Transformation of Dense and Sparse Text Representations

Wenpeng Hu[1,*], **Mengyu Wang**[2,*], **Bing Liu**[2,†],
Feng Ji[3], **Jinwen Ma**[1], **Dongyan Zhao**[2]
[1] Department of Information Science, Peking University
[2] Wangxuan Institute of Computer Technology, Peking University
[3] Alibaba Group
{wenpeng.hu, wangmengyu, dcsliub, jwma, zhaody}@pku.edu.cn zhongxiu.jf@alibaba-inc.com

Abstract

Sparsity is regarded as a desirable property of representations, especially in terms of explanation. However, its usage has been limited due to the gap with dense representations. Most research progresses in NLP in recent years are based on dense representations. Thus the desirable property of sparsity cannot be leveraged. Inspired by Fourier Transformation, in this paper, we propose a novel Semantic Transformation method to bridge the dense and sparse spaces, which can facilitate the NLP research to shift from dense spaces to sparse spaces or to jointly use both spaces. Experiments using classification tasks and natural language inference task show that the proposed Semantic Transformation is effective.

1 Introduction

A sparse vector is a vector that has a large number of zeros or near zeros. Many studies have shown that sparsity is a desirable property of representations, especially for explanation (Fyshe et al., 2014; Faruqui and Dyer, 2015). In this sense, sparse representation may hold the key to solving the explainability problem of deep neural networks. Apart from the interpretability property, sparse representations can also improve the usability of word vectors as features. The embeddings with good sparsity, interpretability or special meanings can also benefit downstream tasks (Guo et al., 2014; Chang et al., 2018). Several tasks have benefited from sparse representations, e.g., part-of-speech tagging (Ganchev et al., 2009), dependency parsing (Martins et al., 2011), and supervised classification (Yogatama and Smith, 2014).

However, much of the research advances so far for NLP tasks are based on dense representations, e.g., text classification (Kim, 2014; Tang et al., 2015; Wu et al., 2017; Wang et al., 2018), natural language inference (Liu et al., 2019; Kim et al., 2019), machine translation (Cheng, 2019; He et al., 2016) and generation (Serban et al., 2017; Zhang et al., 2019; Du and Cardie, 2017). The study of sparse representations is still limited.

There are two key limitations in the study of sparse representations. First, little work has been done to well connect dense and sparse spaces. The two types of representation are rather independent and cannot help each other to achieve synergy. Second, limited work has been done to generate representations of sentences or phrases in the sparse space using sparse word embeddings.

Inspired by Fourier Transformation, this paper proposes a novel method called Semantic Transformation (ST) to address the problems. With the help of ST, dense and sparse spaces can connect with each other and will not be isolated. The proposed transformation consists of two key components, namely, Semantic Forward Transformation (SFT) and Semantic Backward Transformation (SBT) (see Section 2). SFT is designed to transform a dense representation to a sparse representation. That is, we transform any learned dense features to sparse representations and give the model the properties that sparsity possesses. Sparse representations can also be transformed back to dense representations through SBT. Moreover, we can also perform different operations in the sparse space to achieve different goals.

[*] Equal contribution
[†] Corresponding Author. His current affiliation is University of Illinois at Chicago. Email: liub@uic.edu.

Proceedings of the 28th International Conference on Computational Linguistics, pages 3257–3267
Barcelona, Spain (Online), December 8-13, 2020

Another key innovation of this paper is that it proposes a new approach for achieving sparseness. Conventionally, penalties are used to achieve sparseness (Sun et al., 2016; Ng and others, 2011; Subramanian et al., 2018). However, they suffer from the problems of initialization sensitivity and uncontrollable optimization. In this paper, we propose to achieve sparseness through a novel activation function, which gives an effective solution (see Section 2.1). Experimental results show that the proposed activation function works very well.

In this paper, we also explore a combination method to combine words representations into sentence representations in the sparse space directly.[1] Additionally, the proposed transformations and combination method can be paralleled to enable efficient computation.

In summary, this paper makes the following contributions:

• It proposes a semantic transformation method which effectively connects dense and sparse spaces.

• It proposes to use a new activation function to achieve sparseness, which, to the best of our knowledge, has not been used before. The function works very well.

• It proposes a combination method that can encode sentence in the sparse space directly.

• The proposed methods have been evaluated using text classification and natural language inference tasks with promising results. Since the proposed transformations avoid large scale matrix multiplications in the combination procedure, it is also efficient.

2 Semantic Transformation

In this section, we first briefly describe the composition of Semantic Transformation (ST), and then elaborate on each component. The proposed ST has three operations:

1) SFT (Semantic Forward Transformation). It takes a dense representation as input and transforms it into a higher dimensional sparse space.

2) SBT (Semantic Backward Transformation). It is the inverse of SFT, transforming representations from the sparse space back to the dense space.

