IEEE Open Journal of the
Computer Society

Received 24 June 2020; accepted 15 July 2020. Date of publication 23 July 2020;
date of current version 13 August 2020. The review of this paper was arranged by Associate Editor Amiya A. Nayak.

Digital Object Identifier 10.1109/OJCS.2020.3011392

SPQER: Speech Quality Evaluation Using Word Recognition for VoIP Communication in Lossy and Mobile Networks

BERTRAM SCHUETZ AND NILS ASCHENBRUCK (Member, IEEE)

Institute of Computer Science, University of Osnabruck, 49069 Osnabrueck, Germany

CORRESPONDING AUTHOR: BERTRAM SCHUETZ (e-mail: schuetz@uos.de)

ABSTRACT In this paper, we introduce SPQER (pronounced speaker), a novel approach to evaluate the quality of experience for real-time Voice over IP (VoIP) communication in mobile and lossy networks. Traditional speech quality metrics, e.g., Perceptual Evaluation of Speech Quality (PESQ) or the Hearing-Aid Speech Quality Index (HASQI), directly compare frequencies and amplitudes to calculate the received signal distortions. SPQER instead uses machine learning classification to evaluate the percentage of recognizable words in conjunction with a time-based decay function to penalize delay and cross-talking. So instead of evaluating noise, SPQER directly answers the question: What percentage of words is the recipient able to understand? We presented a sensitivity analysis, which is based on testbed experiments for different packet loss rates and simulated delays, to asses the impact of challenging link conditions. A final correlation analysis to a short user study shows that SPQER can better evaluate the amount of understandable words than PESQ and HASQI, while still giving a more precise indication about the voice quality than the Word Error Rate (WER) metric.

INDEX TERMS Lossy networks, machine learning, quality of service, voice over IP.

I. INTRODUCTION

Voice over IP (VoIP) telephony is a central pillar of modern communication. With the growing mobile infrastructure, more available bandwidth, and cheaper data plans, an increasing number of VoIP calls are made from mobile devices. For example, all Long Term Evolution (LTE) voice calls are nowadays already IP-based. Also, an increasing number of command-and-control systems are switching to IP backbones. Since most VoIP applications are based on RTP over UDP, there is no transport layer reliability to recover lost packets. Such loss can occur due to interferences or signal obstruction, which is especially common in mobile scenarios. As a consequence, the rendered audio signal at the receiver contains heavy disruptions, resulting in non-understandable words, ultimately leading to a poor user experience. It has to be noted that advanced VoIP codecs already apply Packet Loss Concealment (PLC) strategies to approach this problem, e.g., waveform substitution, as recommended in ITU G.711 [7]. Yet, these strategies can only mitigate the impact of packet loss and not completely overcome it. To quantify the quality of a VoIP call from a user perspective, traditional Quality of Experience (QoE) metrics are based on signal-comparison, e.g, the Perceptual Evaluation of Speech Quality (PESQ) [8] or the Hearing-Aid Speech Quality Index (HASQI) [11]. These metrics were developed to evaluate the artifacts and distortions, which occur due to compression by the voice encoding. But especially in mobile and tactical scenarios, this minor speech signal degeneration is often overshadowed by packet loss-induced disruptions. For good communication, and, thus, a good user experience, the amount of understandable words is much more important than the negative impact of small interferences, like noise or crackling. Similar effects were not only shown for VoIP applications, but also in fundamental neurology studies regarding speech processing in the human brain, e.g., [12]. This is where the here presented **Sp**eech **Q**uality **E**valuation using Word **R**ecognition (SPQER) approach aims at. Instead of performing a straight signal comparison, SPQER uses machine learning-based word recognition to evaluate the percentage of recognizable words in conjunction with a time-based decay function to also penalize delay and cross-talking. This approach is unique to SPQER. By focusing on the amount of understandable words, SPQER can especially aid in the development of new VoIP solutions for scenarios, where precise communication is necessary, e.g. command-and-control scenarios or search-and-rescue missions. For example, SPQER could be used to automatically evaluate the benefit of robust voice codec parameterizations in terms of QoE without the need for labor and cost intensive user studies. Another use case lies within the evaluation and optimization of new network protocols and robustness strategies. One such example is transport layer forward error correction, which is currently discussed by the Internet Engineering Task Force (IETF) Coding for efficient NetWork Communications Research Group (nwcrg)[1] as an extension for the Quick UDP Internet Connections (QUIC) protocol [19]. In contrast to some other VoIP QoE frameworks, e.g., PESQ, SPQER is Open Source, and, thus, available to the public for usage and further development.[2]

Over the course of this paper, we will present the following main contributions and objectives:

- Formal definition of SPQER, a novel metric to evaluate VoIP QoE, based on word recognition.
- A sensitivity analysis assessing the impact of packet loss and delay on SPQER based on testbed experiments.
- A correlation analysis regarding SPQER, PESQ, HASQI, and WER in comparison to the percentage of understandable words (UserWER) and perceptual voice quality (UserMOS) of a conducted user study.

The organization of this paper follows these main contributions. First, a formal definition of the SPQER metric will be given. A sensitivity analysis based testbed experiments is presented for increasing packet loss ratios and delays. Then, a correlation analysis compares SPQER and related state-of-the-art QoE metrics against the results of a conducted user study. Finally, the limitations are discussed, the conclusions are drawn and an outlook for future work is given.

II. FORMAL DEFINITION SPQER

The following section formally introduces SPQER by showing its development step-by-step. The final definition is given later in Equation (9) with a corresponding pseudo-code presented in Listing 1.

Assume we have an original, error-free speech sample V_A and the corresponding receiver-side recording V_B, which was impacted by packet loss on the link. In contrast to traditional VoIP metrics, e.g., the popular Perceptual Evaluation of Speech Quality (PESQ) metric [8], SPQER is not a common full-reference approach. Thus, instead of matching both samples directly, SPQER compares the word classification results W_A and W_B, which are obtained using speech recognition software. These resulting transcriptions include the classified words a_i, their starting times within the sample t_{a_i} and also the corresponding classification confidence c_{a_i}. Usually, a word recognition metric compares the speech-to-text transcription against the human-annotated ground truth. This approach is very labor intensive and makes traditional metrics like the Word Error Rate (WER) inconvenient to use for previously unknown data. SPQER instead automatically creates the reference transcription, removing the human element from the equation. The reference transcription is then compared against the transcription of the evaluated receiver-side recording. It would also be possible to use SPQER with a human-labeled ground truth as reference, but we wanted to create an automatic metric, which does not need direct human interaction and can be run on a batch of previously unknown recordings. This paper uses the following notions and naming conventions. Most of them are here exemplary shown for V_A, a similar notation is used for V_B:

V_A:	Error-free voice sample [audio file].
V_B:	Error-prone voice sample [audio file].
W_A:	Vector of classified words for V_A [strings].
$\|W_A\|$:	Number of words in vector W_A [int].
a_i:	Word i in vector W_A [string].
t_{a_i}:	Starting time of a_i within the sample V_A [ms].
l_{a_i}:	Word duration of a_i [ms].
c_{a_i}:	Word classification confidence of a_i [float].

