Information Sciences 255 (2014) 1–15

Contents lists available at ScienceDirect

Information Sciences

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

A reconstruction decoder for computing with words

Dongrui Wu *

Machine Learning Lab, GE Global Research, Niskayuna, NY, USA

ARTICLE INFO

Article history:
Received 19 November 2012
Received in revised form 5 August 2013
Accepted 16 August 2013
Available online 2 September 2013

Keywords:
Computing with word
Perceptual Computer
Decoder
2-Tuple representation
Type-1 fuzzy sets
Interval type-2 fuzzy sets

ABSTRACT

The Word decoder is a very important approach for decoding in the Perceptual Computer. It maps the computing with words (CWWs) 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 preserves the shape information of the CWW engine output in a simple form without sacrificing much accuracy. It can be viewed as a generalized Word decoder and is also implicitly a Rank decoder. Moreover, it is equivalent to the 2-tuple representation under certain conditions. The effectiveness of the Reconstruction decoder is verified by three experiments.

1. Introduction

Computing with words (CWW) [47,48] is "*a methodology in which the objects of computation are words and propositions drawn from a natural language*". Usually the words and propositions are modeled by fuzzy sets (FSs) [46]. Many different approaches for CWW using FSs have been proposed so far [8,17,25,22,30,23,18,32,27,44,49,45,10,12,1,28,42,24]. According to Wang and Hao [31], these techniques may be classified into three categories:

(i) *The Extension Principle based models* [1,22,20,3], which operate on the underlying FS models of the linguistic terms using the Extension Principle [46]. Bonissone and Decker proposed the first such model in 1986 [1]. One of the latest developments is the Perceptual Computer (Per-C) [20,22], depicted in Fig. 1. It 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. 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 [19] may be used for word modeling. The outputs of the encoder activate a CWW engine, where the FSs are aggregated by novel weighted averages [39] or perceptual reasoning [38] according to the specific application. The output of the CWW engine 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:
   (a) *Word:* To map a FS into a word, it must be possible to compare the *similarity* between two FSs. The Jaccard similarity measure [37] 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.

* Tel.: +1 213 595 3269.
E-mail address: wud@ge.com

0020-0255/$ - see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.ins.2013.08.050

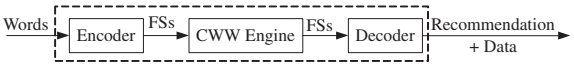

Fig. 1. Conceptual structure of the Perceptual Computer.

(b) *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 FSs is needed. A centroid-based ranking method for T1 and IT2 FSs is described in [37].

(c) *Class:* A classifier is necessary when the output of the CWW engine needs to be mapped into a decision category [21]. Subsethood [33,22,29] 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.

(ii) *The symbolic model* [4,43,6], which makes computations on the indices of the linguistic terms. It first constructs an ordered linguistic term set, $\mathbb{W} = \{W_1, W_2, \ldots, W_N\}$, where $W_i < W_j$ if and only if $i < j$. Convex combination [4] is then used to recursively aggregate the terms. For example, to aggregate W_i and W_j with weight α and β, respectively, it computes

$$W_k = \frac{\alpha W_i + \beta W_j}{\alpha + \beta}, \tag{1}$$

where the term index k is determined as

$$k = i + round\left[\frac{\beta}{\alpha + \beta}(j - i)\right]. \tag{2}$$

To aggregate W_i, W_j and W_p with weight α,β and γ, respectively, i.e., to compute

$$W_{k'} = \frac{\alpha W_i + \beta W_j + \gamma W_p}{\alpha + \beta + \gamma} \tag{3}$$

it rewrites $W_{k'}$ as

$$W_{k'} = \frac{\alpha + \beta}{\alpha + \beta + \gamma} \cdot \frac{\alpha W_i + \beta W_j}{\alpha + \beta} + \frac{\gamma}{\alpha + \beta + \gamma} W_p = \frac{\alpha + \beta}{\alpha + \beta + \gamma} W_k + \frac{\gamma}{\alpha + \beta + \gamma} W_p \tag{4}$$

where W_k is the same as the one in (1) and k is computed by (2). $W_{k'}$ then becomes a two-term aggregation and k' is computed as

$$k' = k + round\left[\frac{\gamma}{\alpha + \beta + \gamma}(p - k)\right] \tag{5}$$

Aggregations involving more terms are computed in a similar recursive way. The intermediate results are numeric values, which must be approximated in each recursion to an integer in $[1,N]$ (e.g., k and k' above), which is the index of the associated linguistic term.

(iii) *The 2-tuple representation based model* [8,9,16,5,6], which is an improvement over the symbolic model. It was first proposed by Herrera and Martinez in 2000 [8] and followed by many others. Instead of representing the aggregation result as a single integer term index in $[1,N]$, it represents the result as a 2-tuple (W_n, α), where n is an integer linguistic term index, and $\alpha \in [-0.5, 0.5)$ is a numeric value representing the symbolic translation, i.e., the translation from the original result to the closest index label in the linguistic term set. More specifically, let $\mathbb{W} = \{W_1, W_2, \ldots, W_N\}$ be a linguistic term set and $\beta \in [1,N]$ be a value representing the result of a symbolic aggregation operation, then the 2-tuple representation (W_n, α) is computed as

$$n = round(\beta) \tag{6}$$

$$\alpha = \beta - n, \alpha \in [-0.5, .5) \tag{7}$$

As a result, the 2-tuple model allows a continuous representation of the linguistic information in its domain. Several aggregation operations using the 2-tuple representation, e.g., arithmetic mean, weighted average, ordered weighted average, have been developed [8].

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 back to the initial expression domain. This results in loss of information and hence the lack of precision [31,2]. 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. In fact, they do not even need a FS model for each word. The only requirement is that the linguistic terms are ordered. They also have the information loss problem because the intermediate results are rounded to integer term indices. 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, e.g., they need to have the same shape and be equidistant [8].

There have been some hybrid approaches, which try to combine the advantages of different models but eliminate their limitations. For example, there is a new version of 2-tuple linguistic representation model [31], 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, which is the focus of this paper. Particularly, we focus on the Word decoder because it is the most widely used decoding method. We propose a Reconstruction decoder for the Per-C, which can be used to replace the Word decoder with smaller information loss.

The remainder of this paper is organized as follows: Section 2 introduces the details of the Reconstruction decoder, its characteristics, and its relationship to the 2-tuple representation. Section 3 demonstrates the performance of the Reconstruction decoder through three experiments. Section 4 draws conclusions.

2. The Reconstruction decoder

The Reconstruction decoder is motivated by how a decimal number can be represented by two successive integers immediately before and after it. For example, the decimal 4.2, which lies between two successive integers 4 and 5, can be represented as $4.2 = \alpha \times 4 + \beta \times 5$, where $\alpha = 0.8$, $\beta = 0.2$. For numbers we always have $\alpha \geqslant 0$, $\beta \geqslant 0$, and $\alpha + \beta = 1$. In the Reconstruction decoder we view each word W_n in the codebook as an "integer," and the CWW engine output Y as a "decimal." The goal is to represent this "decimal" using two successive "integers" with minimum information loss.

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 [14] and its enhanced version, the Enhanced Interval Approach [41], both of which only output normal trapezoidal IT2 FSs. So, in this paper we focus on normal trapezoidal T1 and IT2 FSs. However, very recently it has been shown [26] that FSs with spikes may be generated from some new aggregation functions, although the inputs are still ordinary FSs. At the end of this section we will show how our method can be applied to subnormal FSs, and FSs with arbitrary shapes.

2.1. The Reconstruction decoder for T1 FS Word Models

A normal trapezoidal T1 FS can be represented by four parameters, (a, b, c, d), as shown in Fig. 2. Note that a triangular T1 FS is a special case of the trapezoidal T1 FS when $b = c$. We denote the membership grade of x on a T21 FS Y as $\mu_Y(x)$.