3) SCSS (Semantic Combination in the Sparse Space). It computes the sentence representation using its component word representations in the sparse space.[2]

2.1 Semantic Forward Transformation

SFT aims to discover the latent semantic aspects in a dense representation of word $\mathbf{x}$ and put them in a higher dimensional sparse representation $\mathbf{y}$. We assume M is the number of latent semantic aspects[3], and each latent semantic aspect is represented by a vector, i.e. $\mathbf{b}_m \in R^d$ for the m^{th} base. We define all the latent semantic aspects as the bases of semantemes in the real world, denoted by $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_M\} \in R^{d \ M}$. Given $\mathbf{B}$, the function of SFT is to estimate the semantic distribution of the given dense representation over $\mathbf{B}$.

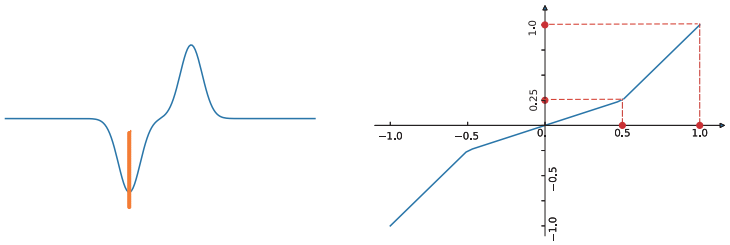
Definition ($\mathbf{y}$): We define $-1 \leq \mathbf{y}_i \leq 1$, meaning that each element of $\mathbf{y}$ has a value in $(-1, 1)$.

The reasons for giving positive and negative values to elements in a sparse representation are that 1) negative values can represent "negative semantemes"; 2) we can eliminate some meanings of elements (positive values) through simple operations between words, i.e., adding. Note that a negative value representing "negative semantics" of a given aspect does not mean that two words with opposite meanings have exactly corresponding positive and negative sparse representations. In the sparse space, we use a composition of semantemes to denote word meanings. This is in line with the human way of using words, e.g., the meaning of "not bad" can be obtained by adding the sparse representations of "not" and "bad". In this sense, the meaning of "not bad" is a composition of several semantemes.

[1] In addition to the combination processing, we can perform different tasks in the sparse space, e.g., filtration and transfer. We leave these tasks to our future work.

[2] Note that although many operations can be done in the sparse space, the purpose of this paper is not to investigate all those operations. This paper mainly focuses on SCSS, the most basic operation in the sparse space for NLP.

[3] We set a limited number of semantemes because those latent semantemes are not all the semantemes in the real world but are the bases for composing real world semantemes.



(a) (b)

Figure 1: Activation curve.

Formulation of SFT: We adopt a multilayer perceptron (MLP)[4] integrated with the base B to build a SFT to perform its function. We first use a MLP $f(\cdot)$ to learn deep features of the dense representation x, and then use the features to compute the sparse distribution over the semantic bases. Formally, the i^{th} layer in $f(\cdot)$ can be written as:

$$\mathbf{p_i} = f_i(\mathbf{p}_{i-1}, \mu) = \sigma(\mathbf{w}_i \mathbf{p}_{i-1}) \tag{1}$$

where σ is the activation function, and $\mathbf{w}_i$ is the parameter of the i^{th} layer denoted by μ; $\mathbf{p}_{i-1}$ is the output of $(i-1)^{th}$ layer and $\mathbf{p}_0 = \mathbf{x}$. We denote the output of the last layer of $f(\cdot)$ as $\mathbf{p}$ and then integrate it with B. The distribution over semantic bases can be computed by:

$$\mathbf{y} = S(\mathbf{p} \cdot \mathbf{w}_f \mathbf{B}) \tag{2}$$

where $\mathbf{w}_f$ is a trainable parameter; $S(\cdot)$ is a specially designed activation function used to control the sparseness of the semantic distribution (discussed later). To sum up, SFT can be written as:

$$\mathbf{y} = \mathcal{SFT}(\mathbf{x}) = S(f(\mathbf{x}) \cdot \mathbf{w}_f \mathbf{B}) \tag{3}$$

Sparse Activation: Sparsity is enforced through penalties in most existing studies, such as ℓ_1 regularizer (Sun et al., 2016), average sparsity penalty (Ng and others, 2011), and partial sparsity penalty (Subramanian et al., 2018). We call those methods *penalty enforcing methods* which push the sparse representation close to either 0 or 1.

However, such penalties suffer from the initialization sensitivity problem as the penalties contain an initial interface which influences the distribution of the learned sparse representation significantly. To overcome the problem, we propose to use an activation function instead. We first give the formulation of the proposed activation function $S(\cdot)$ and then show its activation curve in Figure 1(a).

$$S(x) = e^{-(x-)^2} - e^{-(x+)^2} \tag{4}$$

where β and γ are two hyper-parameters controlling the sparsity of the output. We set $\beta = 1$ and $\gamma = 2$ in our experiments.

Clearly, from Figure 1(a), we can see a large range of inputs of $S(\cdot)$ is mapped to 0, while the positions around $\pm\gamma/\beta$ get high responses. Integrated with an objective loss function (depending on specific types of tasks, e.g., cross entropy for classification), SFT learns to give the relevant aspects/semantemes with predictions around $\pm\gamma/\beta$. In the case under the action of this activation function, we can learn sparse representations through the original objective function, not relying on enforcing penalties. Based on the experimental results, we will see that this activation function works very well on many datasets.

