"What is... Dengue Fever?" Modeling and Predicting Pronunciation Errors in a Text-to-Speech System

Andrew Rosenberg[1], *Raul Fernandez*[2], *Bhuvana Ramabhadran*[2]

[1]Computer Science Department, Queens College (CUNY), New York, USA
[2]IBM TJ Watson Research Lab, Yorktown Heights, New York, USA

andrew@cs.qc.cuny.edu, {fernanra,bhuvana}@us.ibm.com

Abstract

We propose a system to predict baseform-generation errors in a text-to-speech (TTS) front-end, and aid in the process of customizing the synthesis engine to a novel application with a large, open-ended vocabulary. We motivate the use of the system by using data collected during the deployment of the IBM TTS engine in the Watson Deep Question-Answering system customized to play a game of *Jeopardy!*. We propose a set of features derived from a lexeme's orthography and candidate baseform, and use a variety of learning schemes and data sampling algorithms to address the issue of skewed class priors in the training data. We show that 1) these different approaches provide complementary information that can then be exploited by fusion schemes to improve on the baseline performances, and 2) it is possible to use these techniques to retrieve a list of likely incorrect lexemes so as to reduce the number of tokens that must be vetted before finding and fixing an error.

Index Terms: front-end error modeling, speech synthesis

1. Introduction

In the standard text-to-speech (TTS) architecture, a text-processing front-end (FE) module is responsible for extracting a series of symbolic labels from an input stream of graphemes and passing this information to a back-end (BE) module that makes use of it to produce an output waveform. To some extent, FE and BE components operate independently, and it is common for a particular BE system to rely on an existing FE to expedite development of a synthesis application. FE development requires linguistic resources and expertise that may not be readily available or affordable. Importing a legacy FE is therefore a viable solution for flexible development of language solutions. It is, however, unlikely that such systems will be error free, and thus the issue of maintenance (i.e., identifying and fixing errors) is one that the technology developer still needs to address.

In this work, we investigate the performance of the grapheme-to-phoneme conversion, or baseform generation process of a FE, to model any predictability in its error distribution and use this knowledge either to modify the system or to bypass it via an exception mechanism (e.g., via exception dictionaries that are part of most standard TTS architectures). This work was motivated by a deployment of the IBM Speech Synthesis System [1] in the Watson Jeopardy Grand Challenge in early 2011 [2]. Watson is a deep question-answering system that was optimized to play the game of *Jeopardy!* against two leading human contestants and which, in this configuration, delivers the results of its query via synthetic speech. *Jeopardy!* is a US television quiz show with a question-and-answer format where the contestants are given the clues in the form of an answer and must syntactically frame the response as a WH-question [3]. From a TTS point of view, this task entails synthesizing utterances consisting of only a few words which are prosodically fairly simple. The answers, however, make use of a very large, open-ended base vocabulary containing items (such as named entities, technical and foreign terminology) drawn from disparate areas of human knowledge. Thus, a significant portion of the work involved in customizing the TTS engine to perform well on this task consisted of identifying, and correcting, lexical items for which baseform generation failed.

We investigate, in this work, whether we can apply machine learning techniques to automate the process of error identification from a list of lexical candidates. We assume here such a list is given. In practice, this list of lexemes was derived from archival Jeopardy data by extracting thematic categories and category-dependent vocabularies. We pose this problem as a supervised binary classification task where a set of features derived both from a lexeme's orthography and from its candidate baseform can be used to predict whether or not the proposed baseform is correct. Although machine learning techniques have been applied to modeling and predicting errors in TTS systems before, most of that work has focused on BE errors, such as the perceptual quality of joins [4] or abnormal stress patterns [5]. To the best of our knowledge, the use of such techniques to model FE errors is a novel exploration. We envision the use of these techniques as an aid to the application developer to reduce the amount of time involved in vetting the quality of baseforms (via, e.g., listening or baseform proofreading). Our goal in this investigation is to identify an approach that can predict FE errors *a priori* and focus human intervention on that pre-filtered list with highly likely errors. Since human intervention is still needed to *correct* identified FE errors, some amount of false alarms is acceptable. The value of the system we present is in reducing the number of examples that must be evaluated by a listener. In an ideal setting, the listener assesses exactly zero tokens that are correctly pronounced by the FE. However, any reduction in the number of tokens that must be assessed through human intervention is valuable.