A. WORD RECOGNITION

The first step of SPQER is to obtain the word recognition results W_A and W_B for the corresponding reference sample V_A and the error-prone recording V_B. In this paper, we use the Speech-to-Text (S2T) API from Google,[3] which is based on a deep learning neural network. In general, SPQER is not bound to a specific S2T API. It would also be possible to use IBM Watson.[4] A free software alternative is Kaldi [17]. For this work, we have chosen Google S2T over Kaldi to avoid overfitting, because the Kaldi classifier was trained on the LibriSpeech dataset [16], from which a subset was also used in our experiments. We note the calculation of SPQER with other speech recognition frameworks as one aspect of our future work.

In the next step, the words within the two classification results W_A and W_B are compared to find the percentage of correct detected words. Here we use a naive search, which iterates the reference vector W_A and tries to find each word in the recorded vector W_B. While string search in general could be done using dedicated string search algorithms, e.g. Boyer–Moore [2], the here used approach can be extended later to also include delay penalization and classification confidence. This would not be possible with the dedicated string search algorithms. The following equation shows our basic word comparison approach:

$$SPQER_\alpha(V_A, V_B) = \frac{\sum_{i=1}^{|W_A|} \max_{k=1}^{|W_B|} (\mathbb{1}(a_i = b_k))}{|W_A|} \tag{1}$$

[1]https://datatracker.ietf.org/rg/nwcrg/about/

[2]https://sys.cs.uos.de/spqer/index.shtml

[3]https://cloud.google.com/speech-to-text/

[4]https://www.ibm.com/cloud/watson-speech-to-text

The term $\mathbb{1}(a_i = b_k)$ is an indicator function, which returns 1 if a_i and b_k are the same words regarding a string comparison or 0 if this is not the case. Note that we use a maximum function instead of a sum for the inner term to count only the best possible match per reference word. Unfortunately, this naive approach fails if the reference sentence contains duplicates. This problem is shown in the following example:

$$W_A = (\text{This, is, the, sentence, for, the, example})$$

$$W_B = (\text{This, is, sentence, for, the, ample})$$

Both *"the"* in W_A would be mapped to the one *"the"* in W_B, leading to an incorrect result. This challenge will implicitly be solved when using a window-based time constrain function to also penalize delay, which we will show in the next step.

B. INTEGRATION OF DELAY PENALIZATION

One novelty of SPQER is the usage of automatic word recognition while also considering the negative impact of delay on the user experience, i.e. to penalize delay-induced cross-talking. This combination is achieved by integrating a time constrain function T as a multiplicative term, leading to the following work-in-progress equation $SPQER_\beta$:

$$SPQER_\beta(V_A, V_B) = \frac{\sum_{i=1}^{|W_A|} \max_{k=1}^{|W_B|} \left(\mathbb{1}(a_i = b_k) \cdot \boldsymbol{T(t_{b_k}, t_{a_i}, l_{a_i}, \epsilon)}\right)}{|W_A|} \tag{2}$$

By integrating T, an examined word b_k must not only match the reference word in a string-comparison but also have temporal proximity. A word can only be valuable if it starts within the time window of the corresponding reference word. We define the time window of a reference word a_i as the interval from its start time t_{a_i} to the time the word has been fully spoken $t_{a_i} + l_{a_i}$, where l_{a_i} is the duration of a_i. The simplest approach to implement this idea is the usage of an indicator function T_{ind}:

$$T_{ind}(t_{b_k}, t_{a_i}, l_{a_i}) = \mathbb{1}(t_{b_k} \in [t_{a_i}, t_{a_i} + l_{a_i}]) \tag{3}$$

Perfect time-aligned words achieve the maximum value of 1. All words lying outside get a value of 0. This also solves the aforementioned problem of duplicates and wrongfully matched words, even if the string comparison succeeds. If delays should not be further punished, this simple indicator function T_{ind} is sufficient. Depending on how strong delays shall be penalized, a different decay function could also be chosen. This makes SPQER flexible for different scenarios and applications. For more real-time critical VoIP applications, a linear decay function T_{lin} within the time window can be used, e.g.:

$$T_{lin}(t_{b_k}, t_{a_i}, l_{a_i}) = \left(1 - \frac{t_b - t_a}{l_a}\right) \cdot T_{ind}(t_{b_k}, t_{a_i}, l_{a_i}) \tag{4}$$

If delays should be even more severely punished, a quadratic decay could also be used, e.g:

$$T_{quad}(t_{b_k}, t_{a_i}, l_{a_i}) = T_{lin}(t_{b_k}, t_{a_i}, l_{a_i})^2 \tag{5}$$

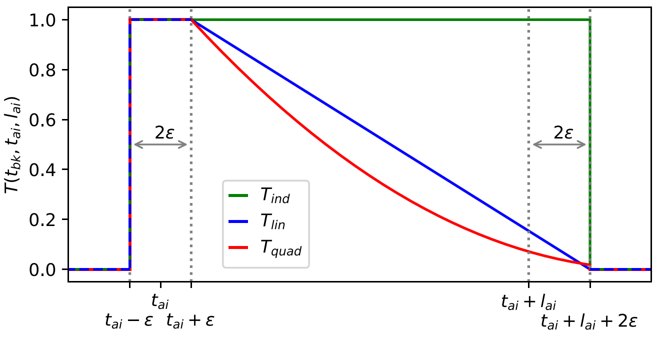


FIGURE 1. Different time window functions T to penalize delay, including error margin ϵ.