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 [37]. The trapezoidal T1 FS model for the nth word is W_n, which is represented by four parameters (a_n, b_n, c_n, d_n) and whose centroid is w_n, $n = 1, 2, \ldots, N$. The basic idea of the Reconstruction decoder is to find a linear combination of two successive codebook words to represent Y with minimum information loss, i.e.,

$$Y \approx W \tag{8}$$

where

$$W = \alpha W_{n'} + \beta W_{n'+1}. \tag{9}$$

The first step is to determine n', the location of the first word in (9). We compute the centroid of Y, y, and then identify an n' such that

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

Essentially, (10) means that we rank $\{W_n\}$ and Y together according to their centroids and then select the two words immediately before and after[2] Y.

[1] Strictly speaking, when trapezoidal T1 FSs are used in the CWW engine, e.g., the novel weighted averages [39,22] or Perceptual Reasoning [38,22], 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. 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 (10); however, this cannot occur in the Per-C if the encoder and the decoder use the same vocabulary and the novel weighted average [39,22] or Perceptual Reasoning [38,22] 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.

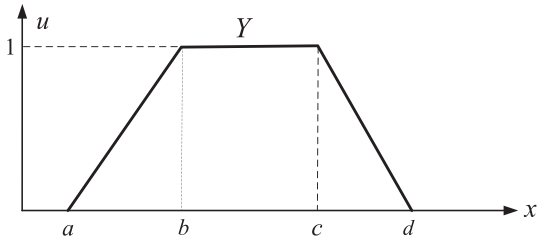

Fig. 2. A trapezoidal T1 FS, determined by four parameters (a, b, c, d).

Having determined the two neighbors of Y, the next step is to determine the coefficients 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.,

(i) 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.
(ii) 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.

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 [37] can work for any T1 FSs. So, in this paper we use the first definition.

Before being able to compute 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 by four parameters (a_w, b_w, c_w, d_w). Based on the Extension Principle [46] and the α-cut Representation Theorem [11], we have

$$a_w = \alpha a_{n'} + \beta a_{n'+1} \tag{11}$$
$$b_w = \alpha b_{n'} + \beta b_{n'+1} \tag{12}$$
$$c_w = \alpha c_{n'} + \beta c_{n'+1} \tag{13}$$
$$d_w = \alpha d_{n'} + \beta d_{n'+1} \tag{14}$$

To solve for α and β, we consider the following constrained optimization problem:

$$\begin{aligned} &\arg\max_{\alpha,\beta} \quad s(Y,W) \\ &\text{s.t.} \quad \alpha \geqslant 0,\ \beta \geqslant 0 \\ &\qquad \alpha + \beta = 1 \end{aligned} \tag{15}$$

where

$$s(Y,W) = \frac{\sum_{i=1}^{I} \min(\mu_Y(x_i), \mu_W(x_i))}{\sum_{i=1}^{I} \max(\mu_Y(x_i), \mu_W(x_i))} \tag{16}$$

is the Jaccard similarity measure between Y and W, and the constraints are motivated from the crisp case, as described at the beginning of this section.

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

(1) Compute w_n, the centroid of W_n, $n = 1, \ldots, N$, and rank $\{W_n\}$ in ascending order. This step only needs to be performed once for a codebook, and it can be done offline.
(2) Compute y, the centroid of Y.
(3) Identify n' according to (10).
(4) Solve the constrained optimization problem in (15) for α and β.
(5) Represent the decoding output as $Y \approx \alpha W_{n'} + \beta W_{n'+1}$.

2.2. 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), $\overline{Y}$, similar to the T1 FS case; however, we need five parameters for the trapezoidal lower membership function (LMF), $\underline{Y}$, 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 $\widetilde{Y}$, 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 [37]. The IT2 FS for the nth word is $\widetilde{W}_n$, which is represented by $(a_n, b_n, c_n, d_n, e_n, f_n, g_n, i_n, h_n)$ and whose

center of centroid [22] is w_n, $n = 1, 2, \ldots, N$. The basic idea of the Reconstruction decoder is again to find a combination of two successive codebook words to represent $\tilde{Y}$ with minimum information loss.

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

$$w_{n'} \leqslant y \leqslant w_{n'+1}. \tag{17}$$

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

$$\begin{aligned} \arg\max_{\alpha,\beta} \quad & s(\tilde{Y}, \widetilde{W}) \\ \text{s.t.} \quad & \alpha \geqslant 0, \ \beta \geqslant 0 \\ & \alpha + \beta = 1 \end{aligned} \tag{18}$$

where

$$\widetilde{W} = \alpha \widetilde{W}_{n'} + \beta \widetilde{W}_{n'+1}. \tag{19}$$

and $s(\tilde{Y}, \widetilde{W})$ is the Jaccard similarity measure between $\tilde{Y}$ and $\widetilde{W}$:

$$s(\tilde{Y}, \widetilde{W}) = \frac{\sum_{i=1}^{I} \min(\mu_{\overline{Y}}(x_i), \mu_{\overline{W}}(x_i)) + \sum_{i=1}^{I} \min(\mu_{\underline{Y}}(x_i), \mu_{\underline{W}}(x_i))}{\sum_{i=1}^{I} \max(\mu_{\overline{Y}}(x_i), \mu_{\overline{W}}(x_i)) + \sum_{i=1}^{I} \max(\mu_{\underline{Y}}(x_i), \mu_{\underline{W}}(x_i))} \tag{20}$$

Clearly, to solve (18), we need to be able to numerically represent $\widetilde{W}$ in (19). 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

$$a_w = \alpha a_{n'} + \beta a_{n'+1} \tag{21}$$
$$b_w = \alpha b_{n'} + \beta b_{n'+1} \tag{22}$$
$$c_w = \alpha c_{n'} + \beta c_{n'+1} \tag{23}$$
$$d_w = \alpha d_{n'} + \beta d_{n'+1} \tag{24}$$

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 [35,36]). Without loss of generality, assume $h_{n'} \leqslant h_{n'+1}$. We then crop the top of the LMF of $\widetilde{W}_{n'+1}$ to make it the same height as the LMF of $\widetilde{W}_{n'}$, as shown in Fig. 4. Representing the cropped version of the LMF of $\widetilde{W}_{n'+1}$ as $(e_{n'+1}, f'_{n'+1}, g'_{n'+1}, i_{n'+1}, h_{n'})$, the LMF of $\widetilde{W}$ is then computed as:

$$e_w = \alpha e_{n'} + \beta e_{n'+1} \tag{25}$$
$$f_w = \alpha f_{n'} + \beta f'_{n'+1} \tag{26}$$
$$g_w = \alpha g_{n'} + \beta g'_{n'+1} \tag{27}$$
$$i_w = \alpha i_{n'} + \beta i_{n'+1} \tag{28}$$
$$h_w = \min(h_{n'}, h_{n'+1}) \tag{29}$$

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

(1) Compute w_n, the centers of centroid of $\widetilde{W}_n$, $n = 1, \ldots, N$, and rank $\{\widetilde{W}_n\}$ in ascending order. This step only needs to be performed once for a codebook, and it can be done offline.
(2) Compute y, the center of centroid of $\tilde{Y}$.
(3) Identify n' according to (17).

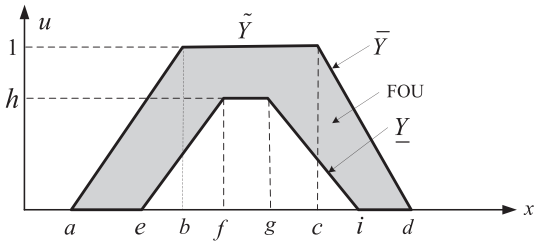

Fig. 3. A normal trapezoidal IT2 FS. (a, b, c, d) determines a normal trapezoidal UMF, and (e, f, g, i, h) determines a trapezoidal LMF with height h.

