0.01, and momentum value of 0.6 were used. For each repetition, the resulting neural net was tested on the examples in the validation set, and the area under the ROC curve was calculated. After 100 repetitions, the neural net that maximized the area under the ROC curve was selected as the optimal neural net classifier obtained by downsizing our training set, and it was tested on the examples on the test set.

8. Solution 2: Oversampling

The second technique that we used to improve the performance of the neural net classifier was oversampling. Oversampling involves balancing the training set by duplicating random examples of the under-represented class [17] until the number of examples in each class is equal. When the target concept (in our case, "is a word accented?") is represented by fewer examples, oversampling can train the classifier to give more weight to features determining the target concept. This method was also studied in [16], and was found to improve the performance of neural net classifiers. As the size of the training set increased, however, downsizing outperformed oversampling.

For our project, we oversampled the training set in this manner 100 times. For each of the 100 repetitions, the neural net was initialized with random weight and bias values; however, in all repetitions, the 30 hidden layer neurons, step size of 0.01, and momentum value of 0.6 were used. For each repetition, the resultant neural net was tested on the examples in the validation set, and the area under the ROC curve was calculated. After 100 repetitions, the neural net that maximized the area under the ROC curve was selected as the optimal neural net classifier obtained by oversampling on our training set, and it was tested on the examples on the test set.

9. Solution 3: Cost-based thresholding

Cost-based classification is the third solution for improving the performance of the neural net classifier on the word accentuation task. This method looks at the different errors made by the classifier in terms of relative cost [18]. Cost-based classification is a post-process algorithm, i.e., the classification may be performed using any classifier that outputs the probability of an instance being positive or negative. However, the probability threshold that distinguishes the positive instance from the negative instance (so far, at 0.5) is not necessarily symmetric. It takes the cost asymmetry into account as follows. Let:

- p, n be the positive and negative classes,
- Y, N be the classifications produced by a classifier,
- $c(Y, n)$ be the cost of a false positive error,
- $c(N, p)$ be the cost of a false negative error, and
- $CR = c(Y, n)/c(N, p)$ be the cost ratio.

For an instance, E, the classifier outputs the probability of being positive, $p(p|E)$. The probability of being negative, $p(n|E)$, is computed as $p(n|E) = 1-p(p|E)$, assuming a two-class dataset. Thus, according to cost-based classification theory, a word is classified as positive if $p(n|E) * CR < p(p|E)$.

For our project, we used the baseline neural net classifier obtained in Section 4. We systematically varied the cost ratio from 0.0001 and 2 with a step size of 0.001. For each value of the cost ratio, we tested the neural net classifier on the validation set and calculated the area under the ROC curve and plotted it on a graph shown in Figure 6. ¿From this graph, we found that the value of the cost ratio that maximizes the area under the ROC curve is 0.1731. Thus, the cost ratio of 0.1731 in combination with the baseline classifier formed the cost-based classifier.

10. Results and discussion

As the table in Figure 5 shows, the baseline neural net had the highest overall accuracy but extremely low rates of both true positives and false positives. We see that the area under the curve is slightly larger than 0.5, indicating accent assignment with this neural net is slightly better than chance. Earlier we speculated that the imbalanced nature of our data, in which only 15% of elements are classified as positive (accented), would lead to these kinds of results, since the classifier could learn to always return negative and still achieve 85% accuracy. We investigated three different ways to alleviate the imbalance of our data: downsizing, oversampling, and cost-based thresholding.

Figure 5 shows that the highest overall accuracy rate among the three balanced alternatives, though still lower than the baseline, comes from the neural network that was balanced using downsizing (Acc = 69.59% in the test set). The false positive rate of the downsized net is the lowest of the three alternative

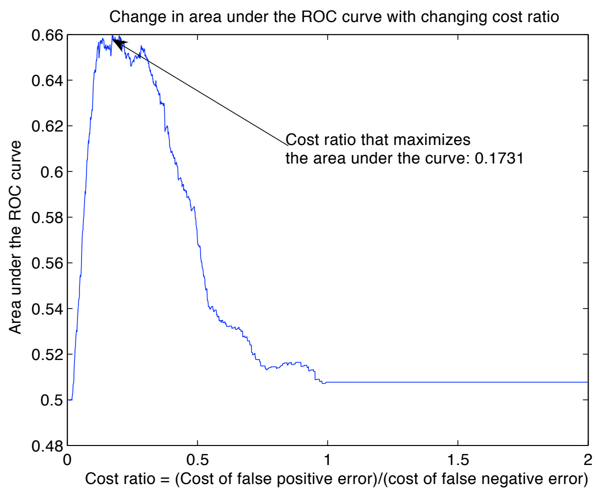

Figure 6: *The change in area under the ROC curve with the change in cost ratio (defined by cost of a false positive error to cost of a false negative error).*

