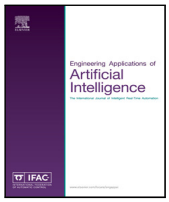
Engineering Applications of Artificial Intelligence 100 (2021) 104189

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai



Analysis of the sensitivity of the End-Of-Turn Detection task to errors generated by the Automatic Speech Recognition process

César Montenegro [a,*], Roberto Santana [a], Jose A. Lozano [a,b]

[a] Intelligent Systems Group, Department of Computer Science and Artificial Intelligence, University of the Basque Country UPV/EHU Paseo Manuel de Lardizabal 1, 20018 Donostia-San Sebastián, Spain

[b] Basque Center for Applied Mathematics (BCAM), Bilbao, Spain

ARTICLE INFO

MSC:
00-01
99-00

Keywords:
Spoken dialogue systems
Automatic speech recognition
End of turn detection
Natural language processing
Neural networks

ABSTRACT

An End-Of-Turn Detection Module (EOTD-M) is an essential component of automatic Spoken Dialogue Systems. The capability of correctly detecting whether a user's utterance has ended or not improves the accuracy in interpreting the meaning of the message and decreases the latency in the answer. Usually, in dialogue systems, an EOTD-M is coupled with an Automatic Speech Recognition Module (ASR-M) to transmit complete utterances to the Natural Language Understanding unit. Mistakes in the ASR-M transcription can have a strong effect on the performance of the EOTD-M. The actual extent of this effect depends on the particular combination of ASR-M transcription errors and the sentence featurization techniques implemented as part of the EOTD-M. In this paper we investigate this important relationship for an EOTD-M based on semantic information and particular characteristics of the speakers (speech profiles). We introduce an Automatic Speech Recognition Simulator (ASR-SIM) that models different types of semantic mistakes in the ASR-M transcription as well as different speech profiles. We use the simulator to evaluate the sensitivity to ASR-M mistakes of a Long Short-Term Memory network classifier trained in EOTD with different featurization techniques. Our experiments reveal the different ways in which the performance of the model is influenced by the ASR-M errors. We corroborate that not only is the ASR-SIM useful to estimate the performance of an EOTD-M in customized noisy scenarios, but it can also be used to generate training datasets with the expected error rates of real working conditions, which leads to better performance.

0. Introduction

Implementing Spoken Dialogue Systems involves solving several difficult machine learning problems. This includes, among others, speech recognition, Natural Language Understanding, semantic disambiguation, and non-trivial response generation. An additional problem is cascading failure, in which an early mistake in any of the system components, will harm the performance of the subsequent components. In particular, mistakes in the Automatic Speech Recognition Module (ASR-M) of a dialogue system based on the architecture illustrated in Fig. 1(a) will have an effect on the performance of the End-Of-Turn Detection Module (EOTD-M) and Natural Language Understanding Module (NLU-M). This consequently affects the overall performance of the system. While different approaches have addressed the question of solving or mitigating the errors produced in the ASR-M (Fernández-Díaz and Gallardo-Antolín, 2020; Graves et al., 2013; Squartini et al., 2012; Zhou et al., 2014; Trentin and Matassoni, 2003; Hannun et al., 2014; Shahamiri and Salim, 2014; Salem et al., 2007; Amrouche et al., 2010), only a few papers analyze the impact of these errors in subsequent components. Voleti et al. (2019) analyzed the effects of word substitution errors on sentence embeddings, and Simonnet et al. (2018) measured the impact of word substitution errors produced by ASR-M on NLU-M. Nevertheless, the question of the relationship between the different types of ASR-M errors and their influence on the EOTD-M has not been addressed. This question is relevant as the deterioration of the performance of EOTD-M due to ASR-M errors can be different as a function of the error: the EOTD-M can be insensitive to some errors but very sensitive to other types of errors. Furthermore, different methods of converting words into numerical information (featurization) exploit different features of speech, consequently the combination of classifier and featurization techniques could also be sensitive to some errors and insensitive to other types of errors. However, investigating this relationship is complicated by the fact that the particular errors that an ASR-M produces depend on the features of human speech, ambient noise, and the performance of the ASR-M itself. It is very difficult to accurately induce specific errors in the ASR-M by manually manipulating these input characteristics. Some studies, such as (Shao and Chang, 2011), manipulate the intensity of different types of noise (Gaussian noise,

* Corresponding author.
E-mail address: cesar.montenegro@ehu.eus (C. Montenegro).

https://doi.org/10.1016/j.engappai.2021.104189
Received 7 October 2020; Received in revised form 25 January 2021; Accepted 1 February 2021
Available online 15 February 2021

pink noise, Volvo engine noise, and speech-like noise) introduced into speech to evaluate the robustness evaluation of an ASR-M. Their ASR-M produces different rates of errors depending on the intensity of the introduced noise, nevertheless, they cannot control what types of errors are generated by the ASR-M.

In this paper, we introduce an ASR Simulator (ASR-SIM) that replicates the different transcription errors produced in an ASR-M due to noise, or due to the particular speech profile, without manipulating human speech or adding noise to an acoustic input. The ASR-SIM allows us to investigate the relationship of these errors with several EOTD-M, with different featurization techniques.

The main contributions of this paper are as follows: It analyzes for the first time how different errors produced by ASR-M can affect the non-trivial task of End-Of-Turn Detection. Secondly, it introduces an ASR-SIM, which is capable of simulating different types of errors produced by the ASR-M, and simulate speaker features that can be used by other modules that form part of a Spoken Dialogue System. There is not other work to our knowledge that has addressed the task of creating such a simulator, the closest comparable works being the above mentioned from Voleti et al. (2019) and Simonnet et al. (2018).

The paper is organized as follows: In Section 1, we present the necessary background on Spoken Dialogue Systems, emphasizing the role of the EOTD-M and ASR-M. In Section 2, we describe the different classes of errors that can be produced by an ASR-M as well as the characteristics of a speech profile. Section 3 introduces a flexible simulator of the ASR-M. In Section 4, we describe the experimental framework and the featurization techniques used. In Section 5 we present and discuss the results of our experiments. Section 6 concludes the paper and discusses future work.

1. Background

1.1. End-Of-Turn detection in Spoken Dialogue Systems

The audio signal received by the ASR-M is a continuous stream of audio. The system must filter the human voice from ambient noise, and estimate the best group of words that corresponds to the audio signal. As a result, the ASR-M outputs a stream of words with timing information, which could be hundreds of words long in a whole conversation. A conversation between two humans consists of a turn-taking transference of information, and replacing one of the humans with a bot requires the detection of the user's End-Of-Turn pauses. The goal of an EOTD-M is to detect this change of turn in a conversation between a human and the system. This triggers the evaluation of the sentence or sentences received by the NLU-M.

The consequences of failing in EOTD-M are:

1. **Anticipation:** When the NLU-M receives an incomplete sentence, the system may potentially answer while the user is still talking, causing overlap between the speech of the human and the system. Some systems close the users microphone (Chang et al., 2017) when answering, missing all the information transmitted by the user during the overlap.
2. **Excessive delay:** when an End-Of-Turn is not detected in time, the time gap between a real End-Of-Turn and the reply from the system is too high, and the user experience is harmed by unnatural waiting times between turns.

Several aspects have to be considered when designing an EOTD-M. Particularly relevant are the architecture of the spoken dialogue (which defines the input to the EOTD-M) and the features used in the classification problem.

The architecture of a Spoken Dialogue System can limit the input resources of an EOTD-M. Figs. 1(a) and 1(b) illustrate how two common architectures differently condition the input of the EOTD-M. In Fig. 1(a), the EOTD-M receives information exclusively from the ASR-M, while in Fig. 1(b) not only can ASR-M information be received, but also raw audio data. We can find studies in the literature that are based on the architecture of Fig. 1(a), such as the work by Razavi et al. (2019), who study the impact of the prediction power of features extracted from pause, prosodic, timing, lexical, syntactic and semantic information. Nevertheless, it is more common to find studies using features extracted from raw audio data, following the architecture in Fig. 1(b). There are different features that can be extracted from raw audio data, Chang et al. (2017) extracted 40-dimensional log-Mel filterbanks with an upper limit of 4 kHz and a frame step of 10 ms using a 25 ms window, while (Maier et al., 2017) and Aldeneh et al. (2018) used raw pitch (F0), smoothed F0 contour, Root Mean Square signal energy, the logarithmized signal energy, intensity, loudness, MFCC and smoothed pitch.

These two architectures exploit only the user's speech information, but different architectures can offer more sources of information, for example the architecture presented by Masumura et al. (2018) uses the user's utterance in conjunction with the interlocutor's utterance.

1.2. Automatic speech recognition in Spoken Dialogue Systems

Automatic speech recognition is the procedure through which a speech signal is converted into a representation of words or other linguistic entities by means of automated algorithms. It has been an active research area for decades, as it has always been considered as an essential tool in human–machine communication (Yu and Deng, 2016).

