Articulatory Feature Asynchrony Analysis and Compensation in Detection-Based ASR

I-Fan Chen[1] *and Hsin-Min Wang*[1,2]

[1]Research Center for Information Technology Innovation, Academia Sinica, Taipei, Taiwan
[2]Institute of Information Science, Academia Sinica, Taipei, Taiwan
{ifanchen, whm}@iis.sinica.edu.tw

Abstract

This paper investigates the effects of two types of imperfection, namely detection errors and articulatory feature asynchrony, of the front-end articulatory feature detector on the performance of a detection-based ASR system. Based on a set of variable-controlled experiments, we find that articulatory feature asynchrony is the major issue that should be addressed in detection-based ASR. To this end, we propose several methods to reduce the asynchrony or the effects of asynchrony. The results are quite promising; for example, currently, we can achieve 67.67% phone accuracy in the TIMIT free phone recognition task with only 11 binary-valued articulatory features.

Index Terms: articulatory feature asynchrony, detection-based ASR, speech recognition

1. Introduction

In recent years, detection-based automatic speech recognition (ASR) has become a popular research topic in fields related to ASR. By simulating human speech recognition (HSR), detection-based ASR systems attempt to reduce the gap between HSR and ASR. Basically, the framework of detection-based ASR can be divided into two key components: a front-end knowledge attribute detector and a backend knowledge integrator [1]. The front-end process collects a wide variety of speech-related knowledge attributes to form knowledge sources; and the backend process integrates the attributes into higher-level speech units, such as phones, syllables, words, and sentences. If the front-end detector targets articulatory features (AFs), two factors may affect the system's overall recognition accuracy: the errors produced by the detector and the asynchrony of AFs.

The asynchrony of AFs is a phenomenon caused by variations in natural speech production; i.e., the onset times of different AFs shift differently in different contextual conditions in speech. Because of this phenomenon, it is generally believed that one of the reasons why AFs are more suitable than phones for ASR is their ability to describe the contextual information in speech. A number of studies have focused on this phenomenon. For example, in [2], AF asynchrony was introduced to the articulatory feature recognizer by applying an embedded training technique; while in [3], the authors used the pattern of AF asynchrony in the AF space to probe for acoustic differences in the onset of Dutch singulars and plurals. However, it is still unclear what role this phenomenon plays in a phone or word recognition task, which maps multiple asynchronous AF streams into a single phone or word stream.

In our previous research [4], we found that the imperfection of the front-end articulatory feature detector (AFDT) can cause a substantial decline in recognition accuracy in detection-based ASR. Here, we investigate the effects of two types of the above imperfection, namely AFDT errors and AF asynchrony, on the backend integrator's performance. To compare the relative importance of the two factors, we design a set of variable-controlled experiments. The results show that AF asynchrony plays a more important role in detection-based ASR than AFDT errors. Based on this finding, we propose a number of ways to improve the system performance. We believe that our experiment results provide further insight into how to design a high-performance detection-based ASR system.

2. Front-end articulatory feature detection and AF asynchrony analysis

2.1. Front-end articulatory feature detection

The AFDT's detection target is the Government Phonology (GP) feature set [5], a phonological feature system in which speech sounds are destructed into a set of primes and can be represented by fusing these primes structurally. We select the GP feature set because the results of our previous study show that it is the most effective for building a high-performance detection-based ASR system [4]. In this work, the GP feature set contains 11 AFs, including 8 AFs (namely "A", "I", "U", "E", "S", "h", "H", and "N") defined in the original GP feature system [5] and 3 AFs ("a", "i", and "u") added by King and Taylor [6].

In addition to the 11 AFs, our front-end AFDT simultaneously detects TIMIT 61 phones for AF asynchrony analysis. The objective of phone detection is to remove the bias introduced by the inconsistency between human alignment and machine alignment. AF asynchrony is defined as the shift between a phone's boundary and the boundaries of its associated AFs. Thus, the different criteria used for AF alignment and phone alignment might lead to inaccurate articulatory feature asynchrony measurements. Even though the TIMIT corpus provides human aligned phone labels, to generate unbiased AF asynchrony data, we need to re-align the phone sequence in speech using our front-end AFDT.

