Towards Scalable and Reliable Capsule Networks for Challenging NLP Applications

Wei Zhao[†], **Haiyun Peng**[‡], **Steffen Eger**[†], **Erik Cambria**[‡] **and Min Yang**[Φ]
[†] Computer Science Department, Technische Universität Darmstadt, Germany
[‡] School of Computer Science and Engineering, Nanyang Technological University, Singapore
[Φ] Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, China
www.aiphes.tu-darmstadt.de

Abstract

Obstacles hindering the development of capsule networks for challenging NLP applications include poor scalability to large output spaces and less reliable routing processes. In this paper, we introduce (i) an agreement score to evaluate the performance of routing processes at instance level; (ii) an adaptive optimizer to enhance the reliability of routing; (iii) capsule compression and partial routing to improve the scalability of capsule networks. We validate our approach on two NLP tasks, namely: multi-label text classification and question answering. Experimental results show that our approach considerably improves over strong competitors on both tasks. In addition, we gain the best results in low-resource settings with few training instances.[1]

1 Introduction

In recent years, deep neural networks have achieved outstanding success in natural language processing (NLP), computer vision and speech recognition. However, these deep models are data-hungry and generalize poorly from small datasets, very much unlike humans (Lake et al., 2015).

This is an important issue in NLP since sentences with different surface forms can convey the same meaning (paraphrases) and not all of them can be enumerated in the training set. For example, *Peter did not accept the offer* and *Peter turned down the offer* are semantically equivalent, but use different surface realizations.

In image classification, progress on the generalization ability of deep networks has been made by capsule networks (Sabour et al., 2017; Hinton et al., 2018). They are capable of generalizing to the same object in different 3D images with various viewpoints.

[1] Our code is publicly available at http://bit.ly/311Dcod

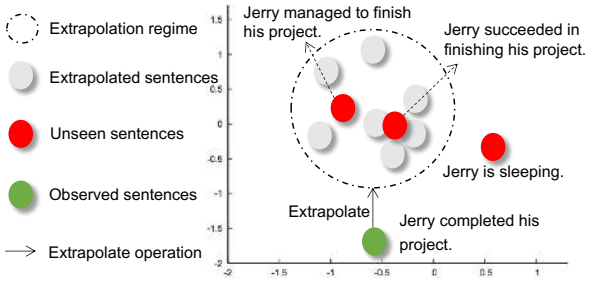

Figure 1: The extrapolation regime for an observed sentence can be found during training. Then, the unseen sentences in this regime may be generalized successfully.

Such generalization capability can be learned from examples with few viewpoints by extrapolation (Hinton et al., 2011). This suggests that capsule networks can similarly abstract away from different surface realizations in NLP applications.

Figure 1 illustrates this idea of how observed sentences in the training set are generalized to unseen sentences by extrapolation. In contrast, traditional neural networks require massive amounts of training samples for generalization. This is especially true in the case of convolutional neural networks (CNNs), where pooling operations wrongly discard positional information and do not consider hierarchical relationships between local features (Sabour et al., 2017).

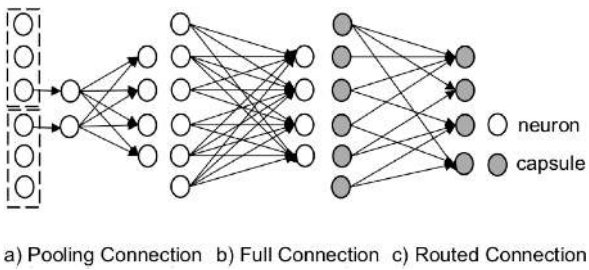

Figure 2: Outputs attend to a) active neurons found by pooling operations b) all neurons c) relevant capsules found in routing processes.

Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 1549–1559
Florence, Italy, July 28 - August 2, 2019.

Capsule networks, instead, have the potential for learning hierarchical relationships between consecutive layers by using routing processes without parameters, which are clustering-like methods (Sabour et al., 2017) and additionally improve the generalization capability. We contrast such routing processes with pooling and fully connected layers in Figure 2.