For most of this evaluation we have chosen to select T_{lin}, which we recommend to use as the default function. A visualization of the different functions is shown in Figure 1. It has to be noted that this approach is only possible, because the used Google S2T API returns the start and end time for each detected word. Not all speech recognition frameworks contain this necessary feature. Yet, after implementing the equation above, we observed that the calculated metrics were lower than expected. Unfortunately, the used API does not predict the starting times with perfect precision. Sometimes a streamed words starting time is predicted earlier than the corresponding reference starting time. If this problem occurs, the time function returns an unnatural low value, leading to a lower total metric. To overcome this problem, we added an error margin ϵ to the window function. The effect of this improvement is shown in Figure 1. Even if the start times of both a_i and b_k are slightly inaccurate, the metric can still achieve the maximum value. Note that the right window bound also covers the rare case that t_{a_i} has an error of -1ϵ while t_{b_k} has an error of $+1\epsilon$. The integration of this error margin leads to the following modified equations:

$$T_{ind}(t_{b_k}, t_{a_i}, l_{a_i}, \boldsymbol{\epsilon}) = \mathbb{1}(t_{b_k} \in [t_{a_i} - \boldsymbol{\epsilon}, t_{a_i} + l_{a_i} + 2\boldsymbol{\epsilon}]) \tag{6}$$

$$T_{lin}(t_{b_k}, t_{a_i}, l_{a_i}, \boldsymbol{\epsilon}) = \begin{cases} 1 - \frac{t_{b_k} - (t_{a_i} + \boldsymbol{\epsilon})}{l_{a_i} + \boldsymbol{\epsilon}} & t_{a_i} + \boldsymbol{\epsilon} < t_{b_k} \le t_{a_i} + l_{a_i} + 2\boldsymbol{\epsilon} \\ T_{ind}(t_{b_k}, t_{a_i}, l_{a_i}, \boldsymbol{\epsilon}) & \text{otherwise} \end{cases} \tag{7}$$

$$T_{quad}(t_{b_k}, t_{a_i}, l_{a_i}, \boldsymbol{\epsilon}) = T_{lin}(t_{b_k}, t_{a_i}, l_{a_i}, \boldsymbol{\epsilon})^2 \tag{8}$$

Based on our experiences we recommend to use $\epsilon = 0.1 * l_{a_i}$. If a different S2T API is used, this parameter may need to be adjusted.

C. INTEGRATION OF CLASSIFICATION CONFIDENCE

Up to this point, the metric just includes if the word recognition was successful, but not how sure the classifier was. Minor signal distortions, like noise or codec compression, can lower the classification confidence. To cover this, we extended our metric by integrating the classification confidence, which is returned by the S2T API for each detected word. This leads to the final formal definition of the SPQER metric:

$$SPQER(V_A, V_B) = \frac{\sum_{i=1}^{|W_A|} \max_{k=1}^{|W_B|} \left(\mathbb{1}(a_i = b_k) \cdot T(t_{b_k}, t_{a_i}, l_{a_i}, \epsilon) \cdot \boldsymbol{c_{b_k}} \right)}{|W_A|} \quad (9)$$

The following python-like code listing 1 shows a pseudo-code snippet for a practical SPQER implementation based on equation 9. A fully functioning python implementation can be found in our repository.

Listing 1: Python-like pseudo-code for SPQER, using a time function T_{lin} with linear decay.

```
def T_ind(tb, ta, la, epsilon):
  if (tb>=ta-epsilon) and (tb<=ta+la+2*epsilon):
    return 1
  else:
    return 0

def T_lin(tb, ta, la, epsilon):
  if (tb>=ta-epsilon) and (tb<=ta+la+2*epsilon):
    return (1-((tb-ta)/la))*T_ind(tb, ta, la, epsilon)
  else:
    return T_ind(tb, ta, la, epsilon)

def SPQER(referenceFilename, streamedFilename):
  VA = readAudiofile(referenceFilename)
  VB = readAudiofile(streamedFilename)
  WA = S2T_API.recognizeWords(VA)
  WB = S2T_API.recognizeWords(VB)
  spqersum = 0
  for a in WA:
  ta = a.startTimestamp
  la = a.duration
  epsilon = 0.1*la
  bmax = 0
  for b in WB:
  tb = b.startTimestamp
  cb = b.confidence
  bmax = max(bmax, strcmp(a,b)
   *T_lin(tb, ta, la, epsilon)*cb)
  spqersum += bmax
  return spqersum / len(WA)
```

III. RELATED WORK

The novelty of SPQER is the bridging between signal-based VoIP metrics and text-based speech recognition metrics. SPQER differs from existing VoIP metrics, i.e., PESQ, POLQA or ViSQOL, because it uses word-recognition instead of direct signal-comparison. SPQER also differs from existing text-based speech recognition metrics, i.e., the Word Error Rate (WER), because it additionally takes the negative impact of delay and the classification confidence (c_{bk}) into consideration. This combination of both worlds makes SPQER novel and unique. The following section distinguishes our contribution from related work in both fields and summarizes the state-of-the-art speech quality metrics, which will be used for comparison in the later presented evaluation.

A. SIGNAL-BASED SPEECH QUALITY METRICS

Evaluation of speech quality is an important area in the field of QoE research. In the last decade, the focus has switched from traditional telephone call quality analysis to voice codec evaluation in IP networks. Yet, common signal-based metrics, e.g., PESQ, still originate from the non-IP-based, public switched telephone network (PSTN) era. It has to be noted that there exist several more speech quality metrics, e.g., XXX. We will only summarize the ones, which we used in our evaluation.

PESQ: The Perceptual Evaluation of Speech Quality (PESQ) is an ITU-T recommendation to analyze the speech quality of telephony systems [8]. PESQ aligns the reference and the recorded signals in time and uses a fast Fourier transform (FFT) for filtering. The raw PESQ value is then calculated based on disturbances in frequency and time, which are aggregated using an Lp norm. The conversion term to convert the raw PESQ values into Mean Opinion Score Listening Quality Objective (MOS-LQO) values is defined in ITU recommendation P.862.1 [9]. While there are several commercial PESQ implementations, for this work we use the ITU reference implementation,[5] which is freely available for research purposes. It has to be noted that the Perceptual Objective Listening Quality Analysis (POLQA, P.863, [10]) is the successor of PESQ and currently, the ITU recommended speech quality metric. Unfortunately, there is no free POLQA implementation available to the public. This is one reason why PESQ is still the reference for us and probably for many other publications.

