root

Go Up

Name	TextVectors
Version	1.0.0
Description	Text Vectorization for words and sentences
License	http://www.apache.org/licenses/LICENSE-2.0
Copyright	Copyright (C) 2019 HPCC Systems
Authors	HPCCSystems
DependsOn	ML_Core 3.2.0
Platform	7.0.0

OVERVIEW

TextVectors

Text vector bundle

Table of Contents

Types

Go Up

DESCRIPTIONS

TYPES Types

	Types

Common Type definitions for TextVectors bundle

Children

1. t_WordId (From Types) : Type definition for a Word Id attribute
2. t_SentId (From Types) : Type definition for a Sentence Id attribute
3. t_TextId (From Types) : Type definition for an attribute that can hold either a Word Id
4. t_Vector (From Types) : Type definition for a vector attribute (used for Word or Sentence Vectors)
5. t_Word (From Types) : Type definition for the text of a Word
6. t_Sentence (From Types) : Type definition for the text of a Sentence
7. t_ModRecType (From Types) : Enumeration of the record type for a Text Model
8. Sentence (From Types) : Dataset Record to hold a Sentence
9. SentInfo (From Types) : Sentence Information record, including the sentence vector
10. Word (From Types) : Dataset Record to hold a Word
11. WordInfo (From Types) : Dataset record to hold information about a word, including its vector, number of occurrences in the corpus, and discard probability
12. Vector (From Types) : Dataset record to hold a sentence or word vector
13. TextMod (From Types) : Record definition for the TextVectors Model
14. Closest (From Types) : Used to return a set of closest items (words or sentences) for each of a given set of items to match

15. TrainStats (From Types) : Record to hold the return values from GetTrainStats function

---

T_WORDID t_WordId

Types \

	t_WordId

Type definition for a Word Id attribute

RETURN **UNSIGNED4** —

T_SENTID t_SentId

Types \

	t_SentId

Type definition for a Sentence Id attribute.

RETURN **UNSIGNED8** —

T_TEXTID t_TextId

Types \

	t_TextId

Type definition for an attribute that can hold either a Word Id. or a Sentence Id.

RETURN **UNSIGNED8** —

T_VECTOR t_Vector

Types \

	t_Vector

Type definition for a vector attribute (used for Word or Sentence Vectors).

RETURN **SET (REAL8)** —

T_WORD t_Word

Types \

	t_Word

Type definition for the text of a Word.

RETURN **STRING** —

T_SENTENCE t_Sentence

Types \

	t_Sentence

Type definition for the text of a Sentence.

RETURN STRING —

T_MODRECTYPE t_ModRecType

Types \

	t_ModRecType

Enumeration of the record type for a Text Model.

RETURN UNSIGNED1 —

SENTENCE Sentence

Types \

	Sentence

Dataset Record to hold a Sentence.

FIELD **sentId** ||| UNSIGNED8 — The numeric record id for this sentence.

FIELD **text** ||| STRING — The text content of the sentence.

SENTINFO SentInfo

Types \

	SentInfo

Sentence Information record, including the sentence vector.

FIELD **sentId** ||| UNSIGNED8 — The numeric record id for this sentence.

FIELD **text** ||| STRING — The text content of the sentence.

FIELD **vec** ||| SET (REAL8) — The Text Vector for the sentence.

WORD Word

Types \

	Word

Dataset Record to hold a Word.

FIELD **id** ||| UNSIGNED4 — The numeric record id for this word.

FIELD **text** ||| STRING — The text content of the word.

WORDINFO WordInfo

Types \

	WordInfo

Dataset record to hold information about a word, including its vector, number of occurrences in the corpus, and discard probability.

FIELD **wordId** ||| UNSIGNED4 — The numeric record id for this word.

FIELD **text** ||| STRING — The text content of the sentence.

FIELD **occurs** ||| UNSIGNED4 — The number of times this word occurs in the Corpus.

FIELD **pdisc** ||| REAL8 — The computed probability of discard, based on the frequency of the word in the corpus.