2. Data Preparation

To customize the TTS system to the various domains covered in Jeopardy games, we developed an interface that allowed us to annotate synthesized material by instructing five different listeners (who are also speech experts) to mark the presence of various types of error, including front- and back-end errors. From these data, we extracted a total of 8836 single-word tokens of which 1107 had been marked as having an incorrect baseform. This final list consists of single word units and excludes entries that are all numbers. We are interested in predicting unigram baseform errors in this work and therefore do not include

(i) incorrect baseforms resulting from improper resolution of context-dependent variants (homographs), and (ii) improperly normalized all-digit strings (e.g., phone number vs zip codes), a function we believe pertains to text normalization rather than baseform generation. Some of the entries on this list, however, contain a combination of alphanumeric characters and punctuation (dash and period are considered legal lexeme characters). This labeled dataset provides the training materials for the supervised learning approach to error modeling.

Foreign words and words of foreign etymology represent a particular challenge for baseform generation; the letter-to-sound rules for different languages vary significantly, and can influence the pronunciation in English while being at odds with the internal letter-to-sound mechanism of the FE. To address this class of problems, we explore an approach inspired by the Parallel Phone Recognition Language Modeling [6], a technique used in language identification. Here, we apply the approach to the orthography of a word, in hopes of identifying whether or not an English FE will generate a correct baseform using features derived from parallel *character*-based language models that attempt to reflect different language orthotactics.

To allow the system to extract these features during the training phase, we built various types of character N-gram models for Dutch, English, French, German, Italian, Portuguese and Spanish. The character set consisted of the standard English alphabet (lower and upper case) plus the apostrophe, resulting in a 53-symbol table (all the multi-lingual data was pre-processed to map diacritic-bearing characters to this set, and eliminate any non-conforming punctuation). Using public-domain texts available for these various languages through Project Gutenberg [7], we extracted language-specific vocabularies containing approximately 28K-37K distinct word units, and used these as training material to extract character sequences and build "forward" and "backward" character-level N-gram models to predict, respectively, $p(c_k|c_{k-1}\cdots c_{k-N+1})$ and $p(c_k|c_{k+1}\cdots c_{k+N-1})$. The value of N ranged from 3 to 5, c_k denotes the k^{th} character in a word, and modified Kneser-Ney smoothing was used during training.

Since we may wish to also consider these character sequences in terms of broader classes (with the aim of, e.g., modeling phenomena such as consonant clusters or unlikely vowel combinations, as reflected in the orthography), we introduced 3 different notions of broadclassing, and repeated the process described above for each of these class mappings. The notions of broadclassing we employ are intended loosely, since they are normally defined with respect to phones not characters (and for English they are notably not equivalent). In deciding what class to map a character to, we opted for the most frequent or canonical phonetic realization of that character (although this estimation was not borne out by statistical analysis). The first was a simple Vowel-Consonant mapping (with *A, E, I, O, U, Y* and lower-case variants treated as "vowels"). The second mapping retained this vowel grouping, but differentiated between consonant characters based on the manner of articulation of the associated phonetic realization (yielding the distinction *stop, fricative, affricate, nasal, liquid,* and *glide*). The third mapping restored the 5 vowel distinctions, treated *Y* as a *glide*, and kept the remaining consonant classifications of mapping 2. This process of building 2 types of N-gram models for 7 different languages, 3 different values of N and 4 distinct character sets yields a total of 168 different orthography-based models.

Yet another class of features exploited by our model makes use of the candidate phonetic representation (PR) predicted by the FE, and how statistics of this PR relate to orthography-based statistics. To facilitate extraction of these features, we built N-gram models from the PRs obtained for the English corpus extracted from Gutenberg texts. As before, forward and backward N-gram models were computed with N ranging from 3 to 4. The basic N-gram model was built on a 78-symbol table reflecting the phone inventory used by the FE, where primary-, secondary- and no-stress variants are included for each vowel and diphthong and a syllable boundary "phone" is also included. As before, broad classes, properly defined on phones this time (syllable boundaries are not mapped), are used to obtain different views of the phone sequences. The first phone mapping encodes the standard Vowel-Consonant difference. The second broad class map differentiates between the categories *stop, fricative, affricate, nasal, liquid, tap,* and *vowels* with 3 different stress levels. The last grouping further refines the *vowel* category into *diphthong* and 5 different *monophthongs* (covering clusters around cardinal points in the vowel triangle). Stress-dependency is retained in this last mapping. A total of 16 phone-based models are obtained from 2 different types of N-gram models, 2 values of N, and 4 different mappings.

3. Experimental Configuration

In this section, we describe the feature sets and classification approaches we use to predict FE errors.