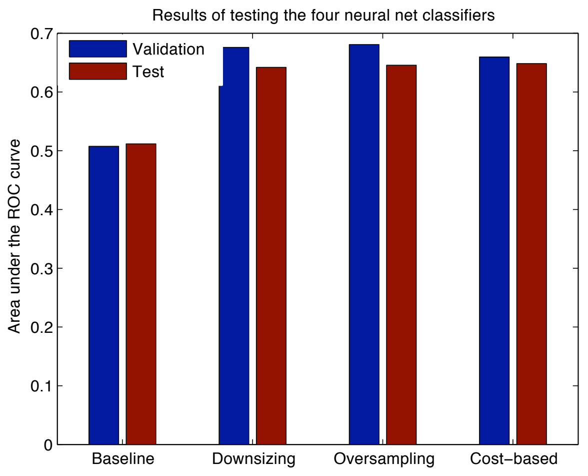

Figure 7: *The performance of the four neural net classifiers in terms of the area under the ROC curve.*

nets. The true positive rate, however, while much higher than that of the baseline neural network, is somewhat lower than the true positive rates of the other two balanced models.

The performance measures of the neural network using oversampling and the neural network using cost-based thresholding are remarkably similar, particularly on the test set. We observe only a slightly higher true positive rate in the cost-based model (64.2447 v. 62.9063), accompanied by a slightly higher false positive rate (34.5788 v. 33.7729). Notably, we see a larger drop in true positive rate from the validation set to the test set in the oversampled neural network than in the other two models. We also observe that the results of the cost-based classifier are more consistent across the validation and test sets that those of the other balanced neural networks.

Figure 7 illustrates perhaps the most interesting observation: our key metric, the area under the ROC curve, is roughly the same for the test set in all three balanced neural networks (0.6419, 0.6457, 0.6483) and is noticeably higher in all three balanced nets than in the baseline (0.5117)

Since all three balanced neural networks had similar AUC measures, choosing the ideal accentuation classifier from the three balanced neural networks will depend on other factors. Accenting a word that should be unaccented is perceived as much more "wrong" than leaving a word unaccented that should be accented. We therefore might prefer the downsized model, since its false positive rate is the lowest of the three. On the other hand, if we are concerned about replicating our findings with new data, we might select the neural network with cost-based thresholding since its performance was more consistent between the validation and test sets.

We had hoped to compare our results to those reported in other recent research in pitch accent prediction using machine learning techniques. Few of these results, however, include measures such as the area under the ROC curve or the false positive and true positive rates. Pan and McKeown [3] report 70-74% accuracy rate with their word-informativeness-based HMM and RIPPER models, a significant improvement over the 52% baseline they assume given the composition of their data set, in which 52% of words are accented. Unfortunately, they do not report other measures of accuracy. In addition, the composition of their data is so different from ours, in which only 15% of words were accented, that a comparison between the two models might not be valid. Similarly, Hirschberg [4] reports overall accuracy of 80-98.3% with both hand-written rule systems and classification and regress trees, but fails to report other measures. Ross and Ostendorf [19], who also used CART techniques, realized similarly high accuracy, but we cannot compare their results to ours since they predicted accent on the syllable-level rather than word level.

One valid point of comparison comes from Marsi et al. [5], who report both higher accuracy (86%) and a higher true positive rate (or recall, 82-88%) than we achieved with our neural network. Their data set, however, was more balanced (one-third of words were accented), and their feature set was far richer.

The most telling results are found in Müller and Hoffman [6], who also used neural networks to model accent prediction. Their word-level neural network classifier achieved 84.5% average overall accuracy, with a low false positive rate (9.3%) and a high true positive rate(85.5%), both of which are noticeably better than the results we found. The HMM they developed using the neural network output for the emission probabilities had very similar accuracy, false positive rate, and true positive rate. We suspect that their training data was more balanced than ours, given that they included secondary and emphatic accents, while we considered only primary accent. In addition, although their feature set was similar to ours, they looked at potentially very long sequences of parts of speech and phrase break locations rather than the simply the properties of the immediately adjacent words.