$S(\cdot)$ is non-linear and differentiable and its derivatives can be written as:

$$S'(x) = (-2\beta^2 x + 2\gamma\beta \cdot \text{Sign}(x)) \cdot S(x) \tag{5}$$

where Sign$(\cdot)$ is Sign function, and Sign$(0) = 0$. Clearly, the derivative of $S(\cdot)$ is easy to compute.

[4]Our approach is not limited to using multilayer perceptron (MLP). Other techniques, e.g., CNN may also be used.

2.2 Semantic Backward Transformation

SBT is the inverse transformation of SFT, which transforms a sparse representation back to a dense representation. Since we hope $\mathbf{y}$ is interpretable enough to be related to the semantemes directly, a straightforward way to achieve SBT is to use the sparse representation to do a weighted sum over the base $\mathbf{B}$. In this way, SBT can be seen as a combination of some specific semantemes. To increase the fitting ability of SBT, similar to SFT, we adopt a MLP $F(\cdot)$ to learn a deep dense representation, i.e.,

$$\mathbf{x} = \mathcal{SBT}(\mathbf{y}) = F(\tanh(\mathbf{w}_b \cdot \mathbf{B}\mathbf{y}^T)) \tag{6}$$

where tanh is the Tanh activation function, and $\mathbf{w}_b$ is a trainable parameter. $F(\cdot)$ is a MLP with its own trainable parameters.

2.3 Semantic Combination in Sparse Space

This section proposes a *Semantic Elimination (SE)* method to complete *semantic combination* in the sparse space.[5] The main idea of SE is to use the negative values in the representation of one word to eliminate another word's semantics. That is also one of the reasons for defining negative values in the sparse representation. In this scenario, the sparse representation has two functions: (1) using positive values (positive semantemes) to denote which semantic meanings a word has and (2) using negative values (negative semantemes) to eliminate the semantemes that should not be present in the word. Below, we detail SE.

Due to the fact that a word's semantemes usually change with the nearby words or just the preceding word in a sentence, given a sentence, we propose to use the i^{th} word's negative values to eliminate the $(i+1)^{th}$ word's positive values (semantemes). We call this elimination method *Preceding Elimination (PE)*. After that, a nonlinear activation function must be followed to avoid the overall operation as a linear operation. Note, the activation function must go through the origin $(0, 0)$ in order to ensure the balance of positive and negative values. In this case, we specially designed an activation function, which we will elaborate it shortly. Then we add the sparse representations of all words in the sentence together after PE as the final sentence sparse representation.[6]

We designed an activation function, called 'leaky' (its curve is shown in Figure 1(b)) to (1) decrease the small values of a sparse representation in order to prevent the system from producing new semantemes that shouldn't exist; (2) make the SE sensitive to word order (in order to consider the information of word order) since the activation function is non-linear which enables non-commutativity of the whole SE over linear and non-linear operations. Note that 'leaky' is used on sparse representations of words after preceding elimination. SE is formulated as:

$$\mathbf{s}_t = \sum_{i=1}^{t} \text{leaky}(-\text{Relu}(-\mathbf{y}_{i-1}) + \text{Relu}(\mathbf{y}_i)) \tag{7}$$

where $\mathbf{s}_t$ is the sparse representation of a sub-sentence from position 1 to t produced by semantic combination in the sparse space. Then, $\mathbf{s}_T$ denotes the sparse representation of a sentence with length T.

2.4 Objective Function

Overall, given a batch of data $\mathcal{D}$, our model is trained to minimize the following objective function:

$$\min \; L(\mathcal{D}) = \text{PL}(\mathcal{D}) + \text{ML}(\mathcal{D}) + \text{BL}(\mathcal{D}) + \text{RL}^o(\mathcal{D}) \tag{8}$$

where $\text{PL}(\mathcal{D})$ denotes the prediction loss over the dataset, it depends on the task that the model is applied to; $\text{ML}(\mathcal{D})$ denotes the margin loss, it is performed to enlarge the margin of distances between sparse representations with different meanings; $\text{BL}(\mathcal{D})$ is a regularization used to constrain the norm of bases;

[5] A sparse representation usually has a large number of dimensions (or aspects) but only a small number of dimensions have none zero values. Inherently, it is inappropriate to combine words' sparse representations into sparse sentence representations by using complex matrix transformation.

[6] In the sum vector, if an element is greater than 1 or less than -1, we reduce its absolute value to 1 without sign change.

$\mathrm{RL}^o(\mathcal{D})$ denotes the reconstruction loss, which is used to do model simulation (see below) and therefore *it is optional*. Note, when applying our method only $\mathrm{PL}(\mathcal{D})$ is necessary, $\mathrm{ML}(\mathcal{D})$ and $\mathrm{BL}(\mathcal{D})$ can be used to improve the model's performance. Next, we discuss these loss functions.

Prediction Loss (PL): $\mathrm{PL}(\mathcal{D})$ is the training loss of the application task. For example, in our case, this loss is Cross Entropy for supervised classification.

