A 40 bps Speech Coding Scheme

Cristina Videira Lopes
School of Information and Computer Science
University of California
Irvine, CA 92697
Email: lopes@uci.edu

Anshuman Chadha
School of Information and Computer Science
University of California
Irvine, CA 92697
Email: achadha@uci.edu

***Abstract*— We describe a method and an implementation for producing a highly compressed representation of speech, in the order of 40 bps. This compression method uses a speech recognition engine to analyze the speech signal at the morphological level, i.e. the words. The words are then coded using a word-level text compression mechanism. After decompression, the speech message is recovered using Text-To-Speech. We report experimental results of our implementation. In particular, we observed that the human listeners were able to recover from errors introduced by the speech recognition engine, and that the human perceptual errors were highly dependent on the content of the messages, especially regarding the familiarity with the topic.**

I. Introduction

Conventional approaches to speech coding, typically based on a source filter model, provide good quality speech at over 2,400 bps. For very low bit rates, this model fails, because there isn't enough information per time unit to account for the speech signal. Speech coders for under 1,000 bps have been proposed that follow different models. Very low speech coders, of the order of 500 bps or less, are typically based on phonetic and segment vocoders [1], [2], [3], [4], [5]. These coders divide the speech signal into a sequence of units (phonetic and acousticly derived segment units, respectively) using speech recognition techniques, and code the unit indexes and durations. At the decoding end, the speech is synthesized according to the encoded information.

It is known that the human-to-human verbal communication data rate is of the order of a few tens of bits per second [6]. In this paper we present a speech coding scheme specifically designed to achieve those extremelly low bit rates, accounting only for the verbal information of speech messages. Our speech coding scheme is of the order of 40 bps, and, like other very low bit rate coders, uses speech recognition/synthesis techniques. The core of our approach is that we don't try to preserve/reconstruct the real-time non-verbal information of the speech signals; we simply target the words. Eventually, speech may be synthesized using the voice features of the speakers, stored statically in the decoder (rather than coded with every message). With our approach, speech coding becomes an issue of English text compression.

While recovering the speaker's voice features at the decoding side may be important, the real-time non-verbal information of speech is, in many situations, irrelevant. For example, the memo recorder capabilities included in most PDAs and cell phones provide the user with a convenient way of sending reminders to him/herself. Given that these devices have limited memory, data compression is a critical issue. The only relevant information in those reminders is the set of words the memos convey. Also for some applications (e.g. military), a voice interface to remote control and communication is faster than a keyboard interface. The recipients of those messages don't necessarily need to receive a perfect, or even a good, rendition of the original speech; they need only to receive the verbal information.

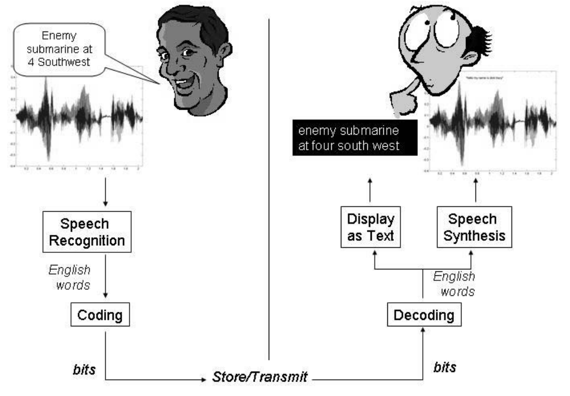

Fig. 1. Illustration of the coding/decoding process.

With those applications in mind, we have implemented and tested this method using off-the-shelf hardware and software. In this paper we report our method, its implementation and experimental results. Further, we discuss the limitations of the method and how it can be improved.

II. Method and Implementation

Fig. 1 illustrates the major components of our method. On the speaker side, a speech recognition engine transforms spoken words into textual words, which are then coded using our morphological coding scheme. On the receiver side, the compressed format is decoded into textual words which can then be fed into a TTS engine, or displayed as text. Two components of this method are noteworthy: the morphological coding and the perceptual system of the human receiver. We explain them next.

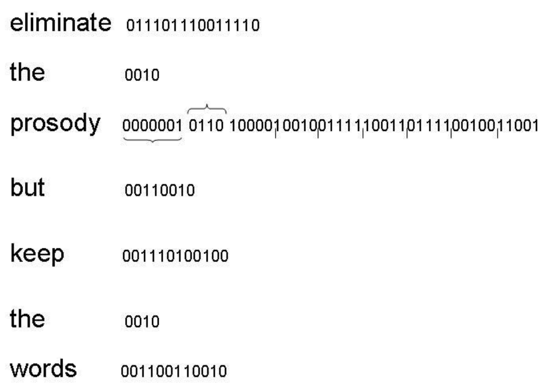
eliminate 011101110011110
the 0010
prosody 0000001 0110 10000 10010 01111 10011 01111 00100 11001
but 00110010
keep 001110100100
the 0010
words 001100110010

Fig. 2. Illustrative example.