Despite some recent success in NLP tasks (Wang et al., 2018; Xia et al., 2018; Xiao et al., 2018; Zhang et al., 2018a; Zhao et al., 2018), a few important obstacles still hinder the development of capsule networks for mature NLP applications.

For example, selecting the number of iterations is crucial for routing processes, because they iteratively route low-level capsules to high-level capsules in order to learn hierarchical relationships between layers. However, existing routing algorithms use the same number of iterations for all examples, which is not reliable to judge the convergence of routing. As shown in Figure 3, a routing process with five iterations on all examples converges to a lower training loss at system level, but on instance level for one example, convergence has still not obtained.

Additionally, training capsule networks is more difficult than traditional neural networks like CNN and long short-term memory (LSTM) due to the large number of capsules and potentially large output spaces, which requires extensive computational resources in the routing process.

In this work, we address these issues via the following contributions:

- We formulate routing processes as a proxy problem minimizing a total negative agreement score in order to evaluate how routing processes perform at instance level, which will be discussed more in depth later.
- We introduce an adaptive optimizer to self-adjust the number of iterations for each example in order to improve instance-level convergence and enhance the reliability of routing processes.
- We present capsule compression and partial routing to achieve better scalability of capsule networks on datasets with large output spaces.
- Our framework outperforms strong baselines on multi-label text classification and question answering. We also demonstrate its superior generalization capability in low-resource settings.

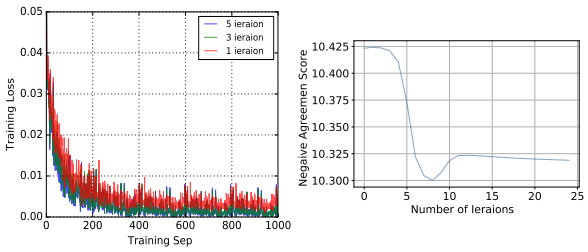

Figure 3: left) System-level routing evaluation on all examples; right) Instance-level routing evaluation on one example.

2 NLP-Capsule Framework

We have motivated the need for better capsule networks being capable of scaling to large output spaces and higher reliability for routing processes at instance level. We now build a unified capsule framework, which we call NLP-Capsule. It is shown in Figure 4 and described below.

2.1 Convolutional Layer

We use a convolutional operation to extract features from documents by taking a sliding window over document embeddings.

Let $\boldsymbol{X} \in \mathbb{R}^{l \times v}$ be a matrix of stacked v-dimensional word embeddings for an input document with l tokens. Furthermore, let $\boldsymbol{W}^a \in \mathbb{R}^{l \times k}$ be a convolutional filter with a width k. We apply this filter to a local region $\boldsymbol{X}^{\intercal}_{i:i+k-1} \in \mathbb{R}^{k \times l}$ to generate one feature:

$$m_i = f(\boldsymbol{W}^a \circ \boldsymbol{X}^{\intercal}_{i:i+k-1})$$

where $\circ$ denotes element-wise multiplication, and f is a nonlinear activation function (i.e., ReLU). For ease of exposition, we omit all bias terms.

Then, we can collect all m_i into one feature map $(m_1, \ldots, m_{(v-k+1)/2})$ after sliding the filter over the current document. To increase the diversity of features extraction, we concatenate multiple feature maps extracted by three filters with different window sizes (2,4,8) and pass them to the primary capsule layer.

2.2 Primary Capsule Layer

In this layer, we use a group-convolution operation to transform feature maps into primary capsules. As opposed to using a scalar for each element in the feature maps, capsules use a group of neurons to represent each element in the current layer, which has the potential for preserving more information.

