Garbage Modeling with Decoys for a Sequential Recognition Scenario

Michael Levit [1], Shuangyu Chang [2], Bruce Buntschuh [3]

Tellme, a Microsoft subsidiary
Mountain View, CA 94041, USA
[1] michael.levit@microsoft.com
[2] shawn.chang@microsoft.com
[3] brubunt@microsoft.com

***Abstract*—This paper is concerned with a speech recognition scenario where two unequal ASR systems, one fast with constrained resources, the other significantly slower but also much more powerful, work together in a sequential manner. In particular, we focus on decisions when to accept the results of the first recognizer and when the second recognizer needs to be consulted. As a kind of application-dependent garbage modeling, we suggest an algorithm that augments the grammar of the first recognizer with those valid paths through the language model of the second recognizer that are confusable with the phrases from this grammar. We show how this algorithm outperforms a system that only looks at recognition confidences by about 20% relative.**

***Index Terms*—parallel and sequential speech recognition, application-dependent garbage modeling**

I. Introduction

It is widely accepted in the scientific and engineering community that technical progress obeys Moore's law which predicts exponential growth for many available resources like processor speed and storage space. However, the other fact of life is that increased technical capabilities lead to more sophisticated services and as a result, our demands for resources grow exponentially as well.

Applied to the field of commercial speech recognition and understanding, this observation emphasizes importance of efficient control mechanisms that maximize recognition accuracy while keeping the required resources at bay. For instance, in the domain of speech-enabled services on mobile devices, certain recognition tasks such as command and control and voice activated dialing can be carried out directly on the device using embedded speech recognizers (which are typically more light-weight but also faster because the network transmission factor is eliminated from the loop). Other requests, like in street address or business name recognition tasks, need to be directed to a network recognizer because recognizing them requires access to more resources than typically offered by a mobile device. One consequence of such separation is that the two ASR systems have different language models, whereby the embedded language model is typically much smaller than the language model of the network ASR, which also contributes to a faster response from the former. Another example is a recognition scenario where prior distribution of sentences in the task language is highly skewed. We can split their language models in two unequal parts: one very small but accounting for a high fraction of all requests, and the other taking care of the long tail of low probability phrases. Here also, one could run two separate recognition processes with these language models.

In a tandem recognition system like that, we can either start by running the light-weight recognizer (*"first recognizer"*) and then run the high-coverage recognizer (*"second recognizer"*) only if needed, or we can kick off two recognition processes at the same time and wait till the first recognizer returns. Assuming the first recognizer returns faster than the second, if needed, we can then wait for the second recognizer to return, otherwise the result of the first one will be accepted.

Here, the crucial point is providing a gainful formalization of the "if needed" condition. Indeed, the first recognizer can only recognize sentences that are in its small language model (in other words, *it only knows what it knows*), but the decision to consult the second recognizer should also take into account the language models of the latter, a kind of knowledge that the first recognizer lacks (*it does not know what it does not know*).

In this paper, we describe an algorithm that helps the decision making process by augmenting the first recognizer's language model with valid paths through the second recognizer language model that are most likely to cause confusions. The augmentation happens offline, prior to the recognition process and can be viewed as application-dependent garbage modeling.

The remainder of this paper is organized as follows: Section II describes the problem and reviews relevant research. Next, in Section III we introduce our approach to use application-dependent garbage model *"decoys"* in a two-stage recognition process. We then suggest alternative ways of decoy generation in Section IV and present results of a pilot experiment to demonstrate the power of decoys in Section V. The paper is concluded with ideas for future work and a summary.

II. Motivation: Garbage Modeling in Automatic Speech Recognition

While Statistical Language Models (SLMs) estimated as n-grams probabilities or PCFG, are essential for natural user-

machine dialogs and can contribute to a better user experience, a lot of the commercial speech recognition for call centers, directory assistance and other applications still relies on directed dialog models with handcrafted, phrase-based grammars for speech recognition due to the practical efficiency of the latter [1]. Grammars specify exactly which user responses are admissible and assume all others to be *out-of-grammar* (OOG). However, even in the case of very clear and restrictive system prompts (e.g. those expecting either *"yes"* or *"no"* answers), there will always be uncooperative users defying these expectations. If we are not prepared to handle such unexpected utterances, as well as side speech that is not actually addressing the system, misrecognitions are likely to occur. To prevent this from happening, garbage models can be introduced in ASR systems that serve as false accept magnets. The more open the prompt (and sophisticated the corresponding grammar), the more opportunities are there for the users to say something out of grammar, and as a consequence, the acuter the need for a good garbage model. Even in the case of SLM recognition, there can still be many words that are out of vocabulary (OOV), which, if not modeled appropriately, can degrade the recognition accuracy of surrounding words. Garbage model is also a natural choice for capturing OOV words.

