Detecting Repetitions in Spoken Dialogue Systems Using Phonetic Distances

José Lopes[1], *Giampiero Salvi*[1], *Gabriel Skantze*[1], *Alberto Abad*[2], *Joakim Gustafson*[1], *Fernando Batista*[3], *Raveesh Meena*[1], *Isabel Trancoso*[2]

[1]KTH Speech, Music and Hearing, Stockholm, Sweden
[2]IST/INESC-ID Lisboa, Lisboa, Portugal
[3]ISCTE/INESC-ID Lisboa, Lisboa, Portugal

jdlopes@kth.se

Abstract

Repetitions in Spoken Dialogue Systems can be a symptom of problematic communication. Such repetitions are often due to speech recognition errors, which in turn makes it harder to use the output of the speech recognizer to detect repetitions. In this paper, we combine the alignment score obtained using phonetic distances with dialogue-related features to improve repetition detection. To evaluate the method proposed we compare several alignment techniques from edit distance to DTW-based distance, previously used in Spoken-Term detection tasks. We also compare two different methods to compute the phonetic distance: the first one using the phoneme sequence, and the second one using the distance between the phone posterior vectors. Two different datasets were used in this evaluation: a bus-schedule information system (in English) and a call routing system (in Swedish). The results show that approaches using phoneme distances over-perform approaches using Levenshtein distances between ASR outputs for repetition detection.

Index Terms: spoken dialogue systems, repetition detection, phonetic distance

1. Introduction

In human dialogues it is a common practice to use repetitions as a mechanism to correct some message that was not correctly understood. While this mechanism is smoothly handled in human communication, the same does not occur when humans talk to machines [1]. The dialogue in Table 1 from the Let's Go system [2] has several examples where repetitions could not be handled by the system and lead to miscommunication.

Several reasons may contribute to miscommunications in the dialogue presented. First, humans tend to adapt their speech in order to increase their chances of being understood, especially after an utterance that was not correctly understood. Instead of computer-directed speech style, strategies such as hyper-articulation (turns U14 and U16) or shouting [3] are commonly used in those situations. Besides hyper-articulation, speech shifts lengthening were also reported as a mechanism for error resolution in human-computer interaction [4]. Current dialogue systems may not be prepared to deal with this behaviour. When facing these strategies, detecting repetitions using the ASR output becomes a very difficult task given the amount of errors introduced by speech recognition. Errors could also be introduced by other modules, such as in turn U2 where the correctly recognized speech was incorrectly parsed. If the system could detect that turn U4 is a repetition of U2, the ASR output from U2 could be used to dynamically adjust the language model that was used to recognize turn U4. The use of state-specific grammars can also influence the speech recognition output. For instance, in utterance U4 the user says "SEVEN", which was not expected by the grammar and thus could not be recognized.

S1: What is the time of your departure?
U2: SEVEN O'CLOCK [SEVEN O'CLOCK, 10:00]
S3: 10 in the evening. Is this correct?
U4: NO [NO SEVEN, *no*]
S5: What is the time of your departure?
U6: ELEVEN P M [SEVEN P M, 11:00]
S7: 11 in the evening. Is this correct?
U8: NO [NO, *no*]
S9: What is the time of your departure?
U10: ELEVEN P M [SEVEN P M, 11:00]
S11: 11 in the evening. Is this correct?
U12: NO [NO, *no*]
S13: What is the time of your departure?
U14: .SEVEN. [SEVEN, -]
S15: What is the time of your departure?
U16: ELEVEN P M [SEVEN P M, 11:00]
S17: 11 in the evening. Is this correct?
U18: NO [NO, *no*]
S19: What is the time of your departure?
U20: SEVEN P M [SEVEN P M, 7:00]
S21: 7 in the evening. Is this correct?
U22: YES [YES, *yes*]

Table 1: Example of repetitions in Let's Go data. Manual transcription and parsing between square brackets.

Repetitions are a sign of problematic turns. Once detected during on-line interaction several actions could be taken: dynamically adjust the language models to improve recognition, narrowing the user intentions or change the system action to avoid hyperarticulation or shouting. They could also be used in off-line data to select dialogues with series of repetitions so that system developers can analyze them.