In Fig. 2 an example architecture of ASR-M, EOTD-M and NLU-M is illustrated. Particularly, in the ASR-M architecture shown, the feature extraction component takes as input the raw audio signal, filters noises that do not correspond to human speech frequencies, and extracts frequency-domain feature vectors that are used to feed the following acoustic model. The acoustic model estimates one or several sets of words that best match with the feature vectors given, where each set of words is an hypothetical sentence based only on acoustics. The acoustic model integrates knowledge about acoustics and phonetics, and for each hypothetical sentence, estimates the similarity score with the audio. The language model estimates another score for each hypothesized sentence, this time, based on correlation between words learned from a training corpora. The language model score can often be estimated more accurately if the training corpora are related to the task domain. These two scores from each hypothesis are combined in the hypothesis search component to output the word sequence with the highest score as the recognized sentence (Yu and Deng, 2016). More recent architectures such as end-to-end ASR-M architectures simplify the conventional ASR-M architecture into a single Deep Neural Network (DNN) architecture. Besides, the end-to-end models require no lexicons and predict graphemes or words directly, which makes the decoding procedure simpler than other hybrid models. To date, the end-to-end ASR-M architectures have gained significant improvement in speech recognition accuracy (Watanabe et al., 2017; Chiu et al., 2018; Amodei et al., 2016). More complex architectures not only use audio as input for the ASR-M, but also use video input to extract characteristics related to lip contour in order to increase robustness as done in Borgstrom and Alwan (2008). These general ASR-M architectures can be implemented for online and offline systems, although there are some differences since online ASR-M can only use present and past contextual information to perform the predictions, while offline ASR-M can use the whole audio as context.

Regarding feature vectors, ever since the introduction of Mel-frequency cepstral coefficients (MFCC), they have been the state-of-the art features in ASR-M, due to their reduced dimensionality and relatively easy procedure (Davis and Mermelstein, 1980). Lately, as a consequence of the implementation of DNN's, more primitive representations can also be considered state-of-the-art features, for instance Mel-frequency spectral coefficients (MFSC) which is the logarithmic scaled Mel-spectrogram from which MFCC are extracted (Martinez and Schädler, 2016).

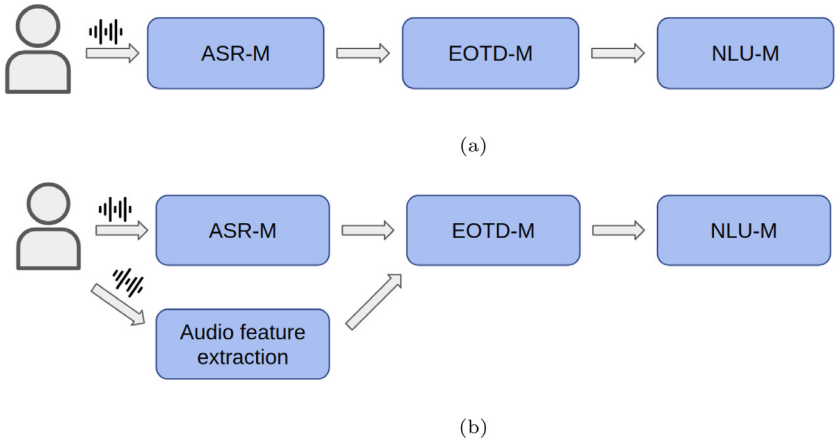


Fig. 1. Subfigure (a) shows an architecture where the EOTD-M uses the output of the ASR-M as input. Subfigure (b) shows an architecture where the EOTD-M uses the output of the ASR-M as input, but also has access to other features extracted from raw audio.

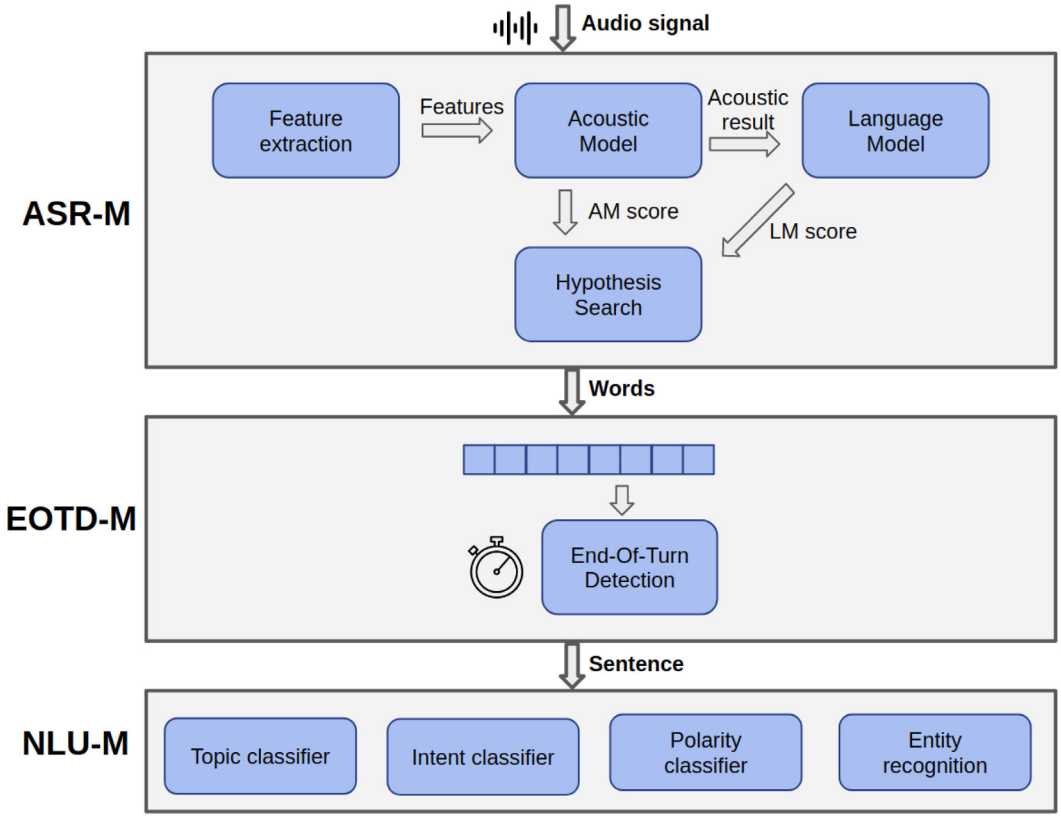


Fig. 2. ASR-M, EOTD-M, and NLU-M architectures.

2. Sources of errors in ASR-M

One of the most challenging aspects of ASR-M is the mismatch between the training and testing conditions, or real life acoustic conditions. During testing, a system may encounter new recording conditions, microphone types, speakers, accents and different sources of background noise. Furthermore, even if the test scenarios are seen during training, there can be significant variability in their statistics (Serdyuk et al., 2016). Without specific noise-robust processing, even state-of-the-art speech recognition degrades rapidly under decreasing Signal-to-Noise Ratios (Narayanan et al., 2006).

These conditions will produce the following errors in the ASR-M transcription result:

1. **Confused word (substitutions):** Due to the pronunciation, noise, or even the accent, some words can be mistranslated. This often occurs when two words are phonetically similar.
2. **Missing word (deletion):** Sometimes due to noise, accent or other speech particularities, word sounds can be confused with ambient noise or unintelligible sounds.
3. **Extra word (insertion):** Although some ambient sounds can be confused with words, the most common source of word insertion occurs when the phoneme of a word can be represented by a tuple of words, instead of the true corresponding word. For example the tuple of words*"Join in"* could replace the word *"Joining"* because they are phonetically similar.

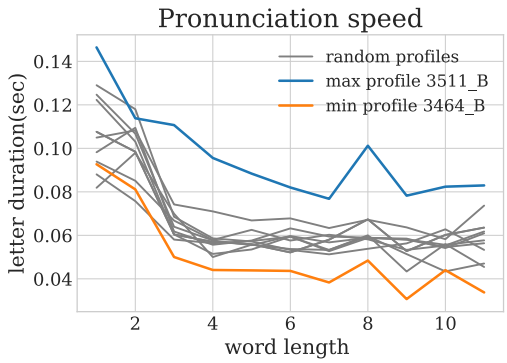


Fig. 3. Pronunciation speed profiles.

The Word Error Rate (WER) (Zechner and Waibel, 2000) defined below (Eq. (1)):

$$WER = \frac{S + D + I}{N} \tag{1}$$

where S, D, I and N are the number of substitutions, deletions, insertions and number of words in the reference respectively, is a common metric used to measure the performance of an ASR-M or machine translation system. The general difficulty of measuring performance lies in the fact that the recognized word sequence can have a different length from the reference word sequence (supposedly the correct one). WER is derived from the Levenshtein distance, working at the word level instead of the phoneme level, and it is a valuable tool for comparing different systems as well as for evaluating improvements within one system. This kind of measurement, however, provides no details on the nature of translation errors (Morris et al., 2004).