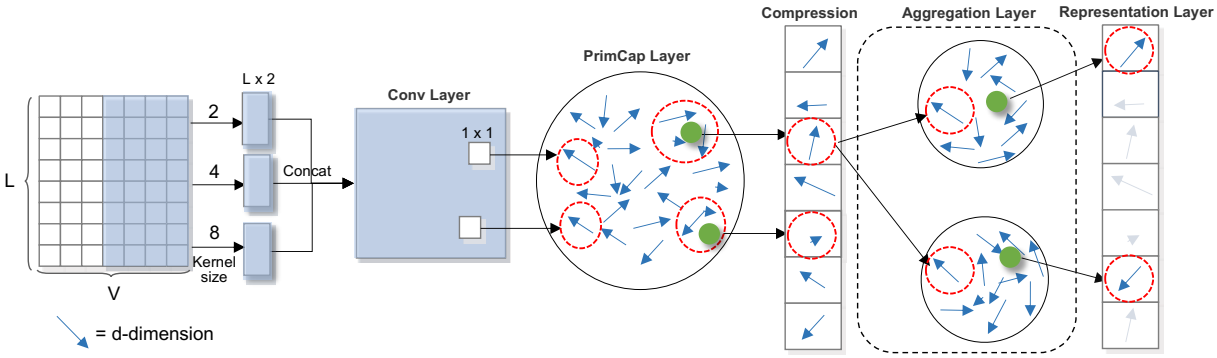

Figure 4: An illustration of NLP-Capsule framework.

Using 1×1 filters $\boldsymbol{W}^b = \{w_1, ..., w_d\} \in \mathbb{R}^d$, in total d groups are used to transform each scalar m_i in feature maps to one capsule $\boldsymbol{p}_i$, a d- dimensional vector, denoted as:

$$\boldsymbol{p}_i = g(p_{i1} \oplus p_{i2} \oplus \cdots \oplus p_{id}) \in \mathbb{R}^d$$

where $p_{ij} = m_i \cdot w_j \in \mathbb{R}$ and $\oplus$ is the concatenation operator. Furthermore, g is a non-linear function (i.e., squashing function). The length $||\boldsymbol{p}_i||$ of each capsule $\boldsymbol{p}_i$ indicates the probability of it being useful for the task at hand. Hence, a capsule's length has to be constrained into the unit interval $[0, 1]$ by the squashing function g:

$$g(\boldsymbol{x}) = \frac{||\boldsymbol{x}||^2}{1 + ||\boldsymbol{x}||^2} \frac{\boldsymbol{x}}{||\boldsymbol{x}||}$$

Capsule Compression One major issue in this layer is that the number of primary capsules becomes large in proportion to the size of the input documents, which requires extensive computational resources in routing processes (see Section 2.3). To mitigate this issue, we condense the large number of primary capsules into a smaller amount. In this way, we can merge similar capsules and remove outliers. Each condensed capsule $\boldsymbol{u}_i$ is calculated by using a weighted sum over all primary capsules, denoted as:

$$\hat{\boldsymbol{u}}_i = \sum_j b_j \boldsymbol{p}_j \in \mathbb{R}^d$$

where the parameter b_j is learned by supervision.

2.3 Aggregation Layer

Pooling is the simplest aggregation function routing condensed capsules into the subsequent layer, but it loses almost all information during aggregation. Alternatively, routing processes are introduced to iteratively route condensed capsules into the next layer for learning hierarchical relationships between two consecutive layers. We now describe this iterative routing algorithm. Let $\{\boldsymbol{u}_1, \ldots, \hat{\boldsymbol{u}}_m\}$ and $\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n\}$ be a set of condensed capsules in layer ℓ and a set of high-level capsules in layer $\ell+1$, respectively. The basic idea of routing is two-fold.

First, we transform the condensed capsules into a collection of candidates $\{\hat{\boldsymbol{u}}_{j|1}, \ldots, \hat{\boldsymbol{u}}_{j|m}\}$ for the j-th high-level capsule in layer $\ell + 1$. Following Sabour et al. (2017), each element $\hat{\boldsymbol{u}}_{j|i}$ is calculated by:

$$\hat{\boldsymbol{u}}_{j|i} = \boldsymbol{W}^c \boldsymbol{u}_i \in \mathbb{R}^d$$

where $\boldsymbol{W}^c$ is a linear transformation matrix.