The output of the speech recognition engine is a text consisting of a set of English words, for example "this is a test". At that particular point of the process, several different data compression / coding schemes could be applied. One of the best text compression algorithms is the Lempel-Ziv algorithm [7], a version of which is implemented by the widely available GNU zip tool. However, we can do better, because we know a lot about the contents of the text. More precisely, we know that (1) the text consists of correctly spelled English words, and (2) case doesn't matter because the words are representations of speech.

Based on this knowledge, we do word-level text compression, instead of letter- or bit-level. The coding is done using a Huffman tree that is built according to the morphological statistics of the English language. We used a 5,000 word vocabulary of the most commonly used English words. For example, the most commonly used English word – "the" – corresponds to the code 0010.

For words not in the vocabulary, our coding scheme falls back to coding the characters. Since we know these can only be the letters of the alphabet, and that case doesn't matter, we use a 5-bit binary word per character. To signal the change from word to letter coding, we use a special binary word followed by a 4-bit character count.

Fig. 2 illustrates our coding method with one example. In this example, the longest word that is still in the list of commonly used words is "eliminate", which, according to the list, is the 3,849th most commonly used English word. That word is coded with a 15-bit binary string. The word "prosody" is not a commonly used word. Our method codes it using 5-bit letter codes, prefixed with the 7-bit marker 0000001 and the 4-bit letter count 0110 (we start counting from 0, which corresponds to 1 letter). Overall, this 41-character, 3 second, sentence corresponds to 101 bits.

The second noteworthy element of our method is how to deal with errors. Occasionally, the speech recognition engine makes mistakes. For example, it may happen that the word "friends" will be recognized as "franz". While this is an error, the process of delivering the message to the human listener often corrects the error, due to human perception and interpretation in context. For example, when used in the sentence "the gods and the giants were not franz," "franz" is always corrected to "friends." We will expand on this in the next section, as we report our results with human listeners.

III. Experimental Results and Discussion

We have implemented our method using off-the-shelf hardware and software. The hardware infrastructure consists of ordinary desktops and laptops, equipped with ordinary microphones and speakers. The software consists of Microsoft's Speech Recognition Engine (MSAPI 5.0) accessed programmatically through CloudGarden's implementation [8] of the Java Speech API [9]. Our coding algorithm - a straightforward implementation of Huffman codes [10] - is implemented in Java. We used a 5,000-word list of the most commonly used English words, along with their probabilities.

In dictation mode, the accuracy of the speech recognition engine we used was no less than 80%, which was acceptable. In order to test our implementation, we used 12 different texts of about 30 seconds each. These texts were chosen from among different domains, namely Computer Science articles, difficult literature and fiction. A speaker read these texts and our system stored the compressed data. Later on, we recovered the information, played it through TTS and had listeners write down what they heard, sentence by sentence, replaying whenever necessary. The listeners did not have access to the original texts and were not told about the general contents of the texts. What follows is an account of our preliminary results obtained with selected listeners in our Computer Science department. More tests will conducted among a different population, to assess the general applicability of the method. [1]

Table I summarizes our results. The most important numbers are in the two rightmost columns: bit rate and perceptual error rate, respectively. The bit rate is the size of the compressed format divided by the length of speech. The perceptual error rate is the number of words in the original texts which were incorrectly identified by the listeners, divided by the number of words in the original texts.

As the table shows, we achieved an average bit rate of 44 bps, ranging from 33 to 54 bps, with perceptual errors between 1% and 30%, depending on the texts. What follows is an explanation of these results.

Texts numbered 1 through 5 were Computer Science texts; 1 and 2 were about graphics, and had unusual words such as "specular" and "frustrum"; 3 through 5 were about operating systems and software engineering, themes and terminology well known by the listeners. Texts 6 and 7 were excerpts from Edgar Allen Poe's poetry and contained obscure words such as "morrow" and "surcease," and unusual use of grammar such as "hesitating then no longer Sir said I or Madam, truly your forgiveness I implore." Text 8 used very simple vocabulary and grammar concerning a student from Japan

[1]All materials supporting the experimental results can be found in www.ics.uci.edu/~lopes/morphcode.

TABLE I
SUMMARY OF RESULTS

Text #	# words	words not in vocab.	length of speech (secs.)	recognition errors	compressed format (bytes)	bit rate	perceptual errors
1	114	15%	42	9%	245	47	10%
2	89	17%	35	19%	220	50	13%
3	96	21%	37	4%	195	42	4%
4	67	24%	29	4%	173	48	1%
5	80	11%	30	5%	147	39	2%
6	115	26%	37	14%	246	53	25%
7	119	18%	39	17%	264	54	30%
8	100	0%	28	7%	116	33	2%
9	122	3%	35	8%	178	41	16%
10	88	10%	28	9%	148	42	14%
11	119	10%	32	11%	188	47	13%
12	109	13%	29	7%	173	48	4%

who's looking for things on campus. Texts 9 through 12 were fictional stories, some more fantastic than others. For example 12 was a straightforward mythical story about gods and giants, while 9 was about little green men eating pills and standing on streets that moved.