FIELD **vec** ||| SET (REAL8) — The Text Vector for the word.

VECTOR Vector

Types \

	Vector

Dataset record to hold a sentence or word vector.

FIELD **id** ||| UNSIGNED4 — The record id for this vector.

FIELD **vec** ||| SET (REAL8) — The contents of the vector.

TEXTMOD TextMod

Types \

	TextMod

Record definition for the TextVectors Model. Text Model contains both the word and sentence vectors for the trained corpus.

FIELD **typ** ||| UNSIGNED1 — The type of the record – Word or Sentence (see t_ModRecType above).

FIELD **id** ||| UNSIGNED8 — The id of the word or sentence.

FIELD **text** ||| STRING — The textual content of the item (word or sentence).

FIELD **vec** ||| SET (REAL8) — The vector for the word or sentence.

CLOSEST Closest

Types \

	Closest

Used to return a set of closest items (words or sentences) for each of a given set of items to match. Closest contains the set of closest items, while Similarity is the cosine similarity between the text sentence and each of the closest items. For example, Similarity[1] is the Cosine similarity between Text and Closest[1]. Likewise for each of the K items in Closest and Similarity.

FIELD **id** ||| UNSIGNED8 — The id of the word or sentence.

FIELD **text** ||| STRING — The text of the word or sentence

FIELD **closest** ||| SET (STRING) — The text of the K closest words or sentence.

FIELD **similarity** ||| SET (REAL8) — The cosine similarity between this word / sentence and each of the K closest words or sentences. This set corresponds 1:1 to the contents of the 'closest' field.

TRAINSTATS TrainStats

Types \

	TrainStats

Record to hold the return values from GetTrainStats function. This records describes the set of parameters used for the current training session.

FIELD **vecLen** ||| UNSIGNED4 — The dimensionality of the word and sentence vectors.

FIELD **nWeights** ||| UNSIGNED4 — The number of weights needed to train the word vectors.

FIELD **nSlices** ||| UNSIGNED4 — The number of slices used to hold the weights.

FIELD **sliceSize** ||| UNSIGNED4 — The number of weights in each weight Slice.

FIELD **nWords** ||| UNSIGNED4 — The number of words in the vocabulary including N-Grams.

FIELD **nSentences** ||| UNSIGNED8 — The number of sentences in the Corpus.

FIELD **maxNGramSize** ||| UNSIGNED4 — The maximum N-Gram size to consider.

FIELD **nEpochs** ||| UNSIGNED4 — The maximum number of epochs for which to train. Zero (default) means auto-compute.

FIELD **negSamples** ||| UNSIGNED4 — The number of negative samples used in training for each training sample.

FIELD **batchSize** ||| UNSIGNED4 — The batch size used to train the vectors. Zero (default) indicates auto-compute.

FIELD **minOccurs** ||| UNSIGNED4 — The minimum number of occurrences in the Corpus in order for a word to be considered part of the vocabulary.

FIELD **maxTextDist** ||| UNSIGNED4 — The maximum number of edits (in edit distance) to make in matching a previously unseen word to a word in the vocabulary.

FIELD **maxNumDist** ||| UNSIGNED4 — The maximum numeric distance to consider one previously unseen number a match for a number in the vocabulary.

FIELD **discardThreshold** ||| REAL4 — Words with frequency below this number are never discarded from training data. Words with frequency above this number are stochastically sampled, based on their frequency.

FIELD **learningRate** ||| REAL4 — The learning rate used to train the Neural Network.

FIELD **upb** ||| UNSIGNED4 — Updates per batch. The approximate number of weights that are updated across all nodes during a single batch.

FIELD **upbPerNode** ||| UNSIGNED4 — Updates per batch per node. The number of weights updated by each node during a single batch.

FIELD **updateDensity** ||| REAL4 — The proportion of weights updated across all nodes during a single batch.

FIELD **udPerNode** ||| REAL4 — The proportion of weights updated by a single node during a single batch.