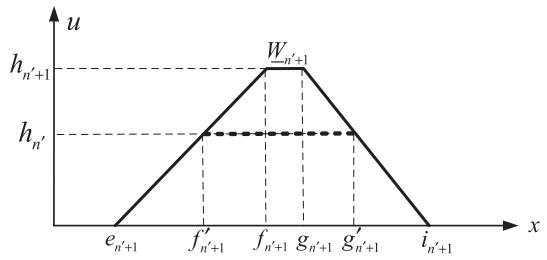

Fig. 4. Illustration of how $\underline{W}_{n'+1}$, the LMF of $\widetilde{W}_{n'+1}$, is cropped to have height $h_{n'}$.

(4) Solve the constrained optimization problem in (18) for α and β.
(5) Represent the decoding output as $\widetilde{Y} \approx \alpha\widetilde{W}_{n'} + \beta\widetilde{W}_{n'+1}$.

Matlab implementation of the Reconstruction decoder for both T1 and IT2 FSs can be found in [34].

2.3. 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 step that becomes more complex is the computation of W in (9) or $\widetilde{W}$ in (19).

Consider W in (9) first. Because $\alpha + \beta = 1$, we can rewrite W as

$$W = \frac{\alpha W_{n'} + \beta W_{n'+1}}{\alpha + \beta} \tag{30}$$

W in the above equation is very similar to a fuzzy weighted average (FWA) [13], whose standard representation is

$$E = \frac{AB + CD}{A + C} \tag{31}$$

where A, B, C, D and E are all T1 FSs. To convert (30) into a FWA, we treat numbers α and β as special T1 FSs $\tilde{\alpha}$ and $\tilde{\beta}$, i.e.,

$$\mu_{\tilde{\alpha}}(x) = \begin{cases} 1, & x = \alpha \\ 0, & \text{otherwise} \end{cases} \tag{32}$$

and

$$\mu_{\tilde{\beta}}(x) = \begin{cases} 1, & x = \beta \\ 0, & \text{otherwise} \end{cases} \tag{33}$$

In terms of the 4-parameter representation, $\tilde{\alpha} = [\alpha, \alpha, \alpha, \alpha]$, and $\tilde{\beta} = [\beta, \beta, \beta, \beta]$. Then

$$W = \frac{\tilde{\alpha} W_{n'} + \tilde{\beta} W_{n'+1}}{\tilde{\alpha} + \tilde{\beta}} \tag{34}$$

can be computed by a FWA procedure [13,22].

Similarly, to compute $\widetilde{W}$ in (19), we can treat number α as a special IT2 FS $\tilde{\tilde{\alpha}}$, whose lower and upper membership functions are both identical to $\tilde{\alpha}$, and number β as a special IT2 FS $\tilde{\tilde{\beta}}$, whose lower and upper membership functions are both identical to $\tilde{\beta}$. In terms of the 9-parameter representation, $\tilde{\tilde{\alpha}} = [\alpha, \alpha, \alpha, \alpha, \alpha, \alpha, \alpha, \alpha, 1]$, and $\tilde{\tilde{\beta}} = [\beta, \beta, \beta, \beta, \beta, \beta, \beta, \beta, 1]$. Then $\widetilde{W}$ in (19) can be rewritten as

$$\widetilde{W} = \frac{\tilde{\tilde{\alpha}} \widetilde{W}_{n'} + \tilde{\tilde{\beta}} \widetilde{W}_{n'+1}}{\tilde{\tilde{\alpha}} + \tilde{\tilde{\beta}}} \tag{35}$$

and computed as a linguistic weighted average [35,36].

2.4. Characteristics of the Reconstruction decoder

The Reconstruction decoder has the following advantages:

(i) *The Reconstruction decoder is a generalized Word decoder.* Take the T1 FS case for example. If we want to represent $Y = \alpha W_{n'} + \beta W_{n'+1}$ by a single word in the codebook, then it is safe to choose $W_{n'}$ if $\alpha > \beta$, or $W_{n'+1}$ if $\alpha < \beta$, because this is almost always consistent with the Word decoder (see the experimental results in the next section). In the rare case of inconsistency, $s(Y, W_{n'})$ and $s(Y, W_{n'+1})$ are very close to each other, so mapping Y to $W_{n'}$ or $W_{n'+1}$ does not make much difference.

(ii) *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 $n' < m'$, then regardless of the values of α_1, β_1, α_2 and β_2, it must be true that $Y_1 \leqslant Y_2$ because $Y_1 \leqslant W_{n'+1} \leqslant W_{m'} \leqslant Y_2$. On the other hand, if we know $Y_1 = \alpha_1 W_{n'} + \beta_1 W_{n'+1}$, $Y_2 = \alpha_2 W_{n'} + \beta_2 W_{n'+1}$ (note that Y_1 and Y_2 have the same n'), and $\alpha_1 < \alpha_2$, then we should have $Y_1 > Y_2$. These properties are especially useful in distinguishing among highly similar Ys, which may be mapped into the same word by the Word decoder.

(iii) *The Reconstruction decoder preserves the shape information of the CWW engine output in a simple form with minimum information loss.* Usually the similarities between the original FSs and the reconstructed FSs are very close to 1. 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. Additionally, as $s(Y,W)$ [or $s(\widetilde{Y}, \widetilde{W})$] is always equal to or larger than the similarity between Y and the word suggested by the Word decoder, replacing Y by W (or $\widetilde{Y}$ by $\widetilde{W}$) almost always results in smaller information loss than replacing it by the word suggested by the Word decoder.

However, we need to point out that a disadvantage of the Reconstruction decoder is its high computational cost in solving the constrained optimization problem.

2.5. Relationship to the 2-tuple representation

The Reconstruction decoder and the 2-tuple representation are closely related, as pointed out by the following[3]:

Theorem 1. *When the codebook $\{W_n\}_{n=1,2,\ldots,N}$ consists of equally spaced trapezoidal T1 FSs with the same shape, a 2-tuple representation (W_m, α) can be converted to the Reconstruction decoder output using the following formula:*

$$(W_m, \alpha) \equiv \begin{cases} (1-\alpha)W_m + \alpha W_{m+1}, & \alpha \geqslant 0 \\ -\alpha W_{m-1} + (1+\alpha)W_m, & \alpha < 0 \end{cases}. \tag{36}$$

Proof. The 4-parameter representation of a trapezoidal T1 FS W_n in an equally spaced codebook can be written as

$$W_n \equiv [a_1 + (n-1)\delta, b_1 + (n-1)\delta, c_1 + (n-1)\delta, d_1 + (n-1)\delta], \quad n = 1, 2, \ldots, N \tag{37}$$

where (a_1, b_1, c_1, d_1) is the 4-parameter representation of W_1, and δ is the distance between two successive T1 FSs. A 2-tuple representation (W_m, α) can be converted to the 4-parameter representation as

$$(W_m, \alpha) \equiv [a_1 + (m+\alpha-1)\delta, b_1 + (m+\alpha-1)\delta, c_1 + (m+\alpha-1)\delta, d_1 + (m+\alpha-1)\delta] \tag{38}$$

When $\alpha \geqslant 0$, (W_m, α) is between W_m and W_{m+1}. Substituting (37) into $(1-\alpha)W_m + \alpha W_{m+1}$ [the first row on the right hand side of (36)], we have

$$\begin{aligned}(1-\alpha)W_m + \alpha W_{m+1} &\equiv (1-\alpha)[a_1 + (m-1)\delta, b_1 + (m-1)\delta, c_1 + (m-1)\delta, d_1 + (m-1)\delta] + \alpha[a_1 + m\delta, b_1 + m\delta, c_1 \\ &\quad + m\delta, d_1 + m\delta] \\ &= [a_1 + (m+\alpha-1)\delta, b_1 + (m+\alpha-1)\delta, c_1 + (m+\alpha-1)\delta, d_1 + (m+\alpha-1)\delta] \equiv (W_m, \alpha)\end{aligned}$$

where the last equation makes use of (38). Thus the first row of (36) is proved.