Following the work in [6], we use a single time delay recurrent neural network [7] to implement the front-end AFDT. The inputs of the neural network are 12 Mel-frequency cepstral coefficients (MFCCs) plus energy, which are extracted by a 25-ms Hamming-windowed frame with 10-ms shifts. The neural network outputs a 72-dimensional vector representing 11 GP AFs and 61 TIMIT phones. The value of each element in the vector ranges from 0 to 1, which can be treated as a posterior probability. The values of the AFs are then discretized[1] with a threshold to form binary-valued AF

[1] Theoretically, CRFs can handle real-valued input; however, we discretize the AFDT's results because the CRF++ toolkit used in our experiments only supports binary-valued input. Moreover, it is easier to define the boundaries of AFs, which are important for asynchrony measurement, by using the discrete AF detection results.

detection results, which are input to the backend CRF integrator. Our AFDT is a recurrent neural network and linkage pruning is not applied. Therefore, the output AF and phone posterior probabilities of the current frame are fed back to the neural network as additional inputs for the subsequent frame. As a result, the AFDT learns the relations and dependencies between AFs and phones.

2.2. AF and phone alignments using AFDT

To analyze the AF asynchrony around each phone boundary, we use the front-end AFDT to obtain the individual frame alignments of AFs and phones. The AFDT generates the posterior probabilities of the TIMIT 61 phones for each frame. Thus, based on the phone label sequence supplied by the TIMIT corpus, we can use the generated posterior probabilities for phone-level forced alignment with the Viterbi algorithm.

AF sequence alignment is more complicated than phone alignment. For each AF, forced alignment is performed individually. The alignment shows whether or not the specific AF is active in each frame, which means there are only two kinds of labels ("*active*" and "*inactive*") to be aligned. The label sequence is derived from the phone label sequence provided by the TIMIT corpus. However, since an AF might be active for some phones and inactive for others, a long phone label sequence might correspond to a short *active*/*inactive* sequence for a specific AF. For example, a long utterance of dozens of phones might correspond to a short sequence /n_**u** **u** n_**u**/ for AF **u**, where n_ denotes the inactivity of the feature. As a result, the forced alignment constraint becomes very relaxed, and any error in the posterior probabilities generated by the AFDT may cause the data to be wrongly aligned. In such cases, it is not possible to analyze the AF asynchrony accurately. To solve this problem, we use the realigned-phone-derived AF labels as prior knowledge to softly constrain the aligned boundaries of AFs to the realigned phone boundaries. Thus, for an articulatory feature AF_K, the smoothed active posterior probability of frame *i* is computed by

$$P_{Smoothed}^{AF_{K\_active}}(i) = \alpha \cdot P_{AFDT}^{AF_{K\_active}}(i) + (1-\alpha) \cdot P_{realignedPhoneDerived}^{AF_{K\_active}}(i), \qquad (1)$$

where $P_{AFDT}^{AF_{K\_active}}(i)$ is the posterior probability generated by the front-end AFDT; $P_{realignedPhoneDerived}^{AF_{K\_active}}(i)$ is the AF_K's value, either 1 (*active*) or 0 (*inactive*), derived from the realigned phone label of frame *i*; and α is a weighting factor for smoothing, which is empirically set to 0.2 in this paper. The smoothed inactive posterior probability of frame *i* is calculated by $P_{Smoothed}^{AF_{K\_inactive}}(i) = 1 - P_{Smoothed}^{AF_{K\_active}}(i)$. Forced alignment is then performed based on the smoothed posterior probabilities for AF_K and the *active-inactive* sequence using the Viterbi algorithm.

3. Backend phone recognition using conditional random fields

In the phone recognition task addressed in this paper, we use conditional random fields (CRFs)[8] as the backend integrator. The observation sequence **x** of our CRF model is comprised of the frame-based GP11 AFs detected by the front-end AFDT, and the target label sequence **y** is a frame-based phone sequence. We build a CRF model to map a frame-based AF sequence to its associated frame-based phone sequence. Then, the latter is merged into a phone sequence as the final phone recognition output. The state feature function in the CRF model is set to $s(y_i, x_{i-1}, x_i, x_{i+1})$, in which the preceding, current, and subsequent phonological feature observations are considered simultaneously. The transition feature function is set to label bi-grams, i.e., $t(y_{i-1}, y_i)$. We use the CRF++ toolkit to implement CRFs [11].

3.1. Training methods

In our previous research, we found that the system performance is highly dependent on how the CRF model is trained [4]. Thus, in this paper, we consider three methods for CRF training. The major difference between the methods is the information contained in the training data.

3.1.1. Oracle data training

The first method uses the phone-derived, frame-based AF value sequence to train the CRF model. Since, the values of the 11 AFs and the phone label for each frame are perfectly matched according to the definition of the phonological feature system, this method is called oracle data training (OT). A CRF model trained in this way does not learn the AFDT errors and the AF asynchrony effect. In our previous research, we found that it can extract more phonological information while performing phone recognition on the detected AF data generated by the front-end detector. However, due to the mismatch between the training and testing conditions, the recognition results suffered from serious insertion errors.