HASQI: The Hearing-Aid Speech Quality Index (HASQI) [11] was developed to measure the effects of distortions on the speech quality for persons in need of a hearing aid. HASQI is calculated by using two components. The first component measures how well the spectral representations of a signal fits its reference, measuring the effects of noise and nonlinear distortion. The second component considers the impact of linear filtering and spectral changes by analyzing the long-term average spectra. The shown HASQI values were generated using a MATLAB code of HASQI Version 2 [11], which can be obtained by contacting the authors.

Another notable VoIP metric is ViSQOL [6], which analyses spectro-temporal signal characteristics to model human perception. According to a study by the authors, ViSQOL achieves a closer correlation to the MoS than PESQ and POLQA. The software was recently made available to the public.[6] Yet, because PESQ is still the commonly used metric, we do not investigate ViSQOL any further. but notice it for future work.

[5]https://www.itu.int/rec/T-REC-P.862-200303-S!Amd1/en
[6]https://github.com/google/visqol

B. TEXT-BASED SPEECH RECOGNITION METRICS

The field of speech recognition has seen major improvements in the last years. While the basic approach has not changed - a model is learned based on a training set, which is then used to classify new data - new algorithms have greatly increased accuracy and recall. Especially the developments in the field of deep-learning neural networks have pushed the capabilities to the current high level. These improvements enable robust word recognition, making an approach like SPQER reliable, which was not possible in the past.

WER: The Word Error Rate (WER) is the common metric to evaluate speech recognition software. It is based on the Levenshtein distance and is defined as follows:

$$WER = \frac{\#Substitutions + \#Deletions + \#Insertions}{\#ReferenceWords} \quad (10)$$

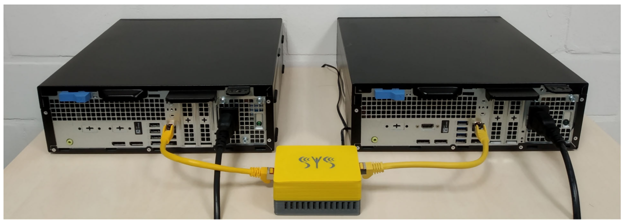
While this metric works for text comparison, it is not suitable to evaluate real-time VoIP communication. The WER equation does neither include delays, nor classification confidence. This is important for IP-based communication, where delays and loss induced distortions can occur and negatively affect the user experience. In this paper, we mostly use $1 - WER$ to be consistent with the other metrics, whose values decrease with increasing packet loss.

There are several publications dedicated to the effect of packet loss on speech recognition, i.e., [3] or [13]. Other research focuses on the improvement of speech recognition in lossy scenarios by using FEC algorithms [4] [1]. All these studies aimed to train better speech recognition models with improved robustness against packet loss. This work focuses instead on the usage of speech recognition for QoE evaluation in networking research and not on the improvement of the recognition algorithms themselves. There is little related work regarding a comparison of text-based speech recognition and signal-based speech quality evaluation. An interesting evaluation for several metrics was done by Hall *et al.* in [5], which among others also examines the impact of packet loss on PESQ and WER.

IV. EVALUATION SETUP

A series of testbed experiments were conducted to evaluate the impact of packet loss and delay on SPQER. The following section presents the testbed setup to record the samples, which were used to perform a sensitivity analysis, conduct a short user study and compare SPQER against the existing metrics, namely PESQ, HASQI, and WER.

The evaluation testbed physically consists of two Dell Optiplex Desktop PCs, whose Ethernet interfaces are connected by a layer-2 link emulation bridge, called Link 'Em [18]. The system uses a three-step approach to stream and then evaluate the recorded voice samples. First, all original voice samples are streamed over a lossy link and recorded in real-time. This process is repeated for each packet loss ratio. Finally, the recordings are used to calculate the values for PESQ, HASQI, WER and SPQER. The same recordings are also used to conduct an initial user study. Figure 2 shows the complete system architecture, including the recording data flow (yellow), the metric calculation flow (green) and the user study flow (violet).



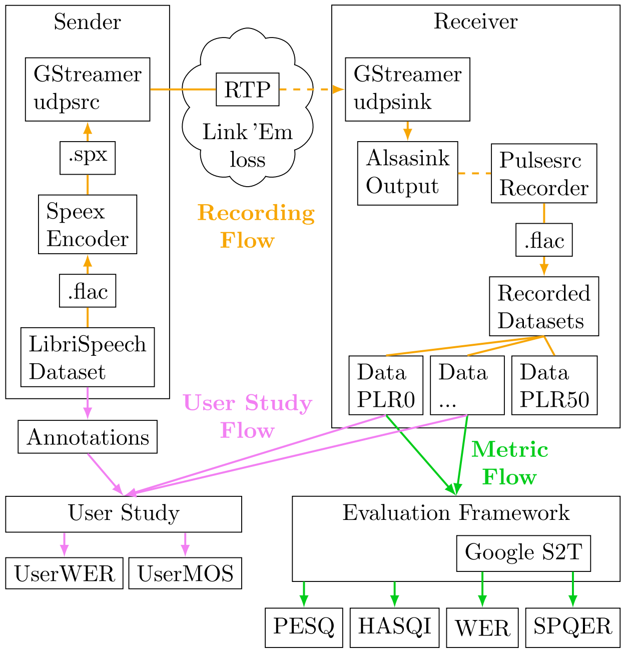


FIGURE 2. Testbed setup and evaluation system architecture to record the streamed samples and calculate all used metrics.

The conducted evaluation is based on a subset of the LibriSpeech dataset [16]. We have chosen to use this dataset, because it is freely available for download and has precise annotations, which is necessary for the conducted evaluation. More information on the LibriSpeech dataset, for example, sample rate or speaker gender distribution and can be found in [16]. To save streaming time and speech recognition costs, a subset of 250 samples was randomly chosen from a list of all samples with at least 10 words. This length constraint was introduced, because Google's *default* speech recognition model was trained on similar samples. For a *command-and-search* use case, the API also offers a model for shorter sequences. An evaluation of the model impact is omitted here but noted for future work.