2.1. Speech profiles

The problems exposed above are related with the conversion of sound waves to phonemes, but there are other characteristics that are useful for communication and are related to the timing and duration of other language resources. These characteristics are: pronunciation speed, speaking rate, and pause duration.

Each person has their own way of speaking. And not even a combination of pronunciation speed, speaking rate, pause length or accent is fixed for a single person, it also varies depending on their mood or fatigue. Henceforth we will refer to the measurable set of these characteristics as **speech profile**. In subsequent sections, we introduce a speech profile representation and propose a way to obtain realistic values of the speech profile representation parameters from the analysis of real ASR-M outputs. For example, in Fig. 3, the average letter duration of multiple speakers is compared, calculated as $letter\_duration = word\_pronunciation\_time/word\_length$. The figure shows the average letter duration grouped by word length. The data is extracted from the Switchboard dataset (Godfrey et al., 1992), which has become the de facto standard experimental testbed for speech recognition, and will be explained in more detail in Section 4. In Fig. 3 it is possible to observe the profiles of the speakers that have the maximum and minimum average letter duration, as well as the profile of another 20 randomly chosen speakers. Fig. 3 reveals that the fastest profile is double the speed of the slowest, illustrating how wide the range of speeds can be in a group of speakers.

3. ASR simulator

As it is not possible to generate all possible types of noise that an ASR-M can receive, our goal is to introduce an ASR-SIM that can be controlled in such a way that the transcribed data exhibits different types and rates of artifacts. A characteristic feature of our simulator is that, instead of using an audio file as input, a dialogue transcription or a plain text can be used. The ASR-SIM converts any conversation transcription into an ASR-M output with the desired probabilities of ASR-M errors, and desired speech profiles.

The ASR-SIM output format is composed of two differentiated parts:

1. Word information: Contains the possible words that the ASR-SIM may estimate that correspond to the audio fragment, and their confidence value.
2. Timing information: Indicates the timestamp of the start of the pronunciation of the word, and its duration.

The transformation from plain text to word and timing information will be determined by a number of internal parameters of the simulator. These parameters can be grouped into two classes: WER probabilities and speech profile parameters.

3.1. WER probabilities

The ASR-SIM allows us to set the probability of each particular error that makes up the WER (probability of a confused word, probability of a missing word, probability of an extra word). Given a sentence, for each word the three error probabilities are evaluated to determine if the word is affected by one of the defined errors. These errors were introduced in Section 2, and the description of the methods implemented to simulate each type of error follows:

1. *Confused word:* the word is substituted by a phonetically similar word. The phonemes of the replacement and replaced words will have a Levenshtein distance smaller than a given threshold. The timing information is calculated using the information of the substituted word.
2. *Missing word:* the word is substituted by a token that represents an unknown phoneme $\langle Unk \rangle$. The timing information is calculated using the information of the substituted word.
3. *Extra word:* the phoneme of the word is randomly split into two sub-phonemes, and each one is replaced by a word with a phoneme with a Levenshtein distance smaller than a given threshold. The timing information is calculated using the original word information, proportionally sharing the word duration between the two replacement words, and with a pause $p = 0$ between replacements.

The implementation of the different error methods should mimic the errors of the ASR-M we want to simulate with the ASR-SIM. Even if an ASR-M uses a common set of features, the combination of the Acoustic model, the Language model and the Hypothesis Search make each ASR-M unique, and therefore the characteristics of the errors generated are unique as well. For example, for an ASR-M that gives more importance to the Acoustic model than to the Language model, when making a *confused word error* the true word might be replaced by a phonetically similar word, even if it causes an unlikely semantic error. In this example implementation the phonemes are obtained using the Refined Soundex algorithm, originated with the implementation of phonetic algorithms included with the Apache Commons library (Fossati and Di Eugenio, 2008). We use the Levenshtein distance (Levenshtein, 1966) to compare word phonemes, as it is commonly used to compute error rates of ASR-M. We have used the implementation in the Pyphonetics library[1] for these tools, and the English dictionary from the Nltk library[2] has been used as a source of replacement words for the *Confused* and *Extra word errors*.

[1] https://pypi.org/project/pyphonetics/.
[2] https://www.nltk.org/.

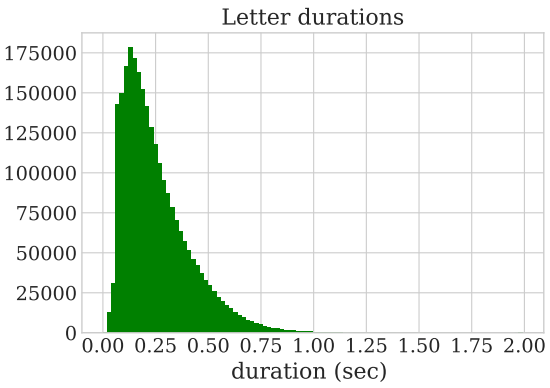


Fig. 4. Letter duration (ms).

3.2. Speech profile parametrization

The parametrized characteristics of the speech profile used by the simulator are:

1. Word duration
2. Pause duration

While in theory these parameters could be arbitrarily set, a realistic output of the ASR-SIM will require a more sensible setting of the parameters. We address this issue using a statistical analysis of real ASR-M outputs. As described in Section 2.1, a speech profile can be defined by a set of characteristics. All these characteristics are measurable, and we can therefore generate a set of variables to simulate a particular speech profile, or simulate multiple speech profiles by modifying these variables. In order to perform the experiments that will be defined in Section 4, we will parametrize word duration performing the study described in Section 3.2.1, and pause duration (Section 3.2.2) based on (Campione and Véronis, 2002). These parameters will directly affect the codification of the sentences, since some codification methods use pauses between words as input information, and pause duration will affect the amount of evaluation points used in the experiments, as detailed in Section 4.1.

3.2.1. Word duration

In order to simulate word duration, we will use the values obtained by calculating the average and standard deviation of the letter duration of the speakers from the Switchboard dataset (Fig. 3). The distribution of the values obtained for the letter duration calculus is illustrated in Fig. 4. The ASR-SIM will use this empirical distribution to estimate the duration of each letter in a word, randomly sampling from the distribution. Although more sophisticated methods could have been used to estimate the pronunciation time of a particular word, this method maintains simplicity, and it still makes words last a proportional, but not fixed, amount of time for their length.

3.2.2. Duration of pauses

Campione and Véronis (2002) present a large-scale study of silent pause duration, based on the analysis of read and spontaneous speech. Although in their study, spontaneous speech analysis is only performed in French, it does not represent an obstacle for our analysis since it has been observed that the language differences with respect to gap duration seem to be minor (Weilhammer and Rabold, 2003). Campione and Véronis (2002) made the hypothesis that the observed pause duration distributions are the result of a combination of three categories of pauses. By using *Generalized Reduced Gradient* (GRG2), they obtained a parametrized probabilistic model of the duration of pauses, which is described by Eq. (2):

$$D(x) = k_1 N(\mu_1, \sigma_1, x) + k_2 N(\mu_2, \sigma_2, x) + k_3 N(\mu_3, \sigma_3, x) \quad (2)$$

Table 1
Pause distributions parameters.

	i	μ_i	σ_i
Between words pause	1	78	1.3
Comma pause	2	426	1.6
Dot pause	3	1585	1.3

where $D(x)$ is the distribution of the duration of pauses, $N(\mu_i, \sigma_i, x)$ is the normal law of mean μ_i, and their standard deviation is σ_i (duration of pauses are log-transformed). The parameters k_1, k_2 and k_3 represent the weight of each component distribution ($k_1 + k_2 + k_3 = 1$).

Based on this work, we match each of these pause duration distributions in increasing order of μ_i, with the pause between words, comma pause and dot pause respectively. The μ and σ values used for each distribution are shown in Table 1. Although μ values are available in the original study, σ values were deduced from the figures in Campione and Véronis (2002).

The ASR-SIM will generate pauses according to these distributions when using plain text inputs. However, if the original pause information is provided, this information will be used.

