IMPROVING SPEECH SYSTEMS BUILT FROM VERY LITTLE DATA

John Kominek, Sameer Badaskar*, Tanja Schultz*§, Alan W Black**
{jkominek, sbadaska, tanja, awb}@cs.cmu.edu

* Language Technologies Institute, Carnegie Mellon University, USA
§ Cognitive Systems Lab, Karlsruhe University, Germany

ABSTRACT

This paper studies two ways for helping non-specialist users develop speech systems from limited data for new languages. Focused web re-crawling finds additional examples of text matching the domain as specified by the user. This improves the language model and cuts word error rate nearly in half. Iterative voice building with interleaved lexicon construction uses the voice from a previous iteration to help construct an improved voice. 4.5 hours of the user's time reduces transcription error rate from 32% to 4%.

1. INTRODUCTION

The SPICE project is an initiative that deploys a web-based toolkit for the rapid development of Automatic Speech Recognition (ASR) and Text-to-Speech (TTS) [1]. The purpose is to make the construction of these speech processing technologies available to the world at large, and in particular to simplify the process for non-specialist users. The first version of this toolkit has been now running as a live server for one year [2], and has been used in laboratory courses taught at Carnegie Mellon University in the U.S. and at Karlsruhe University in Germany. One distinctive characteristic of the SPICE architecture is the ability to interactively and iteratively construct systems customized to new speakers and new languages with an absolute minimum of data and effort, in practice with as little as 5-10 minutes of speech data [3].

In a semester-long assignment, students are asked to choose a language and domain of interest and to use the SPICE system to: i) define a phoneme set, ii) collect a text corpus, iii) build a bigram language model, iv) generate a 200-1000 utterance prompt list, v) record the prompt list from one or more native speakers, vi) construct a lexicon and letter-to-sound rules, vii) build acoustic models for ASR, viii) evaluate the recognizer, ix) build voice models for TTS, and x) evaluate the synthesizer. For end-to-end evaluation, the system provides a "talk-back" function, where the speaker says a sentence which is transmitted to our server, decoded, synthesized, and played back to the user. A pair of systems can be hooked together to provide intermediated communication.

From the laboratories conducted to date, students are successful about half the time. Much of the blame may be attributed to deficiencies in our software. However, we observed that the successful students arrive with substantial experience in speech technologies. Those with less experience lack knowledge of what is reasonable at each stage of data collection, and where the "comfort zone" of each technology component lies. One possible solution is to knowledge-engineer additional constraints into the system, effectively providing sturdier guide-rails for the purpose of preventing users from veering off track. Thus, the major challenge of this approach is how to find and where to place appropriate guide-rails.

A more robust approach is to build a system in an iterative process with integrated user feedback. Here, the user bootstraps the system from 10 minutes of speech for example, evaluates it, and – with system support – mends various deficiencies. Deficiencies typically include insufficient speech data, a weak acoustic or language model, and words missing from the lexicon. Previously in [4] we measured the relative effectiveness of working on the pronunciation dictionary versus simply recording more speech (as it pertains to synthesizer quality), and applied the results to eight non-English languages. We found that during early stage work it is better to collect additional speech, while in later stages improving the lexicon offers greater gain for amount of labor spent. The location of the transition point seems to lie between 30m and 60m of speech, but naturally it depends on the language in question and on the relative speed of recording versus lexicon correction.

This paper reports on initial investigations in iterative system development, as applied to ASR and TTS. The starting point is a "seed application" consisting of a small amount of text, ten minutes of recordings, and a dictionary of a few hundred words. In our tests we use two seed applications, one in English (investigating TTS) and one in Hindi (investigating ASR). Both target the domain of cooking recipes. Using such a seed, we have developed a mechanism for expanding the text corpus using the technique of *focused web re-crawling* [5]. This supports bigram language models with better coverage for ASR and also provides material for additional speech recordings For evaluating TTS, speech is collected in five stages, with

lexicon development interleaved. The voice is evaluated after each stage by transcribing the heldout set.

Two high level ideas guide our investigation into low resource system construction. The first is: how can one improve on an initial system? Users wanting to build usable systems are confronted with the quandary of "now what?". The second idea is: of the many ways the user can devote development time, where can the effort be best expended? This paper provides the beginnings of an answer and some useful hard data to support it.