Margin Loss (ML): $\mathrm{ML}(\mathcal{D})$ is designed to enlarge the margin of distances between sparse representations with different meanings. We need ML to help training because we found that the margin of the learned sparse representation by optimizing PL is not clear or significant for separating positive and negative semantemes, which is undesirable for explanation. We then explore a new method for clear sparse representation learning, called Margin Loss, which makes the sparse representations having different meanings far from each other.

In the scenario of classification, we leverage the class labels as supervising information to group the samples in a batch into each class, and then average the sparse representations of the instances in each class to represent the class. Formally, we assume $\mathbf{y}_{c_i}$ is the averaged representation of the i^{th} class. Then, based on the cosine similarity[7], we define $\mathrm{ML}(\mathcal{D})$ as follows:

$$\min \ \mathrm{ML}(\mathcal{D}) = \mathrm{sum}(\mathbf{W} \odot (\mathbf{Y}_c^T * \mathbf{Y}_c)) \tag{9}$$

where $\mathbf{Y}_c = \{\mathbf{y}_1, \ldots, \mathbf{y}_N\}$, N is the number of classes. $\odot$ denotes Hadamard product. $\mathbf{W} \in \mathbb{R}^{N \ N}$ is hyper-parameter used to control the updating direction and degree. $\mathbf{W}_{ij}$ is set to -1 if $i = j$, or 1 otherwise. This ensures a large margin between different classes by minimizing their inner product. Note that in some scenarios, especially sentiment classification, the distance of different classes belonging to the same positive (or negative) sentiment (e.g., strong and weak positive/negative classes) should not be enlarged much. In this case, we develop an exponential decay function to intuitively set $\mathbf{W}$:

$$\mathbf{W}_{ij} = \begin{cases} \frac{1}{2}^{(N \ \ 1 \ \ ji \ \ jj)=}, & \text{if } i \neq j \\ -1, & \text{otherwise} \end{cases} \tag{10}$$

where τ is half-life, we set it to $(N-1)/2$.

Base Regularization: (BL): Recall in the proposed semantic forward transformation method, base collection $\mathbf{B}$ is the key for obtaining the semantic distribution (semantic representation) of the given dense representation. Clearly, it is a projection procedure. Here, we argue that a larger projection will not ensure a better prediction. That is because representations with a large norm usually get a large projection, which is a point that conventional prediction methods ignore. The proposed Sparse Activation method eliminates this problem by giving large projections small responses. Similarly, inconsistent length of bases in $\mathbf{B}$ will cause different output (response) priors. To tackle this problem, we propose a base regularization to constrain the length of bases in $\mathbf{B}$ to equal to 1. Formally, BL is formulated as:

$$\min \ \mathrm{BL} = \sum_{m=1}^{M} (||\mathbf{b}_m|| - 1)^2 \tag{11}$$

where $\mathbf{b}_m$ is the m^{th} base in $\mathbf{B}$.

Reconstruction Loss (RL^o): The proposed ST can easily do transformations among dense and sparse spaces, and learn sentence representation in the sparse space. In this case, ST could provide a sentence with both dense and sparse representations. One question that may be asked is whether the dense representations produced by ST through back transform can be used in place of dense representations directly learned by models in the dense space, e.g., LSTM? In this case, we propose reconstruction loss to minimize the construction error between the outputs of ST and LSTM. Another purpose of $\mathrm{RL}^o(\mathcal{D})$ is to

[7]Note that $\mathbf{y}_{c_i}$ is not a sparse representation as it is the average of many sparse representations. Cosine similarity is appropriate for Margin Loss.

control the meanings of the same word or sentence/phrase in different spaces to maintain consistency with the representations of a sentence and its phrases produced by LSTM as X, then

$$RL(\mathcal{D}) = \sum_{D} \sum_{i=1}^{\bar{X}} (||x_i - x_i^0||_2^2 + ||s_i - s_i^0||_2^2 + ||X_i - X_i^0||_2^2) \quad (12)$$

where $x_i^0 = \mathcal{SBT}(y_i), X_i^0 = \mathcal{SBT}(s_i), s_i^0 = \mathcal{SFT}(X_i)$; T is the length of the sentence. x, y, s_i have the same meanings as we defined before. X_i^0 denotes the dense representation constructed from sparse representation s_i. This loss helps transform the representations in one space to another while maintaining the semantic information consistency. The last term helps learn similar representations with LSTM.

Table 1: Average accuracy over all tasks. Y and X' are representations for making predictions (X' is the back transformation of Y; Y is the sparse representation). Traditional penalty means the partial sparsity loss in (Subramanian et al., 2018). Helper loss refers to ML or BL. Note that only the experiments using X' as the representations for prediction has RL^o. RL^o is not used when using Y as the prediction feature.