Algorithm 1: Pseudocode of the ASR-SIM main function

```
asr_output = [];
for token in source_text do
    if is_word(token) then
        is_confused = random() > confused_word_threshold
        is_missing = random() > missing_word_threshold
        is_extra = random() > extra_word_threshold
        possible_errors = [ ]
        if is_confused then possible_errors.append("confused")
        if is_missing then possible_errors.append("missing")
        if is_extra then possible_errors.append("extra")
        error = random_selection(possible_errors)
        if error == "confused" then
            confused_word = get_close_match(token)
            asr_output.append(confused_word)
        else if error == "missing" then
            asr_output.append(<unk>)
        else if error == "extra" then
            word1, word2 = generate_extra_word(token)
            asr_output.append(word1)
            asr_output.append(pause_between_words)
            asr_output.append(word2)
        else
            asr_output.append(token)
    else
        if token == "," then
            asr_output.append(comma_pause)
        else if token == "." then
            asr_output.append(point_pause)
        else if token == " " then
            asr_output.append(pause_between_words)
return asr_output;
```

Algorithm 2: Pseudocode of the generate_extra_word(word) function

```
w1_Length = random(1, length(word))
w2_Length = length(word) - w1_Length
w1_segment = word[:w1_Length]
w2_segment = word[w1_Length:]
firstWord = get_close_match(w1_segment)
secondWord = get_close_match(w2_segment)
return firstWord, secondWord
```

3.3. ASR simulator pseudocode

The general procedure to transform a text to the desired ASR-SIM output is the one described by Algorithm 1. The output of the ASR-SIM contains the recognized words, and the associated timing information. Any character that is not a letter is removed from the input text, except for commas and periods which represent pauses in the speech.

In Algorithm 1, the source text is analyzed token by token (line 2). Whenever the token corresponds to a word (line 3), for each error type a random number is generated and compared to the associated threshold value (lines 8–10), the error to apply will be randomly selected between those that exceed the corresponding threshold (line 11), and the error mechanism is executed (lines 12–21). On the other hand, if the token is not a word, it will generate a pause based on the type of token (lines 25–30).

In lines 17–21, the extra word error is generated using an auxiliary function called generate_extra_word, which is in charge of generating two words from one, as explained in Section 2 and described in Algorithm 2. The generate_extra_word function randomly splits the word (lines 1–4) and finds a similar word for each segment with the function get_close_match (lines 5–6). This function is also used in line 10 to generate the confused word error by calculating the Levenshtein distance between the phonemes as explained in Section 3.

The source code is available under GNU General Public License v3.0.[3] in it is source code repository[4]

4. Experiments

In this section we present the experimental framework we have designed to evaluate the influence of the different types of errors in the behavior of the EOTD-M. We conduct this evaluation using the ASR-SIM presented. The parameters of the simulator are changed to produce different combinations of errors. The performance of the EOTD-M prediction task is then tested on this simulated data. The section is organized as follows: Firstly, we describe the EOTD-M prediction task and the features used as input for the classifier. Then we describe the characteristics of the classifier, and the metrics used to evaluate the results. The two sections that follow describe the corpora used and how errors are generated by the ASR-SIM. Finally, we present and discuss the results of the experiments.

4.1. EOTD-M Classification and sets of features

To evaluate the sensitivity of the classifier, we rely on the Prediction at Pauses task described by Skantze (2017). This is a standard turn-taking decision task that takes place at brief pauses during an interaction. The goal is to predict whether the user holding the turn is going to continue speaking (HOLD), or swap turns (SHIFT).

As mentioned in Section 1.1, there are multiple features that can be extracted from raw audio data, but since we use the ASR-SIM, our features will be those that can be extracted from transcriptions. Therefore, in this experiment, we compare the sensitivity of the classifier to the use of the following sets of feature vectors:

- Word Embeddings: multi-dimensional meaning representations of a word. For these experiments, we use the GloVe (Global Vectors for Word Representation (Pennington et al., 2014)) pretrained embeddings[5]. This is a popular vector representation for Natural Language Processing tasks.
- POS Tags: part-of-speech tag for each word is considered to be a good predictor of turn-switches in the literature (Gravano and Hirschberg, 2011). To obtain the tags, we use the tagger from the Nltk library, and generate a one-hot representation.
- Pauses: duration of time gaps between every pair of consecutive words in the sentence.
- Combined: a combination of Word Embeddings, POS Tags and Pauses.

[3] https://choosealicense.com/licenses/gpl-3.0/.
[4] https://github.com/CesarMontenegro/AsrSimulator.
[5] Available online at https://nlp.stanford.edu/projects/glove/.

Table 2
Architecture of the models used, *Layer(type)* is the name of the units used in each particular layer and *Units* is the number of units in the layer.

Layer (type)	Units
LSTM	128
Dense(relu)	128
Dense(sigmoid)	1

Table 3
Parameter of the training procedure for the LSTM based model, *n_batch* samples per gradient update, *epochs* is the number of epochs to train the model, *learning rate* controls how much to modify the model's parameters in response to the estimated error after each epoch, *pad_sequence* is the maximum length in number of words of a sentence (if the sentence is larger than this value, the first words in the sentence are discarded until it reaches the specified length), *earlyStopping monitor* is the variable monitored that will trigger the early stopping of the training procedure, *earlyStopping patience* is the number of epochs with no improvement after which training will be stopped, *loss function* is the optimization score function and *optimizer* is the name of the algorithm used to fit the parameters.

Parameter	Value #
n_batch	1000
epochs	200
learning rate	0.01
pad_sequence length	30
earlyStopping monitor	val_loss
earlyStopping patience	5
loss function	binary_crossentropy
optimizer	adam

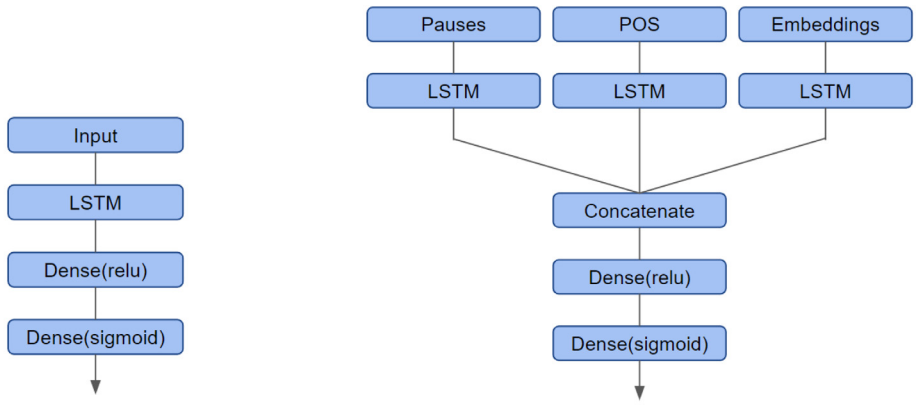
4.2. Characteristics of the classifier

In the literature, the most frequently used models for EOTD-M are models based on LSTM Recurrent Networks (Maier et al., 2017; Aldeneh et al., 2018; Roddy et al., 2018; Liu et al., 2017; Shannon et al., 2017; Masumura et al., 2017; Hara et al., 2019). Despite being combined with other layers or algorithms (*e.g.*, Shannon et al. (2017) add Convolutional Layers to the architecture), the main differences between them are the features they use to train the algorithm. Therefore, for our experiments, we will use the LSTM architectures illustrated in Fig. 5, with the parameters described in Table 2.

For model validation, each scenario generated in these experiments will divide the dataset into three subsets. These three divisions will be called *Train*, *Validation* and *Test*. The algorithm will learn from the Train subset, while Validation is used to avoid overfitting by stopping the training process when the validation loss stops improving. The parametrization of this anti-overfitting mechanism is described by the *earlyStopping* variables in Table 3. The *patience* parameter allows the anti-overfitting mechanism to prevent the training procedure from stopping when it is temporally stuck in a local minima. Nevertheless, for these experiments we have observed that even low values of patience are enough to avoid local minima and overfitting.

4.3. Metrics

The LSTM network will output the probability of a sentence being a SHIFT pause. Using a threshold value, this output can be binarized, thus allowing to calculate the accuracy and other metrics. However, the determination of this threshold can severely affect the result. To avoid this drawback, in these experiments the Area Under the ROC Curve (AUC) will be used to evaluate the performance of the LSTM models.



(a) Single-feature input architecture (b) Multiple-feature input architecture

Fig. 5. Subfigure (a) shows the architecture of the models that use one of the three features vectors as input. Subfigure (b) shows the architecture of the models that use the three types of feature vectors as input.

4.4. Dialogue data corpora

The experiments will be performed with two datasets. The first dataset will be based on the Switchboard dataset, for which we will not generate timing information (word duration and pause duration) since it already has that information available, and we will only induce the correspondent artifacts. This dataset is a telephone-speech corpus that consists of approximately 260 h of speech and was originally collected by Texas Instruments in 1990–1991, under DARPA sponsorship.[6] It is a collection of about 2,400 two-sided telephone conversations among 543 speakers (241 female, 302 male) from all areas of the United States. In these types of conversations, where there is a lack of non-verbal communication, backchannel communication is very present. For End-Of-Turn detection, we are not interested in backchannel turns, we focus on the speech of the speaker who leads the turn, and is making a statement. The backchannel communication made by the listener on a turn is ignored, resulting in a dataset of 35,323 sentences.

