A Reconstruction Decoder for the Perceptual Computer

Dongrui Wu
Machine Learning Lab, GE Global Research, Niskayuna, NY, USA
E-mail: wud@ge.com

***Abstract*—The Word decoder is a very important approach for decoding in the Perceptual Computer. It maps the computing with words (CWW) engine output, which is a fuzzy set, into a word in a codebook so that it can be understood. However, the Word decoder suffers from significant information loss, i.e., the fuzzy set model of the mapped word may be quite different from the fuzzy set output by the CWW engine, especially when the codebook is small. In this paper we propose a Reconstruction decoder, which represents the CWW engine output as a combination of two successive codebook words with minimum information loss by solving a constrained optimization problem. The Reconstruction decoder can be viewed as a generalized Word decoder and it is also implicitly a Rank decoder. Moreover, it preserves the shape information of the CWW engine output in a simple form without sacrificing much accuracy. Experimental results verify the effectiveness of the Reconstruction decoder. Its Matlab implementation is also given in this paper.**

***Index Terms*—Computing with words, Perceptual Computer, decoder, type-1 fuzzy sets, interval type-2 fuzzy sets**

I. Introduction

Computing with words (CWW) [33], [34] is "*a methodology in which the objects of computation are words and propositions drawn from a natural language*." Many different approaches for CWW have been proposed so far [1], [6], [7], [9], [11], [16]–[18], [20], [22], [30], [31], [35]. According to Wang and Hao [21], these techniques may be classified into three categories:

- The Extension Principle based models [1], [3], [13], [16], which operate the underlying fuzzy set (FS) models of the linguistic terms using the Extension Principle [32].
- The symbolic models [4], which operate on the indices of the linguistic terms.
- The 2-tuple representation based models [5], [6]. Each 2-tuple includes a linguistic term and a numeric number in $[-0.5, 0.5)$, which allows a continuous representation of the linguistic information in its domain.

Each category of models has its unique advantages and limitations. The Extension Principle based models can deal with any underlying FS models for the words, but they are computationally intensive. Moreover, their results usually do not match any of the initial linguistic terms, and hence an approximation process must be used to map the results to the initial expression domain. This results in loss of information and hence the lack of precision [2], [21]. The symbolic models are much computationally simpler than the Extension Principle based models, but they do not directly take into account the underlying vagueness of the words [9]. Also, they have the same information loss problem as the Extension Principle based models. The 2-tuple representation based models can avoid the information loss problem, but generally they have constraints on the shape of the underlying FS models for the linguistic terms, i.e., they need to be equidistant [6]. There have also been hybrid approaches, which try to combine the advantages of different models but eliminate their limitations, e.g., a new version of 2-tuple linguistic representation model [21], which combines symbolic models with the 2-tuple representation models to eliminate the "equal-distance" constraint. However, to the best of the author's knowledge, there has not been active research into the information loss problem of the Extension Principle based models.

In this paper we focus on the Extension Principle based models, particularly, the Perceptual Computer (Per-C) [13], [16]. It has the architecture that is depicted in Fig. 1, and consists of three components: encoder, CWW engine and decoder. Perceptions–words–activate the Per-C and are the Per-C output (along with data); so, it is possible for a human to interact with the Per-C using just a vocabulary. A vocabulary is application (context) dependent, and must be large enough so that it lets the end-user interact with the Per-C in a user-friendly manner. The encoder transforms words into FSs and leads to a *codebook*–words with their associated FS models. Both type-1 (T1) and interval type-2 (IT2) FSs [12] may be used for word modeling, but we prefer the IT2 FSs because their footprint of uncertainty (FOU) can model both the intra-personal and inter-personal uncertainties associated with the words [14]. The outputs of the encoder activate a CWW engine whose output is one or more other FSs, which are then mapped by the decoder into a recommendation (subjective judgment) with supporting data.

Thus far, there are three kinds of decoders according to three forms of recommendations:

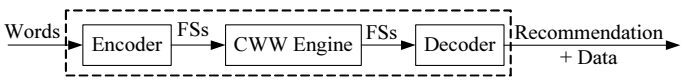

Fig. 1. Conceptual structure of the Perceptual Computer.

1) *Word:* To map a FS into a word, it must be possible to compare the *similarity* between two FSs. The Jaccard similarity measure [26] can be used to compute the similarities between the CWW engine output and all words

in the codebook. Then, the word with the maximum similarity is chosen as the Decoder's output.

2) *Rank:* Ranking is needed when several alternatives are compared to find the best. Because the performance of each alternative is represented by a FS obtained from the CWW engine, a ranking method for IT2 FSs is needed. A centroid-based ranking method for T1 and IT2 FSs is described in [26].
3) *Class:* A classifier is necessary when the output of the CWW engine needs to be mapped into a decision category [15]. Subsethood [16], [19], [23] is useful for this purpose. One first computes the subsethood of the CWW engine output for each of the possible classes. Then, the final decision class is the one corresponding to the maximum subsethood.

The Word decoder suffers a lot from the aforementioned information loss problem because a user-friendly codebook only contains a small number (usually around seven) of words and hence an FOU may be mapped into a codebook word whose FOU looks quite different. In this paper we propose a Reconstruction decoder for the Per-C, which can be used to replace the Word decoder with very little loss of information.

The remainder of this paper is organized as follows: Section II introduces the details of the Reconstruction decoder. Section III presents some experimental results. Section IV draws conclusions. Matlab implementation of the Reconstruction decoder is given in the Appendix.

II. The Reconstruction Decoder for the Per-C

So far almost all FS models used in CWW are normal trapezoidal FSs (triangular FSs are special cases of trapezoidal FSs), no matter whether they are T1 or IT2 FSs. Additionally, the only systematic methods for constructing IT2 FSs from interval survey data are the Interval Approach [10] and its enhanced version, the Enhanced Interval Approach [29], both of which only output normal trapezoidal IT2 FSs. So, in this paper we focus on normal trapezoidal T1 and IT2 FSs for simplicity. We will discuss how our method can be extended to more general FS models like Gaussian FSs or subnormal FSs at the end of this section. Matlab implementation of the Reconstruction decoder is given in the Appendix. It can be used for both T1 and IT2 normal trapezoidal FSs.