Then, we represent a high-level capsule $\boldsymbol{v}_j$ by a weighted sum over those candidates, denoted as:

$$\boldsymbol{v}_j = \sum_{i=1}^{m} c_{ij} \hat{\boldsymbol{u}}_{j|i}$$

where c_{ij} is a coupling coefficient iteratively updated by a clustering-like method.

Our Routing As discussed earlier, routing algorithms like dynamic routing (Sabour et al., 2017) and EM routing (Hinton et al., 2018), which use the same number of iterations for all samples, perform well according to training loss at system level, but on instance level for individual examples, convergence has still not been reached. This increases the risk of unreliability for routing processes (see Figure 3).

To evaluate the performance of routing processes at instance level, we formulate them as a proxy problem minimizing the negative agreement score (NAS) function:

$$\min_{c,\boldsymbol{v}} f(\boldsymbol{u}) = -\sum_{i,j} c_{ij} \langle \boldsymbol{v}_j, \boldsymbol{u}_{j|i} \rangle$$

$$\text{s.t.} \quad \forall i, j : c_{ij} > 0, \quad \sum_j c_{ij} = 1.$$

The basic intuition behind this is to assign higher weights c_{ij} to one agreeable pair $\langle \boldsymbol{v}_j, \boldsymbol{u}_{j|i} \rangle$ if the capsule $\boldsymbol{v}_j$ and $\boldsymbol{u}_{j|i}$ are close to each other such that the total agreement score $\sum_{i,j} c_{ij} \langle \boldsymbol{v}_j, \boldsymbol{u}_{j|i} \rangle$ is maximized. However, the choice of NAS functions remains an open problem. Hinton et al. (2018) hypothesize that the agreeable pairs in NAS functions are from Gaussian distributions. Instead, we study NAS functions by introducing Kernel Density Estimation (KDE) since this yields a non-parametric density estimator requiring no assumptions that the agreeable pairs are drawn from parametric distributions. Here, we formulate the NAS function in a KDE form.

$$\min_{c,\boldsymbol{v}} f(\boldsymbol{u}) = -\sum_{i,j} c_{ij} k(d(\boldsymbol{v}_j, \boldsymbol{u}_{j|i})) \qquad (1)$$

where d is a distance metric with ℓ_2 norm, and k is a Epanechnikov kernel function (Wand and Jones, 1994) with:

$$k(x) = \begin{cases} 1 - x & x \in [0, 1) \\ 0 & x \geq 1 \end{cases}$$

The solution we used for KDE is taking Mean Shift (Comaniciu and Meer, 2002) to minimize the NAS function $f(\boldsymbol{u})$:

$$\nabla f(\boldsymbol{u}) = \sum_{i,j} c_{ij} k'(d(\boldsymbol{v}_j, \boldsymbol{u}_{j|i})) \frac{\partial d(\boldsymbol{v}_j, \boldsymbol{u}_{j|i})}{\partial \boldsymbol{v}}$$

First, $\boldsymbol{v}_j^{\tau+1}$ can be updated while $c_{ij}^{\tau+1}$ is fixed:

$$\boldsymbol{v}_j^{\tau+1} = \frac{\sum_{i,j} c_{ij}^{\tau} k'(d(\boldsymbol{v}_j^{\tau}, \hat{\boldsymbol{u}}_{j|i})) \boldsymbol{u}_{j|i}}{\sum_{i,j} k'(d(\boldsymbol{v}_j^{\tau}, \boldsymbol{u}_{j|i}))}$$

Then, $c_{ij}^{\tau+1}$ can be updated using standard gradient descent:

$$c_{ij}^{\tau+1} = c_{ij}^{\tau} + \alpha \cdot k(d(\boldsymbol{v}_j^{\tau}, \boldsymbol{u}_{j|i}))$$

where α is the hyper-parameter to control step size.