The second dataset will be based on the OpenSubtitles en-es corpus,[7] for which the ASR-SIM will have to estimate all the timing information based on the parameters we have defined. The OpenSubtitles en-es corpus contains 61.4 million speech turns from movie scripts. These speech turns do not belong to real speech, the dialogues are scripted, and therefore the structure and vocabulary vary from natural human-to-human speech. Nevertheless, this dataset provides a complementary validation benchmark for our study, since each communication scenario presents a particular problem, such as telephone conversations, face-to-face conversations or videoconferences. We will train and test the EOTD-M on these types of dialogues without trying to export the models from the scripted-dialogue environment to human-to-human speech. For the experiments, we will use 50.000 randomly selected sentences from this dataset.

According to the EOTD-M problem described in Section 4.1, each speech turn generates a SHIFT-labeled instance, while HOLD instances are generated from turns containing pauses longer than the specified threshold (δ = 1045 ms). This is proposed in Campione and Véronis (2002) as a cut between the distributions that we have associated with comma and dot pauses. A HOLD instance is the sub-sequence of tokens that precedes each pause greater than the threshold in a turn. Illustrative examples of the generation of SHIFT and HOLD instances can be found in Tables 4 and 5, where, from a hypothetical transcribed sentence, we analyze the pause duration to generate HOLD and SHIFT instances. The example in Table 4 uses a threshold value of δ = 1045 ms, and the one in Table 5 uses δ = 1500 ms.

[6] https://catalog.ldc.upenn.edu/LDC97S62.

[7] [dataset] http://opus.nlpl.eu/OpenSubtitles.php.

Table 4
Example of the generation of SHIFT and HOLD instances with δ = 1045 ms.

Hypothetical sentence	
Hello, I would like to buy a necklace, a gold necklace.	
Pause duration	
Hello<1200 ms>I would like to buy a necklace<1600 ms>a gold necklace	
Instances and labels generated from the sentence (δ = 1045 ms):	
Hello I would like to buy a necklace a gold necklace	SHIFT
Hello I would like to buy a necklace	HOLD
Hello	HOLD

Table 5
Example of the generation of SHIFT and HOLD instances with δ = 1500 ms.

Hypothetical sentence	
Hello, I would like to buy a necklace, a gold necklace.	
Pause duration	
Hello<1200 ms>I would like to buy a necklace<1600 ms>a gold necklace	
Instances and labels generated from the sentence (δ = 1500 ms):	
Hello I would like to buy a necklace a gold necklace	SHIFT
Hello I would like to buy a necklace	HOLD

Finally, the datasets are balanced to contain the same amount of SHIFT and HOLD instances. This is done by randomly sampling from each class.

4.5. Word error probabilities

The probabilities of the ASR-SIM errors used to analyze the impact on the quality of the LSTM estimator are: {0.0, 0.1, 0.3, 0.5, 0.7} for the three types of errors. In order to identify which factors influence each feature representation technique, each error probability is analyzed independently. The threshold for the Levenshtein distance of *Confused* and *Extra word errors* is set to $\tau = 3$. To evaluate the impact of the different error types and probabilities, the experiments will be conducted following two strategies: *same_distribution* and *different_distribution*. The *same_distribution* strategy will apply the same error probability in train, validation and test sets, given a particular error type and probability. The *different_distribution* strategy will apply the error to the test set only, while the algorithm will train with free-from-error data. This second strategy simulates the scenario in which the training data is generated in a controlled environment, with a low probability of errors. However, the evaluation data is generated in a real environment, exposed to the errors defined in Section 2. Therefore, the *different_distribution* strategy

will allow us to investigate how important training with the expected test error rates is.

5. Results

Before analyzing how the ASR-SIM transcription errors affect the performance of the LSTM based EOTD-M, we have measured how each error type affects each featurization technique. Inspired on the analysis performed by Voleti et al. (2019), where they investigate the effects of word substitution errors (*Confused word error*) on sentence embeddings, we have measured how much featurized sentences change under the effect of the different errors generated in the ASR-SIM.

5.1. Effects of the ASR-SIM errors on the featurization techniques

The errors considered in this paper can cause variations in the length of a sentence, unlike in Voleti et al. (2019), where only the confusion error is analyzed, which does not change the number of words in a sentence. *Extra word error* adds words to the sentences, and this makes the approach of Voleti et al. (2019) unsuitable for this paper, since it compares the original and modified sentences word to word.

Therefore, to overcome this difference, we treat the sentences as time series of feature vectors, and use the Dynamic Time Warping (DTW) measure (Liu et al., 2007) to compare sentences without errors and sentences with the induced errors. DTW is a measure that finds the optimal alignment between two time series by stretching or shrinking one of the time series along its time axis (Salvador and Chan, 2007). This warping between two time series can then be used to determine the similarity between the two time series by means of a defined distance measure. DTW is often used in speech recognition to determine if two waveforms represent the same spoken phrase (Abdulla et al., 2003).

DTW has been previously used to compare similarity between sentences (Liu et al., 2007), and although it does not guarantee the triangle inequality, it provides an estimation of how the errors affect the vectorized representation of the sentences. For this evaluation we have used the Python *FastDTW* library[8] based on the work of Salvador and Chan (2007), which is an approximate DTW algorithm that provides optimal or near-optimal alignments with an $O(N)$ time and memory complexity.

We have randomly selected 1000 sentences to calculate the average distance to their modified version for each dataset, the distance is calculated by the FastDTW algorithm with the Euclidean distance between each pair of matched words. The distances have been calculated under the effect of the different error probabilities described in Section 4.5. This comparison exercise has been performed 10 times to take into consideration the variability generated in the ASR-SIM, and averaged to illustrate the results in Figs. 6 and 7. These figures are composed of three plots each, one for each featurization technique. Each plot contains information on how a particular featurization technique is affected by the three error types with different error probabilities. The *Y axis* represents the average distance between sentences, and the *X axis* represents the error probability of the modified version of the sentences.

The first plot from left to right in Figs. 6 and 7 shows that all the errors affect similarly to the Embedding featurization. Nevertheless, the error that affects the most is the *confused word error* followed by *extra word error* and *missing word error*. The second plot in Figs. 6 and 7 shows how POS featurization is similarly affected by *extra word error* and *missing word error*, and slightly less affected by *confused word error*. This result responds to the expectations since the distance between every pair of one-hotted POS tags is the same, and occasionally the confused word can have the same POS tag as the original word.

Finally, Pause featurization seems to be unaffected by *confused word error* and *extra word error*, but it is affected by *missing word error*. This is also expected since *missing word error* and *confused word error* do not alter the pause between the duration of words, while *extra word error* creates a pause between the two words added.

[8] https://pypi.org/project/fastdtw/.

5.2. Effects of the ASR-SIM errors on EOTD-M

We compute the predictions made by the LSTM based classifier given different errors in the ASR-SIM transcription, different sentence featurization techniques and different scenarios. The results are shown in Figs. 8, 9, 10 and 11.

For each dataset, there are two figures displaying the results for the *same_distribution* and *different_distribution* strategies. Each of the Figs. 8–11 is composed of three plots, one for each ASR-SIM error type defined in Section 2. Each plot shows the average AUC score obtained from a 10-fold experiment for each featurization technique and the combination of the three techniques, taking into account the variability obtained from randomly generating dataset splits and error generation.

A first analysis of Figs. 8–11 reveals that the Pause feature representation obtains the worst AUC results not only for every error probability on the *same_distribution* experiments, but also for low error probabilities on the *different_distribution* experiments. This poor performance can be explained by the fact that pause duration information is very limited and does not capture semantic aspects of EOTD-M. This seems to be confirmed by the observation that the Combined features produce the highest AUC values, being the most complex representation used in this work.

Therefore, we focus our analysis on the Combined, Embeddings and POS features since, as previously discussed, Pauses features do not produce accurate classifications. Analyzing the effect of the shift of error distributions between train and test sets (*same_distribution* vs *different_distribution*), Figs. 8–11 show that the effect of the change of distributions is remarkable under the effect of the *confused word error* and *missing word error*, and, to a lesser extent, for the *extra word error*. This effect is similar in both datasets, and more remarkable in the Combined and Embeddings experiments, which show fast degradation as the error probabilities grow. The payoff of having the most complex features is that it is the most sensitive to errors, deteriorating to the point of performing worse than other simpler features. This can be appreciated in *Confused* and *Missing word error* of Switchboard results in Fig. 9, and on the Subtitles results shown in Fig. 11.