The recording flow to collect the evaluation samples is indicated by the yellow arrows in figure 2. Before streaming, all *LibriSpeech*. flac files were encoded using *Speex* [20], which is a patent-free voice codec. This pre-processing step was implemented to exclude possible encoding latencies or processing delays from the evaluation. The codec was parameterized using the default configuration of the *Speex* plug-in of *GStreamer*,[7] which is a pipeline-based framework to process and stream video and audio files. The bit rate was set to $27Kb/s$ with a sampling rate of 16 kHz. The encoder produced one Speex frame every 20 ms. It has to be noted that the chosen default configuration uses simple zero insertion instead of a more advanced PLC strategy. For future work, it would be interesting to evaluate the impact of PLC algorithms like waveform substitution, as recommended in ITU G.711 [7]. It has to be noted that *Speex* shall be replaced by *Opus* [15] in the long term. But since relevant existing VoIP solutions still rely on it, we have chosen to use *Speex* in our setup. Also, *Speex* is one of the few codecs which is natively supported by Google's Speech-to-Text API. While *Speex* already uses minimal inter-frame dependency to achieve some level of robustness against packet loss [20], there are more robust audio codecs. Unfortunately, those are either closed source or not supported by the S2T API, e.g. STANAG-4591 [21] for robust tactical communication.

After encoding, each audio sample is then transmitted via *GStreamer* using RTP over UDP. Here, it is used to mimic a real-time VoIP application. The links packet loss rate is emulated using our previous developed *Link'Em* bridge [18] for reproducible networking research. *Link'Em* is essentially a Raspberry Pi 3 to which an additional Realtek 8153 USB Ethernet dongle is added. Both Ethernet interfaces are configured using *brctl* to form a layer-2 bridge. The desired link characteristics are then emulated on the new bridge interface using an extended version of *netem*. A ready-to go image of *Link'Em* and further information can be found on the projects website.[8] If put between a connection of two hosts, the bridge transparently emulates the desired link conditions, here random uniform packet loss, with $PLR \in [0, 50]$. The upper packet loss limit is chosen unusual high compared to most link emulations, but even for high packet loss rates, the later shown user study has verified that a human can still understand a good percentage of words (c.f. Figure 8).

Upon reception, the *GStreamer udpsink* decapsulates the received RTP packets and forwards the data stream to an *alsasink* output. There, the incoming voice signal is live rendered as if a user would hear the VoIP call in real-time. While the received voice sample is played, a parallel running process records the output from the *alsasink* using a *pulsesrc* recorder. The recording is saved in a lossless. flac format to not skew the evaluation by using an additional compression step. This streaming and recording process is repeated for each original voice sample for each emulated packet loss ratio.

Finally, the evaluation framework calculates the WER, PESQ, HASQI and SPQER values for each recorded sample, as depicted by the green flow in figure 2. Notice, we have chosen to use the streamed recordings without packet loss (PLR0) as reference and not the original LibriSpeech source files. This decision was made to not place PESQ and HASQI at a disadvantage due to possible distortions by the additional sender-side audio codec compression while streaming.

[7]https://gstreamer.freedesktop.org/documentation/speex/index.html
[8]https://sys.cs.uos.de/linkem/index.shtml

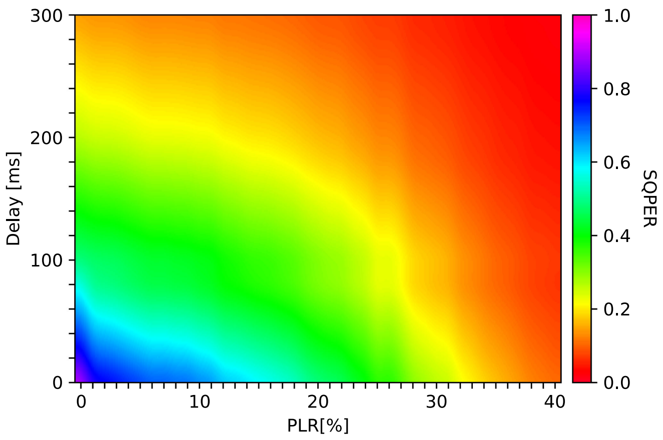


FIGURE 3. **Heatmap of average SPQER values for increasing levels of PLR and delay. T_{lin} used as time function. Values are bicubic interpolated.**

It has to be noted that the recorded experiments were conducted for different PLRs. To also highlight the impact for increasing delays, we have simulated additional delay-prone transmissions for each recording. The simulated delay was increased stepwise by 10 ms and added to each word classification results. The delay was increased stepwise by 10 ms. In a real-world scenario, the impact of this delay can depend on the parameterization of the used applications jitter buffer, which queues delayed and out of order packets to create a steady data stream.

V. SENSITIVITY ANALYSIS

The following section presents SPQERs behavior to increasing levels of packet loss and delay. Figure 3 shows a heatmap of the average SPQER values of all samples for the corresponding link characteristics. The declining scores show that SPQER considers both, PLR and delay, to calculate the quality of experience. In the following sensitivity analysis. we will highlight exactly how the link characteristics affect the different parts of the metric and how strongly this impacts the final score. It has to be noted, that the sensitivity analysis is heavily impacted by the used classification framework. The here conducted evaluation is based on Google's *Cloud Speech-to-Text API* with client library v0.27. All used word recognition results were obtained in July 2019. Since the framework is running in the cloud, it is possible that the underlying classifier can change in the future.

A. IMPACT OF PACKET LOSS RATE

The impact of an increasing packet loss ratio on SPQER for links without additional delay is given in Figure 4. Notably, there is an almost linear relation between loss rate and the SPQER metric. This decline is caused by two parts of the

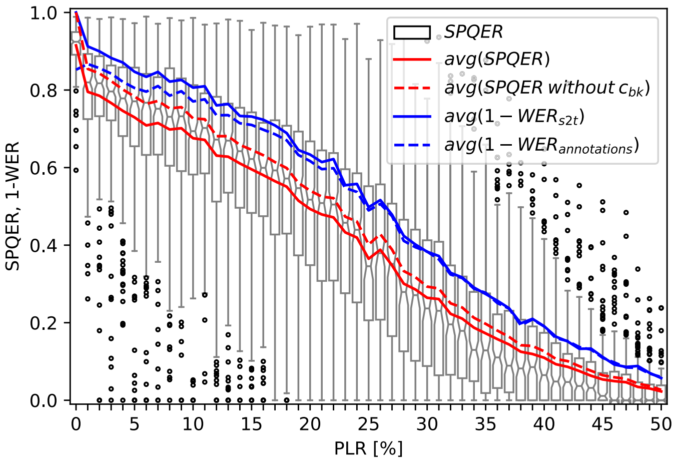


FIGURE 4. Impact of PLR on 1-WER, SPQER and SPQER without the integration of classification confidence c_{b_k}. No additional emulated delay (3±1 ms).

