Investigating Capsule Network and Semantic Feature on Hyperplanes for Text Classification

[1,2]Chunning Du [*], [1,2]Haifeng Sun[*], [1,2]Jingyu Wang[†], [1,2]Qi Qi, [1,2]Jianxin Liao
[1,2]Chun Wang, [1,2]Bing Ma
[1]State Key Laboratory of Networking and Switching Technology,
Beijing University of Posts and Telecommunications, Beijing 100876, China
[2]EBUPT Information Technology Co., Ltd., Beijing 100191, China
{duchunning,sunhaifeng_1,wangjingyu,qiqi}@ebupt.com

Abstract

As an essential component of natural language processing, text classification relies on deep learning in recent years. Various neural networks are designed for text classification on the basis of word embedding. However, polysemy is a fundamental feature of the natural language, which brings challenges to text classification. One polysemic word contains more than one sense, while the word embedding procedure conflates different senses of a polysemic word into a single vector. Extracting the distinct representation for the specific sense could thus lead to fine-grained models with strong generalization ability. It has been demonstrated that multiple senses of a word actually reside in linear superposition within the word embedding so that specific senses can be extracted from the original word embedding. Therefore, we propose to use capsule networks to construct the vectorized representation of semantics and utilize hyperplanes to decompose each capsule to acquire the specific senses. A novel dynamic routing mechanism named 'routing-on-hyperplane' will select the proper sense for the downstream classification task. Our model is evaluated on 6 different datasets, and the experimental results show that our model is capable of extracting more discriminative semantic features and yields a significant performance gain compared to other baseline methods.

1 Introduction

Text classification is a crucial task in natural language processing, which has many applications, such as sentiment analysis, intent identification and topic labeling[Aggarwal and Zhai, 2012; Wang and Manning, 2012a]. Recent years, many studies rely on neural networks and have shown promising performance.

The success of deep learning model for NLP is based on the progress in learning distributed word representations in semantic vector space, where each word is mapped to a vector called a word embedding. The word's representation is calculated relying on the *distributional hypothesis* - the assumption that semantically similar or related words appear in similar contexts [Mikolov *et al.*, 2013; Langendoen, 1959]. Normally, each word's representation is constructed by counting all its context features. However, for the polysemic word which contains multiple senses, the context features of different senses are mixed together, leading to inaccurate word representation. As demonstrated in [Arora *et al.*, 2018], multiple senses of a word actually reside in linear superposition within the word embedding:

$$v \approx \alpha_1 v_{sense1} + \alpha_2 v_{sense2} + \alpha_3 v_{sense3} + \cdots , \quad (1)$$

where coefficients α_i are nonnegative and $v_{sense1}, v_{sense2}...$ are the hypothetical embeddings of different senses. As a result, the word embedding v deviates from any sense, which brings ambiguity for the subsequent task. It demands us to extract the separate senses from the overall word representation to avoid the ambiguity.

Similar to the word embedding, the recently proposed capsule network constructs vectorized representations for different entities[Hinton *et al.*, 2018; Sabour *et al.*, 2017; Hinton *et al.*, 2011]. A dynamic routing mechanism, 'routing-by-agreement', is implemented to ensure that the output of the capsule gets sent to an appropriate parent in the layer above. Very recently, capsule network is applied in the field of NLP where each capsule is obtained from the word embedding. Compared with the standard neural nets using a single scalar (the output of a neural unit) to represent the detected semantics, the vectorized rep-

[*] Authors contributed equally.
[†] Corresponding author.

Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, pages 456–465,
Hong Kong, China, November 3–7, 2019.

resentation in capsule network enables us to utilize hyperplanes to extract the component from the overall representation and get the specific sense.

Therefore, we propose to attach different hyperplanes to capsules to tackle the ambiguity caused by polysemy. Each capsule is decomposed by projecting the output vector on the hyperplanes, which can extract the specific semantic feature. The projected capsule denotes a specific sense of the target words, and a novel dynamic routing mechanism named 'routing-on-hyperplane' will decide which specific senses are selected for the downstream classification and which senses are ignored. Similar to routing-by-agreement [Sabour *et al.*, 2017], we aim to activate a higher-level capsule whose output vector is agreed with the predictions from the lower-level capsules. Differently, before the active capsule at a level makes predictions for the next-level capsules, the capsule's output vector will be projected on the trainable hyperplane. The hyperplanes will be trained discriminatively to extract specific senses. Moreover, in order to encourage the diversity of the hyperplanes, a well-designed penalization term is implemented in our model. We define the cosine similarity between the normal vectors of the hyperplanes as a measure of redundancy, and minimize it together with the original loss.