A. The Reconstruction Decoder for T1 FS Word Models

A normal trapezoidal T1 FS can be represented by four parameters shown in Fig. 2. Note that a triangular T1 FS is a special case of the trapezoidal T1 FS when $b = c$.

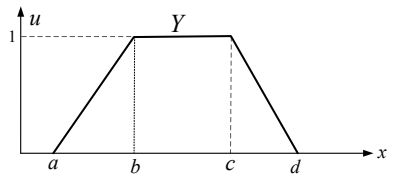

Fig. 2. A trapezoidal T1 FS, determined by four parameters (a, b, c, d).

Assume the output of the CWW Engine is a trapezoidal T1 FS[1] Y, which is represented by four parameters (a, b, c, d). Assume also the codebook consists of N words, which have already been sorted in ascending order using the centroid based ranking method [26]. The trapezoidal T1 FS model for the n^{th} word is W_n, which is represented by four parameters (a_n, b_n, c_n, d_n) and whose centroid is w_n, $n = 1, 2, ..., N$. The Reconstruction decoder tries to find a combination of two successive codebook words to represent Y with minimum information loss, i.e.,

$$Y \approx W \tag{1}$$

where

$$W = \alpha W_{n'} + \beta W_{n'+1}. \tag{2}$$

To determine n', we first compute the centroid of Y, y, and then identify the n' such that

$$w_{n'} \leq y \leq w_{n'+1}. \tag{3}$$

Essentially, this means that we rank $\{W_n\}$ and Y together and then select the two words immediately before and after[2] Y.

The next problem is how to determine the coefficient α and β so that there is minimum information loss in representing Y as W. There can be different definitions of minimum information loss, e.g.,

1) The similarity between Y and W is maximized. This definition is very intuitive, as the more similar Y and W are, the less information loss there is when we represent Y by W.
2) The mean-squared error between the four parameters of Y and W is minimized. This definition is again very intuitive, as generally a smaller mean-squared error means a larger similarity between Y and W, and hence less information loss.

However, one problem with the second approach is that it is difficult to find a set of parameters to define T1 FSs with arbitrary shapes (e.g., not necessarily trapezoidal or Gaussian). On the other hand, the Jaccard similarity measure [26] can work for any T1 FSs. So, in this paper we use the first definition.

Before computing the similarity between Y and W, we first need to compute $W = \alpha W_{n'} + \beta W_{n'+1}$. Because both $W_{n'}$ and $W_{n'+1}$ are normal trapezoidal T1 FSs, W is also a normal trapezoidal T1 FS; so, it can also be represented

[1]Strictly speaking, when trapezoidal T1 FSs are used in the CWW engine, e.g., the novel weighted averages [16], [28] or Perceptual Reasoning [16], [27], the output T1 FS Y is not perfectly trapezoidal, i.e., its waists are slightly curved instead of straight; however, the waists can be approximated by straight lines with very high accuracy. Moreover, as we will see at the end of this section, our method can be applied to FSs with any shape. So, trapezoidal Y is used in the derivation for simplicity.

[2]There may be a concern that Y is smaller than W_1 or larger than W_N so that we cannot find a n' satisfying (3); however, this cannot occur in the Per-C if the encoder and the decoder use the same vocabulary and the novel weighted average [16], [28] or Perceptual Reasoning [16], [27] is used, because both CWW engines are some kind of weighted average, and it is well-known that the output of a weighted average cannot be smaller than the smallest input and cannot be larger than the largest input either.

by four parameters (a_w, b_w, c_w, d_w). Based on the Extension Principle [32] and the α-cut Representation Theorem [8], we have

$$a_w = \alpha a_{n'} + \beta a_{n'+1} \tag{4}$$
$$b_w = \alpha b_{n'} + \beta b_{n'+1} \tag{5}$$
$$c_w = \alpha c_{n'} + \beta c_{n'+1} \tag{6}$$
$$d_w = \alpha d_{n'} + \beta d_{n'+1} \tag{7}$$

To solve for α and β, we consider a constrained optimization problem, i.e.,

$$\begin{aligned} & \arg\max_{\alpha,\beta} s(Y, W) \\ s.t. \quad & \alpha \geq 0,\ \beta \geq 0 \\ & \alpha + \beta = 1 \end{aligned} \tag{8}$$

where

$$s(Y, W) = \frac{\sum_{i=1}^{I} \min(\mu_Y(x_i), \mu_W(x_i))}{\sum_{i=1}^{I} \max(\mu_Y(x_i), \mu_W(x_i))} \tag{9}$$

is the Jaccard similarity measure between Y and W.

Observe that we have some constraints in (8), which are derived from the analogy to the case for crisp numbers. For example, the decimal 4.4, which lies between two successive integers 4 and 5, can be represented as $4.4 = \alpha \cdot 4 + \beta \cdot 5$, where $\alpha = 0.6$, $\beta = 0.4$, and $\alpha + \beta = 1$. In (8) we treat $W_{n'}$ and $W_{n'+1}$ as two successive "integers" and Y as a "decimal," and we want to represent this "decimal" using the two "integers."

In summary, the procedure for the Reconstruction decoder for T1 FS word models is:

1) Compute w_n, the centroid of W_n, $n = 1, ..., N$, and rank $\{W_n\}$ in ascending order. This step only needs to be performed once, and it can be done offline.
2) Compute y, the centroid of Y.
3) Identify n' according to (3).
4) Solve the constrained optimization problem in (8) for α and β.
5) Represent the decoding output as $Y \approx \alpha W_{n'} + \beta W_{n'+1}$.

B. The Reconstruction Decoder for IT2 FS Word Models

In this paper a normal trapezoidal IT2 FS is represented by nine parameters shown in Fig. 3. Note that we use four parameters for the normal trapezoidal upper membership function (UMF), similar to the T1 FS case; however, we need five parameters for the trapezoidal lower membership function (LMF) since usually it is subnormal and hence we need a fifth parameter to specify its height.

Assume the output of the CWW Engine is a trapezoidal IT2 FS[3] $\widetilde{Y}$, which is represented by nine parameters $(a, b, c, d, e, f, g, i, h)$. Assume also the codebook consists of N words, which have already been sorted in ascending order using the centroid based ranking method [26]. The IT2 FS for the n^{th} word is $\widetilde{W}_n$, which is represented by

[3]Similar to the discussions in Footnote 1, here trapezoidal IT2 FSs are used for simplicity. Our method can be applied to IT2 FSs with arbitrary FOUs.