When $\alpha < 0$, (W_m, α) is between W_{m-1} and W_m. Substituting (37) into $-\alpha W_{m-1} + (1+\alpha)W_m$ [the second row on the right hand side of (36)], we have

$$\begin{aligned}-\alpha W_{m-1} + (1+\alpha)W_m &\equiv -\alpha[a_1 + (m-2)\delta, b_1 + (m-2)\delta, c_1 + (m-2)\delta, d_1 + (m-2)\delta] + (1+\alpha)[a_1 + (m-1)\delta, b_1 \\ &\quad + (m-1)\delta, c_1 + (m-1)\delta, d_1 + (m-1)\delta] \\ &= [a_1 + (m+\alpha-1)\delta, b_1 + (m+\alpha-1)\delta, c_1 + (m+\alpha-1)\delta, d_1 + (m+\alpha-1)\delta] \equiv (W_m, \alpha)\end{aligned}$$

Thus the second row of (36) is also proved. □

From Theorem 1, we can also derive the formula to transform a Reconstruction decoder output to a 2-tuple representation:

[3] Here we only consider T1 FSs because the 2-tuple representation has been mainly used for T1 FSs.

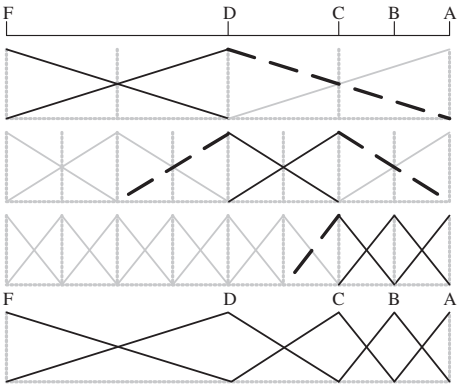

Fig. 5. Semantic representation of the unbalanced grading system in linguistic hierarchies [7].

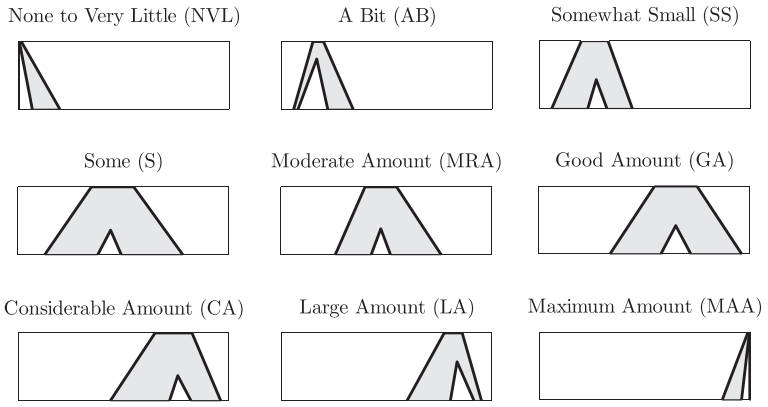

Fig. 6. The 9-word codebook for the SJA.

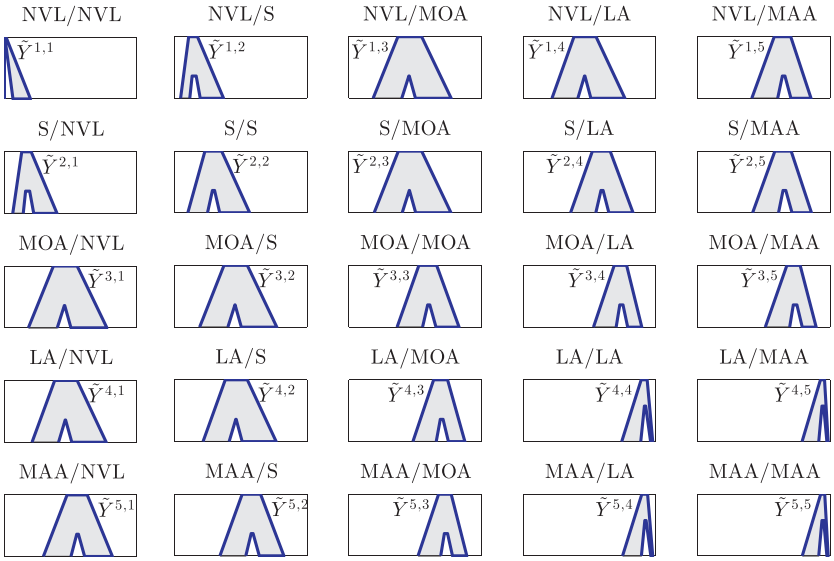

Fig. 7. The 25 rule consequents of the SJA.

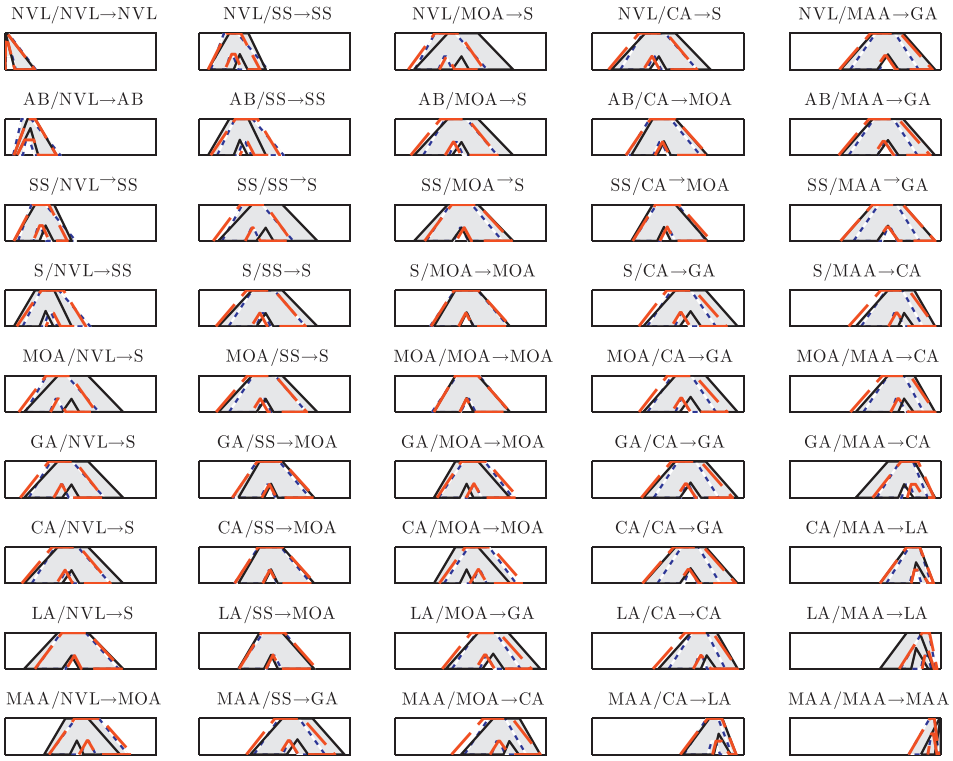

Fig. 8. Comparison of the Word decoder and Reconstruction decoder on the SJA using IT2 FSs. Black solid IT2 FSs are the outputs of Perceptual Reasoning. Blue dotted IT2 FSs are the decoding results of the Word decoder. Red dashed IT2 FSs are the decoding results of the Reconstruction decoder. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

$$\alpha W_m + \beta W_{m+1} \equiv \begin{cases} (W_m, \alpha), & \alpha < 0.5 \\ (W_{m+1}, \alpha - 1), & \alpha \geqslant 0.5 \end{cases}. \tag{39}$$

Of course, the codebook $\{W_n\}_{n=1,2,\ldots,N}$ must consist of equally spaced trapezoidal T1 FSs with the same shape before the above equation can be applied.