Internal

Go Up

Table of Contents

Internal/

Corpus

Go Up

IMPORTS

Types | std.Str | std.system.Thorlib |

DESCRIPTIONS

CORPUS Corpus

	Corpus

```
(DATASET(Sentence) sentences_in=DATASET([], Sentence), UNSIGNED4 wordNGrams = 1,
REAL4 discThreshold = .0001, UNSIGNED4 minOccurs = 5, UNSIGNED4 dropoutK = 3)
```

Analyze a corpus of sentences to support vectorization activities.

Tokenizes sentences into words, provides a Vocabulary of unique words, and supports conversion of sentences into training data.

PARAMETER **wordNGrams** ||| UNSIGNED4 — The maximum sized NGram to generate. 1 indicates unigrams only. 2 indicates unigrams and bigrams. 3 indicates uni, bi, and trigrams. Defaults to 1 (unigrams only).

PARAMETER **discThreshold** ||| REAL4 — Discard threshold. Words with frequency greater than or equal to this number are probabilistically discarded based on their frequency. Words with frequencies below this threshold are never discarded (Default .0001).

PARAMETER **minOccurs** ||| UNSIGNED4 — Words that occur less than this number of times in the corpus are eliminated (Default 5). Words with very few occurrences in the corpus may not get properly trained due to lack of context.

PARAMETER **dropoutK** ||| UNSIGNED4 — The number of NGrams to drop from a sentence (per Sent2Vec paper). Default 3.

PARAMETER **sentences_in** ||| TABLE (Sentence) — No Doc

Children

1. sentences (From Corpus) : Return a dataset of Sentences, distributed evenly
2. getNGrams (From Corpus) : Produce a series of nGrams from the set of words in a sentence
3. sent2wordList (From Corpus) : Convert a sentence to a list of words (including n-grams if requested)
4. tokenizedSent (From Corpus) : Each sentence transformed to a word list
5. VocabSize (From Corpus) : The size of the vocabulary
6. wordCount (From Corpus) : No Documentation Found
7. Vocabulary (From Corpus) : The set of all the unique words in the corpus
8. wordIdList (From Corpus) : No Documentation Found
9. WordList2WordIds (From Corpus) : Convert a list of textual words making up a sentence to a set of ids representing each word's wordId in the vocabulary
10. GetTraining (From Corpus) : Generate training data based on the corpus
11. NegativesTable (From Corpus) : Negatives Table is a record containing the discard probability of each word in the vocabulary as a single SET, indexed by the wordId

---

SENTENCES sentences

Corpus \

	sentences

Return a dataset of Sentences, distributed evenly.

RETURN **TABLE (Sentence)** —

GETNGRAMS getNGrams

Corpus \

SET OF VARSTRING	getNGrams
(SET OF VARSTRING words, UNSIGNED4 ngrams, UNSIGNED4 dropoutk = 0)	

Produce a series of nGrams from the set of words in a sentence. For example, if the parameter ngrams is three, it will produce the set of Bigrams (i.e. 2grams) as well as the set of Trigrams (i.e. 3grams). Ngrams are formatted as _Word1_Word2_Word3. Given the sentence ['the', 'quick', 'brown', 'fox'] and ngrams set to 3, it will return: ['_the_quick', '_quick_brown', '_brown_fox', '_the_quick_brown', '_quick_brown_fox'].

PARAMETER **words** ||| SET (VARSTRING) — No Doc

PARAMETER **ngrams** ||| UNSIGNED4 — No Doc

PARAMETER **dropoutk** ||| UNSIGNED4 — No Doc

RETURN **SET (VARSTRING)** —

SENT2WORDLIST sent2wordList

Corpus \

DATASET(WordList)	sent2wordList
(DATASET(Sentence) sent)	

Convert a sentence to a list of words (including n-grams if requested). Strip out punctuation, cleanup whitespace, and split the words.