In this paper we propose an approach to detect repetitions between turns that does not rely on the ASR output. Our approach is inspired by techniques used in Spoken-Term detection [5]. Our hypothesis is that this method is robust to the noise introduced by the output of speech recognition in the examples described above. We expect to improve repetition detection when compared to methods that compare the ASR outputs directly [6]. The method was tested in two different corpora, in two different languages and in two different applications that deal with real users with very promising results.

The paper is structured as follows. In the next section related work to repetition detection in Spoken Dialogue System (SDS) data will be described. Section 3 describes the datasets and annotation scheme. Section 4 presents the method. Section 5 shows the results. Section 6 discusses the results and Section 7 concludes the paper and points out future work.

2. Related Work

Finding problematic turns in SDS dialogues is a very important resource both to off-line processing and on-line systems. Repetitions were used in [6, 7] as a clue to find miscommunications and correction strategies. In [7], repetition was even the most common correction strategy found in their dataset. Thus detecting repetitions might be useful to detect miscommunication. Martinovsky and Traum in their analysis of breakdowns of human-machine communication [8] refer to repetitions as an example of "continuous tedious miscommunication and also as a cause for the breakdown".

A first step towards detecting repetitions is to find which features can distinguish repetitions from the other utterances. In [1] acoustic-prosodic features in human-computer dialogue were analyzed. Duration, pauses and pitch variability were considered as possible clues to detect corrections, including repetitions. Since repetitions occurrence is highly correlated with hyperarticulation and strong emphasis [6], the properties of hyperarticulated speech in human-compter error resolution [9] could also be relevant to detect repetitions. According to this study, repeated utterances were longer, and had longer and more frequent pauses. The speech rate was also found lower. Average pitch, intonational contour and phonological alternations were found to be significantly different. The repeated utterances were also less disfluent than the original ones.

Two different approaches have been used to built classifiers for detecting repetitions in spoken dialogue system data. In [6] a threshold based classifier using the Levenshtein distance between semantic representations of the utterances was employed. The result was then used as an input to a miscommunication detector. In [7] a classifier for corrections using features that included prosody, ASR related information, the experimental condition of the system and the distance to the correction was trained. When they tried to classify the different types of corrections annotated (instead of binary correction/non-correction classification), repetition detection achieved 33.9% recall and 56% precision.

In our study, we use phonetic-distance based measures between turns as features to detect repetitions. We hypothesize that phonetic-distances can deal with hyper-articulation and lengthening phenomena, and avoid the noise introduced by the speech recognizer.

3. Data Description

In this study we have considered two different datasets. The first subset consists of 41 Let's Go dialogues corresponding to 837 user turns, selected from the data released for the Spoken Dialogue Challenge [10]. Dialogues were selected for having turns with confidence scores below the threshold used by the system to accept the turn as valid.

The second dataset comes from a Swedish commercial call routing system which handles a very large number of calls on a daily basis. A set of 219 dialogues was selected from the whole dataset, corresponding to 1459 turns. Dialogues were selected from the dataset if at least one of the turns was a "NO" and the dialogue was longer than 4 turns, to have more repetitions in the dataset. If there is a "NO" turn, it probably means that there is some information that the system did not understand correctly.

3.1. Annotation

To annotate the data we have used 4 different labels. To introduce them we use examples from the dialogue in Table 1. Turns U6, U10, U16 and U20 have exactly the same content, therefore they are considered **total** repetitions. Turn U14 repeats part of the content of turns U2, U6, U10, U16 and U20. In these cases we used the label **partial**. Turns U6, U10, U16 and U20 repeat "SEVEN" from turn U2. However, since the user also says "NO", we use the **mixed** repetition, instead of partial repetition. All the pairs of utterances that do not have content repeated were labeled as **non-repetitions**.

We adopted these labels since each of them might have a different approach to its detection. The distribution of data per annotation in each data set is presented in Table 2.

Datasets	Total (%)	Partial (%)	Mixed (%)	No Repetition (%)
Let's Go	221 (5.1)	93 (2.1)	52 (1.2)	4005 (91.6)
SweCC	84 (5.7)	47 (3.2)	63 (4.2)	1292 (86.9)

Table 2: Distribution of the repetition types in the datasets.