On the other hand, POS features, despite achieving a worse performance than the Embedding features, are less affected by low error probabilities. Nevertheless, *missing word error*s have a stronger impact than the other errors, as can be seen in Figs. 9 and 11. We may hypothesize that a confused word can still keep the same POS tag, and does not alter the vectorized representation of the sentence as much as a *missing word error*. In the same way, extra word errors generate two words, of which at least one could also have the same POS tag as the original word, despite having an extra tag from the other word. This hypothesis is reinforced by the results illustrated in Figs. 6 and 7, and analyzed in Section 5.1, where *confused word error* is the least severe error in terms of altering the POS featurization of a sentence.

The impact of these induced errors has been also measured in terms of training time. In Figs. 12–13 the training time under each error probability is illustrated. In each figure, given the featurization technique, we can compare the training time for each error type. The X axis in this type of plots (Swarm plots and violin plots) does not correspond to a continuous variable, it acts as an auxiliary variable that helps to plot multiple instances that have the same Y value (training time in this case) without overlapping those instances. These figures show how, for both datasets, neither the error type nor the probability affect the training time significantly. Although the lack of influence of the errors on the training time could be caused by the 200 epoch limit by making all the training processes stop at the same epoch, this is not the case since none of the training processes reached the epoch limit.

Summarizing all the information extracted, we can conclude that *missing word error* is the most potentially harmful error an ASR-M can deliver to the EOTD-M if the classifier is not trained with the expected error probabilities. Not only it modifies the vectorization of a sentence severely, but it is also the error that affects the performance

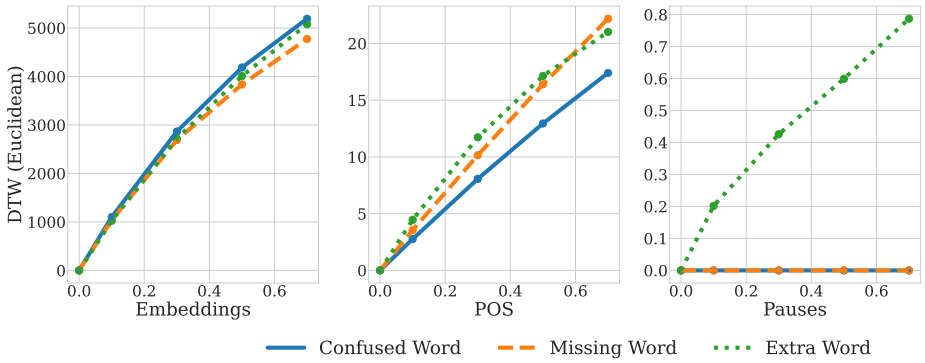


Fig. 6. DTW distances between 1000 Switchboard sentences and their modified versions generated by the ASR-SIM.

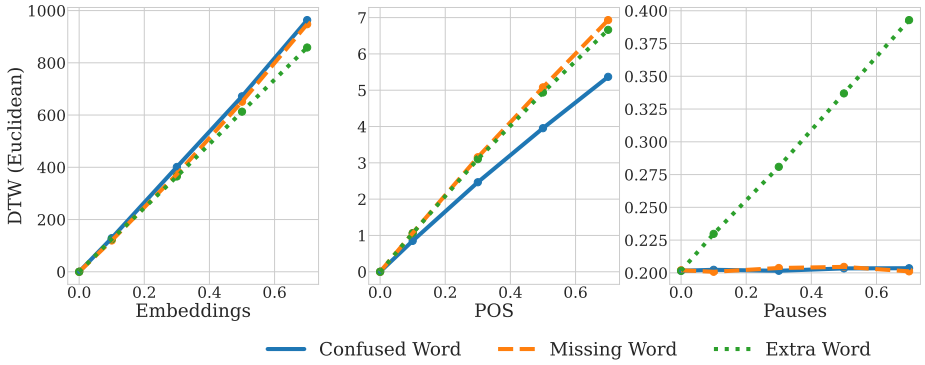


Fig. 7. DTW distances between 1000 subtitle sentences and their modified versions generated by the ASR-SIM.

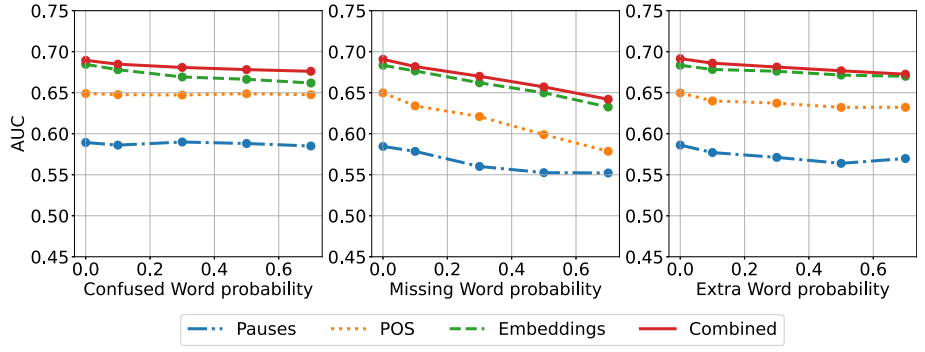


Fig. 8. Results for the Switchboard dataset with equal train–test distribution.

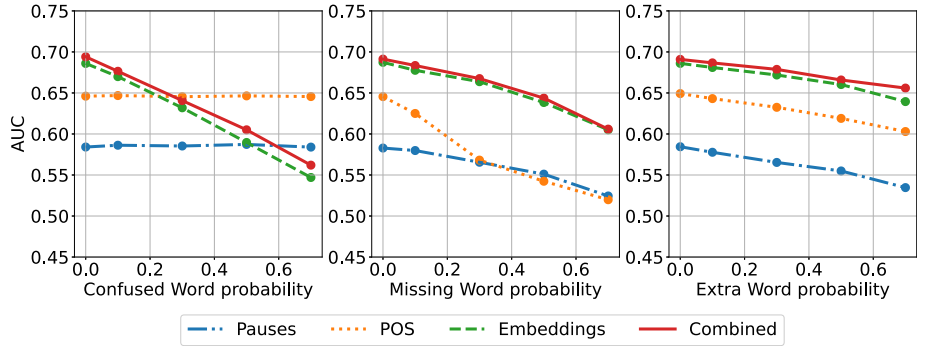


Fig. 9. Results for the Switchboard dataset with different train–test distribution.

of an LSTM based EOTD-M the most. Another important finding is that representations that are less efficient for the EOTD-M under low error probabilities can become more efficient for particular types of errors when the error rate is increased. This is the case of the POS representation, which can outperform the embedding representation for high *confused word and extra word error* probabilities in the different

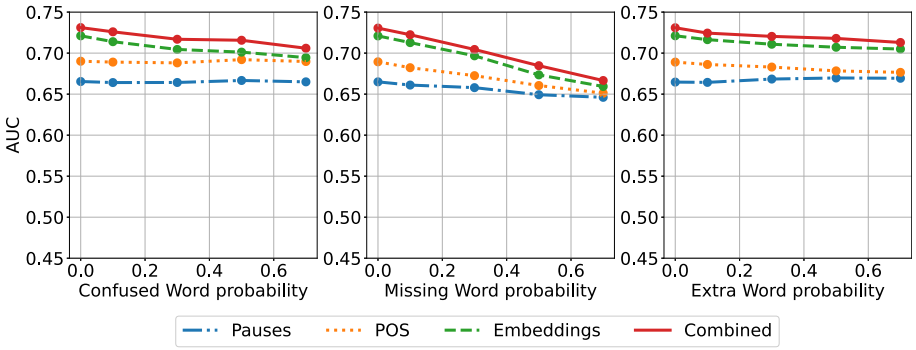


Fig. 10. Results for the Subtitles dataset with equal train–test distribution.

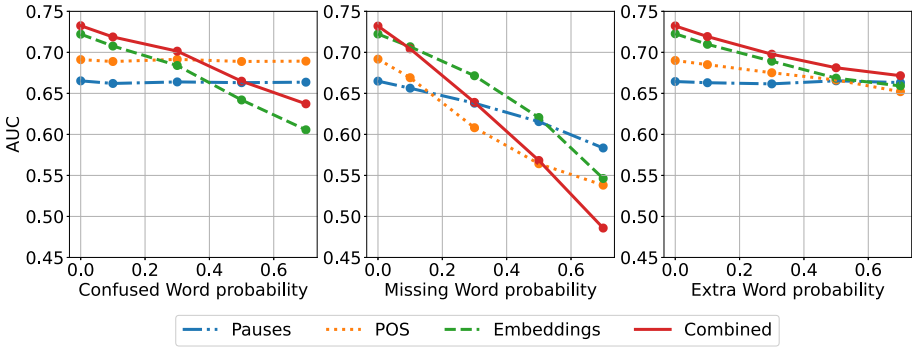


Fig. 11. Results for the Subtitles dataset with different train–test distribution.