We test our model (HCapsNet) on the text classification task and conduct extensive experiments on 6 datasets. Experimental results show that the proposed model could learn more discriminative features and outperform other baselines. Our main contributions are summarized as follows:

- We explore the capsule network for text classification and propose to decompose capsules by means of projecting on hyperplanes to tackle the polysemy problem in natural language.
- Propose routing-on-hyperplane to dynamically select specific senses for the subsequent classification. A penalization term is designed to obtain diversified hyperplanes and offer multiple senses representations of words.
- Our work is among the few studies which prove that the idea of capsule networks have promising applications on natural language processing tasks.

2 Related Work

2.1 Neural Networks for Text Classification

Various neural networks for text classification have been proposed based on the word embedding. Commonly used models include convolutional neural networks [Kim, 2014], recursive neural network [Socher *et al.*, 2013] and recurrent neural networks. There have been several recent studies of CNN for text classification in the large training dataset and deep complex model structures [Schwenk *et al.*, 2017; Johnson and Zhang, 2017]. Some models were proposed to combine the strength of CNN and RNN [Lai *et al.*, 2015; Zhang *et al.*, 2016]. Moreover, the accuracy was further improved by attention-based neural networks[Lin *et al.*, 2017; Vaswani *et al.*, 2017; Yang *et al.*, 2016]. However, these models are less efficient than capsule networks.

As a universal phenomenon of language, polysemy calls much attention of linguists. It has been demonstrated that learning a distinct representation for each sense of an ambiguous word could lead to more powerful and fine-grained models based on vector-space representations [Li and Jurafsky, 2015].

2.2 Capsule Network

Capsule network was proposed to improve the representational limitations of CNN and RNN by extracting features in the form of vectors. The technique was firstly proposed in [Hinton *et al.*, 2011] and improved in [Sabour *et al.*, 2017] and [Hinton *et al.*, 2018]. Vector-based representation is able to encode latent inter-dependencies between groups of input features during the learning process. Introducing capsules also allows us to use routing mechanism to generate high-level features which is a more efficient way for feature encoding.

Several types of capsule networks have been proposed for natural language processing. Yang *et al.* [2018] investigated capsule networks with routing-by-agreement for text classification. They also found that capsule networks exhibit significant improvement when transfer single-label to multi-label text classification. Capsule networks also show a good performance in multi-task learning [Xiao *et al.*, 2018]. Xia *et al.* [2018] discovered the capsule-based model's potential on zero-shot learning. However, existing capsule networks for natural language processing cannot model the

polysemic words or phrases which contain multiple senses.

3 Model

In this section, we begin by introducing the idea of routing-on-hyperplane and formulate it in details. Then the architecture of the HCapsNet is formally presented in the second subsection. Finally, the penalization term and loss function implemented in this paper are explained.

3.1 Routing-on-hyperplane

Suppose that we have already decided on the output vectors of all the capsules in the layer L and we now want to decide which capsules to activate in the layer $L+1$. We should also consider how to assign each active capsule in the layer L to one active capsule in the layer $L+1$. The output vector of capsule i in the layer L is denoted by $\boldsymbol{u}_i$, and the output vector of capsule j in the layer $L+1$ is denoted by $\boldsymbol{v}_j$.

Firstly, for all the capsules in the layer L, we attach the trainable hyperplane to each capsule. Capsule's output vectors will be projected on the hyperplanes before making predictions. More specifically, for capsule i, we define the trainable matrix $\boldsymbol{W}_i^h$, which is used to decide the normal vector $\boldsymbol{w}_i$ of the attached hyperplane. By restricting $\|\boldsymbol{w}_i\|_2 = 1$, we can get the projected capsule's output vector $\boldsymbol{u}_{\perp i}$:

$$\boldsymbol{w}_i = \boldsymbol{W}_i^h \boldsymbol{u}_i. \tag{2}$$

$$\boldsymbol{u}_{\perp i} = \boldsymbol{u}_i - \boldsymbol{w}_i^{\mathsf{T}} \boldsymbol{u}_i \boldsymbol{w}_i. \tag{3}$$

In this way, the output vectors of capsules will be projected on the specific hyperplanes to get different components which denote the specific senses in our task. To retain or ignore a specific sense (projected capsule) will be decided through an iterative procedure. The procedure contains making predictions and calculating the agreement. When one projected capsule's prediction is highly agreed with one target parent capsule, the probability of retaining the projected capsule gets gradually larger. In another word, when a specific sense is highly relevant with the subsequent classification, we choose to keep it and ignore others.