Model	SNLI	MR	SST1	SST2	TREC
CNN (Kim, 2014)	59.71	76.10	36.80	80.60	**90.20**
Transformer (Vaswani et al., 2017)	55.32	75.23	34.80	78.30	81.56
Capsule (Zhao et al., 2018)	54.53	72.57	36.44	77.02	82.31
LSTM (Hochreiter and Schmidhuber, 1997)	66.66	71.04	36.96	75.11	87.60
ST^{ℓ}[X'] (without sparse activation or helper loss)	32.90	61.07	29.97	68.04	63.40
ST^{z}[X'] (with sparse activation, without helper loss)	63.34	70.38	35.79	75.11	80.00
ST^{y}[X'] (using the traditional penalty, without sparse activation or helper loss)	59.89	65.51	33.97	65.25	73.40
ST[X'] (full model)	66.58	71.16	38.69	76.03	87.06
ST^{ℓ}[Y] (without sparse activation or helper loss)	62.46	68.32	35.60	71.33	79.60
ST^{z}[Y] (with sparse activation, without helper loss)	63.53	76.38	38.24	78.33	86.20
ST^{y}[Y] (using the traditional penalty, without sparse activation or helper loss)	62.62	69.17	35.42	71.33	81.60
ST[Y] (full model)	**66.85**	**77.15**	**41.78**	**80.70**	87.80

3 Experiments

We evaluate the proposed method using one natural language inference dataset and four text classification datasets. The tasks act as good quality checks for the learned representations. The five datasets are SNLI, MR, SST1, SST2 and TREC, detailed training/dev/test splits are shown on Table 2:

- SNLI (Bowman et al., 2015): a collection of human-written English sentence pairs manually labeled for balanced classification with labels: entailment, contradiction, and neutral. This is the natural language inference dataset, which is also solved via classification.
- MR v1.0[8]: Movie reviews with one sentence per review labeled positive or negative for sentiment classification.
- SST1[9]: an extension of MR but with fine-grained labels: very positive, positive, neutral, negative, very negative.
- SST2[10]: same as SST1 but with neutral reviews removed and only using positive and negative labels.
- TREC[11]: question samples that classify each question into one of 6 question types: about person, location, numeric information, etc.

Table 2: Summary statistics for the datasets after tokenization. c denotes the number of target classes.

Data	c	Train	Dev	Test
SNLI	3	549367	9842	9842
MR	2	8529	1067	1066
SST1	5	8544	1101	2210
SST2	2	6920	872	1821
TREC	6	5452	500	500

Baseline: Four widely used methods are employed as the baselines:

(1) a 1-layer LSTM (Hochreiter and Schmidhuber, 1997) with 300 hidden units;

[8] https://www.cs.cornell.edu/people/pabo/movie-review-data/

[9] http://nlp.stanford.edu/sentiment/

[10] http://nlp.stanford.edu/sentiment/

[11] https://cogcomp.seas.upenn.edu/Data/QA/QC/

(2) a 3-layer Transformer (Vaswani et al., 2017) with 300 hidden units;

(3) CNN (Kim, 2014): We use exactly the same settings as the paper;

(4) Capsule Network (Zhao et al., 2018). We adopted the code released by the authors and used trainable embeddings.

For our model, we adopt a MLP with 1 hidden layer (300 units) for forward transform and a MLP with 2 hidden layers (300 units) for backward transform. We set the length of semantic base to 1000.

Training details: We adopt uniform settings for all baselines and our model: 1) Adam optimizer for parameter updating with learning rate of 1e-4; trainable embeddings with size 300. 2) A MLP with 1 hidden layer as the classifier. For a fair comparison, the hidden unit size is set to 300 for LSTM, CNN, Transformer and Capsule. For our model, it is set to 64 when we use sparse representation to do the prediction and still 300 when we use back transformation representations as the prediction features.[12] 3) SNLI is the task of identifying the relationships between two given sentences. For each model, we first use it to encode the two sentences into the resulting representations respectively, and then concatenate the two sentence representations for the final prediction. 4) We report the average accuracy over 10 runs of the experiment on the test data. For each run, the maximum accuracy before early stopping is selected as the result of the current run.

3.1 Results and Analysis

Table 1 shows the prediction accuracy of our model and the baselines. Table 3 gives the prediction run time. From Tables 1 and 3, we can make the following observations:

• The proposed Semantic Transform (ST) approach significantly outperforms LSTM on three datasets: SST1, SST2 and MR, and obtain comparable results with LSTM on SNLI and TREC. ST also markedly outperforms Transformer and Capsule on all five datasets, and outperforms CNN on four out of five datasets. Therefore, we can draw the conclusion that ST is an effective method to learn sentence representations in both dense and sparse spaces.

• ST^y (including ST^y[X'] and ST^y[Y]) performs much worse than the proposed sparse activation method, which indicates the effectiveness of the proposed method. ST^z (including ST^z[X'] and ST^z[Y]) shows the proposed sparse activation plays an important role in our system, and it's very effective. And we will show that the proposed sparse activation method can ensure good sparseness of the representation through the analysis below. The relatively worse results of ST^z (including ST^z[X'] and ST^z[Y]) also confirmed the effectiveness of helper losses. This part can be seen as our ablation study. Our activation function, margin loss, and base regularization are all shown to play important roles in our method.

• In terms of efficiency, Table 3 shows that ST is 2-3 times faster than LSTM. ST is also markedly faster than Capsule and Transformer on all datasets. CNN is known as the fastest model and our method achieves comparable speeds with CNN.