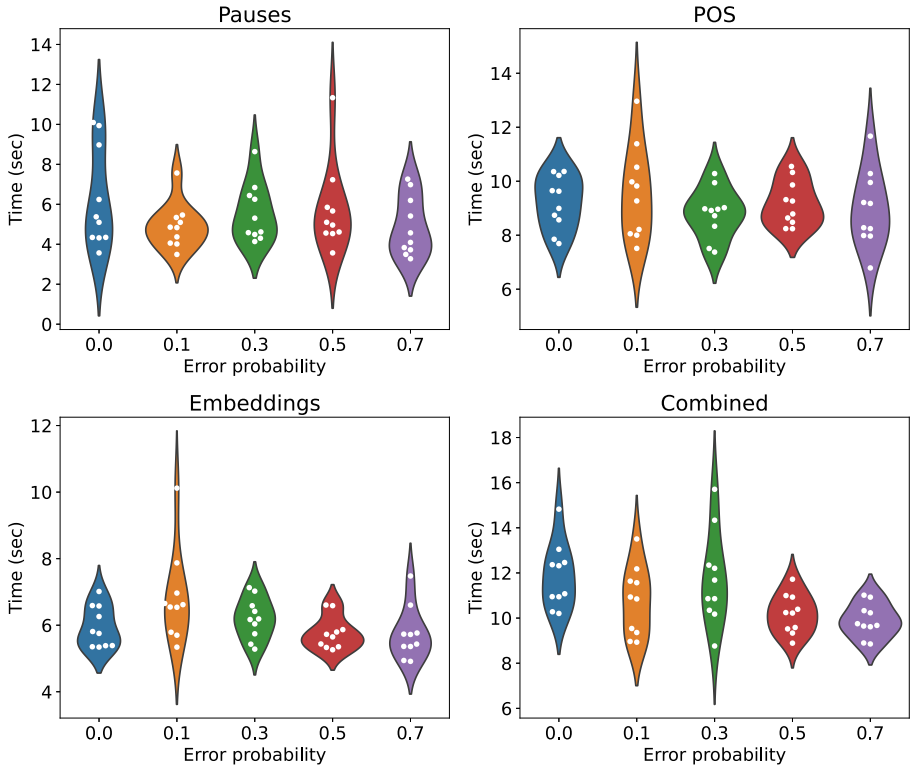


Fig. 12. Training time for the Switchboard dataset.

train–test distribution scenario. Nevertheless, for this to happen in some embedding and error type configurations, the error probability must reach values of 0.5 or above, such is the case of Fig. 9 for *confused word error*. In other studies such as (Voleti et al., 2019) and Simonnet et al. (2018), where the effects of *confused word error* on embeddings and NLU-M respectively are studied, the maximum error probability simulated is 0.5, and real transcription errors, the percentage error was 23%. Nevertheless, in this study we have covered a wider range of

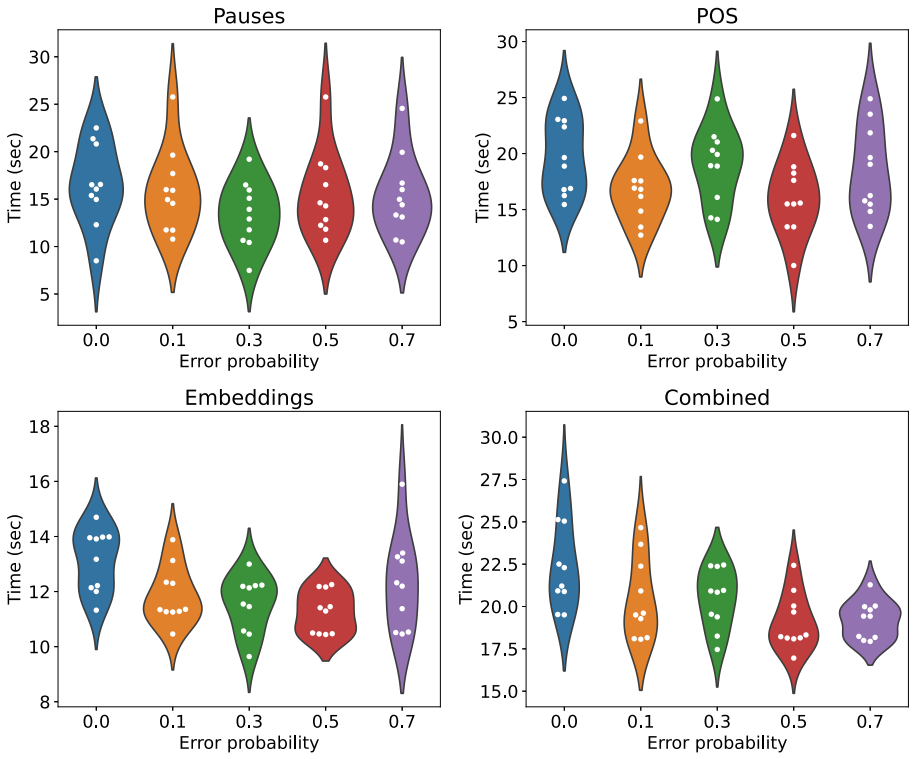


Fig. 13. Training time for the Subtitles dataset.

error probabilities, since the amount of errors depend not only on the ASR-M itself but also the audio conditions, and that is sometimes an uncontrollable factor.

The results obtained from the experiments are similar using both datasets, and the behavior of the classifiers under the influence of the generated errors is coherent. This indicates that the ASR-SIM is suitable for the purpose of simulating ASR-M transcriptions and simulating errors.

6. Conclusions and future work

In this paper we have proposed a method for investigating the influence of the ASR-M output errors on the behavior of the EOTD-M of Spoken Dialogue Systems, which has not been addressed before. The ASR-SIM introduced in this paper generates the transcription of a simulated dialogue starting from plain text, with the amount and type of noise specified by the user. This leads to a realistic simulation of a variety of problems exhibited by ASR-M components. The code of this simulator will remain available in GitHub[9] for future studies as a contribution of this work. The absence of comparable simulators in the literature, is one of the motivations of this work.

Some of the insights from our analysis are the following:

1. The ASR-SIM is suitable for the purpose of simulating ASR-M transcriptions and simulating errors.
2. Word embeddings produce the best overall results for the EOTD-M task. This is consistent with previous reported results.
3. The most influential error across representations is the *missing word error*.
4. In terms of classifier performance, there is an interaction between types of errors and featurization techniques.
5. It is more effective to include ASR-M simulated errors in the train and validation sets in order to make the classifiers more robust.

So far, we have only exploited the capability of the ASR-SIM to vary the three types of noise exposed. However, further research can be done by combining noise with the different speech profiles described in Section 2.1. Moreover, in this early version of the ASR-SIM, errors are generated randomly among the words of a sentence, nevertheless there are probably certain characteristics in some words that make them more prone to errors compare to others. A study on what word characteristics are more influential on the probability of a word to be miss transcribed would help to create more realistic scenarios by the ASR-SIM. Also, in order to increase the amount of algorithms that can benefit from this simulator, the pause and word duration simulations can be extended with other simulations, such as tone and other variables extracted from audio. Doing this, solutions based in the architecture represented in Fig. 1(b) will be able to benefit from the advantages that the ASR-SIM offers.

CRediT authorship contribution statement

César Montenegro: Conceptualization, Software development, Coding, Investigation. **Roberto Santana:** Conceptualization, Writing, Software development, Software reviewing. **Jose A. Lozano:** Conceptualization, Supervision, Investigation, Writing - review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The research presented in this paper has been conducted as part of the project EMPATHIC that has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 769872.

[9] https://github.com/CesarMontenegro/AsrSimulator.

Jose A. Lozano is partially supported by the Basque Government through the BERC 2018–2021 program, IT1244-19 and grant "Artificial Intelligence in BCAM number EXP. 2019/00432" and by the Spanish Ministry of Science, Innovation and Universities: BCAM Severo Ochoa accreditation SEV-2017-0718, TIN2016-78365-R and PID2019-104966GB-I00. And R. Santana acknowledge support by the Spanish Ministry of Science, Innovation and Universities (Project TIN2016-78365-R and PID2019-104966GB-I00), and the Basque Government (IT1244-19 and ELKARTEK Programs).

References

Abdulla, W.H., Chow, D., Sin, G., 2003. Cross-words reference template for DTW-based speech recognition systems. In: TENCON Conference on Convergent Technologies for Asia-Pacific Region, vol. 4. IEEE, pp. 1576–1579.

Aldeneh, Z., Dimitriadis, D., Provost, E.M., 2018. Improving end-of-turn detection in spoken dialogues by detecting speaker intentions as a secondary task. In: International Conference on Acoustics, Speech and Signal Processing. IEEE, pp. 6159–6163.

Amodei, D., Ananthanarayanan, S., Anubhai, R., Bai, J., Battenberg, E., Case, C., Casper, J., Catanzaro, B., Cheng, Q., Chen, G., et al., 2016. Deep speech 2: End-to-end speech recognition in English and Mandarin. In: International Conference on Machine Learning, pp. 173–182.