2. TEXT COLLECTION

One pitfall users of SPICE are inclined to commit is not collecting enough text data. In SPICE these data are used for computing a statistical language model and for automatically selecting a prompt set. The user may upload plain text or point the system to a web page to crawl.

2.1. Focused web re-crawling

The SPICE user specifies the domain implicitly by specifying either the URL of the text or by uploading a text file. The documents crawled from the target URL can be analyzed to identify the domain related terms which could be used to re-query a search engine for additional URLs. The downloaded documents are cleaned of html tags. Then bigrams in the text are scored by the well known "term frequency inverse document frequency" (TF-IDF) weighting scheme. The bigrams with highest TF-IDF score are treated as *domain terms* and used to re-query a search engine (in our case, Google). Additionally, terms with highest TF-IDF score along with their respective scores are stored as a *model* of the domain. For each bigram-query term, the top K (K=5) URLs returned by the search engine are then crawled for more documents. The crawler keeps track of visited pages to prevent duplicate downloads.

To minimize topical (domain) drift of the downloaded documents from the original domain, each of the crawled documents is cleaned up and the cosine similarity with respect to the domain model is computed. Documents whose similarity scores lie above a certain threshold are added to the data obtained initially. This similarity computation and thresholding step ensures purity of the additional text data while removing irrelevant content. We apply focused crawling to automatically gather additional domain related text data for enhancing the language model which in turn affects the performance of the ASR.

2.2. Corpus expansion of recipe domain data

For both our English and Hindi tests we chose the same topic domain of cooking recipes. Sentences typical of this domain are "sprinkle the cavity with salt and pepper," and "in a small bowl, mix flour, beer, and sauce". The English seed application contains 5,261 word tokens; after expansion the count is 215,217. The Hindi seed consists of 192 hand-edited sentences totaling 1,523 words. This was expanded to 159,995 words in one run, and 360,395 in another, depending on the threshold settings. Three Hindi text sets are used to construct and evaluate three language models, as discussed in the following section.

For the English database used to test text-to-speech, 1,111 utterances were recorded from a single speaker. Removing 10% for testing leaves exactly 1,000 for training. Discussion of TTS experiments is deferred to section 3.

2.3. Effect of language model and out-of-vocabulary words on ASR performance

In the Hindi ASR experiments the speech data consists of 192 utterances (comprising 13 minutes) from a single speaker, with 20 utterances held out for testing. This relatively small amount is used to adapt multi-lingual GlobalPhone acoustic models to the speaker [6]. The training/test data was partitioned three times with separate experiments run on each partition

LM	*word count*	*Word Error Rate (WER) (%) perplexity / OOV rate (%)*			
		split 1	*split 2*	*split 3*	*ave.*
1	1523	95.88 5.2/68.7	97.92 6.9/57.9	84.93 7.8/50.0	92.91 6.6/58.9
2	159995	55.15 177/16.8	56.25 93.4/27.4	51.81 165/13.4	54.41 145/19.2
3	360395	54.12 214/15.0	52.08 113/25.0	50.60 187/11.3	52.27 171/17.1

Table 1. WER, perplexity, and OOV rates measured on 3 training/test partitions for each of the 3 language models.

The performance increase from 92.9% to 52.3% shown in table 1 is substantial and mostly a result of significantly reducing the OOV rate on the test set by focused recrawling. We could achieve a comparable result only in a cheating experiment, where we included the test sentences into the LM. Clearly, this is not a solution in practice. Furthermore, overspecialization makes a system less flexible. Recrawling increases the LM perplexity from 6.6 to 171 but this is more than compensated by reducing the OOV rate from 58.9% to 17.1%. Non-technical users easily fall prey to the deadly effect of OOV words. An enhanced recrawler could specifically target this problem by automatically maximizing vocabulary coverage.

3. INCREMENTAL VOICE BUILDING

Incremental voice building is a technique that uses a previous version of a synthesizer to help construct a newer version. Our effort focuses on incremental lexicon building and is based on the observation that synthesized samples of words can assist in the task of pronunciation correction [7]. The procedure is conceptually straightforward.