Therefore, the $\boldsymbol{u}_{\perp i}$ will then be used to make predictions for the $L+1$ layer's capsules and calculate coupling coefficients c_{ij}. When making predictions, the capsules in the layer L will multiply their projected output vector $\boldsymbol{u}_{\perp i}$ by a weight matrix $\boldsymbol{W}_{ij}$:

$$\hat{\boldsymbol{u}}_{\perp j|i} = \boldsymbol{W}_{ij} \boldsymbol{u}_{\perp i}, \tag{4}$$

where $\hat{\boldsymbol{u}}_{\perp j|i}$ denotes the 'vote' of the capsule i for the capsule j. The agreement between the prediction vector $\hat{\boldsymbol{u}}_{\perp j|i}$ with current output vector of parent capsule j will be fed back to the coupling coefficients c_{ij} between the two capsules: increase c_{ij} if highly agreed. Similar with [Sabour *et al.*, 2017] we define the agreement as scalar product between the two vectors. b_{ij} is the accumulation of the agreement after each iteration and the softmax function is implemented to ensure the coupling coefficients between the capsule i and all the capsules in the layer above sum to one:

$$b_{ij} \leftarrow b_{ij} + \hat{\boldsymbol{u}}_{\perp j|i} \cdot \boldsymbol{v}_j \tag{5}$$

$$c_{ij} = \frac{exp(b_{ij})}{\sum_k exp(b_{ik})} \tag{6}$$

Algorithm 1 Routing-on-hyperplane returns the output vector of capsule j in the layer $L+1$ given the output vector of capsule i in the layer L. $\boldsymbol{W}_{ij}$ are trainable parameters denoting the transformation matrix between the two adjacent layers. $\boldsymbol{W}_i^h$ are trainable parameters for each capsule i to calculate the proposed hyperplane's normal vectors $\boldsymbol{w}_i$, we restrict that $\|\boldsymbol{w}_i\|_2 = 1$.

initialize the routing logits:
for all capsule i in the layer L and capsule j in the layer $L+1$: $b_{ij} \leftarrow 0$;
for every capsule i in the layer L: $\boldsymbol{w}_i = \boldsymbol{W}_i^h \boldsymbol{u}_i$
for every capsule i in the layer L: $\boldsymbol{u}_{\perp i} \leftarrow \boldsymbol{u}_i - \boldsymbol{w}_i^{\mathsf{T}} \boldsymbol{u}_i \boldsymbol{w}_i$
for every capsule i in the layer L: $\hat{\boldsymbol{u}}_{\perp j|i} = \boldsymbol{W}_{ij} \boldsymbol{u}_{\perp i}$
for r iterations **do**
  for all capsule i and j: $c_{ij} \leftarrow \frac{exp(b_{ij})}{\sum_k exp(b_{ik})}$
  for all capsule j in the layer $L+1$: $\boldsymbol{s}_{\perp j} \leftarrow \sum_i c_{ij} \hat{\boldsymbol{u}}_{\perp j|i}$
  for all capsule j in the layer $L+1$: $\boldsymbol{v}_j \leftarrow \frac{\|\boldsymbol{s}_{\perp j}\|^2}{1+\|\boldsymbol{s}_{\perp j}\|^2} \frac{\boldsymbol{s}_{\perp j}}{\|\boldsymbol{s}_{\perp j}\|}$
  for all capsule i and capsule j: $b_{ij} \leftarrow b_{ij} + \hat{\boldsymbol{u}}_{\perp j|i} \cdot \boldsymbol{v}_j$
end for
return $\boldsymbol{v}_j$

Each iteration will result in a temporary output vector of the capsule j: the weighted sum over all prediction vectors $\hat{\boldsymbol{u}}_{\perp j|i}$ using coefficient c_{ij}. Moreover, to ensure the length of the output vector of capsule j is able to represent the probability and prevent it from being too big, we use a non-linear 'squashing' function to make the vector's length range from zero to one without changing the vector's direction:

$$\boldsymbol{s}_{\perp j} = \sum_i c_{ij} \hat{\mathbf{u}}_{\perp j|i}. \tag{7}$$

$$\boldsymbol{v}_j = \frac{\|\boldsymbol{s}_{\perp j}\|^2}{1 + \|\boldsymbol{s}_{\perp j}\|^2} \frac{\boldsymbol{s}_{\perp j}}{\|\boldsymbol{s}_{\perp j}\|}. \tag{8}$$

The $\boldsymbol{v}_j$ will then be returned as input to calculate the agreement for the next iteration. The coupling coefficients c_{ij} and the output vector of capsule j gradually converge after several iterations. After the last iteration of the routing process, the coupling coefficients c_{ij} is determined. Hyperplane plays the role to extract specific senses and assist to route the lower-level capsules to the right parent capsules. We detail the whole routing-on-hyperplane algorithm in Algorithm 1.