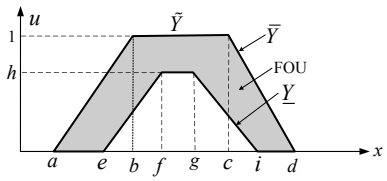

Fig. 3. A normal trapezoidal IT2 FS. (a, b, c, d) determines a normal trapezoidal UMF, and (e, f, g, i, h) determines a trapezoidal LMF with height h.

$(a_n, b_n, c_n, d_n, e_n, f_n, g_n, i_n, h_n)$ and whose center of centroid is w_n, $n = 1, 2, ..., N$. The Reconstruction decoder again tries to find a combination of two successive codebook words to represent $\widetilde{Y}$ with minimum information loss.

Similar to the T1 FS case, we first compute the center of centroid of $\widetilde{Y}$, y, and then identify the n' such that

$$w_{n'} \leq y \leq w_{n'+1}. \tag{10}$$

We then solve the following constrained optimization problem for α and β:

$$\begin{aligned} & \arg\max_{\alpha,\beta} s(\widetilde{Y}, \widetilde{W}) \\ s.t. \quad & \alpha \geq 0,\ \beta \geq 0 \\ & \alpha + \beta = 1 \end{aligned} \tag{11}$$

where

$$\widetilde{W} = \alpha \widetilde{W}_{n'} + \beta \widetilde{W}_{n'+1}. \tag{12}$$

and $s(\widetilde{Y}, \widetilde{W})$ is the Jaccard similarity measure between $\widetilde{Y}$ and $\widetilde{W}$, shown in (13) on top of the next page.

Clearly, to solve (11), we need to be able to numerically represent $\widetilde{W}$ in (12). Assume $\widetilde{W}$ is represented by nine parameters $(a_w, b_w, c_w, d_w, e_w, f_w, g_w, i_w, h_w)$. We then compute the UMF and LMF of $\widetilde{W}$ separately. The UMF computation is very simple. Because the UMFs of both $\widetilde{W}_{n'}$ and $\widetilde{W}_{n'+1}$ are normal, similar to the T1 FS case, we have

$$a_w = \alpha a_{n'} + \beta a_{n'+1} \tag{14}$$
$$b_w = \alpha b_{n'} + \beta b_{n'+1} \tag{15}$$
$$c_w = \alpha c_{n'} + \beta c_{n'+1} \tag{16}$$
$$d_w = \alpha d_{n'} + \beta d_{n'+1} \tag{17}$$

However, the computation of the LMF of $\widetilde{W}$ is not so straightforward, because generally the LMFs of $\widetilde{W}_{n'}$ and $\widetilde{W}_{n'+1}$ have different heights, i.e., $h_{n'} \neq h_{n'+1}$. Based on the Extension Principle, the height of the LMF of $\widetilde{W}$ should be equal to the smaller one of $h_{n'}$ and $h_{n'+1}$ (this fact has also been used in deriving the linguistic weighted averages [24], [25]). Without loss of generality, assume $h_{n'} \leq h_{n'+1}$. We then crop the top of $\underline{W}_{n'+1}$ to make it the same height as $\underline{W}_{n'}$, as shown in Fig. 4. Representing the cropped version of $\underline{W}_{n'+1}$ as $(e_{n'+1}, f'_{n'+1}, g'_{n'+1}, i_{n'+1}, h_{n'})$, the LMF of $\widetilde{W}$ is then

$$s(\widetilde{Y},\widetilde{W}) = \frac{\sum_{i=1}^{I}\min(\mu_{\overline{Y}}(x_i),\mu_{\overline{W}}(x_i)) + \sum_{i=1}^{I}\min(\mu_{\underline{Y}}(x_i),\mu_{\underline{W}}(x_i))}{\sum_{i=1}^{I}\max(\mu_{\overline{Y}}(x_i),\mu_{\overline{W}}(x_i)) + \sum_{i=1}^{I}\max(\mu_{\underline{Y}}(x_i),\mu_{\underline{W}}(x_i))} \tag{13}$$

computed as:

$$e_w = \alpha e_{n'} + \beta e_{n'+1} \tag{18}$$
$$f_w = \alpha f_{n'} + \beta f'_{n'+1} \tag{19}$$
$$g_w = \alpha g_{n'} + \beta g'_{n'+1} \tag{20}$$
$$i_w = \alpha i_{n'} + \beta i_{n'+1} \tag{21}$$
$$h_w = \min(h_{n'}, h_{n'+1}) \tag{22}$$

In fact, if we represent $\widetilde{W}$ as

$$\widetilde{W} = \frac{\alpha\widetilde{W}_{n'} + \beta\widetilde{W}_{n'+1}}{\alpha+\beta} \cdot (\alpha+\beta) \tag{23}$$

and compute the first term on the right hand side as a special linguistic weighted average [24], [25], we can get the same result as presented above.

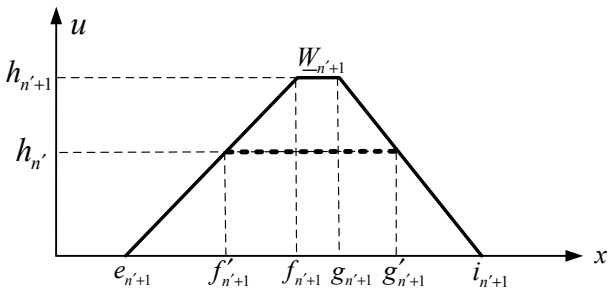

Fig. 4. Illustration of how $\underline{W}_{n'+1}$ is cropped to have height $h_{n'}$.

In summary, the procedure for the Reconstruction decoder for IT2 FS word models is:

1) Compute w_n, the centers of centroid of $\widetilde{W}_n$, $n = 1, ..., N$, and rank $\{\widetilde{W}_n\}$ in ascending order. This step only needs to be performed once, and it can be done offline.
2) Compute y, the center of centroid of $\widetilde{Y}$.
3) Identify n' according to (10).
4) Solve the constrained optimization problem in (11) for α and β.
5) Represent the decoding output as $\widetilde{Y} \approx \alpha\widetilde{W}_{n'} + \beta\widetilde{W}_{n'+1}$.