1. Employing the previous synthesizer and its letter-to-sound rule system, up to four alternate pronunciations are generated for each lexical entry. This is performed as a batch operation prior to involving the user for the correction stage.
2. The alternate pronunciations are presented to the user one entry at a time, with the most likely pronunciation listed first. Presentation includes the word in the native script, phoneme strings for each alternate, and the corresponding wavefile. The user listens to the wavefiles and selects the closest match, or, if none is acceptable, types in an alternate pronunciation. In the SPICE system the typed-in pronunciation is synthesized for playback. Because this task can be tiring, the user was not expected to examine the full lexicon in one sitting. Review sessions lasted 20-30 minutes, and ended when the user noticed encroaching fatigue.
3. The user records an additional set of 200 prompts, spending about 25 minutes to complete this task. We designed the lexicon and recording activities to be interleaved and roughly balanced.
4. The updated lexicon and expanded speech data are used to rebuild the voice.
5. While the voice is being rebuilt the user transcribes a set of heldout test utterances from which the transcription error rate is computed. This number is made available to the user as a measure of voice quality, and is compared to previous values.
6. The user may now begin a new session at step 1. In actual practice, a number of engineering details needed to be attended to between iterations. The experiments reported here were conducted over a span of several days.

3.1. Data characteristics and time usage

The English recipe-domain synthesizer was built in five iterations. Ignoring overhead, the user spent 2 hours to record 1000 short utterances plus 2.5 hours to improve the pronunciation lexicon. With silences trimmed off, each session yielded about 7 ½ minutes of speech data, totaling 36m13s. Table 2 provides time summaries of lexicon building. The column *examine time* gives the average time to handle a lexical entry, viz. selection or type-in correction. Average times vary from 5 to 17s. Notice that selecting the correct pronunciation (if present), is 2-3 times more efficient. Also, the selection times decreased substantially over the sessions. This is due to a combination of factors: familiarization with the task and application, and the fact that the voice quality improves with each iteration.

lexicon words		***time (mm:ss)***		***examine time (s)***	
stage	***total***	***stage***	***total***	***selected***	***type-in***
104	104	20:40	20:40	9.8	15.9
140	244	28:55	49:35	10.1	16.7
193	437	31:10	80:45	7.2	14.4
217	654	26:01	106:46	4.7	14.4
310	964	41:23	148:09	5.9	12.3

Table 2. Five iterations of lexicon expansion on the English test. The seed lexicon of 394 words is not included.

3.2. Lexical coverage of prompts and corpus

The system works on lexical entries ordered by frequency, and one may take the counts from either the prompt list or from the text corpus. Choosing to cover all the words in the prompt list first optimizes model building (because the transcript will be better). Ordering words based on the corpus optimizes coverage of the language domain, at the risk of poorer acoustic models.

Figure 1 compares three word selection strategies. They are 1) prompts before corpus, 2) corpus before prompts, and 3) one from each alternately. In this experiment we adopted the first strategy – that is, seeking a pronunciation for each word in the prompt list first.

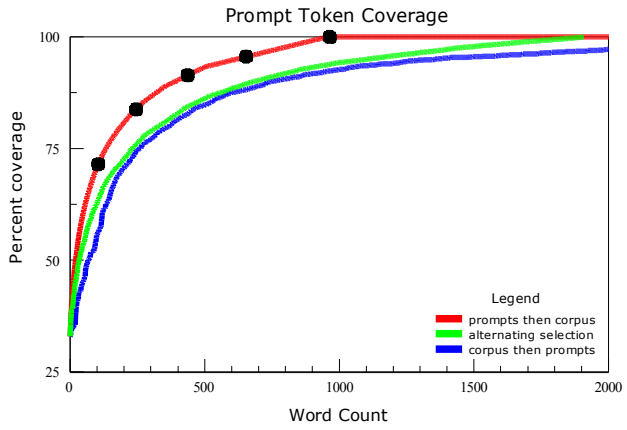

Figure 1. Token Coverage in the prompt set under three selection strategies. The black dots are the actual samples from incremental voice building. The prompt set has 1057 unique words.

3.3. Lexicon construction – usage of wavefiles

When working on the lexicon, each word is accompanied by up to four alternate pronunciations displayed as a phoneme string and synthesized using the voice from the previous stage. To study user behavior we measured the frequency of wavefile playings. The distribution is shown in Figure 2. In difficult cases wavefiles are played eight times or more, but most often once is enough. The average number of counts ranged from 3.7 (stages 1 and 2) to 1.8 (stage 4). Table 3 lists how often each alternate was chosen when no corrections were made. Instead of randomized ordering, our experience suggests that it is better to place the most likely pronunciation first.