3.1.2. Detected data training

The second method treats the function of the front-end AFDT more like acoustic feature extraction than AF detection. The CRF model is trained on the feature vectors generated by the front-end AFDT directly, even if the data contains errors and might be inconsistent with the phonological feature system. This method, called detected data training (DT), is commonly used by researchers [9][10]. In contrast to the oracle data, the detected data provides the CRF model with information about the AFDT errors and the AF asynchrony, which may help the model to handle erroneously detected AFs better in the recognition phase. However, the presence of this information reduces the weight of phonological knowledge in the data. The DT-trained CRF model is thus less sensitive to the composition of AFs, which is important information for a phonological feature system to differentiate between two phones. Our previous study [4] showed that, compared to the OT-trained CRF model, the DT-trained CRF model yields lower insertion errors; however, its recognition results provide less information to the fusion system, which combines the recognition results of different ASR systems.

3.1.3. AFDT aligned data training

In this paper, we propose a new method for training the CRF-based backend integrator. The individual frame alignments of the phone sequence and the 11 AF sequences of the training data (obtained by the alignment methods described in Section 2.2) are used to train the CRF model. Therefore, we call the method AFDT aligned data training (AT). The aligned training data contains the AF asynchrony information but no AFDT errors. From another perspective, AFDT aligned data training can be seen as a compromise between oracle and detected data training. The AF compositions in the aligned data are similar to those in the detected data around phone boundaries, and are close to those in the oracle data in non-boundary regions. Therefore, the CRF model can learn both phonological knowledge and AF asynchrony information from the training data.

By comparing the performance of systems using the three methods described in this section, we can determine the individual effects of AFDT errors and AF asynchrony on the

recognition performance in detection-based ASR. The derived knowledge will help us to further refine our detection-based ASR system.

4. Experiments

4.1. Experiment setup

We conduct experiments on the TIMIT acoustic-phonetic continuous speech corpus, but the dialect utterances (SA1 and SA2) are not used. The database is divided into three parts: a training set (3296 utterances), a development set (400 utterances), and a test set (1344 utterances). The training and development sets are subsets of the standard TIMIT training set, while the test set is the standard TIMIT test set. In the experiments, all the models are trained by the training set, and the configurations and parameters associated with the models are assigned empirically based on the development set. The 61 phones defined in TIMIT are used as recognition units; however, for the performance evaluation, the recognized TIMIT 61-phone results are mapped to the CMU/MIT 39-phone set. No language models are used in recognition.

4.2. The baseline detection-based ASR systems

First, we evaluate the baseline detection-based ASR systems, in which the CRF models are trained by oracle data training (OT) and detected data training (DT). We consider two test conditions: the ideal case and the real case. In the ideal case, the test data is set to match the CRF model's training conditions. In the OT-trained CRF system, the AFs converted from the human phone labels of the testing data are input to the CRF model. In this way, we can measure the performance upper bound of the CRF-based system. In the real case, the inputs to the CRF models are the AFs detected automatically by the AFDT. The phone recognition results of these baseline systems are shown in Table 1, where Corr (correct rate) and Acc (accuracy) are obtained by HTK's HResults tool; and Acc = Corr – insertion rate. The second row shows that if there are no AFDT errors and AF asynchrony, the OT-trained CRF system can achieve a very high phone recognition performance. Clearly, the GP11 AF set demonstrates high potential for detection-based ASR; therefore, we may consider this performance upper bound our ultimate goal in this research. In the real case, since the detected AFs inevitably contain errors and asynchrony, though with a high correct rate, the OT-trained CRF system suffers from serious phone insertion errors, as shown in the third row of Table 1. Although the performance of the DT-trained CRF system is quite stable in terms of the correct rate and accuracy, the values are not satisfactory.

Table 1. The phone recognition results of the oracle data trained (OT) CRF model and the detected data trained (DT) CRF model.