PARAMETER **sent** ||| TABLE (Sentence) — No Doc

RETURN **TABLE ({ UNSIGNED4 sentId , SET (STRING) words })** —

TOKENIZEDSENT tokenizedSent

Corpus \

DATASET(WordList)	**tokenizedSent**

Each sentence transformed to a word list.

VOCABSIZE VocabSize

Corpus \

	VocabSize

The size of the vocabulary

RETURN **INTEGER8** —

WORDCOUNT wordCount

Corpus \

	wordCount

No Documentation Found

RETURN **INTEGER8** —

VOCABULARY Vocabulary

Corpus \

	Vocabulary

The set of all the unique words in the corpus. Note: returned vocabulary is distributed by HASH32(text), and sorted by text.

RETURN **TABLE ({ UNSIGNED4 wordId , STRING text , UNSIGNED4 occurs , REAL8 pdisc , SET (REAL8) vec })** —

WORDIDLIST wordIdList

Corpus \

	wordIdList

No Documentation Found

FIELD **sentid** ||| UNSIGNED4 — No Doc

FIELD **ord** ||| UNSIGNED2 — No Doc

FIELD **wordids** ||| SET (UNSIGNED4) — No Doc

FIELD **pdisc** ||| REAL4 — No Doc

WORDLIST2WORDIDS WordList2WordIds

Corpus \

`DATASET(wordIdList)`	**WordList2WordIds**
`(DATASET(WordList) sent)`	

Convert a list of textual words making up a sentence to a set of ids representing each word's wordId in the vocabulary.

PARAMETER **sent** ||| TABLE (WordList) — No Doc

RETURN **TABLE ({ UNSIGNED4 sentId , UNSIGNED2 ord , SET (UNSIGNED4) wordIds , REAL4 pdisc }) —**

GETTRAINING GetTraining

Corpus \

`DATASET(TrainingDat)`	**GetTraining**

Generate training data based on the corpus. Each record is a main word and a set of context words (words that occur with that word in a sentence).

NEGATIVESTABLE NegativesTable

Corpus \

	NegativesTable

Negatives Table is a record containing the discard probability of each word in the vocabulary as a single SET, indexed by the wordId. It is not currently used but is left here for possible future use.

Internal/
# svTrainNN

Go Up

IMPORTS

Types |

DESCRIPTIONS

SVTRAINNN svTrainNN

STREAMED DATASET(SliceExt)	**svTrainNN**
(STREAMED DATASET(SliceExt) wts, STREAMED DATASET(trainingDat) train, UNSIGNED4 slicesize, UNSIGNED4 nWeights, UNSIGNED4 numwords, UNSIGNED4 dim, UNSIGNED4 mbsize, REAL lr, UNSIGNED4 negsamp)	

Neural Network Training for SentenceVectors.

Train specialized SentenceVector neural network given a batch of training data. Takes in weights as a set of weights slices (SliceExt), and returns a set of weight adjustments, also formatted as slices.

PARAMETER **wts** ||| TABLE (SliceExt) — The weights slices.

PARAMETER **train** ||| TABLE (TrainingDat) — The batch of training data formatted as a main word and set of context words.

PARAMETER **slicesize** ||| UNSIGNED4 — The maximum number of weights in a slice.

PARAMETER **nWeights** ||| UNSIGNED4 — The total number of weights across all slices.

PARAMETER **numwords** ||| UNSIGNED4 — The number of words in the vocabulary.

PARAMETER **dim** ||| UNSIGNED4 — The dimensionality of the vectors being trained

PARAMETER **mbsize** ||| UNSIGNED4 — The number of training records in the mini-batch.

PARAMETER **lr** ||| REAL8 — The learning rate to use for this batch.

PARAMETER **negsamp** ||| UNSIGNED4 — The number of negative samples to choose for each main word.

RETURN **TABLE (SliceExt)** — weight updates as DATASET(SliceExt). Note that these are additive changes to the weights, not the final weight values.