4. Method for Repetition Detection

The proposed method consists of two phases. In the first phase, we compute the pairwise distance between segments from two different utterances using either the phoneme sequence or the phoneme posterior vectors. In the second phase we try to find the best alignment between the two utterances using the distance matrix computed in the first phase. The alignment returns a score that corresponds to the acoustic distance between utterances.

4.1. Distance Matrix Computation

The first step described in the Dynamic Time Warping (DTW) query matching is to compute the distance matrix for each frame of the utterances. In the algorithm proposed in [5] the distance is computed using the cosine distance between the phone posterior vectors produced by a phone recognizer for each frame. We followed the same procedure. The phonetic posteriors for the Let's Go were obtained using the phonetic tokenizer described in [11] for English, that uses the neural networks trained for the Audimus Speech recognizer [12]. The phonetic posteriors for the SweCC data were estimated with a Recurrent Neural Network (RNN) described in [13], trained with the Swedish SpeechDat telephone speech database [14]. To build the matrix, the silence frames from both files were not considered in the computation of the cosine distance between posterior vectors. We did this to avoid that the best alignment provided in the second phase corresponded to silence segments.

Besides the distance computed using the posteriors, we also computed another distance using the phoneme sequence obtained from the phoneme recognizers. To do this, we first compute the confusion matrix following a similar procedure to the one used in [15] to compensate the confusability between phones. For the Let's Go data we used one month of transcribed data. To build the confusion matrix, we used the phoneme sequence recognized from turns where the user utterance had exactly the same content. For instance, from the dialogue from Table 2 turns U6, U10, U16 and U20 would be compared to train the confusion matrix. For the SweCC dataset, the confusion matrix was computed using the correlation between the phone posterior vectors, under the assumption that the more correlated the phone posterior vectors the more difficult it is to distinguish between them.

The resulting distance matrix is an $n \times m$ (where $n > m$) matrix populated with the distance between the two utterances

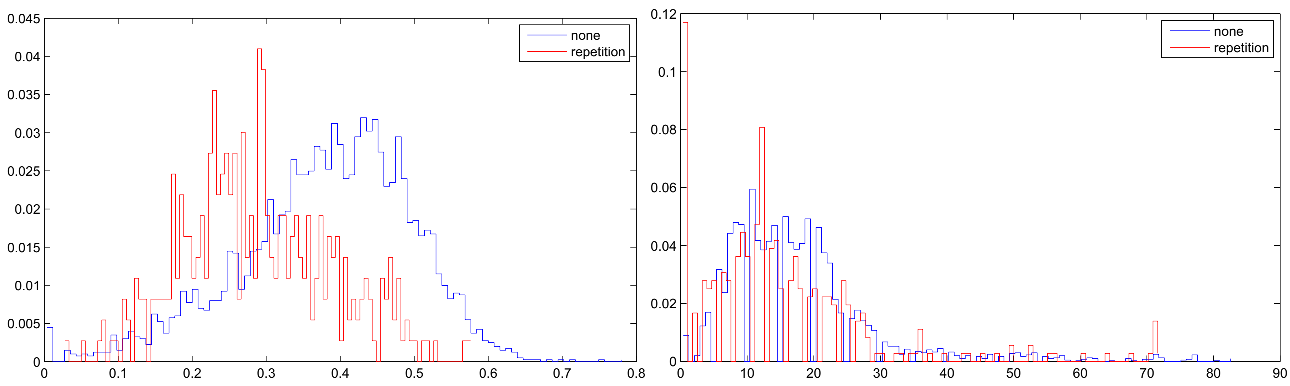

Figure 1: Distribution of the distances in the Let's Go data. Left: computed using extended phoneme sequences and DTW-based matching. Right: computed using the Levenshtein distance between the ASR outputs.

of length n and m respectively. For the phoneme sequence case the element is simply the distance between the pair of phones, whereas for the phoneme posterior case each element is the pairwise distance between posterior vectors.

In Let's Go the phoneme sequence for each utterance was already computed by the phoneme recognizer. In SweCC since only the phoneme posterior vectors are available, the sequence is built based on the maximum posterior found for each frame.