Amrouche, A., Debyeche, M., Taleb-Ahmed, A., Rouvaen, J.M., Yagoub, M.C., 2010. An efficient speech recognition system in adverse conditions using the nonparametric regression. Eng. Appl. Artif. Intell. 23 (1), 85–94.

Borgstrom, B.J., Alwan, A., 2008. A low-complexity parabolic lip contour model with speaker normalization for high-level feature extraction in noise-robust audiovisual speech recognition. IEEE Trans. Syst. Man Cybern. A 38 (6), 1273–1280.

Campione, E., Véronis, J., 2002. A large-scale multilingual study of silent pause duration. In: Speech Prosody, International Speech and Communication Association, pp. 199–202.

Chang, S.-Y., Li, B., Sainath, T.N., Simko, G., Parada, C., 2017. Endpoint detection using grid long short-term memory networks for streaming speech recognition. In: Proceedings of Interspeech, pp. 3812–3816.

Chiu, C.-C., Sainath, T.N., Wu, Y., Prabhavalkar, R., Nguyen, P., Chen, Z., Kannan, A., Weiss, R.J., Rao, K., Gonina, E., et al., 2018. State-of-the-art speech recognition with sequence-to-sequence models. In: International Conference on Acoustics, Speech and Signal Processing. IEEE, pp. 4774–4778.

Davis, S., Mermelstein, P., 1980. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. IEEE Trans. Acoust. Speech Signal Process. 28 (4), 357–366.

Fernández-Díaz, M., Gallardo-Antolín, A., 2020. An attention long short-term memory based system for automatic classification of speech intelligibility. Eng. Appl. Artif. Intell. 96, 103976.

Fossati, D., Di Eugenio, B., 2008. I saw TREE trees in the park: How to correct real-word spelling mistakes. In: LREC, pp. 896–901.

Godfrey, J.J., Holliman, E.C., McDaniel, J., 1992. Switchboard: Telephone speech corpus for research and development. In: International Conference on Acoustics, Speech and Signal Processing, vol. 1. IEEE, pp. 517–520.

Gravano, A., Hirschberg, J., 2011. Turn-taking cues in task-oriented dialogue. Comput. Speech Lang. 25 (3), 601–634.

Graves, A., Mohamed, A.-r., Hinton, G., 2013. Speech recognition with deep recurrent neural networks. In: International Conference on Acoustics, Speech and Signal Processing. IEEE, pp. 6645–6649.

Hannun, A., Case, C., Casper, J., Catanzaro, B., Diamos, G., Elsen, E., Prenger, R., Satheesh, S., Sengupta, S., Coates, A., et al., 2014. Deep speech: Scaling up end-to-end speech recognition. arXiv preprint arXiv:1412.5567.

Hara, K., Inoue, K., Takanashi, K., Kawahara, T., 2019. Turn-taking Prediction Based on Detection of Transition Relevance Place. In: Proceedings of Interspeech, pp. 4170–4174.

Levenshtein, V.I., 1966. Binary codes capable of correcting deletions, insertions, and reversals. Sov. Phys. Dokl. 10 (8), 707–710.

Liu, C., Ishi, C., Ishiguro, H., 2017. Turn-taking estimation model based on joint embedding of lexical and prosodic contents. In: Proceedings of Interspeech. pp. 1686–1690.

Liu, X., Zhou, Y., Zheng, R., 2007. Sentence similarity based on dynamic time warping. In: International Conference on Semantic Computing. IEEE, pp. 250–256.

Maier, A., Hough, J., Schlangen, D., 2017. Towards deep end-of-turn prediction for situated spoken dialogue systems. In: Proceedings of Interspeech, pp. 1676–1680.

Martinez, A.M.C., Schädler, M.R., 2016. Why do ASR systems despite neural nets still depend on robust features. In: Proceedings of Interspeech, pp. 1883–1887.

Masumura, R., Asami, T., Masataki, H., Ishii, R., Higashinaka, R., 2017. Online end-of-turn detection from speech based on stacked time-asynchronous sequential networks. In: Proceedings of Interspeech, pp. 1661–1665.

Masumura, R., Tanaka, T., Ando, A., Ishii, R., Higashinaka, R., Aono, Y., 2018. Neural dialogue context online end-of-turn detection. In: Proceedings of the 19th Annual SIGDial Meeting on Discourse and Dialogue, pp. 224–228.

Morris, A.C., Maier, V., Green, P., 2004. From WER and RIL to MER and WIL: improved evaluation measures for connected speech recognition. In: Eighth International Conference on Spoken Language Processing, pp. 2765–2768.

Narayanan, S., Georgiou, P.G., Sethy, A., Wang, D., Bulut, M., Sundaram, S., Ettelaie, E., Ananthakrishnan, S., Franco, H., Precoda, K., et al., 2006. Speech recognition engineering issues in speech to speech translation system design for low resource languages and domains. In: International Conference on Acoustics, Speech and Signal Processing, vol. 5. IEEE, pp. 1209–1212.

Pennington, J., Socher, R., Manning, C.D., 2014. GloVe: Global vectors for word representation. In: Empirical Methods in Natural Language Processing (EMNLP). pp. 1532–1543, URL: http://www.aclweb.org/anthology/D14-1162.

Razavi, S.Z., Kane, B., Schubert, L.K., 2019. Investigating linguistic and semantic features for turn-taking prediction in open-domain human-computer conversation. In: Proceedings of Interspeech, pp. 4140–4144.

Roddy, M., Skantze, G., Harte, N., 2018. Investigating speech features for continuous turn-taking prediction using LSTMs. arXiv preprint arXiv:1806.11461.

Salem, Z.N.B., Boougrain, L., Alexandre, F., 2007. Spatio-temporal biologically inspired models for clean and noisy speech recognition. Neurocomputing 71 (1–3), 131–136.

Salvador, S., Chan, P., 2007. Toward accurate dynamic time warping in linear time and space. Intell. Data Anal. 11 (5), 561–580.

Serdyuk, D., Audhkhasi, K., Brakel, P., Ramabhadran, B., Thomas, S., Bengio, Y., 2016. Invariant representations for noisy speech recognition. arXiv preprint arXiv:1612.01928.

Shahamiri, S.R., Salim, S.S.B., 2014. Real-time frequency-based noise-robust automatic speech recognition using multi-nets artificial neural networks: A multi-views multi-learners approach. Neurocomputing 129, 199–207.

Shannon, M., Simko, G., Yiin Chang, S., Parada, C., 2017. Improved end-of-query detection for streaming speech recognition. In: Proceedings of Interspeech, pp. 1909–1913.

Shao, Y., Chang, C.-H., 2011. Bayesian Separation with sparsity promotion in perceptual wavelet domain for speech enhancement and hybrid speech recognition. IEEE Trans. Syst. Man Cybern. A 41 (2), 284–293.

Simonnet, E., Ghannay, S., Camelin, N., Estève, Y., 2018. Simulating ASR errors for training SLU systems. In: LREC, pp. 3157–3162.

Skantze, G., 2017. Towards a general, continuous model of turn-taking in spoken dialogue using LSTM recurrent neural networks. In: Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue, pp. 220–230.

Squartini, S., Principi, E., Rotili, R., Piazza, F., 2012. Environmental robust speech and speaker recognition through multi-channel histogram equalization. Neurocomputing 78 (1), 111–120.

Trentin, E., Matassoni, M., 2003. Noise-tolerant speech recognition: the SNN-TA approach. Inform. Sci. 156 (1–2), 55–69.

Voleti, R., Liss, J.M., Berisha, V., 2019. Investigating the effects of word substitution errors on sentence embeddings. In: International Conference on Acoustics, Speech and Signal Processing. IEEE, pp. 7315–7319.

Watanabe, S., Hori, T., Kim, S., Hershey, J.R., Hayashi, T., 2017. Hybrid CTC/attention architecture for end-to-end speech recognition. IEEE J. Sel. Top. Signal Process. 11 (8), 1240–1253.

Weilhammer, K., Rabold, S., 2003. Durational aspects in turn taking. In: Proceedings of the International Conference of Phonetic Sciences, pp. 2145–2148.

Yu, D., Deng, L., 2016. Automatic Speech Recognition. Springer Publishing Company, Incorporated.

Zechner, K., Waibel, A., 2000. Minimizing word error rate in textual summaries of spoken language. In: 1st Meeting of the North American Chapter of the Association for Computational Linguistics, Seatle, pp. 186–193.

Zhou, J., Liang, R., Zhao, L., Tao, L., Zou, C., 2014. Unsupervised learning of phonemes of whispered speech in a noisy environment based on convolutive non-negative matrix factorization. Inform. Sci. 257, 115–126.