Table 3: Average running time over all test sets (Minute)

Model	SNLI	MR	SST1	SST2	TREC
CNN	1.190	0.108	0.083	0.079	0.035
Transformer	1.810	0.151	0.140	0.137	0.041
Capsule	3.590	0.303	0.220	0.206	0.057
LSTM	2.096	0.186	0.168	0.142	0.049
ST	1.404	0.093	0.088	0.071	0.025

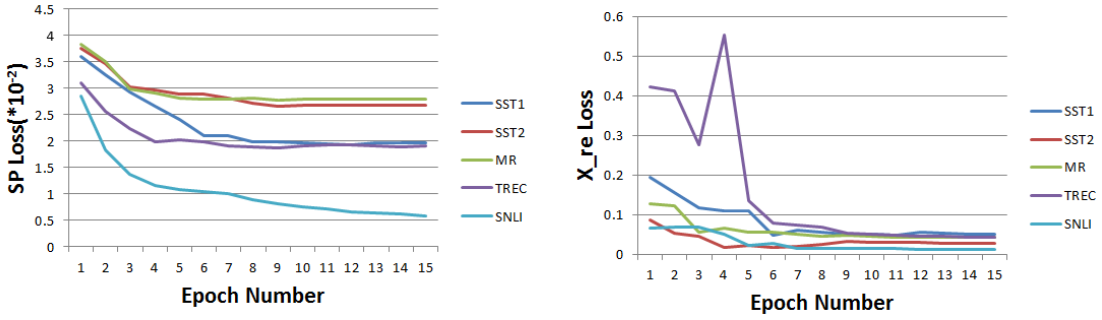
In summary, considering that our work is only the first attempt, it performs quite well compared with highly researched and optimized LSTM, CNN, Capsule and Transformer models. We foresee that future work will significantly optimize our method.

Sparsity Analysis: Figure 2(a) shows the sparsity of the word sparse representations of all datasets. Sparsity is evaluated using the Sparse Evaluation (SE) function. We proposed this method because previous methods were not designed for sparse representations with both pos-

Table 4: Distribution of values in the sparse representations (%). $V > 0.6$ ($V < 0.05$) shows the frequency of the values greater (less) than 0.6 (0.05)

Metrics	SNLI	MR	SST1	SST2	TREC
$V > 0.6$	0.14	1.38	1.31	1.71	1.38
$V < 0.05$	99.68	97.39	97.21	96.36	97.16

[12] In detail, the number of parameter of the classifiers for baselines and our model using back transformation representations is 300*300=90,000; while the number for our model using sparse representation is 1000*64=64,000.



(a) (b)

Figure 2: a) Sparsity evaluation of sparse word representations (the legend is explained below);b) Evaluation of the construction of X.

itive and negative values:

$$SE(\mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{i=1}^{j\mathcal{D}j} (\sin(\pi y_i))^2 \tag{13}$$

As function $(\sin(\pi y))^2$ has only three minimum points, -1, 0, 1, it is suitable for measuring the concentration degree of the components of sparse representations. Figure 2(a) shows a clear decline of SPLoss, which indicates a high concentration degree. Table 4 also gives the statistics about the distributions of the sparse representations. We can see that 'zero' ($V < 0.05$) takes a large portion of the sparse representations, which is desirable. We can conclude that the learned sparse representations are indeed sparse.

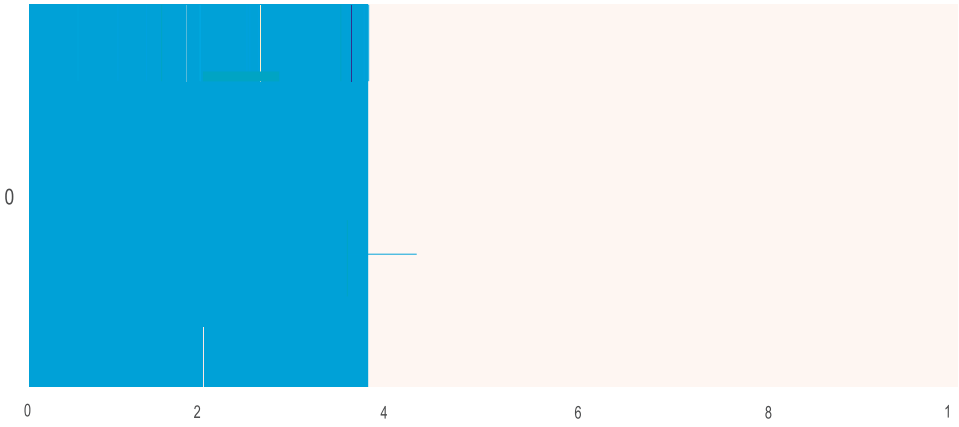


Figure 3: Visualization of learned sparse representations.