To evaluate the impact of hyperarticulation in repetition detection we have also used a frame-based phoneme sequence with the corresponding phoneme for each 20 ms frame.

4.2. Matching the utterances

Once we had the distance matrix, we tried to find the least costly path in the matrix. We followed the DTW-based matching algorithm proposed in [5] when using either phoneme posterior and phoneme sequences. For the phoneme sequence case we have also used a modified edit distance where the costs of substitutions, deletions and insertions were taken from the confusion matrix. An extra penalty factor was added for consecutive deletions and insertions, since in our data, insertions and deletions (i.e., replacing/being replaced by silence) were more frequent than substitutions.

5. Experimental Results

Several versions of the proposed method were compared to a baseline approach based on the Levenshtein distance between ASR outputs.

In this study all the total and partial repetitions were treated as *repetitions* and the non-repetitions were labeled as *none*. Figure 1 shows the normalized distributions of the distance scores obtained using the phoneme sequence and DTW-based matching and the scores obtained using the Levenshtein distance between ASR outputs. The phoneme sequence with one phoneme per frame was used to compute the distance matrix and DTW-based matching was used to find the best path.

Although there is still an overlap in the distribution in left part of Figure 1, there is a clear separation between the distances obtained for the *repetition* and the *none* categories, whereas the same cannot be observed in the right part of the same figure.

A similar comparison is made in Figure 2 for the SweCC data. Unlike the results for Let's Go, the score computed using the distance between posterior vectors achieved the best performance in the SweCC data. Once more, the scores obtained using our method separate the two sets under analysis performed better than the Levenshtein distance.

5.1. Building a repetition detector

The fact that there is a visible separation in the datasets does not mean that repetition can be detected automatically. In order to verify it, we compared different methods for detecting repetitions where we used different combinations of the scores derived from the methods proposed with other features available from system logs. For each corpus we present results with a phoneme posterior vector based score (PP), a phoneme sequence based score (PS), combined with system independent features (SI), number of words and turn duration; and system dependent features (SD), system act and grammar used.

To train the repetition detection we used JRip and SVMs available in Weka [16]. We first trained the classifiers using 10-fold cross-validation scheme (10-f CV). Since our data was skewed, we also split the dataset into 70% for training and 30% for testing and oversampled (OS) the training data using the algorithm described in [17] . We also report the results obtained using a simple threshold tuning (TT) procedure for the different scores tested. The threshold was tuned using the 70% of the data used for training (without oversampling) and the results are reported in the test set.

Figure 3 compares the different Unweighted Average Recall (UAR) [18] for the different features sets and evaluation methods. This performance measure was chosen since our datasets were skewed. The results for Let's Go show that the distance based on PS performed better than the one based on PP. When combined, the detection performance increases. The combination of the distance features with features from the system logs also improves the performance by 10% absolute UAR. Compared with the performance achieved using only the system dependent features we have a 5% absolute UAR improvement.

In the SweCC case the gains in performance are not so clear when adding system log features. In fact, using a simple TT procedure with the phoneme posterior based distance the UAR obtained is 81%. Using the OS procedure and combining SD and SI features, the performance increases to 86%. The combination of all the features only improves the UAR over the SD features when using the original dataset without oversampling.