As most 2-tuple representations use codebooks consisting of equally spaced trapezoidal T1 FSs with the same shape, their results can be completely duplicated using the Reconstruction decoder. However, the Reconstruction decoder is much more general in the sense that it can also be applied to codebooks consisting of *arbitrary* FSs, both T1 and IT2. To the author's knowledge there has been only one effort [7] to make the 2-tuple representations applicable to nonuniform and nonsymmetric T1 FSs (called unbalanced linguistic terms sets in [7]). For example, to handle the unbalanced grading system evaluation term set $\{A, B, C, D, F\}$ shown in the first row of Fig. 5, one first needs to construct linguistic hierarchies shown in the middle three rows of Fig. 5, and then select individual FSs from different hierarchies to form the desired terms. This process is not easy if the distances between the terms do not have very nice relationship (in the first row of Fig. 5 $|DF| = 2|CD| = 4|BC| = 4|AB|$). Additionally, the resulting FSs still have many constraints, e.g., they must be triangular and their support and apex are defined by the grids used in linguistic hierarchies. On the contrary, the Reconstruction decoder can be easily applied to any codebook of arbitrary FSs without any constraints, as demonstrated next.

3. Experimental results

Three experiments were performed to verify the performance of the Reconstruction decoder. The results are presented in this section.

3.1. Application to the social judgement advisor (SJA): IT2 FSs

In [40] we used the Per-C to construct a single-input social judgement advisor (SJA). In Chapter 8 of [22] we used the Per-C to construct two single-input SJAs and a two-input SJA. In both works the Word decoder was employed. The two-input SJA was used in this experiment to compare the performance of the Reconstruction decoder and the Word decoder.

Table 1
Experimental results for the SJA using IT2 FSs. For each row, $\widetilde{W} = \alpha\widetilde{W}_{n'} + \beta\widetilde{W}_{n'+1}$, where n' is determined by (17).

Touching/eye contact	$s(\widetilde{Y}, \widetilde{W})$	α	β	$s(\widetilde{Y}, \widetilde{W}_{n'})$	$s(\widetilde{Y}, \widetilde{W}_{n'+1})$
NVL/NVL	0.83	1	0	0.81	0.13
NVL/SS	0.85	0.39	0.61	0.50	0.73
NVL/MOA	0.88	**0.56**	0.44	0.45	**0.52**
NVL/CA	0.84	0.26	0.74	0.25	0.73
NVL/MAA	0.72	0.09	0.91	0.35	0.70
AB/NVL	0.75	1	0	0.72	0.42
AB/SS	0.86	0.73	0.27	0.60	0.41
AB/MOA	0.82	0.31	0.69	0.25	0.70
AB/CA	0.86	0.27	0.73	0.72	0.79
AB/MAA	0.73	0.16	0.84	0.38	0.70
SS/NVL	0.84	0.28	0.72	0.43	0.78
SS/SS	0.87	0.50	0.50	0.39	0.57
SS/MOA	0.83	0.46	**0.54**	**0.76**	0.72
SS/CA	0.92	0.87	0.13	0.88	0.39
SS/MAA	0.68	0	1	0.31	0.68
S/NVL	0.86	0.72	0.28	0.60	0.42
S/SS	0.81	0.23	0.77	0.22	0.75
S/MOA	0.96	0.14	0.86	0.75	0.94
S/CA	0.76	0.37	0.63	0.48	0.63
S/MAA	0.69	0.46	0.54	0.58	0.67
MOA/NVL	0.89	**0.52**	0.48	0.42	**0.55**
MOA/SS	0.81	0.21	0.79	0.21	0.77
MOA/MOA	0.95	0	1	0.72	0.95
MOA/CA	0.76	0.35	0.65	0.47	0.64
MOA/MAA	0.69	0.36	0.64	0.56	0.67
GA/NVL	0.89	0.38	0.62	0.33	0.65
GA/SS	0.91	0.30	0.70	0.76	0.83
GA/MOA	0.90	0.72	0.28	0.73	0.46
GA/CA	0.69	0.17	0.83	0.36	0.65
GA/MAA	0.74	0.30	**0.70**	**0.60**	0.59
CA/NVL	0.86	0.25	0.75	0.26	0.75
CA/SS	0.95	0.18	0.82	0.74	0.90
CA/MOA	0.84	0.55	0.45	0.59	0.54
CA/CA	0.64	0.07	0.93	0.31	0.64
CA/MAA	0.79	0	1	0.44	0.78
LA/NVL	0.93	0.46	0.54	0.82	0.82
LA/SS	0.95	0.92	0.08	0.91	0.39
LA/MOA	0.75	0.30	0.70	0.43	0.66
LA/CA	0.64	0.36	0.64	0.52	0.62
LA/MAA	0.87	0.53	0.47	0.36	0.13
MAA/NVL	0.93	0.59	0.41	0.65	0.52
MAA/SS	0.78	0.32	0.68	0.46	0.67
MAA/MOA	0.67	**0.64**	0.36	0.61	**0.63**
MAA/CA	0.76	0.10	0.90	0.53	0.73
MAA/MAA	1	0.30	0.70	0.18	0.26

The two-input SJA is a fuzzy logic system describing the relationship between touching/eye contact and flirtation. It uses a 9-word codebook {None to Very Little (NVL), A Bit (AB), Somewhat Small (SS), Some (S), Moderate Amount (MOA), Good Amount (GA), Considerable Amount (CA), Large Amount (LA), Maximum Amount (MAA)}, shown in Fig. 6. Five of them (NVL, S, MOA, LA, and MAA) were used in a survey [15] to obtain the following 25 rules:

$R^{1,1}$: IF touching is *NVL* and eye contact is *NVL*, THEN flirtation is $\widetilde{Y}^{1,1}$.
⋮
$R^{1,5}$: IF touching is *NVL* and eye contact is *MAA*, THEN flirtation is $\widetilde{Y}^{1,5}$.
⋮
$R^{5,1}$: IF touching is *MAA* and eye contact is *NVL*, THEN flirtation is $\widetilde{Y}^{5,1}$.
⋮
$R^{5,5}$: IF touching is *MAA* and eye contact is *MAA*, THEN flirtation is $\widetilde{Y}^{5,5}$.

where the 25 consequent IT2 FSs are shown in Fig. 7.

The SJA can be used to indicate the flirtation level linguistically given the linguistic description of touching and eye contact levels. It makes use of *Perceptual reasoning (PR)* [38,22], whose details are not relevant to this paper and hence are omitted.

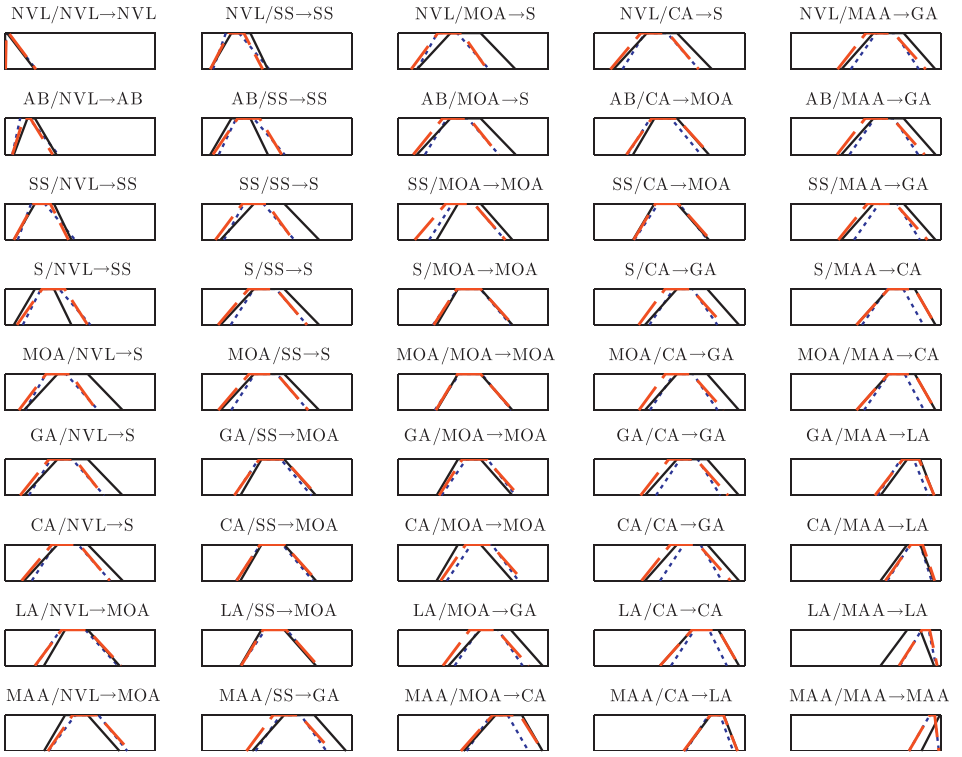

Fig. 9. Comparison of the Word decoder and Reconstruction decoder on the SJA using T1 FSs. Black solid T1 FSs are the outputs of Perceptual Reasoning, which are the same as the black UMFs in Fig. 8. Blue dotted T1 FSs are the decoding results of the Word decoder. Red dashed T1 FSs are the decoding results of the Reconstruction decoder. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