C. The Reconstruction Decoder for Arbitrary FS Shapes

We have explained the Reconstruction decoder for normal trapezoidal T1 and IT2 FS word models. Our method can also be applied to T1 and IT2 FSs with arbitrary shapes. The procedure is essentially the same. The only difference is in computing W or $\widetilde{W}$. Take W as an example. If $W_{n'}$ and $W_{n'+1}$ have different heights, then the method for computing $\underline{W}$ in the previous subsection can be used for computing W, i.e., we fist crop the higher T1 FS to make it the same height as the lower one. If $W_{n'}$ and $W_{n'+1}$ are not trapezoidal, then W cannot be represented using only four parameters; however, it can still be computed using the α-cut Decomposition Theorem [8], [24], [25], one α-cut at a time.

III. Experimental Results

Experimental results on verifying the performance of the Reconstruction decoder are presented in this section. We consider T1 FS and IT2 FS cases separately.

A. T1 FS Case

In [29] we computed the FOUs of 32 IT2 FSs using the Enhanced Interval Approach. The UMFs of these 32 IT2 FSs, shown as the black solid curves in Fig. 5, are used in this experiment as our T1 FSs. They have been sorted in ascending order according to their centroids. In the experiment, we select a T1 FS (W_n) from these 32 words as Y, the output of the CWW engine, and use the remaining 31 words as our codebook. The Reconstruction decoder is then used to represent Y as a combination of two IT2 FSs in the codebook. Note that once a W_n is selected as Y, it is excluded from the codebook because otherwise Y would be mapped to W_n directly, which is not interesting and usually does not happen in practice. By excluding W_n in the codebook we force the Reconstruction decoder to represent W_n as a combination of W_{n-1} and W_{n+1} and hence we can test the performance of the Reconstruction decoder. Note also that W_1 and W_{32} are not selected as Y because, as explained in Footnote 2, if the Encoder and the Decoder use the same codebook, then the CWW engine output can never be smaller than the smallest word in the codebook, and also can never be larger than the largest word in the codebook. If we select W_1 as Y and use $W_2 - W_{32}$ as the codebook, then Y is smaller than all words in the codebook, which cannot happen in practice. So, we only repeat the experiment for W_n, $n = 2, 3, ..., 31$.

The reconstructed W are shown in Fig. 5 as the red dashed curves. Observe that most of them are almost identical to the original W_n. The Jaccard similarities between Y and W are shown in the second column of Table I. Observe that 16 of the 30 similarities are larger than or equal to 0.95, 25 are larger than or equal to 0.9, and all 30 similarities are larger than 0.65. The corresponding α and β for constructing W are given in the third part of Table I, and the Jaccard similarities between Y and W_{n-1} and W_{n+1} are shown in the fourth part of Table I. It is interesting to examine whether the Reconstruction decoder preserves the order of similarity, i.e., if $s(Y, W_{n-1}) > s(Y, W_{n+1})$, then we would expect that $\alpha > \beta$ and vice versa. We call this property *consistency*. The inconsistencies words are marked in bold in Table I. Observe that only two of the 30 words have an inconsistency.

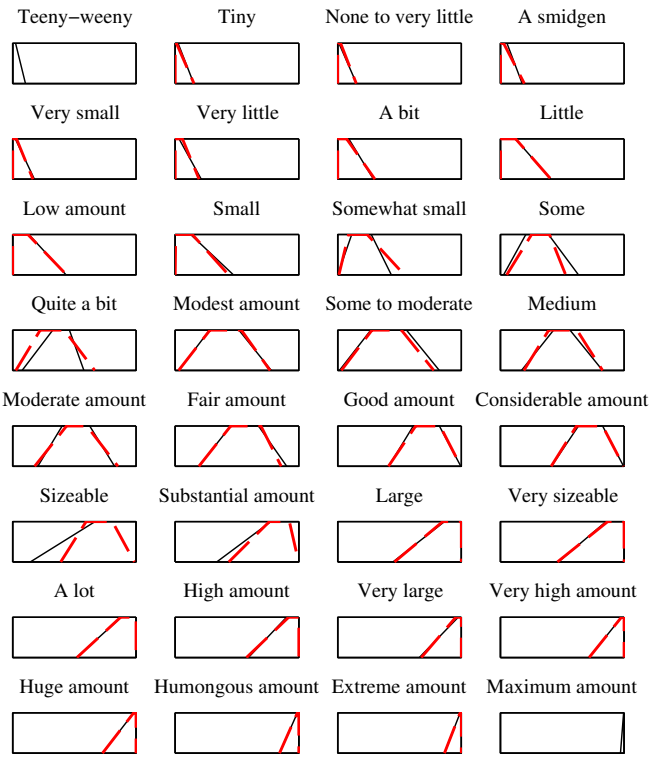

Fig. 5. Reconstruction results for T1 FS models. The black solid curves show the original T1 FSs, and the red dashed curves show the reconstructed T1 FSs.

TABLE I
EXPERIMENTAL RESULTS FOR T1 FS WORD MODELS. FOR EACH ROW, $Y = W_n$ AND $W = \alpha W_{n-1} + \beta W_{n+1}$.

Y	$s(Y,W)$	α	β	$s(Y,W_{n-1})$	$s(Y,W_{n+1})$
W_2	0.93	0	1	0.73	0.93
W_3	0.93	1	0	0.93	0.72
W_4	0.90	**0.53**	0.47	0.72	**0.87**
W_5	0.94	0.50	0.50	0.87	0.82
W_6	0.91	0.83	0.17	0.82	0.64
W_7	0.96	**0.54**	0.46	0.64	**0.71**
W_8	1	0.17	0.83	0.71	0.94
W_9	0.99	0.57	0.43	0.94	0.94
W_{10}	0.94	0.95	0.05	0.94	0.67
W_{11}	0.84	0.57	0.43	0.67	0.52
W_{12}	0.79	0.34	0.66	0.52	0.72
W_{13}	0.73	0.76	0.24	0.72	0.64
W_{14}	0.98	0.22	0.78	0.64	0.90
W_{15}	0.90	1	0	0.90	0.70
W_{16}	0.90	0.07	0.93	0.70	0.89
W_{17}	0.94	0.65	0.35	0.89	0.83
W_{18}	0.96	0.82	0.18	0.83	0.47
W_{19}	0.97	0.01	0.99	0.47	0.97
W_{20}	0.98	0.98	0.02	0.97	0.65
W_{21}	0.66	0.10	0.90	0.65	0.75
W_{22}	0.82	0.43	0.57	0.75	0.77
W_{23}	0.98	0.06	0.94	0.77	0.98
W_{24}	0.98	0.82	0.18	0.98	0.89
W_{25}	0.99	0.57	0.43	0.89	0.84
W_{26}	0.97	0.59	0.41	0.84	0.71
W_{27}	0.95	0.32	0.68	0.71	0.84
W_{28}	1	0.17	0.83	0.84	0.96
W_{29}	1	0.91	0.09	0.96	0.57
W_{30}	1	0.15	0.85	0.57	0.87
W_{31}	1	0.84	0.16	0.87	0.21