Garbage modeling is usually addressed in the literature from either acoustic or language modeling perspective. The acoustic modeling approach [2], [3] is aiming at building special filler models to represent non-speech and task-irrelevant speech audio. For instance, in [3], to assist a single digit recognition task in a challenging environment, special HMMs are trained on noise and non-keywords to minimize a number of error metrics.

The language model based garbage modeling [4], [5], [6] focuses on relevant keyword phrases but attempts to recognize every word (or phoneme) in an utterance even where it is not relevant for the task. One advantage of such methods is that they alleviate the need for additional acoustic model training when porting across different tasks. Instead, n-grams at word or subword levels are trained to account for the utterance parts surrounding keywords and phrases.

Related to this approach is application-dependent garbage modeling where the system acts on the assumption that the application itself determines how to construct garbage models [7], [8]. For instance, for a connected digit verification task, Rahim et al. in [7] explored usability of generic filler models (one for non-speech and another for non-digits) as well as a number of anti-keyword models, each of them being trained on all digits but a specific one.

Sometimes garbage models combine acoustic and language model elements. For instance, in [9] acoustic models for phonemic or syllabic fillers are trained and then introduced into a language model via training data where they are used to replace occurrences of out-of-vocabulary words. In [10] a relatively small number of phone-based filler models introduced in a bigram language model produced competitive keyword spotting results compared to a much slower LVCSR recognition system.

The approach we are suggesting in this paper is vastly different from the ones listed above as we are not limiting ourselves to word- or phone- filler models but instead select entire phrases as false recognition magnets, guided by the known application structure.

Finally, we should mention the related task of detection of spoken out-of-domain utterances, where determination of admissible versus unsupported requests is based not on whether they are covered by the employed recognition grammar (which in these cases is rather a large SLM) but on a broader concept of *topic*. The decision is usually made after the recognition has taken place and can be facilitated either by explicit modeling of out-of-domain utterances (e.g. [11]) or, if obtaining out-of-domain data is cumbersome, by considering topic classification scores (e.g [12]).

III. Application-dependent Garbage Modeling

For the two-recognizer scenario that we are focusing on in this paper, the need for garbage modeling is eminent, since the task definition itself implies that there will be many OOGs for the first grammar-based recognizer, namely all those requests that are not in its grammar, but can be understood by the second recognizer[1]. Because of this prior knowledge, we can do more than employing standard garbage modeling techniques such as filler models that essentially "average" acoustic characteristics of the language and therefore can be viewed as a mesh of random false accept magnets. The schematic representation of a recognizer without a garbage model and of the standard garbage modeling is shown in Figure 1(a,b). In contrast to this, the two-recognizer scenario allows for an early insight into the language "outside" the first grammar but still within the scope of the second language model. From the magnitude of the phrases of this language we can select only those phrases that appear confusable with the phrases from the first grammar. We call these confusable phrases *decoys*. The application-dependent garbage modeling with decoys is schematically depicted in Figure 1(c). The number of decoys can vary but it would typically be comparable in size with the number of the phrases in the grammar of the first recognizer in order to preserve the advantage of fast recognition.

Note that application-dependent and generic garbage models can be efficiently combined together to take advantage of the domain competence of the former and the robustness of the latter. See Section V for details.

In the explanations above, we have yet not addressed the important question of what "confusability" means in the context of decoy selection. The next section will offer alternative definitions of confusability and suggest practical ways of its computation.

IV. Determining Decoys

The notion of confusability we are interested in should be defined in the following context. Given a set A of phrases $\{a_i\}$

[1]Note that we make no assumptions about the nature of the language model of the second recognizer which is not restricted to phrase-based grammars but can be a statistical language model (SLM) as well.