Accuracy of Transformation: We asked a question about the ability of ST to construct LSTM when we introduced the RL^o. Here, we analyze the transformation accuracy of the proposed method and give a positive answer to that question. From Table 1, we can see that ST[X'] achieves very similar results to those of LSTM. From the results, we can draw the conclusion that the dense representation generated by ST through backward transformation can achieve very similar results to those of LSTM. Further, we propose a measure to gauge the construction accuracy, named Construction Accuracy Metric (CAM), to evaluate the accuracy of transformation. CAM is formulated as the following function (results are shown in Figure 2(b)):

$$CAM(\mathcal{C}) = \frac{1}{J|\mathcal{C}|} \sum_{i=1}^{j\mathcal{C}j} \sum_{j=1}^{J} \frac{|X_{ij} - X_{ij}^0|_2^2}{0.5 * |X_{ij}|_2^2 + 0.5 * |X_{ij}^0|_2^2} \tag{14}$$

where X_{ij} is the original dense representation of a sub-sentence (generated by LSTM) and X_{ij}^0 is the

backward transformation result of its sparse representation; $\mathcal{C}$ denotes the test set, and J is the length of the sentence. Clearly, this function can evaluate the similarity between X and X^0 as CAM will raise with the increasing of distance between X and X^0. Figure 2(b) shows that the difference between X and X^0 is only about 5%. Therefore, we can conclude that our model can construct the outputs of LSTM well.

Interpretability Analysis: Interpretability is one of the most desirable properties of sparse representations. Figure 3 shows the average sparse representation of five classes (tested on the test set of SST1) with different sentiment polarities (-2, -1, 0, 1, 2). Positive numbers refer to positive sentiment, and negative numbers refer to negative sentiment. In order to clearly visualize the differences in the learned representations over the five classes, we sort the bases based on the ascending order of the sparse representation values of +2 (very positive) class.

From Figure 3, we can see that there is a clear color difference for sentiment polarity class +2 and class -2. We can also see a similar phenomenon for sentiment polarity class +1 and class -1 but less pronounced as the their polarities are more similar. These observations demonstrate that the same bases obtain opposite values for classes of opposite sentiments. The bases generating distinct responses for classes with different sentiment polarities can be regarded as primary sentiment bases as they clearly indicate the semantic differences of the classes. In other words, the primary sentiment bases can be explained as sentiment bases. For example, the bases give positive response to positive classes but negative responses to negative classes are the positive sentiment bases, which directly indicate the sentiment polarities.

Comparing with positive and negative classes, neutral class shows relative mixed responses. That means neutral class has similar semantemes to those of both positive and negative classes. This demonstrates that the neutral class is more difficult to identify.

4 Related Work

Sparse embeddings have been used in image (Ji et al., 2019; Zhou et al., 2016; Zhang and Patel, 2016), signal (Caiafa and Cichocki, 2013; Huang and Aviyente, 2007), and NLP (Subramanian et al., 2018; Kober et al., 2016) applications.

Several sparse models have been proposed to produce sparse embeddings. For example, some previous works trained word embeddings with sparse or non-negative constraints (Murphy et al., 2012; Luo et al., 2015). Linguistically inspired dimensions (Faruqui et al., 2015) is another way to increase sparsity and interpretability. SPINE (SParse Interpretable Neural Embeddings) (Subramanian et al., 2018), a variant of denoising k-sparse autoencoder, can generate efficient and interpretable distributed word representations. Our method is different from these approaches. We not only construct sparse representations but also transform between dense and sparse spaces. We also combine word sparse representations to produce sentence representations. Some recent studies tried to achieve sparsity in novel ways (Park et al., 2017). We also proposed a novel method in this paper and experimentally verified its effectiveness.

5 Conclusion and Future Works

This paper proposed a novel method to transform representations between dense and sparse spaces, and a technique to combine semantics in the sparse space. It also proposed and experimentally verified a new activation function that can be used to achieve sparseness. Natural language inference and text classification tasks were used to evaluate the proposed transformations with promising results. Based on this study, many other interesting directions can be pursued in the future, e.g.,

(1) As we discussed in the paper, the proposed method can construct the output of LSTM well. One future work is to apply ST to language modeling. In this case, the results can be used in many down stream tasks such as machine translation and dialogue systems.

(2) With the help of ST, we can investigate the style transfer on similar tasks in the sparse space by direct semantic reversing. Also, we can use ST to filter out noises or undesirable information.

(3) Based on sparse representations, we can also explore semantic pattern recognition and transformation.

Acknowledgement

This work was partially supported by the National Key Research and Development Program of China under grant 2018AAA0100205.

References

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*.

Cesar F Caiafa and Andrzej Cichocki. 2013. Computing sparse representations of multidimensional signals using kronecker bases. *Neural computation*, 25(1):186–220.

Ting-Yun Chang, Ta-Chung Chi, Shang-Chi Tsai, and Yun-Nung Chen. 2018. xsense: Learning sense-separated sparse representations and textual definitions for explainable word sense networks. *arXiv preprint arXiv:1809.03348*.

Yong Cheng. 2019. Semi-supervised learning for neural machine translation. In *Joint Training for Neural Machine Translation*, pages 25–40. Springer.

Xinya Du and Claire Cardie. 2017. Identifying where to focus in reading comprehension for neural question generation. In *EMNLP*, pages 2067–2073.

Manaal Faruqui and Chris Dyer. 2015. Non-distributional word vector representations. *arXiv preprint arXiv:1506.05230*.