To address the issue of convergence not being reached at instance level, we present an adaptive optimizer to self-adjust the number of iterations for individual examples according to their negative agreement scores (see Algorithm 1). Following Zhao et al. (2018), we replace standard softmax with leaky-softmax, which decreases the strength of noisy capsules.

Algorithm 1 Our Adaptive KDE Routing

procedure ROUTING($\boldsymbol{u}_{j|i}$, ℓ)
Initialize $\forall i, j : c_{ij} = 1/n_{\ell+1}$
while true **do**
  foreach capsule i, j in layer ℓ, $\ell + 1$ **do**
    $c_{ij} \leftarrow$ leaky-softmax(c_{ij})
  foreach capsule j in layer $\ell + 1$ **do**
    $\boldsymbol{v}_j \leftarrow \frac{\sum_i c_{ij} k'(d(\boldsymbol{v}_j, \boldsymbol{u}_{j|i})) \hat{\boldsymbol{u}}_{j|i}}{\sum_{i=1}^{n} k'(d(\boldsymbol{v}_i, \boldsymbol{u}_{j|i}))}$
  foreach capsule i, j in layer ℓ, $\ell + 1$ **do**
    $c_{ij} \leftarrow c_{ij} + \alpha \cdot k(d(\boldsymbol{v}_j, \boldsymbol{u}_{j|i}))$
  foreach capsule j in layer $\ell + 1$ **do**
    $a_j \leftarrow |v_j|$
  NAS $= \log(\sum_{\mathrm{i,j}} \mathrm{c_{ij}} \mathrm{k}(\mathrm{d}(\boldsymbol{v}_{\mathrm{j}}, \boldsymbol{u}_{\mathrm{j|i}})))$
  if $|\mathrm{NAS} - \mathrm{Last\_NAS}| < \epsilon$ **then**
    break
  else
    Last_NAS $\leftarrow$ NAS
return v_j, a_j

2.4 Representation Layer

This is the top-level layer containing final capsules calculated by iteratively minimizing the NAS function (See Eq. 1), where the number of final capsules corresponds to the entire output space. Therefore, as long as the size of an output space goes to a large scale (thousands of labels), the computation of this function would become extremely expensive, which yields the bottleneck of scalability of capsule networks.

Partial Routing As opposed to the entire output space on data sets, the sub-output space corresponding to individual examples is rather small, i.e., only few labels are assigned to one document in text classification, for example. As a consequence, it is redundant to route low-level capsules to the entire output space for each example in the training stage, which motivated us to present a partial routing algorithm with constrained output spaces, such that our NAS function is described as:

$$\min_{c,\boldsymbol{v}} -\sum_i \Big(\sum_{j \in D^+} c_{ij} \langle \boldsymbol{v}_j, \boldsymbol{u}_{j|i} \rangle + \lambda \cdot \sum_{k \in D^-} c_{ik} \langle \boldsymbol{v}_k, \boldsymbol{u}_{k|i} \rangle \Big)$$

where D^+ and D^- denote the sets of real (positive) and randomly selected (negative) outputs for each example, respectively. Both sets are

far smaller than the entire output space. λ is the hyper-parameter to control aggregation scores from negative outputs.

3 Experiments

The major focus of this work is to investigate the scalability of our approach on datasets with a large output space, and generalizability in low-resource settings with few training examples. Therefore, we validate our capsule-based approach on two specific NLP tasks: (i) multi-label text classification with a large label scale; (ii) question answering with a data imbalance issue.

3.1 Multi-label Text Classification

Multi-label text classification task refers to assigning multiple relevant labels to each input document, while the entire label set might be extremely large. We use our approach to encode an input document and generate the final capsules corresponding to the number of labels in the representation layer. The length of final capsule for each label indicates the probability whether the document has this label.

Dataset	#Train/Test/Labels	Avg-docs
RCV1	23.1K/781.2K/103	729.67
EUR-Lex	15.4K/3.8K/3.9K	15.59

Table 1: Characteristics of the datasets. Each label of RCV1 has about 729.67 training examples, while each label of EUR-Lex has merely about 15.59 examples.