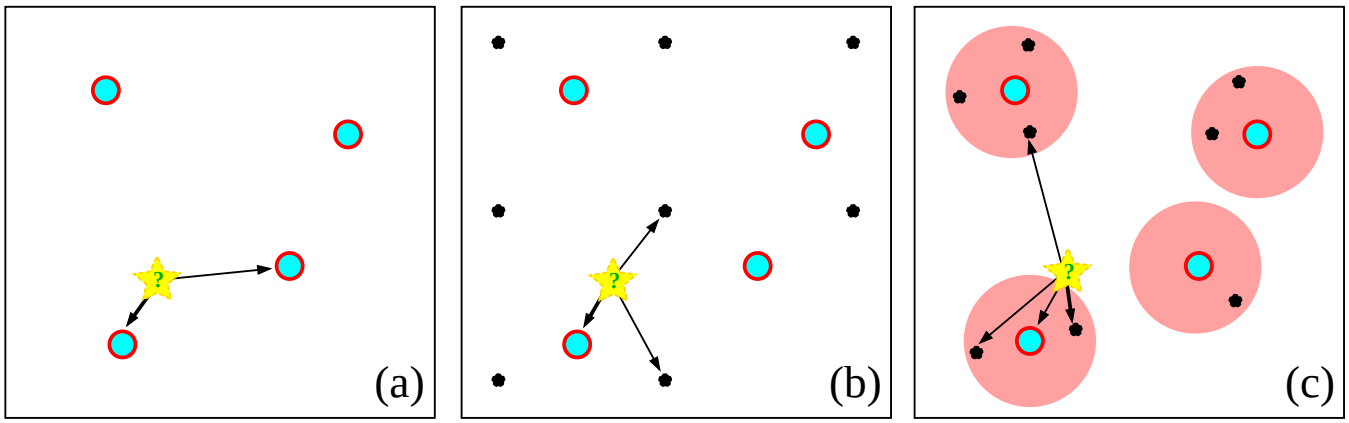

Fig. 1. Schematic representation of speech recognition (a) without garbage model (b) with traditional garbage model (c) application-dependent garbage model. New audio is represented by a star, recognition candidates from the first grammar are circles and garbage model is represented by dots that in the application-dependent scenario are located in the vicinity of the recognition candidates. Arrows indicate recognition candidates for the new audio and the bold arrow marks the winner candidate. Assuming that the new audio in this example is an OOG, only the decoys will correctly identify it as such.

from the grammar of the first recognizer, and a potentially much larger set B of phrases $\{b_j\}$ from the language of the second recognizer, for each $a_i \in A$ we need to find a subset $B^i \subset B$ of phrases b_j that the recognizer would have difficulties telling apart from a_i. There are several alternative ways to estimate the confusability measure and hence to select decoys candidates:

1) **Acoustic (phonetic) similarity**: produce pronunciations of all candidates b_j and select those with phonetic distances to a_i within a given threshold. Weighted or Levenshtein edit distances between individual phones can be used to compute distances between phrases, and dynamic programming can be employed for agglomeration.
2) **Observed recognizer behavior**: collect statistics about misrecognitions (or n-best competitors) from an existing application that uses both language models; select all phrases b_j from the second language model that were involved in competitions with some utterances of a_i.
3) **Directed recognition test**: recognize some acoustic realization of a_i one or several times with the second recognizer; accept recognition results as decoys.

All three alternatives have to struggle with practical feasibility issues. It might be reasonable to assume that the number of phrases $a_i \in A$ is relatively small and we can indeed iterate over the entire list, but making the same assumption about the second set B would be very presumptuous, since the second recognizer can operate with a large SLM. Thus, we would not be able to consider all phrase pairs $(a_i, b_j), i = 1 \ldots I, j = \ldots J$ to compare their phonetic representations. Similarly, we need to dismiss the second option as it implies that we have access to a transcribed corpus large enough to include each a_i at least once (better several times). The third option, does not make any assumptions about an existing data set; instead it deliberately creates one. Furthermore, it drops the first language model from the recognition, in order to forestall interference of language model weights, while boosting the chance of getting more confusable candidates at the same time.

Yet even this option has the bottleneck of possibly not having acoustic realizations for all $a_i \in A$. Next we suggest two approximations to get around this problem: the employed ASR in a simulation mode and a text-to-speech engine.

A. ASR in the Simulation Mode

While we cannot rely on availability of audio samples for all phrases from the first grammar, the ASR engine itself has an idea of what acoustic representation to expect for each word phrase. Using its internal acoustic models, it can draw one or several samples from the distributions that define them and thus generate feature vectors that can then be recognized as if they were coming from regular "naturally obtained" speech [13]. We use Microsoft proprietary ASR engine to carry out the feature vector generation process which we call *simulation*. It is worth mentioning that using the simulated data, one can also compute acoustic confusability directly from the acoustic models (HMMs), without a recognition pass [14].