3.2 HCapsNet Model Architecture

We propose a model named HCapsNet for text classification based on the theory of capsule network and routing-on-hyperplane. The architecture is illustrated in Figure 1. The model consists of three layers: one bi-directional recurrent layer, one convolutional capsule layer, and one fully connected capsule layer. The input of the model is a sentence S consisting of a sequence of word tokens $t_1, t_2, ..., t_n$. The output of the model contains a series of capsules. Each top-level capsule corresponds to a sentence category. The length of the top-level capsule's output vector is the probability p that the input sentence S belongs to the corresponding category.

The recurrent neural network can capture long-distance dependencies within a sentence. For this strength, a bi-directional recurrent neural network is the first layer of HCapsNet. We concatenate the left context and the right context as the word's elementary representation x_i, which is the input to the second layer:

$$c_l(t_i) = \overrightarrow{RNN}(t_i), \tag{9}$$

$$c_r(t_i) = \overleftarrow{RNN}(t_i), \tag{10}$$

$$x_i = [c_l(t_i), c_r(t_i)]. \tag{11}$$

The second layer is a convolutional capsule layer. This is the first layer consisting of capsules, we call capsules in this layer as primary capsules. Primary capsules are groups of detected features which means piecing instantiated parts together to make familiar wholes. Since the output of the bi-directional recurrent neural network is not in the form of capsules, no routing method is used in this layer.

The final layer is fully connected capsule layer. Each capsule corresponds to a sentence class. All the capsules in this layer receive the output of the lower-level capsules by the routing-on-hyperplane method as we described in Section 3.1. The length of the top-level capsule's output vector represents the probability that the input sentence belongs to the corresponding category.

3.3 Penalization Term

The HCapsNet may suffer from redundancy problem if the output vectors of capsules are always getting projected on the similar hyperplanes at the routing-on-hyperplane procedure. Thus, we need a penalization term to encourage the diversity of the hyperplanes. We introduce an easy penalization term with low time complexity and space cost. Firstly, we construct a matrix X_i the columns of which is the normal vectors w of the hyperplanes for the ith word. The dot product of X_i and its transpose, subtracted by an identity matrix is defined as a measure of redundancy. The penalization term is the sum of all the words' redundancy:

$$P_i = ||(X_i X_i^{\mathsf{T}} - I)||_F^2. \tag{12}$$

$$P = \sum_i P_i, \tag{13}$$

where $||\bullet||_F$ stands for the Frobenius norm of a matrix. Similar to adding the L2 regularization term, this penalization term P will be multiplied by a coefficient, and we minimize it together with the original loss.

Let's consider the two columns w^a and w^b in X_i, which are two normal vectors of hyperplanes for the ith word. We have restricted that $||w|| = 1$ as described in Algorithm 1. For any non-diagonal elements x_{ab} $(a \neq b)$ in the $X_i X_i^{\mathsf{T}}$ matrix, it corresponds to the cosine similarity between the two

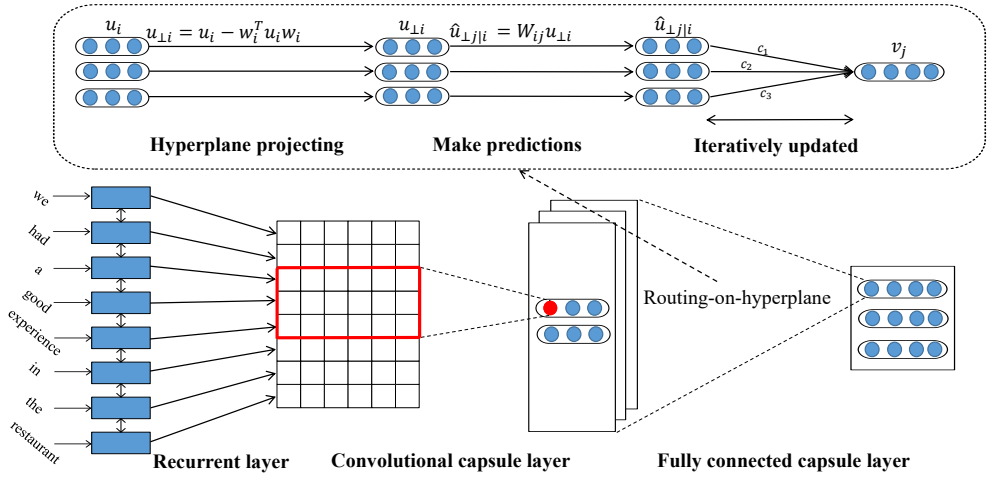

Figure 1: The architecture of HCapsNet

normal vectors:

$$-1 < x_{ab} = \sum_k w_k^a w_k^b < 1. \quad (14)$$

where w_k^a and w_k^b are k-th element in the w^a and w^b vectors, respectively. In the most extreme case, where the two normal vectors of hyperplanes are orthometric with each other, i.e. the word is projected to two extremely different meanings, the corresponding x_{ab} is 0. Otherwise, the absolute value will be positive. In the other most extreme case, where the two normal vectors of hyperplanes are identical, i.e. the word is projected to the same vector, the corresponding absolute value of x_{ab} is 1. The diagonal elements x_{ab} $(a = b)$ in the $X_i X_i^\mathsf{T}$ matrix is the normal vectors' cosine similarity with themselves, so they are all 1. The $X_i X_i^\mathsf{T}$ is subtracted by an identity matrix I so as to eliminate the meaningless elements. We minimize the Frobenius norm of P_i to encourage the non-diagonal elements in P_i to converge to 0, in another word, to encourage word vector to be projected on orthometric hyperplanes and get diversified explanation.

3.4 Loss Function

In HCapsNet, each top-level capsule corresponds to a sentence category. The length of the top-level capsule's output vector represents the probability that the input sentence belongs to the corresponding category. We would like the top-level capsule for the category k to have a long output vector if the input sentence belongs to the category k and have a short output vector if the input sentence does not belong to the category k. Similar with [Sabour *et al.*, 2017], We use a separate margin loss, L_k for each top-level capsule k. The total loss L is simply the sum of the losses of all top-level capsules:

$$L = \sum_k \{T_k \max(0, m^+ - ||\boldsymbol{v_k}||)^2 + \lambda_1 (1 - T_k) \max(0, ||\boldsymbol{v_k}|| - m^-)^2\} + \lambda_2 P, \quad (15)$$

where T_k will be 1 if the sentence belongs to the k class, or else T_k will be 0. $||\boldsymbol{v_k}||$ is the length of the output vector of capsule k. We introduce λ_1 to reduce the penalization to avoid shrinking the length of the capsules' output vectors in the initial learning stage. P is the penalization term introduced in Section 3.3. In our experiments, $m^+ = 0.9$, $m^- = 0.1$, $\lambda_1 = 0.5$.

4 Experiments

We compare our method with the widely used text classification methods and baseline models (listed in Table 1).

4.1 Datasets

HCapsNet is evaluated on six widely studied datasets including three common text classification tasks: sentiment analysis, question classification and topic classification. These datasets are Stanford Sentiment Treebank [Socher *et al.*, 2013], Movie Review Data [Pang and Lee, 2005], Subjectivity dataset [Pang and Lee, 2004], TREC [Li and Roth, 2002] and AG's corpus of news articles [Zhang *et al.*, 2015b]. Summary statistics of the datasets are listed in Table2.

Method	SST-2	SST-5	MR	Subj	TREC	AG's news
SVM [Socher *et al.*, 2013]	79.4	40.7	-	-	-	-
NB [Socher *et al.*, 2013]	81.8	41.0	-	-	-	-
NBSVM-bi [Wang and Manning, 2012b]	-	-	79.4	93.2	-	-
Standard-LSTM	80.6	45.3	75.9	89.3	86.8	86.1
bi-LSTM	83.2	46.7	79.3	90.5	89.6	88.2
RCNN [Lai *et al.*, 2015]	-	47.21	-	-	-	
SNN [Zhao *et al.*, 2018]	-	50.4	82.1	93.9	**96**	-
CNN-non-static [Kim, 2014]	87.2	48.0	81.5	93.4	93.6	92.3
VD-CNN [Schwenk *et al.*, 2017]	-	-	-	88.2	85.4	91.3
CL-CNN [Zhang *et al.*, 2015a]	-	-	-	88.4	85.7	92.3
Capsule-B [Yang *et al.*, 2018]	86.8	-	82.3	93.8	93.2	92.6
HCapsNet	**88.7**	**50.8**	**83.5**	**94.2**	94.2	**93.5**

Table 1: Experimental results of our model compared with other models. Performance is measured in accuracy (%). Models are divided into 3 categories. The first part is baseline methods including SVM and Naive Bayes and their variations. The second part contains models about recurrent neural networks. The third part contains models about convolutional neural networks.

Dataset	Class	Len	V	Train	Dev	Test
SST-2	2	54	16185	6920	872	1821
SST-5	5	54	17836	8544	1101	2210
MR	2	58	18765	9596	-	1066
Subj	2	121	21323	9000	-	1000
TREC	6	37	9592	5452	-	500
AG's news	4	197	51379	120k	-	7.6k

Table 2: Summary statistics for the datasets.