Experimental Setup We conduct our experiments on two datasets selected from the extreme classification repository:[2] a regular label scale dataset (RCV1), with 103 labels (Lewis et al., 2004), and a large label scale dataset (EUR-Lex), with 3,956 labels (Mencia and Fürnkranz, 2008), described in Table 1. The intuition behind our datasets selection is that EUR-Lex, with 3,956 labels and 15.59 examples per label, fits well with our goal of investigating the scalability and generalizability of our approach. We contrast EUR-Lex with RCV1, a dataset with a regular label scale, and leave the study of datasets with extremely large labels, e.g., Wikipedia-500K with 501,069 labels, to future work.

Baselines We compare our approach to the following baselines: non-deep learning approaches using TF-IDF features of documents as inputs: FastXML (Prabhu and Varma, 2014), and PD-Sparse (Yen et al., 2016), deep learning approaches using raw text of documents as inputs: FastText (Joulin et al., 2016), Bow-CNN (Johnson and Zhang, 2014), CNN-Kim (Kim, 2014), XML-CNN (Liu et al., 2017)), and a capsule-based approach Cap-Zhao (Zhao et al., 2018). For evaluation, we use standard rank-based measures (Liu et al., 2017) such as Precision@k, and Normalized Discounted Cumulative Gain (NDCG@k).

Implementation Details The word embeddings are initialized as 300-dimensional GloVe vectors (Pennington et al., 2014). In the convolutional layer, we use a convolution operation with three different window sizes (2,4,8) to extract features from input documents. Each feature is transformed into a primary capsule with 16 dimensions by a group-convolution operation. All capsules in the primary capsule layer are condensed into 256 capsules for RCV1 and 128 capsules for EUR-Lex by a capsule compression operation.

To avoid routing low-level capsules to the entire label space in the inference stage, we use a CNN baseline (Kim, 2014) trained on the same dataset with our approach, to generate 200 candidate labels and take these labels as a constrained output space for each example.

Experimental Results In Table 2, we can see a noticeable margin brought by our capsule-based approach over the strong baselines on EUR-Lex, and competitive results on RCV1. These results appear to indicate that our approach has superior generalization ability on datasets with fewer training examples, i.e., RCV1 has 729.67 examples per label while EUR-Lex has 15.59 examples.

In contrast to the strongest baseline XML-CNN with 22.52M parameters and 0.08 seconds per batch, our approach has 14.06M parameters, and takes 0.25 seconds in an acceleration setting with capsule compression and partial routing, and 1.7 seconds without acceleration. This demonstrates that our approach provides competitive computational speed with fewer parameters compared to the competitors.

Discussion on Generalization To further study the generalization capability of our approach, we vary the percentage of training examples from 100% to 50% on the entire training set, leading to the number of training examples per label de-

[2] https://manikvarma.github.io

Datasets	Metrics	FastXML	PD-Sparse	FastText	Bow-CNN	CNN-Kim	XML-CNN	Cap-Zhao	NLP-Cap	Impv
RCV1	PREC@1	94.62	95.16	95.40	96.40	93.54	96.86	96.63	**97.05**	+0.20%
	PREC@3	78.40	79.46	79.96	81.17	76.15	81.11	81.02	**81.27**	+0.20%
	PREC@5	54.82	55.61	55.64	**56.74**	52.94	56.07	56.12	56.33	-0.72%
	NDCG@1	94.62	95.16	95.40	96.40	93.54	96.88	96.63	**97.05**	+0.20%
	NDCG@3	89.21	90.29	90.95	92.04	87.26	92.22	92.31	**92.47**	+0.17%
	NDCG@5	90.27	91.29	91.68	92.89	88.20	92.63	92.75	**93.11**	+0.52%
EUR-Lex	PREC@1	68.12	72.10	71.51	64.99	68.35	75.65	-	**80.20**	+6.01%
	PREC@3	57.93	57.74	60.37	51.68	54.45	61.81	-	**65.48**	+5.93%
	PREC@5	48.97	47.48	50.41	42.32	44.07	50.90	-	**52.83**	+3.79%
	NDCG@1	68.12	72.10	71.51	64.99	68.35	75.65	-	**80.20**	+6.01%
	NDCG@3	60.66	61.33	63.32	55.03	59.81	66.71	-	**71.11**	+6.59%
	NDCG@5	56.42	55.93	58.56	49.92	57.99	64.45	-	**68.80**	+6.75%