Test Data Type	System	Corr	Acc
Ideal (upper bound)	OT CRF	98.31	98.28
Detected (real case)	OT CRF	70.55	34.38
	DT CRF	57.30	56.14

4.3. The AFDT aligned data trained system

To identify the degrees of AFDT errors and AF asynchrony that cause the substantial difference in the performance of the OT-trained CRF model between the ideal case and the real case, we applied AFDT aligned data training (AT) in the experiments. Table 2 shows the ideal and real performances of the AT-trained CRF system, where the ideal performance (upper bound) is obtained by feeding the individual frame alignments of the 11 AF sequences of the test data (obtained by the alignment methods described in Section 2.2) to the AT-trained CRF model. Comparing the upper bound performance of the AT-trained CRF model in Table 2 with that of the OT-trained CRF model in Table 1, it is clear that the introduction of AF asynchrony to the backend CRF integrator has a major impact on the performance (26.82% drop in the correct rate and 27.97% drop in accuracy). The effect of the front-end AFDT errors can be measured by the difference between the performances of the AT-trained CRF model in the ideal and real cases. Specifically, the differences are only 6.62% for the correct rate and 7.99% for accuracy. We also compare the real case performance of the DT-trained CRF model (the fourth row in Table 1) with that of the AT-trained CRF model (the third row in Table 2). Clearly, the CRF model trained with AF asynchrony information alone outperforms the CRF model trained with both AF asynchrony and AFDT errors. The above evidences show that AF asynchrony plays a much more important role than AFDT errors in detection-based ASR. In other words, we can conclude that, to build a high-performance detection-based ASR system, it is better to use AFDT aligned data training for the backend CRF integrator; and employ as many methods as possible to reduce or conceal AF asynchrony from the front-end AFDT.

Table 2. The phone recognition results of the AFDT aligned data trained (AT) CRF model.

Test Data Type	System	Corr	Acc
Ideal (upper bound)	AT CRF	71.49	70.31
Detected (real case)	AT CRF	64.87	62.32

4.4. AF asynchrony compensation

In this section, we perform several preliminary experiments to determine whether AF asynchrony can be reduced or concealed. AF asynchrony compensation can be applied in the front-end AFDT or the backend CRF integrator.

4.4.1. AF asynchrony compensation in the AFDT using long-term information

Contextual variation in natural speech is one of the major reasons for AF asynchrony [2]. One way to reduce the output asynchrony is to let the AFDT learn the contextual variation directly. This can be done by introducing long-term information in speech to the AFDT. It has already been shown that such information is helpful for speech recognition [12]. We believe that using long-term speech information in the AFDT makes the detection of AFs more stable than using only local speech information; hence, the AF boundary shift around the phone boundary can be reduced.

Table 3. The mean AF-phone boundary distances (in frames) of GP11 AFs generated by the original AFDT (using MFCC as input) and the AFDT using long-term information.

AFDT sys	a	A	E	h	H	i	I	N	S	u	U
MFCC	0.94	4.27	1.18	0.94	0.59	1.07	2.44	1.74	0.56	1.88	7.16
Long Term	0.75	3.57	0.94	0.77	0.38	0.91	2.36	1.15	0.43	1.32	4.55

Following the approach in [12], we use the Mel-frequency filter bank vectors in a 310-ms window jointly as input to implement an AFDT with long-term speech information. Table 3 shows the statistics of the AF asynchrony of the original AFDT and the AFDT with long-term information. Clearly, the introduction of long-term information reduces the AF asynchrony for all 11 AFs significantly.

4.4.2. AF asynchrony compensation in the CRF integrator using long-term information

Even though the front-end AFDT generates AF asynchrony, there are some ways to conceal the effect of this disturbance on the backend CRF integrator. One way is to increase the time range used for CRF feature extraction. In our original CRF model, the state feature function considers the AF observations in the current frame as well as those in the adjacent frames (±1). When processing frames around a phone boundary, where AF asynchrony occurs, the CRF model might be more affected by the asynchrony due to the short time range. To mitigate the asynchrony effect, we can use the AF values in the n-th preceding frame and the n-th subsequent frame (±n) instead of the adjacent frames in the state feature function. The results of experiments on the development set show that it is adequate to set n= 3.

4.4.3. The AT-trained CRF system with AF asynchrony compensation

Table 4 shows the phone recognition performance of the AT-trained CRF system after applying the AF asynchrony compensation methods. Comparing the ideal case of the AT-trained CRF systems with and without AF asynchrony compensation, we find that the two compensation methods improve the upper bound as expected. We also find that compensation in the front-end AFDT is more effective than in the backend integrator. It seems that the effects of these two methods are cumulative. By combining them, we can improve the accuracy upper bound from 70.31% to 74.97%. From Table 4, we also observe that AF asynchrony compensation improves the real performance of detection-based ASR systems. This supports our conclusion in Section 4.3.