4.2 Hyperparameters

In our experiments, we use 300-dimensional word2vec [Mikolov *et al.*, 2013] vectors to initialize word representations. In the first bi-directional RNN layer of HCapsNet, we use Long Short Term Memory network, the dimension of the hidden state is 256. The second layer contains 32 channels of primary capsules and the number of capsules in one channel depends on the sentence length. Each primary capsule contains 8 atoms which means that the dimension of the primary capsules is 8. The top-level capsules are obtained after 3 routing iterations. The dimension of the output vector of top-level capsules is 16. For all the datasets, we conduct mini-batch with size 25. We use Adam [Kingma and Ba, 2014] as our optimization method with $1e-3$ learning rate. λ_2 is 0.01.

4.3 Results and Discussions

Table 1 reports the results of our model on different datasets comparing with the widely used text classification methods and state-of-the-art approaches. We can have the following observations.

Our HCapsNet achieves the best results on 5 out of 6 datasets, which verifies the effectiveness of our model. In particular, HCapsNet outperforms vanilla capsule network Capsule-B[Yang *et al.*, 2018] by a remarkable margin, which only utilizes the dynamic routing mechanism without hyperplane projecting.

HCapNet does not perform best on the TREC dataset. One main reason maybe TREC dataset is used for question type classification, where samples are all question sentences. The task is mainly determined by interrogative words. For example, the sentence containing 'where' will probably be classified to 'location'. The ability to tackle polysemy doesn't play an important role. So, our model gets a similar result with Capsule-B.

4.4 Ablation Study

To analyze the effect of different components including hyperplane projection, penalization term, and routing iterations, we report the results of variants of HCapsNet in Table 4.

The results show that capsule network performs best when conducting 3 routing iterations, which stays in line with the conclusion in [Sabour *et*

Sentence	Projected representation for polyseme	Capsule-B	HCapsNet
like old myths and **[wonder]** tales spun afresh.	[0.4632, 0.1429, -0.2311, 0.4664, 0.2448, -0.6256, -0.0975, 0.1784]	P$^{\checkmark}$	P$^{\checkmark}$
as your relatives swap one mundane story after another , you begin to **[wonder]** if they are ever going to depart.	[-0.059, -0.3657, -0.3788, -0.0716, 0.4276, -0.5122, 0.5092, 0.092]	N$^{\times}$	P$^{\checkmark}$
you **[wonder]** why enough was n't just a music video rather than a full-length movie .	[0.0137, -0.2311, -0.4361, 0.0194, 0.5012, -0.5311, 0.4617, -0.0082]	P$^{\times}$	N$^{\checkmark}$
trouble every day is a success in some sense , but it 's hard to like a film so **[cold]** and dead	[0.1758, -0.3037, -0.3679, 0.0992, 0.3996, -0.6481, 0.3668, 0.1267]	N$^{\checkmark}$	N$^{\checkmark}$
the **[cold]** and dreary weather is a perfect metaphor for the movie itself , which contains few laughs and not much drama	[-0.3810, -0.3923, -0.3016, -0.3045, 0.2417, -0.3109, 0.5999, -0.0391]	N$^{\times}$	P$^{\checkmark}$

Table 3: Projected primary capsule's representations for polysemic words. P and N denote positive and negative classification results, respectively. ✓ denotes the right classification and × denotes the incorrect classification.

- Almost every scene in this film is a gem that could stand alone, a perfectly realized observation of mood, behavior and intent.
- A spunky, original take on a theme that will resonate with singles of many ages.
- The story drifts so inexorable into cliches about tortured lrb and torturing rrb artists and consuming but impossible love that you can't help but become more disappointed as each overwrought new sequence plods on.
- The premise is in extremely bad taste, and the film's supposed insights are so poorly thought out and substance free that even a high school senior taking his or her first psychology class could dismiss them.

Figure 2: Examples of routing results for SST-2.

al., 2017; Yang *et al.*, 2018]. Compared with the vanilla capsule network (row 3), applying routing-on-hyperplane brings a noticeable improvement (row 2). This demonstrates the necessity of integrating hyperplane projecting at the routing procedure to tackle the polysemy problems. Moreover, the penalization term described in Section 3.3 also marginally improves the accuracy, which proves that the orthogonal constraint on hyperplane is beneficial for text classification.

Hyperplane	Pena.	Iterations	Accuracy
✓	✓	3	**83.5**
✓	×	3	83.2
×	×	3	82.5
✓	✓	1	81.5
✓	✓	5	82.6

Table 4: Ablation study on MR dataset. "Pena." denotes Penalization Term in Section 3.3.