The extracted features are based on both the orthography of a given token and its proposed baseform. The initial set of features include surface information drawn from the orthographic form of the token. These include the number of characters, the number of vowels (the set *A, E, I, O, U, Y* and its lower-case variant) and consonants, the ratio of vowels to consonants, the number of non-alphabetic characters, and the number of clustered consonants and vowels. Here a *clustered* consonant or vowel is one that immediately follows another consonant, or vowel, respectively. A similar set of features are generated based on the proposed baseform. Similar to the character-based features, we extract the number of phones, the number of vowels and consonants, the ratio of vowels to consonants and the number of clustered consonants and vowels, irrespective of syllable boundaries. Naturally, in this feature set the definition of "vowel" and "consonant" are based on the phonemic identities contained in the proposed baseform. Moreover, the number of syllables, and the numbers of primary, secondary and unstressed vowels are extracted. We also calculate the ratio of orthographic vowels to baseform syllables, and orthographic to baseform consonants. In addition to these, we extract a set of features based on the character- and phone-based language models (LM) described in Section 2. These include the LM perplexity of 3-, 4-, and 5-gram character models trained on material in each of the seven languages using both forward and reversed character sequences. We calculate the ratio of perplexities between each language pair, holding the order and direction of the model fixed. Each of these orthographic LM features are also extracted using the broad classing strategies described in Section 2. The phone LM perplexities are calculated under 3- and 4-gram models of forward and reverse phone sequences. The phone LMs are constructed only on English material. We calculate the ratio of phone to character LM perplexities for both the 3- and 4-gram forward and reverse models. We repeat this feature extraction using the Vowel/Consonant broad-class representations. These orthographic to phonotactic ratio features measure how much a surface form "looks" like one language against the "Englishness" of the hypothesized baseform.

We explore the use of three classifiers in the prediction of front-end errors: logistic regression, support vector machines

	No Sampling		Under Sampling		SMOTE		Ensemble Sampling	
Classifier	Accuracy	F-measure	Accuracy	F-measure	Accuracy	F-measure	Accuracy	F-measure
J48	83.93±1.12	33.92±4.11	52.62±2.59	29.71±2.25	82.49±0.76	33.82±3.73	70.04±2.04	**36.31±2.74**
Logistic	86.21±1.15	27.33±5.41	53.58±1.77	29.73±1.97	85.86±0.85	27.68±2.37	68.61±2.45	34.84±5.01
SVM	**87.56±0.83**	1.62±2.16	49.20±0.95	30.59±1.97	87.54±0.82	4.25±2.69	70.18±3.06	**36.58±2.66**

Table 1: *Accuracy and F-measure (in %) from classification experiments, including mean and standard deviation from 10-fold CV.*

with linear kernels, and J48 decision trees. We use the Weka [8] implementation of each of these algorithms. We also investigated the use of radial basis functions in the SVM kernel, but found no significant differences in performance when compared to the linear kernel SVM. In the interest of space, we omit presentation and discussion of these results.

In situations where class distributions are heavily skewed, sampling techniques are often used to improve detection of minority class instances. In this work, we explore three techniques: under-sampling, SMOTE [9], and ensemble sampling. Let m_a be the number of majority class training instances and m_i be the number of minority class instances. When under-sampling, a randomly selected set of m_i majority instances are used in training. The SMOTE algorithm over-samples each minority instance with perturbation based on its nearest neighbors to generate an approximately even class distribution in the training data. Ensemble sampling [10] combines qualities of over- and under-sampling. Given a set of training data $N = \left\lfloor \frac{m_a}{m_i} \right\rfloor$ partitions of the majority instances are constructed. These partitions are combined with all m_i minority instances to construct N sets of training data. These training sets are used to train N classifiers. Hypotheses are generated through weighted majority voting decisions from the N ensemble classifiers.

In running these experiments with three classifiers and four sampling settings, we train 12 classifiers to predict FE errors. Using these 12 classifiers, we explore three approaches to combining their individual predictive power into a single decision. Classifiers can be felicitously combined into an ensemble when their predictions are not tightly correlated. In each of three fusion approaches, we combine predictions using weighted majority voting based on classifier confidence scores. First, we combine *across* each of the three classifiers *within* each sampling setting. Second, we combine *within* each classifier and *across* sampling settings. Finally, we combine the predictions of all 12 classifiers into a single decision.