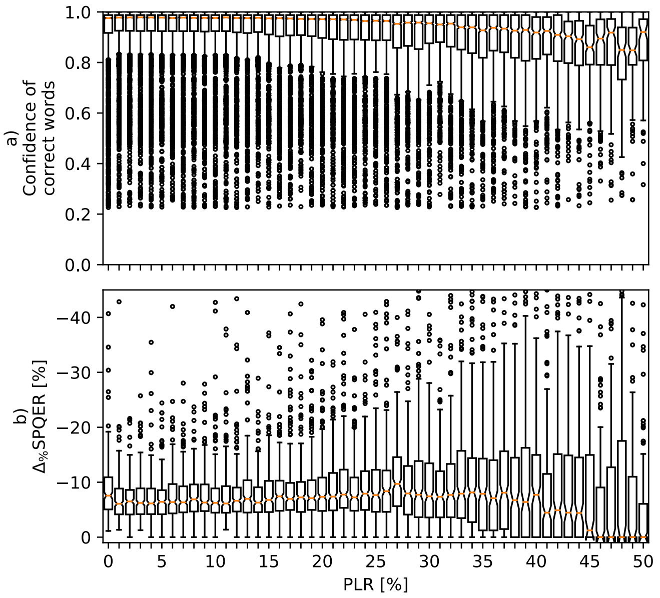


FIGURE 5. (a) Classification confidence of correct recognized words. (b) Percentage change of SPQER and SPQER without classification confidence c_{b_k}. No additional emulated delay (3±1 ms).

SPQER equation (Eq. (9)): the word recognition rate and the classification confidence.

1) WORD RECOGNITION RATE

As shown in Figure 4, an increasing packet loss rate leads to a lower word recognition rate. This is not only true for the traditional WER metric, but also for the word recognition done by SPQER. Both metrics have a similar trend. Notice, the figure shows $1 - WER$ for better comparability. Without the integration of classification confidence and without the occurrence of delay, SPQER mainly does word recognition, and, thus, logically behaves similarly to the WER metric.

In contrast to the traditional WER approach, SPQER does not need human-annotated ground truth data, which makes a direct comparison of these metrics inherent unfair. However, it is possible to also automatically calculate WER values by using a S2T transcription of the lossless recording as reference. This is the reason why Figure 4 shows two different WER graphs. The $WER_{annotations}$ (c.f., dashed blue line) values are calculated using the human annotated ground truth of the *LibriSpeech* data set. The WER_{s2t} (c.f., dashed blue line) graph instead was generated by the fully automatic approach. This difference in approach explains the gap between the two metrics at $PLR = 0$. A similar gap occurs for SPQER, if used with (c.f., solid red line) and without considering the classification confidence (c.f., dashed red line). Integrating the classification confidence will mitigate this problem (c.f., solid red line), as we will discuss in-depth in the next section.

2) CLASSIFICATION CONFIDENCE

To consider minor signal distortions, like noise or codec compression, which can lower the quality of experience, SPQER also includes the classification confidence c_{b_k} of a detected word. This also mitigates the aforementioned problem when evaluating a lossless sample against the reference. The impact of this extension can be seen by comparing the two red lines in Figure 4. For $PLR = 0$, SPQER gives a perfect score of 1, if the classification confidence c_{b_k} is ignored. Minor distortions due to voice codec compression are not reflected in this perfect score. While using the classification confidence as a multiplicative term does not disruptively change the metric's value, it leads to minor diminutions, and, thus, solving the issue. A deeper evaluation of the confidence impact is shown in Fig. 5. The upper subplot shows the confidence score c_{b_k} of each best matching word $b \in B$ that succeeded the string comparison ($a_i = b_k$) and had the best time alignment score $max(T(t_{b_k}, t_{a_i}, l_{a_i}, \epsilon))$. Words, which do not succeed both checks, always achieve a score of 0 anyway. Therefore, their classification confidence is not needed. As seen in Fig. 5 a, the classification confidence slightly decreases with higher loss levels. The resulting percentage change by applying these confidence values to the SPQER metric is shown in the lower subplot. While this change does not significantly impact the final SPQER score for higher error rates, it is especially important if no packet loss occurs. The lossless average value without confidence is 1.0, incorporating the confidence reduces it to 0.92 (c.f. Fig. 4, PLR0). This penalization of smaller artifacts allows to finer differentiate recordings that achieve the same word recognition percentage. For future work, it would be interesting to analyze if it is helpful in some scenarios to use a steeper confidence function to increase its impact, e.g., use $c_{b_k}^2$.

B. IMPACT OF DELAY

One novelty of SPQER is to penalize delay by using a time constrain function. To evaluate the impact for increasing delays, we have simulated additional delay-prone transmissions for the previously streamed recordings. Fig. 6 shows the

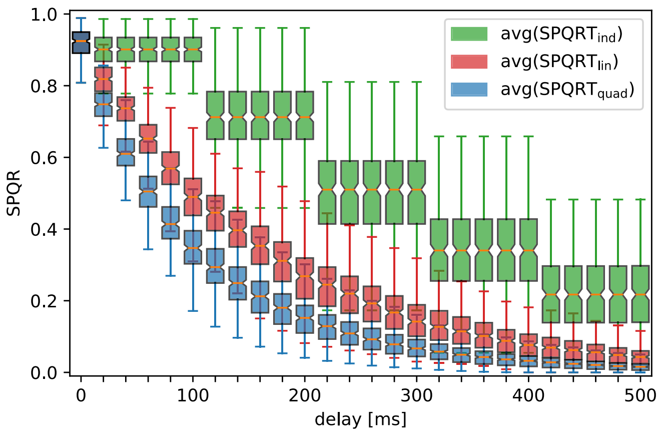


FIGURE 6. Impact of delay on SPQER for different T functions. All recordings with 0% PLR.