In the experiment we used Touching = {NVL, AB, SS, S, MOA, GA, CA, LA, MAA} and Eye Contact = {NVL, SS, MOA, CA, MAA}[4]. Their full combination has 45 input pairs. We used PR to compute the output IT2 FSs for these 45 cases, which are shown as the black solid IT2 FSs in Fig. 8. The Word decoder and the Reconstruction decoder were then used separately to decode these 45 IT2 FSs. The results from the Word decoder are shown as the blue dotted IT2 FSs in Fig. 8, and the results from the Reconstruction decoder are shown as the red dashed IT2 FSs. Recall that the Word decoder maps each PR result, $\widetilde{Y}$, into a single word $\widetilde{W}'$ in the codebook. The name of that word is indicated in the title of each subfigure. For example, the title of the first subfigure, NVL/NVL → NVL, means that when Touching is NVL and Eye Contact is NVL, the Word decoder maps the PR result into the word NVL.

The Jaccard similarities between $\widetilde{Y}$ and the reconstructed IT2 FS $\widetilde{W}, s(\widetilde{Y}, \widetilde{W})$, are shown in the second column of Table 1. Observe that 5 of the 45 similarities are larger than or equal to 0.95, 10 similarities are larger than or equal to 0.9, 28 similarities are larger than or equal to 0.8, and all 45 similarities are larger than 0.6.

The corresponding α and β for constructing $\widetilde{W}$ are given in the third part of Table 1, and the Jaccard similarities between $\widetilde{Y}$ and $\widetilde{W}_{n'}$ and $\widetilde{W}_{n'+1}$ are shown in the fourth part. The output of the Word decoder is $\widetilde{W}_{n'}$ if $s(\widetilde{Y}, \widetilde{W}_{n'}) > s(\widetilde{Y}, \widetilde{W}_{n'+1})$, and $\widetilde{W}_{n'+1}$ otherwise, where n' is determined by (17). Observe that $s(\widetilde{Y}, \widetilde{W})$ is always larger than or equal to the larger one of $s(\widetilde{Y}, \widetilde{W}_{n'})$ and $s(\widetilde{Y}, \widetilde{W}_{n'+1})$, which means the information loss of the Reconstruction decoder is always smaller than or equal to that of the Word decoder. The mean similarity from the Word decoder is 0.6949, and the mean similarity from the Reconstruction decoder is 0.8211, which represents a 18% improvement over the Word decoder. To examine whether the performance improvement is statistically significant, we performed paired t-test on these 45 pairs of $s(\widetilde{Y}, \widetilde{W})$ and $s(\widetilde{Y}, \widetilde{W}')$ using $\alpha = 0.05$. It gives $df = 44$, $t = 5.72$, and $p < 0.0001$, which means the performance improvement of the Reconstruction decoder over the Word decoder is statistically significant.

It is also interesting to examine whether the Reconstruction decoder preserves the order of similarity, i.e., if $s(\widetilde{Y}, \widetilde{W}_{n-1}) > s(\widetilde{Y}, \widetilde{W}_{n+1})$, then we would expect that $\alpha > \beta$, and vice versa. We call this property *consistency*. Cases with inconsistency are marked in bold in Table 1. Observe that only five of the 45 cases have inconsistency, and for all these five cases, $s(\widetilde{Y}, \widetilde{W}_{n'})$ and $s(\widetilde{Y}, \widetilde{W}_{n'+1})$ are close to each other. So mapping $\widetilde{Y}$ to $\widetilde{W}_{n'}$ or $\widetilde{W}_{n'+1}$ does not make much difference.

As it is mentioned in Section 2.4, the Reconstruction decoder also implies the ranking of the outputs, so it is able to distinguish between cases that a Word decoder cannot. For example, observe from the first row of Fig. 8 that, when Touching is

[4] We could have used Eye Contact = {NVL, AB, SS, S, MOA, GA, CA, LA, MAA} but in this case there would be 81 different combinations of Touching/Eye Contact pairs. The subfigures in Fig. 8 would be too small, and Table 1 would be too long to fit into one page.

Table 2
Experimental results for the SJA using T1 FSs. For each row, $W = \alpha W_{n'} + \beta W_{n'+1}$, where n' is determined by (10).

Touching/eye contact	$s(Y,W)$	α	β	$s(Y,W_{n'})$	$s(Y,W_{n'+1})$
NVL/NVL	0.96	0.94	0.06	0.90	0.17
NVL/SS	0.88	0.34	0.66	0.62	0.81
NVL/MOA	0.91	**0.55**	0.45	0.50	**0.56**
NVL/CA	0.85	0.26	0.74	0.26	0.76
NVL/MAA	0.75	0.33	0.67	0.39	0.70
AB/NVL	0.85	0.17	0.83	0.19	0.77
AB/SS	0.89	0.73	0.27	0.65	0.43
AB/MOA	0.85	0.34	0.66	0.29	0.72
AB/CA	0.85	0.33	0.67	0.73	0.78
AB/MAA	0.76	0.36	0.64	0.41	0.71
SS/NVL	0.88	0.19	0.81	0.54	0.85
SS/SS	0.87	**0.51**	0.49	0.43	**0.60**
SS/MOA	0.82	0.2	0.8	0.21	0.76
SS/CA	0.94	0.95	0.05	0.93	0.42
SS/MAA	0.71	0.27	0.73	0.34	0.68
S/NVL	0.90	0.68	0.32	0.61	0.46
S/SS	0.84	0.24	0.76	0.24	0.76
S/MOA	0.97	0.15	0.85	0.78	0.94
S/CA	0.78	**0.53**	0.47	0.51	**0.68**
S/MAA	0.73	0	1	0.62	0.73
MOA/NVL	0.90	**0.51**	0.49	0.47	**0.59**
MOA/SS	0.84	0.21	0.79	0.23	0.77
MOA/MOA	0.97	0.03	0.97	0.76	0.97
MOA/CA	0.78	**0.51**	0.49	0.50	**0.69**
MOA/MAA	0.72	0	1	0.61	0.72
GA/NVL	0.88	0.38	0.62	0.36	0.70
GA/SS	0.91	0.30	0.70	0.77	0.85
GA/MOA	0.91	0.80	0.20	0.79	0.51
GA/CA	0.71	0.36	0.64	0.38	0.66
GA/MAA	0.82	0.24	0.76	0.60	0.71
CA/NVL	0.87	0.25	0.75	0.27	0.78
CA/SS	0.94	0.19	0.81	0.77	0.90
CA/MOA	0.85	0.65	0.35	0.64	0.60
CA/CA	0.67	0.28	0.72	0.32	0.63
CA/MAA	0.88	0.89	0.11	0.82	0.05
LA/NVL	0.94	0.45	0.55	0.83	0.85
LA/SS	0.97	0.93	0.07	0.93	0.43
LA/MOA	0.77	0.46	0.54	0.47	0.70
LA/CA	0.67	0	1	0.56	0.67
LA/MAA	0.93	0.55	0.45	0.42	0.15
MAA/NVL	0.95	0.67	0.33	0.71	0.57
MAA/SS	0.81	0.49	0.51	0.51	0.71
MAA/MOA	0.71	0.18	0.82	0.63	0.7
MAA/CA	0.86	0.03	0.97	0.56	0.86
MAA/MAA	1	0.30	0.70	0.21	0.33

NVL, Eye Contact at two different levels (MOA and CA) are mapped into the same word S by the Word decoder, so it is impossible to distinguish between the two cases. When the Reconstruction decoder is used, the output for NVL/MOA is reconstructed as 0.56SS+0.44S, and the output for NVL/CA is reconstructed as 0.26SS+0.74S. So we know that the output for NVL/MOA is smaller than the output for NVL/CA, which is reasonable.