Table 2: Comparisons of our NLP-Cap approach and baselines on two text classication benchmarks, where '-' denotes methods that failed to scale due to memory issues.

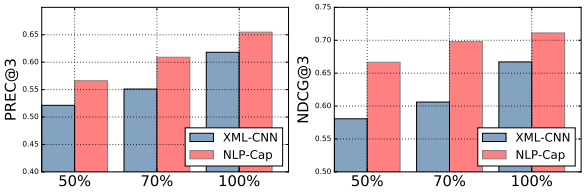

Figure 5: Performance on EUR-Lex by varying the percentage of training examples (X-axis).

Method	#Training	PREC@1	PREC@3	PREC@5
XML-CNN	100% examples	75.65	61.81	50.90
NLP-Capsule	50% examples	73.69	56.62	44.36
	60% examples	74.83	58.48	46.33
	70% examples	77.26	60.90	47.73
	80% examples	77.68	61.06	48.28
	90% examples	79.45	63.95	50.90
	100% examples	**80.20**	**65.48**	**52.83**
Method	#Training	NDCG@1	NDCG@3	NDCG@5
XML-CNN	100% examples	75.65	66.71	64.45
NLP-Capsule	50% examples	73.69	66.65	67.36
	60% examples	74.83	67.87	68.62
	70% examples	77.26	69.79	69.65
	80% examples	77.67	69.43	69.27
	90% examples	79.45	**71.64**	**71.06**
	100% examples	**80.21**	71.11	68.80

Table 3: Experimental results on different fractions of training examples from 50% to 100% on EUR-Lex.

creasing from 15.59 to 7.77. Figure 5 shows that our approach outperforms the strongest baseline XML-CNN with different fractions of the training examples.

This finding agrees with our speculation on generalization: the distance between our approach and XML-CNN increases as fewer training data samples are available. In Table 3, we also find that our approach with 70% of training examples achieves about 5% improvement over XML-CNN with 100% of examples on 4 out of 6 metrics.

Routing Comparison We compare our routing with (Sabour et al., 2017) and (Zhang et al., 2018b) on EUR-Lex dataset and observe that it performs best on all metrics (Table 4). We speculate that the improvement comes from enhanced reliability of routing processes at instance level.

3.2 Question Answering

Question-Answering (QA) selection task refers to selecting the best answer from candidates to each question. For a question-answer pair (q, a), we use our capsule-based approach to generate two final capsules $\boldsymbol{v}_q$ and $\boldsymbol{v}_a$ corresponding to the respective question and answer. The relevance score of question-answer pair can be defined as their cosine similarity:

$$s(q,a) = \cos(\boldsymbol{v}_{\mathrm{q}}, \boldsymbol{v}_{\mathrm{a}}) = \frac{\boldsymbol{v}_{\mathrm{q}}^{\intercal}\boldsymbol{v}_{\mathrm{a}}}{||\boldsymbol{v}_{\mathrm{q}}|| \cdot ||\boldsymbol{v}_{\mathrm{a}}||}$$

Experiment Setup In Table 5, we conduct our experiments on the TREC QA dataset collected from TREC QA track 8-13 data (Wang et al., 2007). The intuition behind this dataset selection is that the cost of hiring human annotators to collect positive answers for individual questions can be prohibitive since positive answers can be conveyed in multiple different surface forms. Such issue arises particularly in TREC QA with only 12%