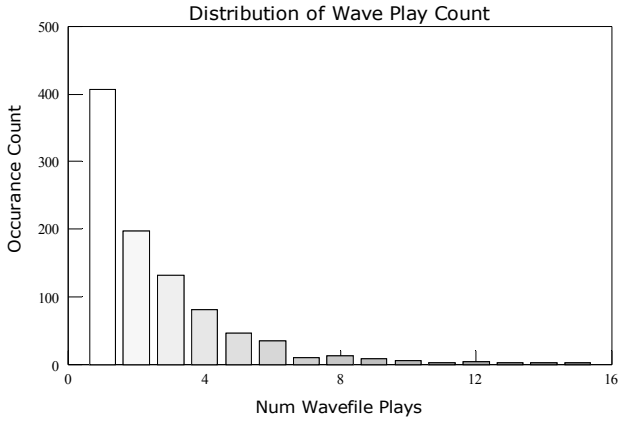

Figure 2. Distribution of the number of times that the user plays the wavefiles for a word.

	distribution of pronunciation selections				
stage	***1st***	***2nd***	***3rd***	***4th***	***% 1st***
1	54	9	5	2	77.1
2	61	17	12	1	67.0
3	93	27	2	2	75.0
4	132	18	4	7	82.0
5	155	26	22	5	74.5

Table 3. Distribution of selection choices of each stage.

3.4. Transcription listening tests

Transcription word error rates on a held out test set is a direct measure of comprehensibility. After 4.5h of user effort the final result is 33 errors out of 724 words, or 4.6%, down from an initial rate of 32%. Ten of these may be considered "soft" (*the* for *a*), while remainder are "hard" errors (*bowl* for *dough*). In the final voice building iteration, 41 minutes of lexicon work resulted in an 2.62% reduction in absolute WER, or 3.8% per hour. In the previous stage 51 minutes of total effort (25 recording, 26 lexicon) reduced WER by 3.32%, or 3.9% per hour. This suggests that the bulk of improvement is due to an improved lexicon. Our experiments in [4] found that when a voice is small (less than 30m) collecting more speech data is most efficient path to improvement, then after some threshold lexicon work wins out. Truly separating the two effects, though, requires that substantially more conditions be built and evaluated.

Voice			***Transcription Errors***			
utts	***lexicon***	***effort***	***INS***	***DEL***	***SUB***	***WER***
200	356	0:25	18	45	167	31.77
400	460	1:08	19	48	111	24.59
600	600	2:00	16	24	90	17.96
800	793	2:56	7	10	59	10.50
1000	1010	3:47	6	6	40	7.18
1000	1320	4:28	5	1	27	4.56

Table 4. Transcription error counts and rates for a 724 word test set. Total effort is given in hours and minutes.

4. CONCLUSION

To improve existing seed ASR and TTS systems we have prototyped two innovations for SPICE: 1) focused web recrawling to enhance the language model, and 2) iterative voice building with interleaved lexicon construction. We conjecture that if users can experience tangible improvement while working, they will be much more willing to devote the effort required to develop speech system for new languages.

6. REFERENCES

[1] SPICE, http://cmuspice.org.

[2] Schultz, T., Black, A., Badaskar, S., Hornyak, M., Kominek, J., *SPICE: Web-based Tools for Rapid Language Adaptation in Speech Processing Systems*, Interspeech 2007, Antwerp.

[3] Kominek, J., Schultz, T., Black, A. *Voice Building from Insufficient Data – Classroom Experiences with Web-based Language Development Tools*, ISCA Speech Synthesis Workshop 6, Bonn, German, 2007.

[4] Kominek, J., Schultz, T., Black, A. *Synthesizer Voice Quality of New Languages Calibrated with Mean Mel Cepstral Distortion*, SLTU-2008 Workshop, Hanoi, Vietnam.

[5] Chakrabarti, S., van den Berg, M., Dom, Byron. *Focused crawling: a new approach to topic-specific (web) resource discovery,* Computer Networks, vol. 31, no. 11-16, 1999.

[6] Schultz, T., *GlobalPhone: A Multilingual Speech and Text Database developed at Karlsruhe University*. ICSLP, Denver CO, USA, 2002.

[7] Davel, M., Barnard, E. *The Efficient generation of pronunciation dictionaries: human factors during bootstrapping*, Interspeech 2004, Jeju, Korea.