3.2. Application to the SJA: T1 FSs

We also tested the performance of the Reconstruction decoder on the SJA for T1 FSs, which were chosen as the UMFs of the corresponding IT2 FSs in the previous subsection. The experimental procedure was the same. In short, we applied both the Reconstruction decoder and the Word decoder to each black UMF in Fig. 8, and the codebook consisted of the nine UMFs in Fig. 6. The results are shown in Fig. 9 and Table 2.

The Jaccard similarities between Y and the reconstructed IT2 FS W, $s(Y,W)$, are shown in the second column of Table 2. Observe that 6 of the 45 similarities are larger than or equal to 0.95, 15 similarities are larger than or equal to 0.9, 33 similarities are larger than or equal to 0.8, and all 45 similarities are larger than 0.65.

The corresponding α and β for constructing W are given in the third part of Table 2, and the Jaccard similarities between Y and $W_{n'}$ and $W_{n'+1}$ are shown in the fourth part. The output of the Word decoder is $W_{n'}$ if $s(Y,W_{n'}) > s(Y,W_{n'+1})$, and $W_{n'+1}$ otherwise, where n' is determined by (10). Observe that $s(Y,W)$ is always larger than or equal to the larger one of $s(Y,W_{n'})$ and $s(Y,W_{n'+1})$, which means the information loss of the Reconstruction decoder is always smaller than or equal to that of

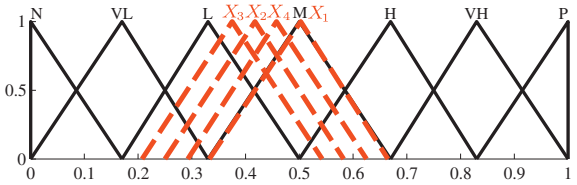

Fig. 10. The 7-term codebook (black solid lines) used in the evaluation and the CWW Engine outputs, X_1-X_4 (red dashed lines). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 3
Evaluations of the four candidates against the four criteria [8].

	x_1	x_2	x_3	x_4
p_1	VL	M	M	L
p_2	M	L	VL	H
p_3	H	VL	M	M
p_4	H	H	L	L

Table 4
Selection results using four different methods.

Method	X_1	X_2	X_3	X_4	Selections
Word decoder	M	M	L	M	x_1, x_2, x_4
Reconstruction decoder	M	.5L + .5M	.75L + .25M	.25L + .75M	x_1
Symbolic representation	M	M	L	M	x_1, x_2, x_4
2-Tuple representation	(M,0)	(M,−.5)	(L,−.25)	(M,−.25)	x_1

the Word decoder. The mean similarity from the Word decoder is 0.7333, and the mean similarity from the Reconstruction decoder is 0.8500, which represents a 16% improvement over the Word decoder. To examine whether the performance improvement is statistically significant, we performed paired t-test on these 45 pairs of $s(Y,W)$ and $s(Y,W')$ using $\alpha = 0.05$. It gives $df = 44$, $t = 5.73$, and $p < 0.0001$, which means the performance improvement of the Reconstruction decoder over the Word decoder is statistically significant.

We also studied inconsistency, as introduced in the previous experiment. Cases with inconsistency are marked in bold in Table 2. Observe that only five of the 45 cases have inconsistency, and for all these five cases, $s(Y, W_{n'})$ and $s(Y, W_{n'+1})$ are very close to each other. So mapping Y to $W_{n'}$ or $W_{n'+1}$ does not make much difference.

3.3. Group decision making using T1 FSs

In this subsection we use the group decision making example introduced in [8] to compare the four decoders introduced in this paper: the Word decoder, the Reconstruction decoder, the symbolic representation, and the 2-tuple representation.

In this example [8], a consulting company is helping another company select its computing system from four candidates: x_1-UNIX, x_2-WINDOWS, x_3-AS/400, and x_4-VMS. The consulting company has four departments to evaluate each candidate from four different perspectives: p_1-Cost, p_2-System, p_3-Risk, and p_4-Technology. The evaluations are assessed using the equally-spaced 7-term codebook shown in Fig. 10. The evaluation results are shown in Table 3. The four evaluations for each candidate are then weighted equally to obtain the overall score of that candidate.

When the Per-C approach is used, the CWW Engine is a special fuzzy weighted average, and the outputs, X_1–X_4, are shown as the red dashed lines in Fig. 10. Observe that in this special case X_1–X_4 assume the same shape as the codebook words. When the Word Decoder is used, X_1–X_4 are mapped into the four words shown in the first row of Table 4. Observe that X_1, X_2 and X_4 are mapped into the same word M, so they are not distinguishable by the Word decoder.[5] The results for the Reconstruction decoder are shown in the second row of Table 4. Observe that the words for X_1–X_4 are different. Using the results presented in Section 2.4, it is easy to conclude that X_1 is the best, which is correct. The results for the symbolic and 2-tuple representations are shown in the last two rows of Table 4, and the detailed computations can be found in [8]. Similar to the Word decoder, the symbolic representation cannot distinguish among X_1, X_2 and X_4. The 2-tuple representation suggests that x_1 is the best, which is also correct.

[5] We suggest a Rank decoder for this application [22]. The Word decoder is used here just to illustrate its difference from other approaches. It was also used in [8] under a different name.

In summary, this example demonstrates that the Reconstruction decoder and the 2-tuple representation are better able to distinguish among similar outputs than the Word decoder and the symbolic representation. This is not surprising, as we have shown that the Reconstruction decoder and the 2-tuple representation are equivalent when the codebook consists of equally spaced trapezoidal T1 FSs with the same shape, which is true in this example.

4. Conclusions

The Word decoder is a very important approach for decoding in the Per-C. It maps the CWW engine output, a FS, 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 preserves the shape information of the CWW engine output in a simple form without sacrificing much accuracy. It can be viewed as a generalized Word decoder and is also implicitly a Rank decoder. Moreover, it is equivalent to the 2-tuple representation under certain conditions. The effectiveness of the Reconstruction decoder has been verified by three experiments.