4.5 Case Study

Table 3 shows some sample cases from SST validation dataset, which are movie reviews for sentiment analysis. We analyze the attended primary capsule representation for the polysemic words in brackets. Specifically, we report the output vectors of the projected primary capsule, which is mostly attended by the routing mechanism.

The word 'wonder' in the first sample sentence means something that fills you with surprise and admiration, which shows a very positive sentiment. However, the polysemic word 'wonder' in the second and third sentences means to think about something and try to decide what is true, which is neutral in sentiment. We can observe that for the same word, the attended projected capsule representations are quite different according to different word senses. The projected representations for the same sense are similar, the Euclidean distance is 0.23 (row 2,3). On the contrary, for the different senses, the Euclidean distance is 1.12 (row 1,2). This property helps our model to make the predictions all correctly, while Capsule-B [Yang *et al.*, 2018] can not handle the latter two sentences. Similarly, the word 'cold' conveys two different senses in the last two samples (row 4-5), which means cruel and low temperature, respectively. The corresponding projected vectors are also quite different, which verifies the ability to tackle polysemy by routing-on-hyperplane.

4.6 Visualizing Routing Results

After several iterations of the routing algorithm, each primary capsule and the top-level capsule will be connected via a calculated coupling coefficient. The coupling coefficient corresponds to how much contribution of a low-level capsule to a specific high-level capsule. Routing-on-hyperplane can also be viewed as a parallel attention mech-

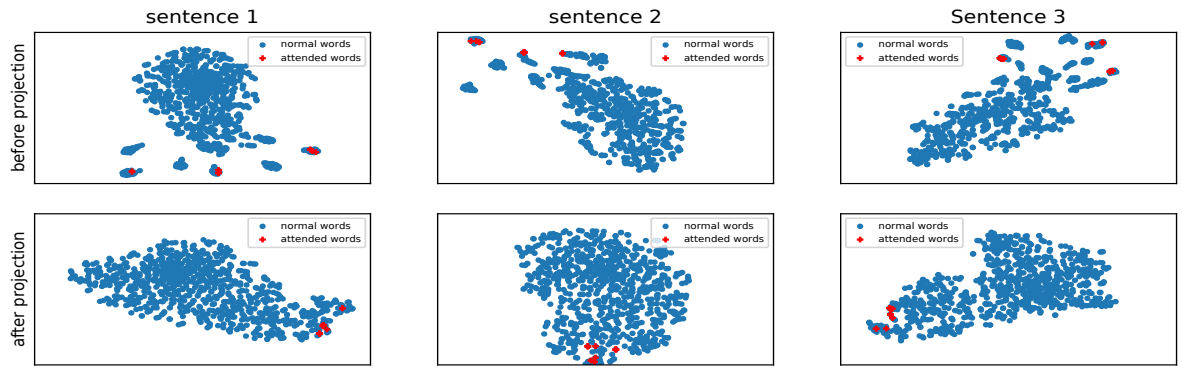

Figure 3: Illustrating the effect of the hyperplane.

anism that allows each capsule at one level to attend to some active capsules at the level below and to ignore others. We can thus draw a heat map to figure which phrases are taken into account a lot, and which ones are skipped by the routing-on-hyperplane in the task of text classification.

We randomly select 4 examples of reviews from the test set of SST, when the model has a high confidence (>0.8) in predicting the label. As shown in Figure 2, the words whose coupling coefficient greater than 0.7 are marked. It is easy to conclude that our routing method can effectively extract the sentimental words that indicate strongly on the sentiment behind the sentence and assign a greater coupling coefficient between the corresponding capsules. For example, '*gem*', '*spunky*', '*disappointed*', '*bad taste*' etc.

4.7 Visualizing Effects of The Hyperplane

In order to assess the effect of the hyperplane, we randomly select 3 examples in SST dataset and draw the distribution maps of primary capsules' output vectors before and after the projection operation respectively. As the dimension of the primary capsule's output vector is 8, T-Distributed Stochastic Neighbor Embedding (t-SNE) is performed on the vectors to reduce the dimension for visualization. As illustrated in Figure 3, the three pictures in the first line show the distribution before the projection operation for the three example sentences respectively. And the three pictures in the second line show the distribution after the projection. The blue points in the distribution maps denote the normal words and the red crosses denote the words attended by the routing algorithm which are defined in Section 4.6.

The relationship between the semantic capsules can be estimated by analyzing the distribution of the low-dimensional data in Figure 3. We find that originally scattered points which denote attended words converge after the projection. The attended words' projected vectors are close with each other, showing that they contain similar senses which are beneficial for the subsequent task. On the contrary, the capsules before projection contain multiple senses and show a scattered pattern. This demonstrates that the hyperplanes can effectively extract the guided senses and get attended by the routing-on-hyperplane mechanism.