4. Results and Discussion

In this section, we describe the results of experiments presented in Section 3. We evaluate each classifier, J48, Logistic Regression and SVM, using ten-fold cross validation. The mean and standard deviation of accuracy and F-measure from each experiment is reported in Table 1. Since the ten-folds have a degree of correlation, the variance may be underestimated, but this still gives some measure of classifier consistency. The ten folds are held constant across all evaluations. This decision allows us to fuse classification hypotheses from any subset of experiments.

Without any modification of training data, SVM classification is unable to reliably detect the minority class, while J48 and Logistic Regression classifiers show improved F-measures. Using undersampling allows each classifier to predict the minority class with approximately the same F-measure, but the reduction to accuracy is dramatic, indicating a significant over-prediction of the minority class. In these experiments we find that SMOTE performs approximately as well as no sampling. On all classifiers, we find that ensemble sampling is able to improve F-measure while incurring a more modest reduction to accuracy than is found with undersampling. Across all sampling approaches, we find Logistic Regression to generate average performance – it does not overpredict the majority class to the degree that SVM classification does, yet it does not yield F-measures as high as J48.

One remarkable result we find in these experiments is the robustness of J48 classification to skewed training data. The performance without any sampling obtained by this decision tree classifier is almost as high that which we observe in any sampling setting. This is due to the objective function employed by the J48 classifier. Both SVMs and Logistic Regression classifiers optimize classification accuracy by optimizing either through minimum classification error with a hinge loss function or through the log odds ratio, respectively. Because high accuracy can be obtained while ignoring the minority class, skewed class distributions pose difficulty for these classifiers when high minority class F-measure is desired. On the other hand, J48 optimizes information gain at each decision tree node. Information gain is calculated as the difference in the entropy of class distributions before and after the decision. Since entropy incorporates the class distribution prior to the decision, it is more sensitive to skewed data than classification error or log-odds are. We observe the impact of the optimization of information gain in improved minority class detection by J48.

We now turn our attention to the results of fusing these twelve classification scenarios into a single decision. In each scenario, predictions are fused using weighted majority voting, where the weights are classifier confidence scores. The results of these experiments can be found in Table 2. We find that in

	Fusion Across Classifiers	
Sampling	Accuracy	F-measure
None	87.94±0.76	17.37±4.22
Under	50.65±2.15	30.95±2.08
SMOTE	87.85±0.77	19.77±3.62
Ensemble	71.28±2.05	38.36±3.25

	Fusion Across Sampling Approaches	
Classifier	Accuracy	F-measure
J48	81.72±1.04	39.16±3.07
Logistic	78.87±1.54	37.54±3.51
SVM	87.53±0.82	4.26±2.69

Fusion Across Sampling and Classifiers	
Accuracy	F-measure
84.04±1.11	**43.25±3.67**

Table 2: *10-fold Accuracy and F-measure from classifier fusion experiments. Mean and standard deviation are reported. All figures in %.*

the unsampled and SMOTE settings, the F-measure is lower than the best performing ensemble member, J48. In both of these settings, SVMs predict the majority class with near uniformity. Therefore, for a minority class fusion prediction, both J48 and Logistic Regression must overwhelm this SVM confidence value. In the undersampling setting, we see only modest improvement through fusion, possibly indicating that the decisions are redundant. In the ensemble sampling scenario, we find an improvement of 1.80% to F-measure by fusing predictions. This suggests that despite having similar aggregate performance, the three ensemble classifiers generate complementary predictions.

Examining the performance when classifying across sampling strategies, we find that the J48 and Logistic Regression classifiers benefit from the fusion of predictions. Note that in this fusion setting, the classifier algorithm is held constant. Each fusion member is trained on different samplings of the training data. This is similar to the insight that leads to the robust performance of ensemble sampling, but now observed on a broader scale with less tightly controlled construction of training data samples. Here we observe again that merging the decisions of classifiers trained on different samples of the training data can lead to a more robust, and in some cases, better performing, ensemble classifier.

Finally, we find that the best performance on this task is observed when combining the predictions from all 12 classifiers. We observe an aggregate F-measure of 43.25% corresponding to a precision of 39.11% and a recall of 48.67%.