Manaal Faruqui, Yulia Tsvetkov, Dani Yogatama, Chris Dyer, and Noah Smith. 2015. Sparse overcomplete word vector representations. *arXiv preprint arXiv:1506.02004*.

Alona Fyshe, Partha P Talukdar, Brian Murphy, and Tom M Mitchell. 2014. Interpretable semantic vectors from a joint model of brain-and text-based meaning. In *ACL*, volume 2014, page 489. NIH Public Access.

Kuzman Ganchev, Ben Taskar, Fernando Pereira, and Joao Gama. 2009. Posterior vs parameter sparsity in latent variable models. In *Advances in Neural Information Processing Systems*.

Jiang Guo, Wanxiang Che, Haifeng Wang, and Ting Liu. 2014. Revisiting embedding features for simple semi-supervised learning. In *EMNLP*, pages 110–120.

Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tie-Yan Liu, and Wei-Ying Ma. 2016. Dual learning for machine translation. In *Advances in Neural Information Processing Systems*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Ke Huang and Selin Aviyente. 2007. Sparse representation for signal classification. In *Advances in neural information processing systems*, pages 609–616.

MingShu Ji, Hong Rao, ZhiXun Li, Jian Zhu, and Ning Wang. 2019. Partial multi-view clustering based on sparse embedding framework. *IEEE Access*.

Seonhoon Kim, Inho Kang, and Nojun Kwak. 2019. Semantic sentence matching with densely-connected recurrent and co-attentive information. In *AAAI*, volume 33, pages 6586–6593.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Thomas Kober, Julie Weeds, Jeremy Reffin, and David Weir. 2016. Improving sparse word representations with distributional inference for semantic composition. *arXiv preprint arXiv:1608.06794*.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Multi-task deep neural networks for natural language understanding. *arXiv preprint arXiv:1901.11504*.

Hongyin Luo, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2015. Online learning of interpretable word embeddings. In *EMNLP*.

André FT Martins, Noah A Smith, Pedro MQ Aguiar, and Mário AT Figueiredo. 2011. Structured sparsity in structured prediction. In *EMNLP*, pages 1500–1511. Association for Computational Linguistics.

Brian Murphy, Partha Talukdar, and Tom Mitchell. 2012. Learning effective and interpretable semantic models using non-negative sparse embedding. *Proceedings of COLING 2012*, pages 1933–1950.

Andrew Ng et al. 2011. Sparse autoencoder. *CS294A Lecture notes*, 72(2011):1–19.

Sungjoon Park, JinYeong Bak, and Alice Oh. 2017. Rotated word vector representations and their interpretability. In *EMNLP*.

Iulian Vlad Serban, Tim Klinger, Gerald Tesauro, Kartik Talamadupula, Bowen Zhou, Yoshua Bengio, and Aaron Courville. 2017. Multiresolution recurrent neural networks: An application to dialogue response generation. In *AAAI*.

Anant Subramanian, Danish Pruthi, Harsh Jhamtani, Taylor Berg-Kirkpatrick, and Eduard Hovy. 2018. Spine: Sparse interpretable neural embeddings. In *AAAI*.

Fei Sun, Jiafeng Guo, Yanyan Lan, Jun Xu, and Xueqi Cheng. 2016. Sparse word embeddings using l1 regularized online learning. In *IJCAI*.

Duyu Tang, Bing Qin, Furu Wei, Li Dong, Ting Liu, and Ming Zhou. 2015. A joint segmentation and classification framework for sentence level sentiment classification. *TASLP*, 23(11).

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Yequan Wang, Aixin Sun, Jialong Han, Ying Liu, and Xiaoyan Zhu. 2018. Sentiment analysis by capsules. In *WWW*, pages 1165–1174.

Fangzhao Wu, Jia Zhang, Zhigang Yuan, Sixing Wu, Yongfeng Huang, and Jun Yan. 2017. Sentence-level sentiment classification with weak supervision. In *SIGIR*, pages 973–976.

Dani Yogatama and Noah A Smith. 2014. Linguistic structured sparsity in text categorization. In *ACL*, volume 1, pages 786–796.

He Zhang and Vishal M Patel. 2016. Sparse representation-based open set recognition. *IEEE transactions on pattern analysis and machine intelligence*, 39(8):1690–1696.

Yizhe Zhang, Xiang Gao, Sungjin Lee, Chris Brockett, Michel Galley, Jianfeng Gao, and Bill Dolan. 2019. Consistent dialogue generation with self-supervised feature learning. *arXiv preprint arXiv:1903.05759*.

Wei Zhao, Jianbo Ye, Min Yang, Zeyang Lei, Suofei Zhang, and Zhou Zhao. 2018. Investigating capsule networks with dynamic routing for text classification. *arXiv preprint arXiv:1804.00538*.

Xiaowei Zhou, Menglong Zhu, Spyridon Leonardos, and Kostas Daniilidis. 2016. Sparse representation for 3d shape estimation: A convex relaxation approach. *IEEE transactions on pattern analysis and machine intelligence*, 39(8):1648–1661.