B. IT2 FS Case

The 32 FOUs in [29] are again used in the experiment. The UMFs and LMFs are shown as the black solid curves in Fig. 6. The experiment setup is very similar to that in the previous subsection, except that here we use the IT2 FSs instead of only the UMFs. The reconstructed $\widetilde{W}$ are shown in Fig. 6 as the red dashed curves. Observe again that most of them are almost identical to the original $\widetilde{W}_n$. The Jaccard similarities between $\widetilde{Y}$ and $\widetilde{W}$ are shown in the second column of Table II. Observe that 12 of the 30 similarities are larger than or equal to 0.95, 18 are larger than or equal to 0.9, 26 are larger than or equal to 0.8, and all 30 similarities are larger than 0.6. The corresponding α and β for constructing W are given in the third part of Table II, and the Jaccard similarities between $\widetilde{Y}$ and $\widetilde{W}_{n-1}$ and $\widetilde{W}_{n+1}$ are shown in the fourth part of Table II. The inconsistency is marked in bold in Table II. Observe that only one of the 30 words has an inconsistency.

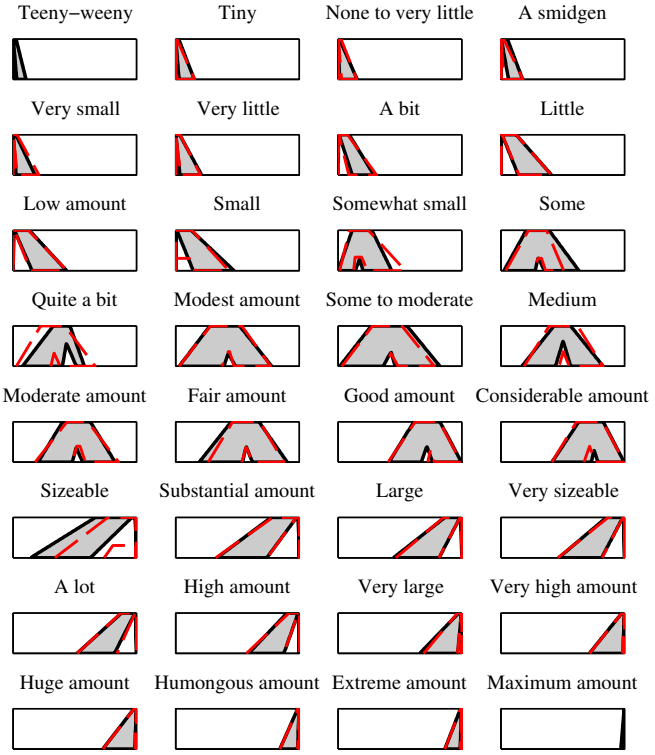

Fig. 6. Reconstruction results for IT2 FS models. The black solid curves show the original IT2 FSs, and the red dashed curves show the reconstructed IT2 FSs.

C. Discussions

The Reconstruction decoder has the following advantages, evidenced from its derivation and the experimental results:

1) *The Reconstruction decoder is a generalized Word decoder.* Take the T1 FS case for example. If $\alpha > \beta$, then the output $Y = \alpha W_{n'} + \beta W_{n'+1}$ reads "Y *is a word between* $W_{n'}$ *and* $W_{n'+1}$*, and it is closer to* $W_{n'}$." Similarly, if $\alpha < \beta$, then the output $Y = \alpha W_{n'} + \beta W_{n'+1}$ reads "Y *is a word between* $W_{n'}$ *and* $W_{n'+1}$*, and it*

TABLE II
EXPERIMENTAL RESULTS FOR IT2 FS WORD MODELS. FOR EACH ROW, $\widetilde{Y} = \widetilde{W}_n$ AND $\widetilde{W} = \alpha\widetilde{W}_{n-1} + \beta\widetilde{W}_{n+1}$.

$\widetilde{Y}$	$s(\widetilde{Y}, \widetilde{W})$	α	β	$s(\widetilde{Y}, \widetilde{W}_{n-1})$	$s(\widetilde{Y}, \widetilde{W}_{n+1})$
$\widetilde{W}_2$	0.86	0.02	0.98	0.76	0.86
$\widetilde{W}_3$	0.86	1	0	0.86	0.60
$\widetilde{W}_4$	0.76	0	1	0.60	0.76
$\widetilde{W}_5$	0.80	0	1	0.76	0.80
$\widetilde{W}_6$	0.94	0.79	0.21	0.80	0.55
$\widetilde{W}_7$	0.91	0.50	0.50	0.55	0.73
$\widetilde{W}_8$	0.99	0.17	0.83	0.73	0.93
$\widetilde{W}_9$	0.97	**0.57**	0.43	0.93	**0.94**
$\widetilde{W}_{10}$	0.85	0.98	0.02	0.94	0.54
$\widetilde{W}_{11}$	0.80	0.55	0.45	0.54	0.50
$\widetilde{W}_{12}$	0.78	0.48	0.52	0.50	0.65
$\widetilde{W}_{13}$	0.66	0.76	0.24	0.65	0.63
$\widetilde{W}_{14}$	0.97	0.22	0.78	0.63	0.89
$\widetilde{W}_{15}$	0.89	1	0	0.89	0.64
$\widetilde{W}_{16}$	0.86	0.08	0.92	0.64	0.85
$\widetilde{W}_{17}$	0.91	0.62	0.38	0.85	0.78
$\widetilde{W}_{18}$	0.89	0.71	0.29	0.78	0.44
$\widetilde{W}_{19}$	0.94	0.04	0.96	0.44	0.93
$\widetilde{W}_{20}$	0.93	1	0	0.93	0.50
$\widetilde{W}_{21}$	0.64	0.04	0.96	0.50	0.73
$\widetilde{W}_{22}$	0.99	0.41	0.59	0.73	0.75
$\widetilde{W}_{23}$	0.98	0.01	0.99	0.75	0.98
$\widetilde{W}_{24}$	0.98	0.83	0.17	0.98	0.90
$\widetilde{W}_{25}$	0.97	0.58	0.42	0.90	0.80
$\widetilde{W}_{26}$	0.97	0.62	0.38	0.80	0.62
$\widetilde{W}_{27}$	0.93	0.19	0.81	0.62	0.79
$\widetilde{W}_{28}$	0.99	0.17	0.83	0.79	0.96
$\widetilde{W}_{29}$	1	0.91	0.09	0.96	0.59
$\widetilde{W}_{30}$	1	0.15	0.85	0.59	0.88
$\widetilde{W}_{31}$	1	0.84	0.16	0.88	0.27