One problem with using ASR acoustic models to generate speech is the bias that the models are pre-wired with. For instance, if there is not enough data to train a particular distribution, then all acoustic representations involving it will likely contain errors which will be further amplified during recognition.

B. Using TTS to Produce Acoustic Representations

To avoid the issues with simulation, we need an independent source of audio representation for all phrases in the first grammar, and in Text-to-Speech engines we will find the most natural candidate for that. Unlike the ASR simulation mode, TTS tends to generate reproducible speech output and thus, in order to obtain several audio samples, we can either change voices within a TTS engine, or try alternative engines. The approach we will be evaluating in Section V will have both.

V. Experiments and Results

In this section we present a pilot experiment to deliver a proof of concept for the decoy framework. We set up baselines and evaluate advantage of application-dependent garbage modeling with decoys.

The notion of *recognition confidence* is central to understanding the experiments. Confidence values are trained to predict the expected average utterance accuracy for all utterances recognized with the same confidence. Statistical regression methods are often used to derive confidences from a number of features such as maximum-likelihood scores of the highest scoring hypotheses, number of the hypotheses etc.

All of our experiments consist of two stages. First, we employ the first language model (possibly augmented with garbage models) to recognize all utterances. Those recognition results that possess high confidence values are accepted and the rest is recognized using the second language model, after which the two recognition confidences are compared to select the final result for the utterances in question.

A. Data and Experimental Setup

For our pilot experiments we selected two domains: Voice Activated Dialing (VAD) and business search. VAD is a perfect candidate to provide a grammar for the first recognizer as a typical personal address book is small to moderate in size (1081 distinct entries in our experiments). On the contrary, business listings can become very large in size, a great choice of a language model for the second recognition stage. For our experiments, we compiled a grammar that comprises some 68K company names.

Our test set contains 570 utterances, both domains represented by an equal number of examples. In addition, for the sake of a better generalization, we required that no names appear in the corpus more than once.

Both language models were represented by grammars in SRGS format [15] and employed by Microsoft speech recognizer [16] that was also used to produce decoys in the simulation mode. For TTS experiments we used a proprietary Microsoft TTS engine conforming to SAPI API [17] and AT&T Natural Voices [18].

B. Evaluation Metrics

We use *semantic error rate* as the principle evaluation metric. Since the objective of any practical application is to successfully accomplish its task, using the traditional measure of word error rate is often inadequate. Instead, recognized text is compressed to its meaning by dismissing occasional non-salient elements like hesitations, pre-fillers and post-fillers like ”please”, “I need to” etc. and compensating for spelling alternatives. In the pilot experiments we are reporting on in this paper, all utterances contain only salient information and meaning is just the recognized word string.

Then, we compute *false accept* (FA) and *false reject* (FR) rates in terms of the meaning-representations:

$$\mathrm{FA} = \frac{\#\text{meaning is wrong}}{\#\text{utterances}}$$

$$\mathrm{FR} = \frac{\#\text{no recognition}}{\#\text{utterances}}$$

and compute the final error rate as a weighted average of the two[2].

By raising the confidence threshold for skipping the second recognition (see above) we increase the percentage of utterances that are sent to the second recognition stage, and thus improve final error rate. We plot the final error rate against the fraction of utterances for which second recognizer had to be consulted, and call it *Speed-to-Error (S2E)* curve. The further down-and-to-the-left the curve the better the overall performance of the combined system.

C. Generating Decoys

Because of a simple structure of the first grammar (a disjunction of 1081 name alternatives), seeding decoys with entries from the first grammar is quite simple. Using the ASR simulation mode, up to 21 alternative recognitions have been obtained for each of the 1081 seeds (five decoys per seed on average), producing a total of 2136 distinct decoys after eliminating redundancies.

For TTS-based decoys, we used four voices (one male and one female per TTS engine) to vocalize each of the seeds and recognized them in a 5-best mode producing a total of 2617 decoys.

The generated decoys are then compiled into a new grammar as a disjunction with individual entry weights as determined by the second language model that they all came from.