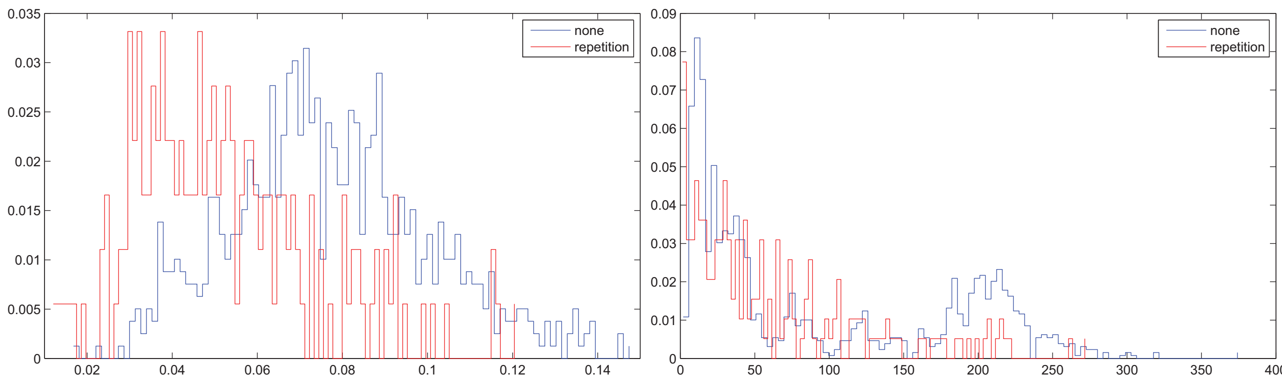

Figure 2: Distribution of the distances in the SweCC data. Left: computed using phonetic posteriors vectors distance and DTW-based matching. Right: computed using the Levenshtein distance between the ASR outputs.

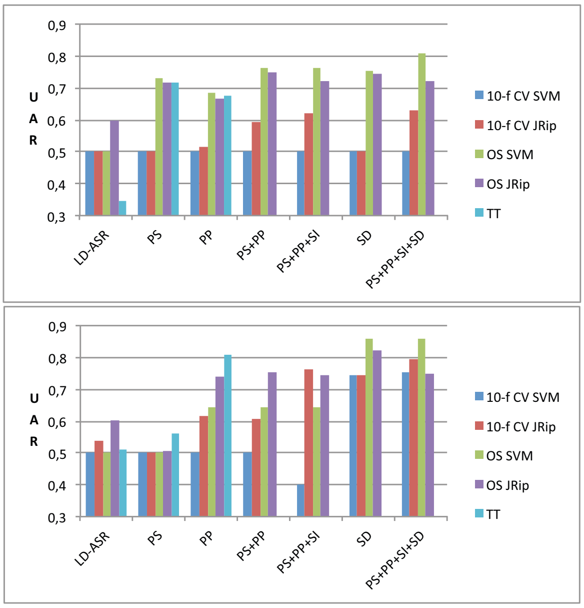

Figure 3: Results for repetition detection. Top: Let's Go. Bottom: SweCC.

6. Discussion

The presented results confirm that the phonetic distances are robust to the noise introduced by the ASR and improve repetition detection compared to the previously proposed methods that use ASR outputs to find repetitions

Interestingly, different versions of the PS and PP lead to better results depending on the dataset. We believe this is due to the different phoneme recognizers used. Whereas in Let's Go the phoneme sequence is restricted by the phonotactic information, in SweCC the phoneme sequence is extracted directly from the phoneme posterior vector making it more noisy and hindering the performance of the PS-based distance in this dataset.

Figures 1 and 2, show that DTW-based matching using acoustic based distances globally improves repetition detection. The fact that the method accommodates partial repetition detection might have contributed to this result.

Another restriction that we added when using the PP score that improved the performance was to discard all the alignments that were shorter than 750 ms. Since we are interested in detecting repetitions of content that can be used to fill the system slots, this threshold eliminates all the false alarms that correspond to irrelevant information for slot filling.

In cases where we do not have access to SD dependent features, our method can be very useful. In SweCC if we use only the PP features the MRR is only 5% below the performance achieved using only SD features. In Let's Go if we combine PP+PS features the performance is even better than using just SD features. The SD features seem to be more powerful in SweCC which is an indication that repetitions may have occurred in the same dialogue state using the same grammar.

Finally, it is interesting to observe that the results using the Levenshtein distance between ASR outputs are better in the SweCC data (0.51 vs. 0.35). Considering the figures for WER in both datasets SweCC is 21.3% and Let's Go 33.3%. This means that using the Levenshtein distance between ASR outputs might be appropriate in systems where the ASR performance is better. However, current state-of-the art speech recognition in SDSs is far from being perfect, which suggests that our method might be more appropriate for some systems.

7. Conclusions and Future Work

In this paper we have presented an approach for repetition detection in SDSs. The performance for detection of repetitions greatly improves when compared to the performance obtained using the Levenshtein distance between ASR outputs. This confirms our hypothesis the method presented is robust to the noise introduced by the ASR output. The performance of the method was comparable to the performance using system dependent features in SweCC data and even better in Let's Go data.