is closer to $W_{n'+1}$." If we want to represent Y by a single word, then it is safe to choose $W_{n'}$ if $\alpha > \beta$, or $W_{n'+1}$ if $\alpha < \beta$, because we have shown through experiments that this is almost always consistent with the Word decoder.

2) *The Reconstruction decoder is implicitly a Rank decoder.* Again take the T1 FS case for example. If we know that $Y_1 = \alpha_1 W_{n'} + \beta_1 W_{n'+1}$, $Y_2 = \alpha_2 W_{m'} + \beta_2 W_{m'+1}$, and $n' < m'$, regardless of the values of α_1, β_1, α_2 and β_2, it must be true that $Y_1 \leq Y_2$ because $Y_1 \leq W_{n'+1} \leq W_{m'} \leq Y_2$.
3) *The Reconstruction decoder preserves the shape information of the CWW engine output in a simple form with minimum information loss.* So, if we want to use Y or $\widetilde{Y}$ in future computations, we can always approximate it by W or $\widetilde{W}$ without sacrificing much accuracy. As an evidence, observe from the second column of Tables I and II that the similarities between the original FSs and the reconstructed FSs are very close to 1. Additionally, replacing Y by W (or $\widetilde{Y}$ by $\widetilde{W}$) is almost always better than replacing Y (or $\widetilde{Y}$) by the word suggested by the Word decoder, because in Table I we almost always have $s(Y, W) \geq s(Y, W_{n-1})$ and $s(Y, W) \geq s(Y, W_{n+1})$, and in Table II we almost always have $s(\widetilde{Y}, \widetilde{W}) \geq s(\widetilde{Y}, \widetilde{W}_{n-1})$ and $s(\widetilde{Y}, \widetilde{W}) \geq s(\widetilde{Y}, \widetilde{W}_{n+1})$.

IV. CONCLUSIONS

The Word decoder is a very important approach for decoding in the Per-C. It maps the CWW engine output into a word in a codebook so that it can be understood. However, it suffers from significant information loss, i.e., the FS of the mapped word may be quite different from the FS output by the CWW engine, especially when the codebook is small. In this paper we have proposed a Reconstruction decoder for the Per-C, which represents the CWW engine output as a combination of two successive codebook words with minimum information loss by solving a constrained optimization problem. The Reconstruction decoder can be viewed as a generalized Word decoder and it is also implicitly a Rank decoder. Moreover, it preserves the shape information of the CWW engine output in a simple form without sacrificing much accuracy. Experimental results verified the effectiveness of our proposed method. We also give the Matlab implementation of the Reconstruction decoder in the Appendix.

Our future research includes studying the relationship between the Reconstruction decoder and the 2-tuple representation based models.

APPENDIX A
MATLAB IMPLEMENTATION OF THE RECONSTRUCTION DECODER

Matlab implementation of the Reconstruction decoder is given in this Appendix. It can be used for both T1 and IT2 normal trapezoidal FSs.

The input parameters are:

- **Y**, which is a matrix containing Y or $\widetilde{Y}$. Each row is a separate Y or $\widetilde{Y}$. Each Y is represented by four parameters (a, b, c, d) in Fig. 2, and each $\widetilde{Y}$ is represented by nine parameters $(a, b, c, d, e, f, g, i, h)$ in Fig. 3.
- **CB**, which is a matrix containing the codebook. Each row is a separate W_n or $\widetilde{W}_n$. Each W_n is represented by four parameters, and each $\widetilde{W}_n$ is represented by nine parameters.
- **Cs**, which is a vector containing the centroids of W_n, or the centers of centroid of $\widetilde{W}_n$. **Cs** will be computed automatically if it is not provided or if it is empty.

The output parameters are:

1) **alpha**, which is the α in the paper. Note that $\beta = 1 - \alpha$.
2) **W**, which is a matrix containing the reconstructed FSs. It has the same number of rows as the input **Y**. Each row is a separate W or $\widetilde{W}$. Each W is represented by four parameters, and each $\widetilde{W}$ is represented by nine parameters.

```
function [alpha,W]=reconstruction1(Y,CB,Cs)

M=size(Y,1);  W=zeros(M,size(Y,2));
if nargin==2 || isempty(Cs)
    Cs=centroid(CB);
end
```

```
[Cs,index]=sort(Cs);
CB=CB(index,:);  alpha=nan(M,1);
switch size(Y,2)
    case 4 % T1 FSs
        for m=1:M
            c=centroid(Y(m,:));
            i=find(Cs<c,1,'last');
            MFl=CB(i,:); MFr=CB(i+1,:);
            A=[-1; 1]; b=[0; 1];
            f=@(x)myfun1(x,MFl,MFr,Y(m,:));
            alpha(m)=fmincon(f,.5,A,b);
            W(m,:)=MFl*alpha(m)+...
                     MFr*(1-alpha(m));
        end
    case 9 % IT2 FSs
        for m=1:M
            c=centroid(Y(m,:));
            i=find(Cs<c,1,'last');
            [h,index]=min([CB(i,9),CB(i+1,9)]);
            MFl=CB(i,:); MFr=CB(i+1,:);
            MFl(9)=h; MFr(9)=h;
            switch index
                case 1
                    MFr(6)=CB(i+1,5)+...
                  h/CB(i+1,9)*(CB(i+1,6)...
                  -CB(i+1,5));
                    MFr(7)=CB(i+1,8)-...
                   h/CB(i+1,9)*(CB(i+1,8)...
                   -CB(i+1,7));
                case 2
                    MFl(6)=CB(i,5)+h/...
                  CB(i,9)*(CB(i,6)-CB(i,5));
                    MFl(7)=CB(i,8)-h/...
                  CB(i,9)*(CB(i,8)-CB(i,7));
            end
            A=[-1; 1]; b=[0; 1];
            f=@(x)myfun2(x,MFl,MFr,Y(m,:));
            alpha(m)=fmincon(f,.5,A,b);
            W(m,:)=[MFl(1:8)*alpha(m)+...
                MFr(1:8)*(1-alpha(m)) h];
        end
end


function f=myfun1(x,MFl,MFr,Y)

% Objective function for T1 FS model

f=-Jaccard(Y,x(1)*MFl+(1-x(1))*MFr);


function f=myfun2(x,MFl,MFr,Y)

% Objective function for IT2 FS model

f=-Jaccard(Y,[x(1)*MFl(1:8)+(1-x(1))*...
          MFr(1:8) MFl(9)]);


function CA=centroid(A)

% Compute the centroid of a T1 or IT2 FS

CA=zeros(size(A,1),1);
for i=1:size(A,1)
    Xs=linspace(A(i,1),A(i,4),100);
    UMF=mg(Xs,A(i,1:4),[0 1 1 0]);
    switch size(A,2)
        case 4
            CA(i)=Xs*UMF'/sum(UMF);
        case 9
            LMF=mg(Xs,A(i,5:8),...
                   [0 A(i,9) A(i,9) 0]);
            CA(i)=EIASC(Xs,Xs,LMF,UMF,0);
    end
end


function S=Jaccard(A,B)

% The Jaccard similarity measure

minX=min(A(1),B(1)); maxX=max(A(4),B(4));
X=linspace(minX,maxX,100);
upperA=mg(X,A(1:4)); upperB=mg(X,B(1:4));
if length(A)==4
    S=sum(min([upperA;upperB]))/...
      sum(max([upperA;upperB]));
else
    lowerA=mg(X,A(5:8),[0 A(9) A(9) 0]);
    lowerB=mg(X,B(5:8),[0 B(9) B(9) 0]);
    S=sum([min([upperA;upperB]),...
     min([lowerA;lowerB])])/...
     sum([max([upperA;upperB]),...
     max([lowerA;lowerB])]);
end


function [y,yl,yr,l,r]=EIASC(Xl,Xr,Wl,Wr,needSort)

% Implements the EIASC type-reduction
% algorithm proposed in:
% D. Wu and M. Nie, "Comparison and Practical
% Implementation of Type-Reduction Algorithms
% for Type-2 Fuzzy Sets and Systems," IEEE
% International Conference on Fuzzy Systems,
% Taipei, Taiwan, June 2011.

ly=length(Xl); XrEmpty=isempty(Xr);
if XrEmpty;  Xr=Xl; end
if max(Wl)==0
    yl=min(Xl); yr=max(Xr);
    y=(yl+yr)/2;  l=1; r=ly-1; return;
end
if nargin==4;  needSort=1; end
% Compute yl
if  needSort
    [Xl,index]=sort(Xl); Xr=Xr(index);
    Wl=Wl(index); Wr=Wr(index);
    Wl2=Wl; Wr2=Wr;
end
if ly==1
    yl=Xl;  l=1;
else
    yl=Xl(end); l=0;
    a=Xl*Wl'; b=sum(Wl);
    while l<ly && yl > Xl(l+1)
        l=l+1;
        t=Wr(l)-Wl(l);
        a=a+Xl(l)*t;
        b=b+t;
        yl=a/b;
    end
end
```

```
% Compute yr
if ~XrEmpty && needSort==1
    [Xr,index]=sort(Xr);
    Wl=Wl2(index); Wr=Wr2(index);
end
if ly==1
    yr=Xr; r=1;
else
    r=ly; yr=Xr(1);
    a=Xr*Wl'; b=sum(Wl);
    while r>0 && yr < Xr(r)
        t=Wr(r)-Wl(r);
        a=a+Xr(r)*t;
        b=b+t;
        yr=a/b;  r=r-1;
    end
end
y=(yl+yr)/2;


function u=mg(x,xMF,uMF)

% Compute the membership grade of x on a
% T1 FS represented by xMF and uMF, where
% xMF are samples in the x domain and uMF
% are the corresponding membership grades

if nargin==2;  uMF=[0 1 1 0];  end
[xMF,index]=sort(xMF);
uMF=uMF(index); u=zeros(size(x));
for i=1:length(x)
    if x(i)<=xMF(1) || x(i)>=xMF(end)
        u(i)=0;
    else
        left=find(xMF<x(i),1,'last');
        right=left+1;
        u(i)=uMF(left)+(uMF(right)...
        -uMF(left))*(x(i)-xMF(left))...
        /(xMF(right)-xMF(left));
    end
end
```

REFERENCES

[1] P. Bonissone and K. Decker, "Selecting uncertainty calculi and granularity: an experiment in tradeing-off precision and complexity," in *Uncertainty in Artificial Intelligence*, L. Kanal and J. Lemmer, Eds. Amsterdam, The Netherlands: North-Holland, 1986, pp. 217–247.

[2] C. Carlsson and R. Fuller, "Benchmarking and linguistic importance weighted aggregations," *Fuzzy sets and systems*, vol. 114, no. 1, pp. 35–42, 2000.

[3] R. Degani and G. Bortolan, "The problem of linguistic approximation in clinical decision making," *International Journalof Approximate Reasoning*, no. 2, pp. 143–162, 1988.

[4] M. Delgado, J. L. Verdegay, and M. A. Vila, "On aggregation operations of linguistic labels," *International Journal of Intelligent Systems*, vol. 8, pp. 351–370, 1993.