---

svUtils

Go Up

IMPORTS

Types |

DESCRIPTIONS

SVUTILS svUtils

	svUtils

Various utility functions used by TextVectors

Children

1. normalizeVector (From svUtils) : Normalize a vector by dividing by the its length to create a unit vector
2. calcSentVector (From svUtils) : Calculate a Sentence Vector by taking the average of the word vectors for all words in the sentence
3. cosineSim (From svUtils) : Cosine similarity a and b are unit vectors
4. isNumeric (From svUtils) : Returns TRUE if a string represents a number (integer)
5. numDistance (From svUtils) : Calculates the numeric distance between two numeric strings as ABS(n1 - n2)
6. addVecs (From svUtils) : Implements vec1 + (vec2 * multiplier) Allows (potentially) scaled addition of vectors as well as subtraction (using a negative multiplier

NORMALIZEVECTOR normalizeVector

svUtils \

t_Vector	**normalizeVector**
(t_Vector vec)	

Normalize a vector by dividing by the its length to create a unit vector

PARAMETER **vec** ||| SET (REAL8) — No Doc

RETURN **SET (REAL8)** —

CALCSENTVECTOR calcSentVector

svUtils \

t_Vector	**calcSentVector**
(t_Vector wordvecs, UNSIGNED2 veclen)	

Calculate a Sentence Vector by taking the average of the word vectors for all words in the sentence.

PARAMETER **wordvecs** ||| SET (REAL8) — A concatenated set of vectors for all the words in the sentence.

PARAMETER **veclen** ||| UNSIGNED2 — The length of each word vector and the resulting sentence vector.

RETURN **SET (REAL8)** —

COSINESIM cosineSim

svUtils \

REAL8	**cosineSim**
`(t_Vector a_in, t_Vector b_in, UNSIGNED4 veclen)`	

Cosine similarity a and b are unit vectors. Theta is the angle between vectors. Cosine similarity is Cos(theta). Cos(theta) = (a . b) / (L2Norm(a) * L2Norm(b)) Note: a . b = L2Norm(a) * L2Norm(b) * Cos(theta) Since we assume the inputs to be unit vectors, the norms will be 1. We therefore simplify the calculation to a . b.

PARAMETER **a_in** ||| SET (REAL8) — No Doc

PARAMETER **b_in** ||| SET (REAL8) — No Doc

PARAMETER **veclen** ||| UNSIGNED4 — No Doc

RETURN **REAL8** —

ISNUMERIC isNumeric

svUtils \

BOOLEAN	**isNumeric**
`(STRING instr)`	

Returns TRUE if a string represents a number (integer). Otherwise FALSE.

PARAMETER **instr** ||| STRING — No Doc

RETURN **BOOLEAN** —

NUMDISTANCE numDistance

svUtils \

UNSIGNED4	numDistance
(VARSTRING str1, VARSTRING str2)	

Calculates the numeric distance between two numeric strings as ABS(n1 - n2).

PARAMETER **str1** ||| VARSTRING — No Doc

PARAMETER **str2** ||| VARSTRING — No Doc

RETURN **UNSIGNED4** —

ADDVECS addVecs

svUtils \

t_Vector	addVecs
(t_Vector vec1, t_Vector vec2, UNSIGNED4 multiplier = 1)	

Implements vec1 + (vec2 * multiplier) Allows (potentially) scaled addition of vectors as well as subtraction (using a negative multiplier.

PARAMETER **vec1** ||| SET (REAL8) — No Doc

PARAMETER **vec2** ||| SET (REAL8) — No Doc

PARAMETER **multiplier** ||| UNSIGNED4 — No Doc

RETURN **SET (REAL8)** —

Internal/

Weights

Go Up

IMPORTS

Types | std.system.Thorlib |

DESCRIPTIONS

WEIGHTS weights

	weights
(SET OF INTEGER4 shape)	

Module to perform calculations to manage the weights, and their storage as slices.

Weights are stored in fixed size slices for ease of distribution and management. Currently only supports 3 layer Neural Network weights as used in word vectorization. Will need to be extended to handle a general Neural Network shape.