This performance suggests that potential FE errors can be identified at rates much higher than chance, though with far from perfect performance. However, this approach to repairing FE errors operates in concert with a human user responsible for the baseform correction. Therefore, an additional measure of the value of this system is in a reduction in the number of tokens that must be inspected manually in order to identify errors. To illustrate the value of this work, we plot the number of tokens that need to be inspected in order to find a single error, a measure equal to inverse precision. Figure 1 contains this information based on the unsampled J48 classifier, the ensemble sampled SVM, and the fusion classifier based on all 12 members. The baseline performance will present one error every 7.98 tokens as determined by the class distribution. In this

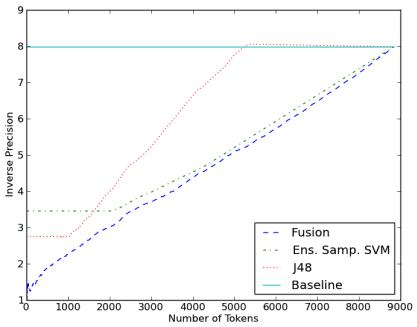

Figure 1: *Inverse precision at N tokens for J48, ensemble sampled SVM, and fusion classifiers.*

figure, we can visualize the improved performance of the classification schemes as they describe the amount of work that a human listener is required to do as compared to the expected reward of that work, specifically, correcting an error.

5. Conclusions

In this paper we have explored a variety of modeling approaches to predict FE errors in a TTS system, and reached the following main findings:

- Baseform generation errors are predictable by a variety of machine learning methods from a set of text-based features that can be easily extracted from both the surface orthography and the candidate baseform hypothesized by the FE. We envision these techniques being useful as an aid to a developer in identifying problem areas when customizing the FE of a TTS engine to a new domain or when handling very large open-ended vocabulary tasks, and have motivated the approach by our own work customizing the IBM TTS system to be part of the Watson Deep QA system deployed to play *Jeopardy!*.
- A result consistent with the theory, but which bears repeating here, is the observed different behavior of various classifiers when modeling data with skewed distributions. SVMs, learning schemes designed to directly optimize overall accuracy, achieve poor F-measure by over-predicting the majority class and are not as helpful when a performance metric with good precision on the minority class is desired or applicable to the task, as is the case here. Decision trees, on the other hand, are found to perform nearly as well as the best system in terms of F-measure, a result due to training with an information-gain criterion.
- Classifier fusion is a powerful approach to improve minority class F-measure. We have explored fusing results of classifier outputs within a sampling scheme, as well as by fusing across sampling approaches, and shown that the highest gains are obtained by fusing across sampling methods, a result that strongly suggests there is non-redundancy in the different views of the data afforded by the various sampling schemes and that these can be used complementarily in some fashion. Lastly, the biggest gain is obtained by fusing outputs both across classifiers and sampling schemes

As a continuation of this work, we will explore whether feature selection makes an impact on the results reported here. The use of a supervised meta-learner for fusion, trained directly from the classifiers outputs on held-out data, is another potentially fruitful idea to improve on the combination figures we have reported in this paper.

6. Acknowledgements

The authors would like to thank Andy Aaron for his help in collecting and analyzing domain dependent multi-category word lists, pronunciations, and running listening tests; Maria Smith for offering her front-end expertise and consulting on many aspects of this project; and Larry Sansone for his help in running listening tests.

7. References

[1] J. Pitrelli, R. Bakis, E. Eide, R. Fernandez, W. Hamza, and M. Picheny, "The IBM Expressive Text-to-Speech Synthesis System for American English." *IEEE Trans. Audio, Speech and Lang. Processing*, vol. 14, no. 4, pp. 1099–1108, July 2006.

[2] "IBM Watson," http://www.research.ibm.com/deepqa/.

[3] "Jeopardy!" http://www.jeopardy.com/.

[4] H. Lu, Z.-H. Ling, S. Wei, L. Dai, and R. Wang, "Automatic error detection for unit selection speech synthesis using log likelihood ratio based SVM classifier," in *Proc. Interspeech*, Japan, 2010, pp. 162–165.

[5] Y.-J. Kim and M. Beutnagel, "Automatic detection of abnormal stress patterns in unit selection synthesis," in *Proc. Interspeech*, Japan, 2010, pp. 170–173.

[6] V. Ramasubramanian, A. K. V. Sai Jayram, and T. V. Sreenivas, "Language identification using parallel phone recognition," in *Workshop on Spoken Language Processing*, 2003.

[7] "Project Gutenberg," http://www.gutenberg.org/.

[8] I. Witten, et al., "Weka: Practical machine learning tools and techniques with java implementation," in *ICONIP/ANZIIS/ANNES International Workshop: Emerging Knowledge Engineering and Connectionist-Based Information Systems*, 1999, pp. 192–196.

[9] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.

[10] Y. Rong, Y. Liu, R. Jin, and A. Hauptmann, "On predicting rare cases with SVM ensembles in scene classification," in *ICASSP*, vol. III, 2003, pp. 21–24.