D. Effect of Decoys on Semantic Error Rate

First of all, we have observed that the first recognizer was only marginally slower due to the added decoys (about 5% relative), which is inline with our goal to improve overall recognition latency. In Figure 2 we plot the S2E error curves for four experiments where the first recognition is conducted with:

1) only VAD grammar
2) VAD augmented with a traditional acoustic garbage model
3) VAD and simulation-generated decoys
4) VAD and TTS-generated decoys

The plots show that while generic garbage model and simulation-based decoys both contributed to lower error rates, the TTS-based decoys provided by far the best results, especially in the “interesting” range of re-recognition percentage

[2]Typically, business logics require that false accepts have slightly lower weight than false rejects (0.7:1.0 in all of our experiments).

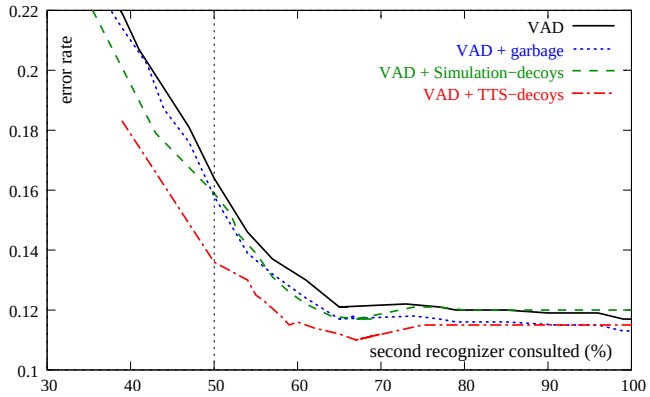

Fig. 2. Effect of decoys on S2E-curves.

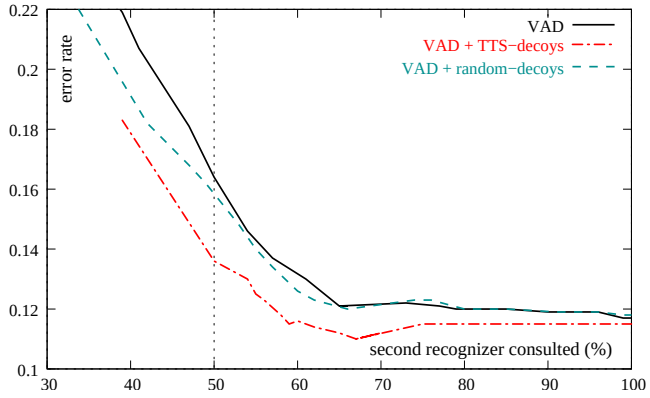

Fig. 3. Importance of a good decoys selection strategy.

around the prior (=0.5) where the original error rate was cut by almost 18%. We explain the improvement of the TTS-generated decoys over the simulation-based ones by the ability of the former to provide ASR-independent and thus better representative audio samples.

As an additional proof that decoy selection strategy matters, in Figure 3 we re-plotted the S2E curves for the VAD only case and the TTS-generated decoys, and added results of another experiment where 2600 decoys were selected at random from the second language model. Again, the TTS-generated decoys exhibited a much better recognition accuracy.

E. Combining Decoys and Acoustic Garbage Models

In line with our expectations, the previous experiment showed that a traditional acoustic garbage model can also improve recognition. On the other hand, the decoys, though effective as demonstrated, can still make mistakes, for instance, due to imperfect TTS. We thus expect that the generic garbage model trained on a large amount of representative data for the language will be a good counterpart to decoys that focus specifically on confusable phrases. As a next step, we try combining the generic garbage model with our application-dependent decoys. .

Figure 4 demonstrates that the combination of the two

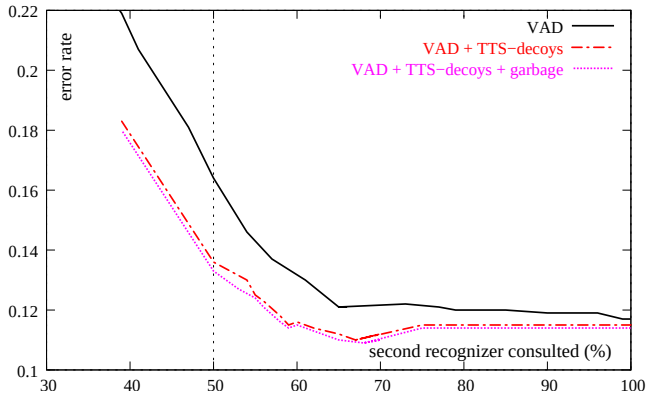

Fig. 4. Combining application-dependent decoys and language-specific garbage model.

models does indeed make sense, and an additional small improvement can be achieved due to this combination that brings the overall reduction of the original error rate by almost 20%.