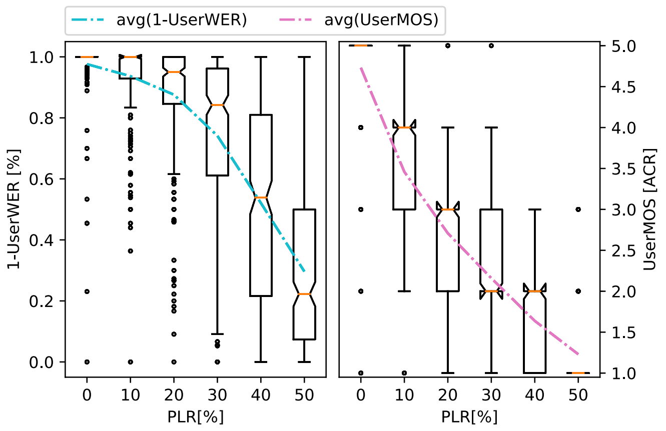


FIGURE 7. 1-UserWER and UserMOS values of the conducted user study. Averages represented as lines.

results for the lossless recordings. We have also simulated delay-prone transmissions for different PLR levels. But since the impact was similar, an additional figure is omitted.

Most notably are the steps when using T_{ind}. These are invoked by rounding errors within the classification software. The used Google S2T API returns the time values as a tuple of seconds and nanoseconds. Yet, the nanoseconds are always rounded to centiseconds. Thus, the steps always occur at multiple of 100 ms. Unfortunately, it is currently not possible to fix this problem, because the software is closed source. We have already contacted the Google development team and hope it will be possible to turn of this rounding in the future. Alternatively we could switch to an open-source speech recognition framework like Kaldi [17], where such changes could be made.

Interestingly, despite the selection of a linear decay function T_{lin}, the delay impact is quadratic instead of linear. By applying a steeper function like T_{quad}, the impact is weighted even more. Such a stronger delay penalization could be useful to evaluate VoIP applications with stricter latency constrains, which highlights the flexibility of SPQER. Yet, we recommend using T_{lin} as the default function.

VI. USER STUDY & RELATED WORK COMPARISON

In the following section, we will compare SPQER to related state-of-the-art speech quality metrics, namely PESQ and HASQI. We will also present a preliminary user study to give an indication of the correlation to human perception.

A. CONDUCTED USER STUDY

A short user study was conducted to get a MOS and WER reference baseline. To distinguish from the other metrics, we will use the naming convention $UserMOS$ and $UserWER$ to indicate these user study-based metrics. The study included 8 participants in the age between 20 and 38 years (avg. 25.3). All participants were computer scientists, students or research assistants. Most speakers were non-native but fluent in English on a scientific level. For each participant, a random subset of original files was chosen. The recordings of these files were then played over a headphone to the participant for each $PLR \in \{0, 10, 20, 30, 40, 50\}\%$. During the trial, the order of all audio files was randomized to minimize bias. After hearing a sample, the participant first classified the speech quality using the ACR scale. Then the $LibriSpeech$ annotations were shown and the user typed in the number of non-understandable or misheard words. It took on average 90 minutes to complete the study. The resulting audio file classifications per participant led to a total volume of 1800 $UserMOS$ and 1800 $UserWER$ classifications. Fig. 7 shows the total scores for $UserMOS$ and $UserWER$. While there is a high variance in the data per PLR, the mean values per user form a smooth curve. The study shows that the perceived speech quality decreases much faster than the percentage of understandable words, which was also mentioned by most participants after completing the study. One reason for this is the inter-frame packet loss concealment of the used $Speex$ voice codec. The algorithm stretches the impact of packet loss, thus, instead of heavy but short interruptions, a higher degree of noise can be heard over a sequence of frames. This strategy fits well with with the evolution of the human brain to understand words even in the presence of noise. Similar observations are described in fundamental neurology studies, e.g., [12]. We are aware that the conducted small study is not fully representative due to its small number of participants. But this paper focuses on the introduction of SPQER and its technical details and not on the presentation of a full-size user study. Also, the study was designed to evaluate different PLR levels and not increasing delays. We are looking forward to extend our user study in our future work and also include native English speakers to get rid of potential second language effects.

B. CORRELATION ANALYSIS

The following section compares the impact of packet loss on SPQER to related work metrics, namely PESQ, HASQI, and WER in correlation to the UserWER and UserMOS scores of the conducted user study. The results of this study are shown

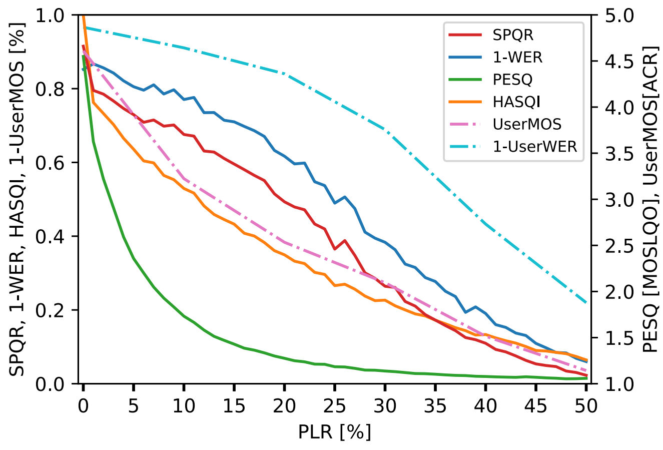


FIGURE 8. Impact of increasing PLR on speech quality metric averages. Constant delay of 3±1 ms.

TABLE 1. Pearson Correlation Scores Between Metrics and User Study, Corresponding to the Graphs in Fig. 8

Correlation		1-UserWER			UserMOS		
		r^2	🏆	p	r^2	🏆	p
🟩	PESQ	0.36	(4)	< 0.001	0.75	(4)	< 0.001
🟧	HASQI	0.75	(3)	< 0.001	0.98	(1)	< 0.001
🟦	1-WER	0.95	(1)	< 0.001	0.90	(3)	< 0.001
🟥	SPQER	0.91	(2)	< 0.001	0.95	(2)	< 0.001

in Figure 8 with the corresponding Pearson correlation scores presented in Table 1.