In the future we plan to evaluate the method including mixed repetitions. We also plan to implement the method in a live system and test recovering strategies using the repetition detection information during live interaction.

8. Acknowledgements

The authors would like to thank Luis J. Fuentes-Rodriguez for providing the implementation of the DTW-based search and suggestions. This research is supported by the EU project SpeDial, EU grant # 611396. This work was partially supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UID/CEC/50021/2013.

9. References

[1] G. Levow, "Characterizing and recognizing spoken corrections in human-computer dialogue," in *Proceedings of COLING/ACL*, 1998.

[2] A. Raux, B. Langner, D. Bohus, A. W. Black, and M. Eskenazi, "Let's Go public! Taking a spoken dialog system to the real world." in *INTERSPEECH*. ISCA, 2005, pp. 885–888.

[3] L. Bell and J. Gustafson, "Repetition and its phonetic realizations: Investigating a Swedish database of spontaneous computer directed speech," in *Proceedings of ICPhS-99*, 1999, pp. 1221–1224.

[4] S. Oviatt, G. Levow, M. MacEachern, and K. Kuhn, "Modeling hyperarticulate speech during human-computer error resolution," in *Proceedings of ICSLP*, 1996.

[5] L. J. Rodriguez-Fuentes, A. Varona, M. Penagarikano, G. Bordel, and M. Diez, "High-performance query-by-example spoken term detection on the SWS 2013 evaluation," in *ICASSP 2014*, 2014.

[6] A. Batliner, K. Fischer, R. Huber, J. Spilker, and E. Nöth, "How to find trouble in communication," *Speech Communication*, no. 40, pp. 117–143, 2003.

[7] D. Litman, M. Swerts, and J. Hirschberg, "Characterizing and predicting corrections in spoken dialogue systems," *Computational Linguistics*, vol. 32, no. 3, pp. 417–438, 2006.

[8] B. Martinovsky and D. Traum, "The error is the clue: breakdown in human-machine interaction," in *Proceedings of the ISCA Tutorial and Research Workshop Error Handling in Spoken Dialogue Systems*, Chteau d'Oex, Vaud, Switzerland, Aug 2003.

[9] S. Oviatt, M. MacEachern, and G. A. Levow, "Predicting hyperarticulate speech during human-computer error resolution," *Speech Communication*, vol. 24, no. 2, pp. 87–110, may 1998.

[10] A. W. Black, S. Burger, B. Langner, G. Parent, and M. Eskenazi, "Spoken dialog challenge 2010." in *SLT*. IEEE, 2010, pp. 448–453.

[11] A. Abad, "The L2F language recognition system for Albayzin 2012 evaluation," in *IberSPEECH*, Nov 2012.

[12] H. Meinedo, A. Abad, T. Pellegrini, I. Trancoso, and J. Neto, "The L2F broadcast news speech recognition system," in *Fala2010*, Nov 2010, pp. 93–96.

[13] G. Salvi, "Dynamic behaviour of connectionist speech recognition with strong latency constraints," *Speech Communication*, vol. 48, no. 7, pp. 802–818, Jul. 2006.

[14] K. Elenius, "Experience from collecting two Swedish telephone speech databases," *International Journal of Speech Technology*, vol. 3, pp. 119–127, 2000.

[15] L. Qin and A. I. Rudnicky, "Finding recurrent out-of-vocabulary words." in *INTERSPEECH*. ISCA, 2013, pp. 2242–2246.

[16] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: An update," *SIGKDD Explorations*, vol. 11, no. 1, 2009.

[17] N. V. Hawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *J. Artif. Int. Res.*, vol. 16, no. 1, pp. 321–357, jan 2002.

[18] R. Higashinaka, Y. Minami, K. Dohsaka, and T. Meguro, "Modeling user satisfaction transitions in dialogues from overall ratings," in *Proceedings of the SIGDIAL 2010 Conference*. Tokyo, Japan: Association for Computational Linguistics, Sep 2010, pp. 18–27.