We also experimented with language model based garbage models (such as syllable loops) and were able to achieve further improvements for re-recognition percentages around the domain prior.

VI. Conclusion and Future Work

We presented an algorithm to efficiently combine two uneven ASR systems that cover different discourse domains. Assuming that the first recognizer only understands a limited number of phrases, and the language models of the second recognizer is much more inclusive, we would like to minimize the need for the second recognizer after the results of the first recognizer become available. Our algorithm augments the first recognizer's grammar with "decoys", i.e. those valid paths through the language model of the second, slower recognizer, that are confusable with phrases from the first recognizer. The approach is tantamount to application-dependent garbage modeling. Determining decoys by computing acoustic confusability for TTS-generated audio representations of the phrases from the first grammar resulted in the best tradeoff between accuracy and the fraction of times where the second recognizer had to be used. In combination with a generic garbage model and compared to a baseline that decides to call the second recognizer based on the first recognizer's recognition confidence, our approach reduced the error rate by almost 20%. As a next step, we plan to extend the algorithm to support arbitrary language models for the first recognizer.

REFERENCES

[1] R. Pieraccini and J. Huerta, "Where Do We Go from Here? Research and Commercial Spoken Dialog Systems," in *Proceedings of 6th SIGdial Workshop on Discourse and Dialog*, Lisbon, Portugal, September 2005, pp. 1–10.

[2] J. G. Wilpon, L. R. Rabiner, C. H. Lee, and E. R. Goldman, "Automatic recognition of keywords in unconstrained speech using hidden markov models," *Acoustics, Speech, and Signal Processing, IEEE Transactions on*, vol. 38, no. 11, pp. 1870–1878, 1990.

[3] L. Villarrubia and A. Acero, "Rejection techniques for digit recognition in telecommunication applications," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1993, pp. 455–458.

[4] Q. Lin, D. Lubensky, M. Picheny, and P. S. Rao, "Key-phrase spotting using an integrated language model of n-grams and finite-state grammar," in *Proceedings of 5th European Conference on Speech Communication and Technology*, 1997, pp. 255–258.

[5] I. Bazzi and J. Glass, "Modeling out-of-vocabulary words for robust speech recognition," in *Proceedings of International Conference on Spoken Language Processing*, 2000, pp. 401–404.

[6] D. Yu, Y. Ju, Y.-Y. Wang, and A. Acero, "N-gram based filler model for robust grammar," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2006, pp. 565–568.

[7] M. Rahim, C. Lee, and B. Juang, "Robust utterance verification for connected digits recognition," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1995, pp. 285–288.

[8] C. Shu, "Selected phoneme rejection grammar for a speech recognition system," in *Proceedings of 4th International Conference on Signal Processing*, 1998, pp. 646–649.

[9] R. El Meliani and D. O' Shaughnessy, "Accurate keyword spotting using strictly lexical fillers," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1997, pp. 907–910.

[10] A. Manos and V. Zue, "A segment-based wordspotter using phonetic filler models," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1997, pp. 899–902.

[11] P. Haffner, G. Tur, and J. Wright, "Optimizing SVMs for Complex Call Classification," in *Proceedings of ICASSP*, vol. 1, April 2003, pp. 632–535.

[12] I. Lane, T. Kawahara, T. Matsui, and S. Nakamura, "Out-of-Domain Utterance Detection Using Classification Confidences of Multiple Topics," vol. 15, no. 1, pp. 150–161, January 2007.

[13] D. McAllaster, L. Gillick, F. Scattone, and M. Newman, "Fabricating Conversational Speech Data with Acoustic Models: a Program to Examine Modeldata Mismatch," in *Proceedings of International Conference on Spoken Language Processing*, Sydney, Australia, November 1998, pp. 1847–1850.

[14] H. Printz and P. Olsen, "Theory and Practice of Acoustic Confusability," in *ASR*, 2000, pp. 77–84.

[15] *Speech Recognition Grammar Specification Version 1.0*, W3C, URL: http://www.w3.org/TR/speech-grammar.

[16] J. Odell and K. Mukerjee, "Architecture, User Interface, and Enabling Technology in Windows Vistas Speech Systems," vol. 56, no. 9, 2007.

[17] "SAPI text-to-speech (SAPI 5.3)," URL: http://msdn.microsoft.com/en-us/library/ms720571(VS.85).aspx.

[18] "AT&T Natural Voices™ Text-to-Speech System," URL: http://www.naturalvoices.att.com.