PARAMETER **shape** ||| SET (INTEGER4) — The number of neurons in each layer of the Neural Network. For example, [100, 10, 200] describes a neural network with 100 neurons in the input layer, 10 in the hidden layer, and 200 in the output layer.

Children

1. nWeights (From weights) : The total number of weights in the network
2. toFlatIndex (From weights) : Convert the compound index: (layer, j, i) to a contiguous flat index into a set of weights

3. fromFlatIndex (From weights) : Convert a flat index into a list of weights into a compound index into the weights of the neural network: (layer, j, i)
4. slicesPerNode (From weights) : The number of slices needed for each node
5. nSlices (From weights) : The total number of slices used to hold the weights
6. sliceSize (From weights) : The number of weights in each slice
7. nWeightSlots (From weights) : The number of slots to hold weights across all slices
8. initWeights (From weights) : Return an initial set of weight slices with weights set to random values
9. distributeAllSlices (From weights) : Copy weights to all nodes and assign the node id to the copy on each node
10. toSliceExt (From weights) : Make Extended Weights
11. fromSliceExt (From weights) : Take a set of Extended Weight slices (i.e
12. slices2Linear (From weights) : Convert a dataset of replicated slices (SliceExt) into a single SliceExt replicated on each node and containing one linear array (i.e
13. compressOne (From weights) : Compress a set of weights (assumed to be sparse) by converting to a sparse representation [...] packed into a DATA field
14. decompressOne (From weights) : Decompress a set of compressed weights in sparse format (i.e
15. compressWeights (From weights) : Compress a set of extended slices (e.g
16. decompressWeights (From weights) : Decompress a set of compressed slices into the native extended slice format

---

NWEIGHTS nWeights

weights \

	nWeights

The total number of weights in the network

RETURN **INTEGER8** —

TOFLATINDEX toFlatIndex

weights \

UNSIGNED4	toFlatIndex
(UNSIGNED2 l, UNSIGNED4 j, UNSIGNED4 i)	

Convert the compound index: (layer, j, i) to a contiguous flat index into a set of weights.

PARAMETER l ||| UNSIGNED2 — No Doc

PARAMETER j ||| UNSIGNED4 — No Doc

PARAMETER i ||| UNSIGNED4 — No Doc

RETURN **UNSIGNED4** —

FROMFLATINDEX fromFlatIndex

weights \

wIndex	fromFlatIndex
(UNSIGNED4 indx)	

Convert a flat index into a list of weights into a compound index into the weights of the neural network: (layer, j, i).

PARAMETER indx ||| UNSIGNED4 — No Doc

RETURN **ROW (wIndex)** —

SLICESPERNODE slicesPerNode

weights \

	slicesPerNode

The number of slices needed for each node

RETURN **INTEGER8** —

NSLICES nSlices

weights \

	nSlices

The total number of slices used to hold the weights.

RETURN **INTEGER8** —

SLICESIZE sliceSize

weights \

	sliceSize

The number of weights in each slice.

RETURN **INTEGER8** —

NWEIGHTSLOTS nWeightSlots

weights \

	nWeightSlots

The number of slots to hold weights across all slices. This may be different from nWeights because nWeights does not always divide exactly into nSlices.

RETURN **INTEGER8** —

INITWEIGHTS initWeights

weights \

DATASET(slice)	**initWeights**

Return an initial set of weight slices with weights set to random values.

DISTRIBUTEALLSLICES distributeAllSlices

weights \

DATASET(SliceExt)	**distributeAllSlices**
(DATASET(SliceExt) slices)	

Copy weights to all nodes and assign the node id to the copy on each node.

If running on 7.2 or greater, use the DISTRIBUTE(.., ALL) facility. Otherwise, use NORMALIZE to make copies of each and assign nodeId, then DISTRIBUTE by nodeId.