References

[1] P. Bonissone, K. Decker, Selecting uncertainty calculi and granularity: an experiment in trading-off precision and complexity, in: L. Kanal, J. Lemmer (Eds.), Uncertainty in Artificial Intelligence, North-Holland, Amsterdam, The Netherlands, 1986, pp. 217–247.
[2] C. Carlsson, R. Fuller, Benchmarking and linguistic importance weighted aggregations, Fuzzy Sets and Systems 114 (1) (2000) 35–42.
[3] R. Degani, G. Bortolan, The problem of linguistic approximation in clinical decision making, International Journal of Approximate Reasoning (2) (1988) 143–162.
[4] M. Delgado, J.L. Verdegay, M.A. Vila, On aggregation operations of linguistic labels, International Journal of Intelligent Systems 8 (1993) 351–370.
[5] Y. Dong, Y. Xu, S. Yu, Computing the numerical scale of the linguistic term set for the 2-tuple fuzzy linguistic representation model, IEEE Transaction on Fuzzy Systems 17 (6) (2009) 1366–1378.
[6] F. Herrera, E. Herrera-Viedma, S. Alonso, F. Chiclana, Computing with words in decision making: foundations, trends and prospects, Fuzzy Optimization and Decision Making 8 (2009) 337–364.
[7] F. Herrera, E. Herrera-Viedma, L. Martinez, A fuzzy linguistic methodology to deal with unbalanced linguistic term sets, IEEE Transaction on Fuzzy Systems 16 (2) (2008) 354–370.
[8] F. Herrera, L. Martinez, A 2-tuple fuzzy linguistic representation model for computing with words, IEEE Transaction on Fuzzy Systems 8 (6) (2000) 746–752.
[9] F. Herrera, L. Martinez, A model based on linguistic 2-tuples for dealing with multigranular hierarchical linguistic contexts in multi-expert decision-making, IEEE Transaction on Systems, Man, and Cybernetics – B (2) (2001) 227–234.
[10] J. Kacprzyk, S. Zadrożny, Computing with words in intelligent database querying: standalone and internet-based applications, Information Sciences 34 (2001) 71–109.
[11] G.J. Klir, B. Yuan, Fuzzy Sets and Fuzzy Logic: Theory and Applications, Prentice-Hall, Upper Saddle River, NJ, 1995.
[12] J. Lawry, A methodology for computing with words, International Journal of Approximate Reasoning 28 (2001) 51–89.
[13] F. Liu, J.M. Mendel, Aggregation using the fuzzy weighted average, as computed using the Karnik–Mendel Algorithms, IEEE Transaction on Fuzzy Systems 12 (1) (2008) 1–12.
[14] F. Liu, J.M. Mendel, Encoding words into interval type-2 fuzzy sets using an Interval Approach, IEEE Transaction on Fuzzy Systems 16 (6) (2008) 1503–1521.
[15] B. Luscombe, Why we flirt, Time Magazine 171 (4) (2008) 62–65.
[16] L. Martinez, F. Herrera, An overview on the 2-tuple linguistic model for computing with words in decision making: extensions, applications and challenges, Information Sciences 207 (1) (2012) 1–18.
[17] S. Massanet, J.V. Riera, J. Torrens, E. Herrera-Viedma, A new linguistic computational model based on discrete fuzzy numbers for computing with words, Information Sciences, in press.
[18] J.M. Mendel, The perceptual computer: An architecture for computing with words, in: Proceedings of the IEEE Int'l Conference on Fuzzy Systems, Melbourne, Australia, 2001.
[19] J.M. Mendel, Uncertain Rule-Based Fuzzy Logic Systems: Introduction and New Directions, Prentice-Hall, Upper Saddle River, NJ, 2001.
[20] J.M. Mendel, An architecture for making judgments using computing with words, International Journal of Applied Mathematics and Computer Science 12 (3) (2002) 325–335.
[21] J.M. Mendel, D. Wu, Computing with words for hierarchical and distributed decision making, in: D. Ruan (Ed.), Computational Intelligence in Complex Decision Systems, Atlantis Press, Paris, France, 2010.
[22] J.M. Mendel, D. Wu, Perceptual Computing: Aiding People in Making Subjective Judgments, Wiley-IEEE Press, Hoboken, NJ, 2010.
[23] S.K. Pal, L. Polkowski, A. Skowron (Eds.), Rough-neural Computing: Techniques for Computing with Words, Springer-Verlag, Heidelberg, Germany, 2003.
[24] W. Pedrycz, Granular Computing: Analysis and Design of Intelligent Systems, CRC Press, Boca Raton, FL, 2013.
[25] M.R. Rajati, J.M. Mendel, Novel weighted averages versus normalized sums in computing with words, Information Sciences 235 (2013) 130–149.
[26] J.T. Rickard, J. Aisbett, New classes of threshold aggregation functions based upon the Tsallis q-exponential with applications to perceptual computing, IEEE Transaction on Fuzzy Systems (in press).
[27] S.H. Rubin, Computing with words, IEEE Transaction on Systems, Man, and Cybernetics – B 29 (4) (1999) 518–524.
[28] K.S. Schmucker, Fuzzy Sets, Natural Language Computations, and Risk Analysis, Computer Science Press, Rockville, MD, 1984.
[29] I. Vlachos, G. Sergiadis, Subsethood, entropy, and cardinality for interval-valued fuzzy sets – an algebraic derivation, Fuzzy Sets and Systems 158 (2007) 1384–1396.
[30] H. Wang, D. Qiu, Computing with words via Turing machines: a formal approach, IEEE Transaction on Fuzzy Systems 11 (6) (2003) 742–753.
[31] J.-H. Wang, J. Hao, A new version of 2-tuple fuzzy linguistic representation model for computing with words, IEEE Transaction on Fuzzy Systems 14 (3) (2006) 435–445.
[32] P. Wang (Ed.), Computing With Words, John Wiley & Sons, New York, 2001.
[33] D. Wu, Intelligent Systems for Decision Support, Ph.D. Thesis, University of Southern California, Los Angeles, CA, May 2009.
[34] D. Wu, A reconstruction decoder for the Perceptual Computer, in: Proceedings of the IEEE World Congress on Computational Intelligence, Brisbane, Australia, 2012.

[35] D. Wu, J.M. Mendel, Aggregation using the linguistic weighted average and interval type-2 fuzzy sets, IEEE Transaction on Fuzzy Systems 15 (6) (2007) 1145–1161.
[36] D. Wu, J.M. Mendel, Corrections to aggregation using the linguistic weighted average and interval type-2 fuzzy sets, IEEE Transaction on Fuzzy Systems 16 (6) (2008) 1664–1666.
[37] D. Wu, J.M. Mendel, A comparative study of ranking methods, similarity measures and uncertainty measures for interval type-2 fuzzy sets, Information Sciences 179 (8) (2009) 1169–1192.
[38] D. Wu, J.M. Mendel, Perceptual reasoning for perceptual computing: a similarity-based approach, IEEE Transaction on Fuzzy Systems 17 (6) (2009) 1397–1411.
[39] D. Wu, J.M. Mendel, Computing with words for hierarchical decision making applied to evaluating a weapon system, IEEE Transaction on Fuzzy Systems 18 (3) (2010) 441–460.
[40] D. Wu, J.M. Mendel, Social judgment advisor: an application of the perceptual computer, in: Proceedings of the IEEE World Congress on Computational Intelligence, Barcelona, Spain, 2010.
[41] D. Wu, J.M. Mendel, S. Coupland, Enhanced interval approach for encoding words into interval type-2 fuzzy sets and its convergence analysis, IEEE Transaction on Fuzzy Systems 20 (3) (2012) 499–513.
[42] R. Yager, A new methodology for ordinal multiobjective decisions based on fuzzy sets, Decision Sciences 12 (4) (1981) 589–600.
[43] R. Yager, An approach to ordinal decision making, International Journal of Approximate Reasoning 12 (1995) 237–261.
[44] R. Yager, Approximate reasoning as a basis for computing with words, in: L.A. Zadeh, J. Kacprzyk (Eds.), Computing With Words in Information/Intelligent Systems 1: Foundations, Physica-Verlag, Heidelberg, 1999, pp. 50–77.
[45] M. Ying, A formal model of computing with words, IEEE Transaction on Fuzzy Systems 10 (5) (2002) 640–652.
[46] L.A. Zadeh, Fuzzy sets, Information and Control 8 (1965) 338–353.
[47] L.A. Zadeh, Fuzzy logic = computing with words, IEEE Transaction on Fuzzy Systems 4 (1996) 103–111.
[48] L.A. Zadeh, From computing with numbers to computing with words – from manipulation of measurements to manipulation of perceptions, IEEE Transaction on Circuits and Systems I 46 (1) (1999) 105–119.
[49] L.A. Zadeh, J. Kacprzyk (Eds.), Computing with Words in Information/Intelligent Systems: 1. Foundations, 2. Applications;;, Physica-Verlag, Heidelberg, 1999.