5 Conclusion and Future Work

In this paper, we explore the capsule network for text classification and propose to decompose the capsule by means of projecting on hyperplanes to tackle the polysemy problem in natural language. Routing-on-hyperplane, a dynamic routing method, is implemented to select the sense-specific projected capsules for the subsequent classification task. We assess the effect of the hyperplane by case study and analyzing the distribution of the capsules' output vectors. The experiments demonstrate the superiority of HCapsNet and our proposed routing-on-hyperplane method outperforms the existing routing method in the text classification task.

In future, we would like to investigate the application of our theory in various tasks including reading comprehension and machine translating. We believe that capsule networks have broad applicability on the natural language processing tasks. Our core idea that decomposing the semantic capsules by projecting on hyperplanes is a necessary complement to capsule network to tackle the polysemy problem in various natural language processing tasks.

Acknowledgements

This work was jointly supported by: (1) National Natural Science Foundation of China (No. 61771068, 61671079, 61471063, 61372120, 61421061); (2) Beijing Municipal Natural Science Foundation (No.4182041, 4152039); (3) the National Basic Research Program of China (No. 2013CB329102); (4) Fundamental Research Funds for the Central Universities under Grant 2018RC20; (5) BUPT Excellent Ph.D. Students Foundation.

References

Charu C. Aggarwal and ChengXiang Zhai. A survey of text classification algorithms. In *Mining Text Data*. 2012.

Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. Linear algebraic structure of word senses, with applications to polysemy. *TACL*, 2018.

Geoffrey E. Hinton, Alex Krizhevsky, and Sida D. Wang. Transforming auto-encoders. In *Artificial Neural Networks and Machine Learning*, 2011.

Geoffrey Hinton, Nicholas Frosst, and Sara Sabour. Matrix capsules with em routing. 2018.

Rie Johnson and Tong Zhang. Deep pyramid convolutional neural networks for text categorization. In *ACL*, 2017.

Yoon Kim. Convolutional neural networks for sentence classification. In *EMNLP*, 2014.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, 2014.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. Recurrent convolutional neural networks for text classification. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

D. Terence Langendoen. Studies in linguistic analysis. *International Journal of American Linguistics*, 1959.

Jiwei Li and Dan Jurafsky. Do multi-sense embeddings improve natural language understanding? In *EMNLP*, 2015.

Xin Li and Dan Roth. Learning question classifiers. In *19th International Conference on Computational Linguistics, COLING*, 2002.

Zhouhan Lin, Minwei Feng, Cícero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding. *international conference on learning representations*, 2017.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *Computer Science*, 2013.

Bo Pang and Lillian Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, 2004.

Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *ACL 2005*, 2005.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *NAACL HLT*, 2018.

Sara Sabour, Nicholas Frosst, and Geoffrey E. Hinton. Dynamic routing between capsules. In *NIPS*, 2017.

Holger Schwenk, Loïc Barrault, Alexis Conneau, and Yann LeCun. Very deep convolutional networks for text classification. In *EACL*, 2017.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*, 2013.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, 2017.

Sida I. Wang and Christopher D. Manning. Baselines and bigrams: Simple, good sentiment and topic classification. In *ACL*, 2012.

Sida I. Wang and Christopher D. Manning. Baselines and bigrams: Simple, good sentiment and topic classification. In *ACL*, 2012.

Congying Xia, Chenwei Zhang, Xiaohui Yan, Yi Chang, and Philip S. Yu. Zero-shot user intent detection via capsule neural networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018.

Liqiang Xiao, Honglun Zhang, Wenqing Chen, Yongkun Wang, and Yaohui Jin. Mcapsnet: Capsule network for text with multi-task learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alexander J. Smola, and Eduard H. Hovy. Hierarchical attention networks for document classification. In *NAACL HLT*, 2016.

Min Yang, Wei Zhao, Jianbo Ye, Zeyang Lei, Zhou Zhao, and Soufei Zhang. Investigating capsule networks with dynamic routing for text classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems*, 2015.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *NIPS*, 2015.

Rui Zhang, Honglak Lee, and Dragomir R. Radev. Dependency sensitive convolutional neural networks for modeling sentences and documents. In *NAACL HLT 2016*, 2016.

Jianyu Zhao, Zhiqiang Zhan, Qichuan Yang, Yang Zhang, Changjian Hu, Zhensheng Li, Liuxin Zhang, and Zhiqiang He. Adaptive learning of local semantic and global structure representations for text classification. In *Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018, Santa Fe, New Mexico, USA, August 20-26, 2018*, 2018.