We were very conservative in estimating the length of speech. The values correspond to the time it took for a synthesized voice to speak the words continuously and in a relatively fast pace. Human speakers are slower, and make a fair use of pauses between sentences. But these are the values we used to calculate the final bit rate. If we account for the fact that humans usually take more time to speak, then the bit rates would be even lower, on the order of 30 bps.

In half of the test cases (2, 3, 4, 5, 8 and 12) the human listeners in the decoding side introduced no additional errors on top of the speech recognition errors in the coding side. In other words, they correctly understood what was being spoken by the synthesized voice, even if, at points, the words didn't make a lot of sense in context and might affect the perception of the following words.

The interesting results is that in five of those cases (namely 2, 4, 5, 8 and 12) the human listeners were able to compensate for the mistakes made by the speech recognizer. For example, this happened in the last sentence of text 2 "this is all done through the model view matrix." By this time the listeners had some idea of the topic of the text – graphics – and were able to recover from the errors introduced by the speech recognizer, which had the sentence as "this is all done through the mall view made tricks."

In four of the test cases (6, 7, 9, 10) the human listeners performed very poorly, tests 6 and 7 being the worst. This result is reasonable. Edgar Allen Poe's texts (6 and 7) were extremely hard to understand, and errors would probably occur even if the listeners would hear those texts directly from a human speaker with no pronunciation mistakes. Given the relatively high error rates introduced by the speech recognizer, the task for the human listeners was made a lot harder, as they needed to reconstruct meaning from a grammatically unorthodox sequence of unusual words, some of which introduced non-sense.

Test 9 was the least expected result, given that the text consisted of frequently occurring words placed together in common grammatical sentences. The human listeners doubled the errors of the recognizer. Our explanation for it is that this was due to the relatively bizarre and unexpected contents of the story.

In summary, our results show that it is feasible to code speech at extremely low bit rates, by doing morphological analysis of the speech signals. The errors introduced by the recognition engine are not an obstacle. In fact, our results lead us to believe that a shared context between the speaker and the listener is the most important factor to correctly recover the messages, as it compensates for errors introduced by the speech recognition engine. This may be similar to the process that occurs during face to face communication, when listeners have a hard time understanding speakers who use unfamiliar terms or talk about unfamiliar subjects.

One problem with this method is that the synthesized voice is not the voice of the speaker. Ideally, we should be able to model and store the features of the speakers' voices so that the words would be synthetically "put in their mouths" in the receiving end.

IV. CONCLUSION

We presented a method for coding speech at about 40 bps, by doing morphological analysis of the speech signals and then using word-level text compression. Our experimental results show perceptual recognition errors between 1% and 30%. These errors are strongly related to the listeners' familiarity with the contents of the messages. In half of the test cases the listeners did no worse than the speech recognition engine. In five of those cases, the listeners were able to recover from errors introduced by the speech recognition engine, and the perceptual error rate was lower than the recognition error rate.

Overall, this method is acceptable for certain applications, but not for general person-to-person communication. For example, it is acceptable for memo recording and retrieving, where the speaker and the listener are the same person and therefore share the context of the messages. Also for communications in specialized domains such as military command and control with a relatively small vocabulary and a shared context among the participants. We view this approach as a way to maximally remove bits from the wire by doing the most possible pre- and post-processing at the sending and receiving ends.

REFERENCES

[1] S. Roucos, R. M. Schwartz and J. Makhoul, *A segment vocoder at 150 b/s*, in proc. ICASSP-83, 1983, pp.61–64.
[2] F. K. Soong, *A phonetically labeled acoustic segment (PLAS) approach to speech analysis-synthesis*, in proc. ICASSP-89, 1989, pp.584–587.
[3] Y. Hirata and S. Nakagawa, *A 100b/s speech coding using a speech recognition technique*, in proc. EUROSPEECH-89, 1989, pp.290–293.
[4] K. Tokuda, T. Masuko and J. Hiroi, T. Kobayashi and T. Kitamura, *A very low bit rate speech coder using HMM-based speech recognition synthesis techniques*, in proc. ICASSP-98, 1998, pp.609–612.
[5] K.-S. Lee and R. V. Cox, *A segmental speech coder based on a concatenative TTS*, in Speech Communication v.38, Elsevier Science, 2002, pp.89–100.
[6] L. Rabiner and B.-H. Juang, *Fundamentals of Speech Recognition*, Prentice Hall Signal Processing Series, 1993.
[7] J. Ziv and A. Lempel, *Compression of individual sequences via variable-rate coding*, IEEE Trans. Inform. Theory, 24 (1978), pp. 530–536.
[8] http://www.cloudgarden.com
[9] http://java.sun.com/products/java-media/speech/
[10] D. A. Huffman, *A Method for the Construction of Minimum Redundancy Codes*, in proc. IRE, Vol. 40, No. 9, 1952, pp. 1098–1101.