[5] F. Herrera and L. Martinez, "A model based on linguistic 2-tuples for dealing with multigranular hierarchical linguistic contexts in multi-expert decision-making," *IEEE Trans. on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 31, no. 2, pp. 227–234, 2001.

[6] F. Herrera and L. Martinez, "A 2-tuple fuzzy linguistic representation model for computing with words," *IEEE Trans. on Fuzzy Systems*, vol. 8, no. 6, pp. 746–752, 2000.

[7] J. Kacprzyk and S. Zadrożny, "Computing with words in intelligent database querying: Standalone and internet-based applications," *Information Sciences*, vol. 34, pp. 71–109, 2001.

[8] G. J. Klir and B. Yuan, *Fuzzy Sets and Fuzzy Logic: Theory and Applications*. Upper Saddle River, NJ: Prentice-Hall, 1995.

[9] J. Lawry, "A methodology for computing with words," *International Journal of Approximate Reasoning*, vol. 28, pp. 51–89, 2001.

[10] F. Liu and J. M. Mendel, "Encoding words into interval type-2 fuzzy sets using an Interval Approach," *IEEE Trans. on Fuzzy Systems*, vol. 16, no. 6, pp. 1503–1521, 2008.

[11] J. M. Mendel, "The perceptual computer: An architecture for computing with words," in *Proc. IEEE Int'l Conf. on Fuzzy Systems*, Melbourne, Australia, December 2001, pp. 35–38.

[12] J. M. Mendel, *Uncertain Rule-Based Fuzzy Logic Systems: Introduction and New Directions*. Upper Saddle River, NJ: Prentice-Hall, 2001.

[13] J. M. Mendel, "An architecture for making judgments using computing with words," *International Journal of Applied Mathematics and Computer Science*, vol. 12, no. 3, pp. 325–335, 2002.

[14] J. M. Mendel, "Computing with words: Zadeh, Turing, Popper and Occam," *IEEE Computational Intelligence Magazine*, vol. 2, pp. 10–17, 2007.

[15] J. M. Mendel and D. Wu, "Computing with words for hierarchical and distributed decision making," in *Computational Intelligence in Complex Decision Systems*, D. Ruan, Ed. Paris, France: Atlantis Press, 2010.

[16] J. M. Mendel and D. Wu, *Perceptual Computing: Aiding People in Making Subjective Judgments*. Hoboken, NJ: Wiley-IEEE Press, 2010.

[17] S. K. Pal, L. Polkowski, and A. Skowron, Eds., *Rough-neural Computing: Techniques for Computing with Words*. Heidelberg, Germany: Springer-Verlag, 2003.

[18] S. H. Rubin, "Computing with words," *IEEE Trans. on Systems, Man, and Cybernetics–B*, vol. 29, no. 4, pp. 518 – 524, 1999.

[19] I. Vlachos and G. Sergiadis, "Subsethood, entropy, and cardinality for interval-valued fuzzy sets – an algebraic derivation," *Fuzzy Sets and Systems*, vol. 158, pp. 1384–1396, 2007.

[20] H. Wang and D. Qiu, "Computing with words via Turing machines: A formal approach," *IEEE Trans. on Fuzzy Systems*, vol. 11, no. 6, pp. 742–753, 2003.

[21] J.-H. Wang and J. Hao, "A new version of 2-tuple fuzzy linguistic representation model for computing with words," *IEEE Trans. on Fuzzy Systems*, vol. 14, no. 3, pp. 435–445, 2006.

[22] P. Wang, Ed., *Computing With Words*. New York: John Wiley & Sons, 2001.

[23] D. Wu, "Intelligent systems for decision support," Ph.D. dissertation, University of Southern California, Los Angeles, CA, May 2009.

[24] D. Wu and J. M. Mendel, "Aggregation using the linguistic weighted average and interval type-2 fuzzy sets," *IEEE Trans. on Fuzzy Systems*, vol. 15, no. 6, pp. 1145–1161, 2007.

[25] D. Wu and J. M. Mendel, "Corrections to 'Aggregation using the linguistic weighted average and interval type-2 fuzzy sets'," *IEEE Trans. on Fuzzy Systems*, vol. 16, no. 6, pp. 1664–1666, 2008.

[26] D. Wu and J. M. Mendel, "A comparative study of ranking methods, similarity measures and uncertainty measures for interval type-2 fuzzy sets," *Information Sciences*, vol. 179, no. 8, pp. 1169–1192, 2009.

[27] D. Wu and J. M. Mendel, "Perceptual reasoning for perceptual computing: A similarity-based approach," *IEEE Trans. on Fuzzy Systems*, vol. 17, no. 6, pp. 1397–1411, 2009.

[28] D. Wu and J. M. Mendel, "Computing with words for hierarchical decision making applied to evaluating a weapon system," *IEEE Trans. on Fuzzy Systems*, vol. 18, no. 3, pp. 441–460, 2010.

[29] D. Wu, J. M. Mendel, and S. Coupland, "Enhanced Interval Approach for encoding words into interval type-2 fuzzy sets and its convergence analysis," *IEEE Trans. on Fuzzy Systems*, 2012, in press.

[30] R. Yager, "Aproximate reasoning as a basis for computing with words," in *Computing With Words in Information/ Intelligent Systems 1: Foundations*, L. A. Zadeh and J. Kacprzyk, Eds. Heidelberg: Physica-Verlag, 1999, pp. 50–77.

[31] M. Ying, "A formal model of computing with words," *IEEE Trans. on Fuzzy Systems*, vol. 10, no. 5, pp. 640–652, 2002.

[32] L. A. Zadeh, "Fuzzy sets," *Information and Control*, vol. 8, pp. 338–353, 1965.

[33] L. A. Zadeh, "Fuzzy logic = Computing with words," *IEEE Trans. on Fuzzy Systems*, vol. 4, pp. 103–111, 1996.

[34] L. A. Zadeh, "From computing with numbers to computing with words – From manipulation of measurements to manipulation of perceptions," *IEEE Trans. on Circuits and Systems I*, vol. 46, no. 1, pp. 105–119, 1999.

[35] L. A. Zadeh and J. Kacprzyk, Eds., *Computing with Words in Information/Intelligent Systems: 1. Foundations, 2. Applications*. Heidelberg: Physica-Verlag, 1999.