11. Conclusions

The three techniques we used to balance our unbalanced data set yielded noticeable improvements over our baseline neural network in the measure of area under the ROC curve and the true positive rate, two of our key metrics. We expect to see further improvement by expanding our feature set to include higher-level syntactic properties, such as parent node or depth in the syntactic tree, more precise information about the location of the previous accented word and phrase break [6], and semantic features, such as word informativeness [3, 5] or the features used with success in data-to-speech systems [20]. In addition, we might see gains from using training data that are nearly but not perfectly balanced, given Estabrooks' [21] findings that downsizing and oversampling achieve their lowest error rates when the training set is slightly imbalanced. Finally, we would like to investigate training our neural net with other corpora. Our corpus was labeled to maximize the naturalness of TTS output, which seems to have resulted in an imbalance not seen in the corpora used in other machine learning approaches to accent prediction. Our goal in choosing this labeling process was to create a word accentuation model that would translate elegantly to real-world speech synthesis applications, but perhaps by training with a less skewed set of training data, we can realize the recall and accuracy reported elsewhere in the literature.

12. References

[1] Bolinger, D., "Accent Is Predictable (If You're a Mind-Reader)." Language, 48:3, 633-644, 1972.

[2] Altenberg, B., "Prosodic Patterns in Spoken English: Studies in the Correlation between Prosody and Grammar for Text-to-Speech Conversion", Lund Studies in English, vol. 76, Lund: Lund University Press, 1987.

[3] Pan, S., and McKeown, K., "Word Informativeness and Automatic Pitch Accent Modelling", Proceedings of the Joint SIGDAT Conference on EMNLP and VLC, 148-157, 1999.

[4] Hirschberg, J., "Pitch Accent in Context: Predicting Intonational Prominence from Text", Artificial Intelligence, 63:1-2, 305-340, 1993.

[5] Marsi, E., Busser, G. J., Daelemans, W., Hoste, V., Reynaert, M., and van den Bosch, A., "Combining information sources for memory-based pitch accent placement", Proceedings of the International Conference on Spoken Language Processing, 1273-1276, 2002.

[6] Müller, A., and Hoffmann, R., "A neural network and a hybrid approach for accent label prediction", Proceedings of the 4th ISCA Tutorial and Research Workshop on Speech Synthesis, paper 102, 2001.

[7] McCulloch, W. and Pitts, W., "A logical calculus of the ideas immanent in nervous activity", Bulletin of Mathematical Biophysics 5, 115-33, 1943.

[8] Rosenblatt, F., "The Perceptron: A probabilistic model for information storage and organization in the brain", Psychological Review 65, 386-408, 1958

[9] Werbos, P., The Roots of Backpropagation: From Ordered Deriatives to Neural Networks and Political Forecasting, New York: Wiley, 1994

[10] Black, A., and Taylor, P., "Festival Speech Synthesis System: System documentation (1.1.1)", Human Communication Research Centre Technical Report HCRC/TR-83, Edinburgh, U.K, 1997.

[11] MathWorks, Inc. NEWFF. MATLAB function. Obtained from the MATLAB command 'type newff'. August 2005.

[12] MathWorks, Inc. TANSIG. MATLAB function. Obtained from the MATLAB command 'type tansig. August 2005.

[13] MathWorks, Inc. LOGSIG. MATLAB function. Obtained from the MATLAB command 'type logsig. August 2005.

[14] MathWorks, Inc. TRAINGDX. MATLAB function. Obtained from the MATLAB command 'type traingdx'. August 2005.

[15] Kubat, M., and Matwin, S., "Addressing the curse of imbalanced training sets: one-sided selection", Proceedings of the 14th International Conference on Machine Learning, 179-186, 1997.

[16] Japkowicz, N., "The Class Imbalance Problem: Significance and Strategies", Proceedings of the 2000 International Conference on Artificial Intelligence 1, 111-117, 2000.

[17] Ling, C., and Li, C., "Data mining for direct marketing: Problems and solutions", Proceedings of the 4th International Conference on Knowledge Discovery in Databases (KDD-98), New York, 1998.

[18] Pazzani, M., Merz, C., Ali, K., and Hume, T., "Reducing Misclassification Costs", Proceedings of the International Conference on Machine Learning 1994.

[19] Ross, K., and Ostendorf, M., "Prediction of abstract prosodic labels for speech synthesis", Computer Speech and Language, 10(3), 155-185, 1996.

[20] Theune, M., "Parallelism, Coherence, And Contrastive Accent", Proceedings of Eurospeech 1999, 555-558, 1999.

[21] Estabrooks, A., "A combination scheme for inductive learning from imbalanced data sets", Master's Thesis, Dalhousie University - Daltech, 2000.