Table 4. The phone recognition results of the AT-trained CRF system with AF asynchrony compensation

Test Data Type	System	Corr	Acc
-	CI-HMM	69.02	63.45
-	CD-HMM	75.76	65.78
Detected (real case)	OT CRF(±3)	75.24	47.97
	Long Term AFDT + DT CRF(±3)	64.58	63.12
Ideal (upper bound)	Long Term AFDT + AT CRF	74.96	73.64
	MFCC AFDT + AT CRF(±3)	72.87	71.62
	Long Term AFDT + AT CRF(±3)	76.41	74.97
Detected (real case)	Long Term AFDT + AT CRF	69.83	66.97
	MFCC AFDT + AT CRF(±3)	66.21	63.16
	Long Term AFDT + AT CRF(±3)	**71.01**	**67.67**

The AFDT with long-term information are more complicated than the original AFDT, and the CRF model using ±3 frames also increases the number of trainable model parameters from 9,089 to 11,773. Thus, the improved performance might be a result of the more complicated AFDT and CRF model. To validate that AF asynchrony compensation is the major reason for the improved performance, we implement the OT-trained and DT-trained CRF systems with the new AFDT and CRF settings. The results are shown in the fourth and fifth rows of Table 4. Although the new settings improve the performance of both systems, the AT-trained CRF system still outperforms them. This demonstrates that AF asynchrony is a very important research issue in detection-based ASR. For reference, the performances of HMM-based ASR systems are shown in the second and third rows of Table 4. It is obvious that the best detection-based ASR system outperforms the HMM-based ASR systems.

5. Conclusion and future work

In this work, we describe a set of variable-controlled experiments designed to identify potential difficulties in detection-based ASR. The results show that if AF asynchrony is not handled properly, it can cause serious problems in mapping AF multi-streams into a single phone or word stream. Although including information about AF asynchrony in the backend integrator training can resolve the problems to some extent, the asynchrony still degrades the upper bound of the system performance substantially. Thus, there is a need for a component or mechanism that can compensate for AF asynchrony in the current detection-based ASR framework, in order to imitate HSR. The need for this component is shown by the fact that humans can use phoneme systems to represent word pronunciations easily in spite of co-articulation effects.

Our study shows that AF asynchrony compensation would be a worthwhile avenue for future research, and that methods for reducing/concealing AF asynchrony are promising. For the front-end AFDT, AF asynchrony minimization training and AF-phone synchronization could also be investigated. Meanwhile, for the backend integrator, a more advanced method for concealing AF asynchrony would also be worth studying. Information like prosody or higher level knowledge might also help us tackle this problem in the front-end detector of detection-based ASR systems. We believe that all of these methods are crucial to improving the performance of detection-based ASR.

6. Acknowledgement

This work was supported by the National Science Council of Taiwan under Grants: NSC 97-2221-E-001-022-MY3 and NSC98-2631-001-013.

7. References

[1] Lee, C.-H., Clements, M.A., Dusan, S., Fosler-Lussier, E., Johnson, K., Juang, B.-H. and Rabiner, L.R., "An Overview on Automatic Speech Attribute Transcription (ASAT)," in *Proc. INTERSPEECH 2007*.

[2] Wester, M., Frankel, J. and King, S., "Asynchronous Articulatory Feature Recognition Using Dynamic Bayesian Networks," in *Proc. IEICI Beyond HMM Workshop*, 2004.

[3] Bosch, L. ten, Baayen, H. and Ernestus, M., "On Speech Variation and Word Type Differentiation by Articulatory Feature Representations," in *Proc. INTERSPEECH 2006*.

[4] Chen, I.-F. and Wang, H.-M., "An Investigation of Phonological Feature Systems used in Detection-Based ASR," in *Proc. ISCSLP 2008*.

[5] Harris, J., English Sound Structure, Blackwell, 1994.

[6] King, S. and Taylor, P., "Detection of Phonological Features in Continuous Speech using Neural Networks," Computer Speech and Language, vol. 14, pp. 333-353, 2000.

[7] Strom, N., "The NICO Artificial Neural Network Toolkit," http://nico.nikkostrom.com

[8] Lafferty, J. McCallum, A. and Pereira, F., "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data," in *Proc. ICML2001*.

[9] Morris, J. and Fosler-Lussier, E., "Combining Phonetic Attributes Using Conditional Random Fields," in *Proc. INTERSPEECH, 2006*.

[10] Siniscalchi, S.M., Svendsen, T. and Lee, C.-H, "Towards Bottom-Up Continuous Phone Recognition," in *Proc. ASRU 2007*.

[11] Kudo, T., "CRF++: Yet Another CRF Toolkit," http://crfpp.sourceforge.net

[12] Schwarz, P., Matejka, P. and Cernocky, J., "Towards Lower Error Rates in Phoneme Recognition," in *Proc. TSD2004*, Brno, Czech Republic, 2004.