First, we will focus on the impact of packet loss on voice quality, which is represented by the correlation to the User-MOS score (c.f., cyan line Fig. 8). Most notably, an increasing packet loss ratio leads to an exponential decay for PESQ, while all other metrics show a less steep regression. Even for recordings with a very low PESQ value, the user study has shown that a human can still recognize a large percentage of words. This heavy impact on PESQ is evoked by its MOS-LQO conversion, which was parameterized based on a user study to evaluate the small but perceptible impact of voice encoding distortion. Packet loss distortions instead are more disruptive. Thus, PESQ is not well suited to evaluate the amount of understandable words, because it overcompensates if loss occurs. Very impressive is the almost perfect ($r^2 = 0.98$) correlation of HASQI to the UserMOS value. Yet, HASQI correlates just average ($r^2 = 0.75$) to the amount of understandable words, represented by the 1-UserWER score. The word recognition-based metrics SPQER and 1-WER on the other hand correlate well to both user study scores, with SPQER having a marginal better sum. It has to be noted that the presented study only considers packet loss and does not further include the impact of delay, which is a unique feature of SPQER and would distinguish it further from the WER metric. We are looking forward to also include the impact of delay when extending our user study in future work.

VII. CONCLUSION & FUTURE WORK

In this paper, we have introduced SPQER, a novel word recognition-based QoE metric for the evaluation of VoIP applications. SPQER is fully automatic and can be applied to an unknown batch of recordings without the need for additional human interaction to classify a ground truth. The conducted sensitivity analysis shows that SPQER does not only consider the impact of packet loss but also factors in delay. Thus, SPQER is able to asses time-critical VoIP applications more precisely, compared to existing text-bound metrics, such as the Word Error Rate. The final correlation analysis to a short user study shows that SPQER can better evaluate the percentage of understandable words than related state-of-art speech quality metrics, namely PESQ, HASQI, while still giving a good indication of the voice quality.

It has to be noted, that the presented evaluation results depend heavily on the chosen speech-to-text software and used voice codec. The main goal of this work was to introduce SPQER, highlight the unique benefits, and discuss possible limitations. In general, SPQER is not bound to be used with a specific classifier or codec. It would be interesting to evaluate SPQER in different setups, using other classification frameworks, e.g., IBM Watson or Kaldi, and voice codecs, e.g., Opus. Yet, the sensitivity analysis must be repeated if a different setup is used. Also, with the groundbreaking progress in the field of word recognition, classifiers may achieve a better word recognition rate than humans. If this should be the case, it would be necessary to train a dedicated classifier, which mimics the human perception. For future work, our next objective is to extend the user study to further strengthen our presented results. We are also looking forward to analyze the impact of burst losses, i.e., by applying a Gilbert-Elliot model, which could be parameterized using real-world traces.

REFERENCES

[1] M. Assefi, M. Wittie, and A. Knight, “Impact of network performance on cloud speech recognition,” in *Proc. Int. Conf. Comput. Commun. Netw.*, 2015, pp. 1–6.

[2] R. S. Boyer and J. S. Moore, “A fast string searching algorithm,” *Commun. ACM*, vol. 20, no. 10, pp. 762–772, 1977.

[3] C. Boulis, M. Ostendorf, E. A. Riskin, and S. Otterson, “Graceful degradation of speech recognition performance over packet-erasure networks,” *IEEE Trans. Speech Audio Process.*, vol. 10, no. 8, pp. 580–590, Nov. 2002.

[4] A. M. Gómez, A. M. Peinado, V. Sánchez, and A. J. Rubio, “Combining media-specific FEC and error concealment for robust distributed speech recognition over loss-prone packet channels,” *IEEE Trans. Multimedia* vol. 8, no. 6, pp. 1228–1238, Dec. 2006.

[5] T. A. Hall, “Objective speech quality measures for internet telephony,” in *Proc. Int. Symp. Convergence IT Commun.*, 2001, vol. 4522, pp. 128–136.

[6] A. Hines, J. Skoglund, A. C. Kokaram, and N. Harte, “ViSQOL: An objective speech quality model,” *EURASIP J. Audio, Speech, Music Process.*, vol. 13, pp. 1–18, 2015.

[7] ITU-T, “G.711 : Pulse code modulation (PCM) of voice frequencies,” Recommendation G.711, International Telecommunication Union, Geneva, 1988.

[8] ITU-T, “P.862: Perceptual evaluation of speech quality (PESQ),” Recommendation P.862, International Telecommunication Union, Geneva, 2001.

[9] ITU-T, “P.862.1: Mapping function for transforming p.862 raw result scores to MOS-LQO,” Recommendation P.862.1, International Telecommunication Union, Geneva, 2003.

[10] ITU-T, “P.863: Perceptual objective listening quality prediction,” Recommendation P.863, International Telecommunication Union, Geneva, 2018.

[11] J. M. Kates, and K. H. Arehart, “The hearing-aid speech quality index (HASQI) version 2,” *J. Audio Eng. Soc.*, vol. 62, no. 3, pp. 99–117, 2014.

[12] W. Marslen-Wilson and L. K. Tyler, “The temporal structure of spoken language understanding,” *Cognition*, vol. 8, no. 1, pp. 1–71, 1980.

[13] P. Mayorga, L. Besacier, R. Lamy, and J.-F. Serignat, “Audio packet loss over IP and speech recognition,” in *Proc. IEEE Workshop Autom. Speech Recognit. Understanding*, 2003, pp. 607–612.

[14] G. Mittag and S. Möller, “Non-intrusive speech quality assessment for super-wideband speech communication networks,” in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2019, pp. 7125–7129.

[15] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, “Librispeech: an ASR corpus based on public domain audio books,” in *Proc. IEEE Int. Conf. Acoust., Speech Sig. Process.*, 2015, pp. 5206–5210.

[16] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, “Librispeech: An ASR corpus based on public domain audio books,” in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2015, pp. 5206–5210.

[17] D. Povey *et al.*, “The Kaldi speech recognition toolkit,” in *Proc. IEEE Workshop Autom. Speech Recognit. Understanding*, 2011.

[18] B. Schuetz *et al.*, “Link’em: An open source link emulation bridge for reproducible networking research,” in *Proc. Int. Conf. Netw. Syst.*, 2019, pp. 1–3.

[19] I. Swett, M.-J. Montpetit, V. Roca, and F. Michel, “Coding for QUIC,” IETF NWCRG Internet-Draft, draft-swett-nwcrg-coding-for-quic-04, Work in Progress, Sep. 2020.

[20] J-M. Valin, “Speex: A free codec for free speech,” 2016, *arXiv:1602.08668*.

[21] T. Wang, K. Koishida, V. Cuperman, A. Gersho, and J. S. Collura, “A 1200/2400 bps coding suite based on MELP,” in *Proc. IEEE Workshop Speech Coding*, 2002, pp. 90–92.