PARAMETER slices ||| TABLE (SliceExt) — No Doc

RETURN **TABLE ({ UNSIGNED2 nodeId , UNSIGNED2 sliceId , REAL4 loss , REAL4 minLoss , UNSIGNED4 minEpoch , UNSIGNED4 maxNoProg , UNSIGNED8 batchPos , SET (REAL8) weights })** —

TOSLICEEXT toSliceExt

weights \

DATASET(SliceExt)	toSliceExt
(DATASET(Slice) weights)	

Make Extended Weights. Return a dataset of weight slices that have been replicated to all nodes and converted to SliceExt record type that includes a node id.

PARAMETER **weights** ||| TABLE (Slice) — No Doc

RETURN **TABLE ({ UNSIGNED2 nodeId , UNSIGNED2 sliceId , REAL4 loss , REAL4 minLoss , UNSIGNED4 minEpoch , UNSIGNED4 maxNoProg , UNSIGNED8 batchPos , SET (REAL8) weights })** —

FROMSLICEEXT fromSliceExt

weights \

DATASET(Slice)	fromSliceExt
(DATASET(SliceExt) extWeights)	

Take a set of Extended Weight slices (i.e. replicated to all nodes) and return a dataset of Weight slices that are distributed by sliceId. The duplicated copies are filtered out except on the node that owns each slice.

PARAMETER **extweights** ||| TABLE (SliceExt) — No Doc

RETURN **TABLE ({ UNSIGNED2 sliceId , SET (REAL8) weights })** —

SLICES2LINEAR slices2Linear

weights \

SliceExt	slices2Linear
(DATASET(SliceExt) slices)	

Convert a dataset of replicated slices (SliceExt) into a single SliceExt replicated on each node and containing one linear array (i.e. SET) of weights.

PARAMETER <u>**slices**</u> ||| TABLE (SliceExt) — No Doc

RETURN **ROW ({ UNSIGNED2 nodeId , UNSIGNED2 sliceId , REAL4 loss , REAL4 minLoss , UNSIGNED4 minEpoch , UNSIGNED4 maxNoProg , UNSIGNED8 batchPos , SET (REAL8) weights })** —

COMPRESSONE compressOne

weights \

DATA	compressOne
(t_Vector wts, UNSIGNED4 slicesize)	

Compress a set of weights (assumed to be sparse) by converting to a sparse representation [. . .] packed into a DATA field.

PARAMETER <u>**wts**</u> ||| SET (REAL8) — No Doc

PARAMETER <u>**slicesize**</u> ||| UNSIGNED4 — No Doc

RETURN **DATA** —

DECOMPRESSONE decompressOne

weights \

t_Vector	decompressOne
(DATA cwts, UNSIGNED4 slicesize)	

Decompress a set of compressed weights in sparse format (i.e. [. . .] into a dense set of weights.

PARAMETER **cwts** ||| DATA — No Doc

PARAMETER **slicesize** ||| UNSIGNED4 — No Doc

RETURN **SET (REAL8)** —

COMPRESSWEIGHTS compressWeights

weights \

DATASET(CSlice)	**compressWeights**
(DATASET(SliceExt) slices)	

Compress a set of extended slices (e.g. SliceExt) into CSlice format.

PARAMETER **slices** ||| TABLE (SliceExt) — No Doc

RETURN **TABLE ({ UNSIGNED2 nodeId , UNSIGNED2 sliceId , REAL4 loss , REAL4 minLoss , UNSIGNED4 minEpoch , UNSIGNED4 maxNoProg , UNSIGNED8 batchPos , DATA cweights })** —

DECOMPRESSWEIGHTS decompressWeights

weights \

DATASET(SliceExt)	**decompressWeights**
(DATASET(CSlice) cslices)	

Decompress a set of compressed slices into the native extended slice format.

PARAMETER **cslices** ||| TABLE (CSlice) — No Doc

RETURN **TABLE ({ UNSIGNED2 nodeId , UNSIGNED2 sliceId , REAL4 loss , REAL4 minLoss , UNSIGNED4 minEpoch , UNSIGNED4 maxNoProg , UNSIGNED8 batchPos , SET (REAL8) weights })** —