Method	PREC@1	PREC@3	PREC@5
XML-CNN	75.65	61.81	50.90
NLP-Capsule + Sabour's Routing	79.14	64.33	51.85
NLP-Capsule + Zhang's Routing	80.20	65.48	52.83
NLP-Capsule + Our Routing	**80.62**	**65.61**	**53.66**
Method	**NDCG@1**	**NDCG@3**	**NDCG@5**
XML-CNN	75.65	66.71	64.45
NLP-Capsule + Sabour's Routing	79.14	70.13	67.02
NLP-Capsule + Zhang's Routing	80.20	71.11	68.80
NLP-Capsule + Our Routing	**80.62**	**71.34**	**69.57**

Table 4: Performance on EUR-Lex dataset with different routing process.

Dataset	#Questions	#QA Pairs	%Positive
Train/Dev/Test	1229/82/100	53417/1148/1517	12%

Table 5: Characteristic of TREC QA dataset. %Positive denotes the percentage of positive answers.

positive answers. Therefore, we use this dataset to investigate the generalizability of our approach.

Baselines We compare our approach to the following baselines: CNN + LR (Yu et al., 2014b) using unigrams and bigrams, CNN (Severyn and Moschitti, 2015) using additional bilinear similarity features, CNTN (Qiu and Huang, 2015) using neural tensor network, LSTM (Tay et al., 2017) using single and multi-layer, MV-LSTM (Wan et al., 2016), NTN-LSTM and HD-LSTM (Tay et al., 2017) using holographic dual LSTM and Capsule-Zhao (Zhao et al., 2018) using capsule networks. For evaluation, we use standard measures (Wang et al., 2007) such as Mean Average Precision (MAP) and Mean Reciprocal Rank (MRR).

Implementation Details The word embeddings used for question answering pairs are initialized as 300-dimensional GloVe vectors. In the convolutional layer, we use a convolution operation with three different window sizes (3,4,5). All 16-dimensional capsules in the primary capsule layer are condensed into 256 capsules by the capsule compression operation.

Experimental Results and Discussions In Table 6, the best performance on MAP metric is achieved by our approach, which verifies the effectiveness of our model. We also observe that our approach exceeds traditional neural models like CNN, LSTM and NTN-LSTM by a noticeable margin.

This finding also agrees with the observation we found in multi-label classification: our approach has superior generalization capability in low-resource setting with few training examples. In contrast to the strongest baseline HD-LSTM with 34.51M and 0.03 seconds for one batch, our approach has 17.84M parameters and takes 0.06 seconds in an acceleration setting, and 0.12 seconds without acceleration.

Method	MAP	MRR
CNN + LR (unigram)	54.70	63.29
CNN + LR (bigram)	56.93	66.13
CNN	66.91	68.80
CNTN	65.80	69.78
LSTM (1 layer)	62.04	66.85
LSTM	59.75	65.33
MV-LSTM	64.88	68.24
NTN-LSTM	63.40	67.72
HD-LSTM	67.44	**75.11**
Capsule-Zhao	73.63	70.12
NLP-Capsule	**77.73**	74.16

Table 6: Experimental results on TREC QA dataset.

4 Related Work

4.1 Multi-label Text Classification

Multi-label text classification aims at assigning a document to a subset of labels whose label set might be extremely large. With increasing numbers of labels, issues of data sparsity and scalability arise. Several methods have been proposed for the large multi-label classification case.

Tree-based models (Agrawal et al., 2013; Weston et al., 2013) induce a tree structure that recursively partitions the feature space with non-leaf nodes. Then, the restricted label spaces at leaf nodes are used for classification. Such a solution entails higher robustness because of a dynamic hyper-plane design and its computational efficiency. FastXML (Prabhu and Varma, 2014) is one such tree-based model, which learns a hierarchy of training instances and optimizes an NDCG-based objective function for nodes in the tree structure.

Label embedding models (Balasubramanian and Lebanon, 2012; Chen and Lin, 2012; Cisse et al., 2013; Bi and Kwok, 2013; Ferng and Lin, 2011; Hsu et al., 2009; Ji et al., 2008; Kapoor et al., 2012; Lewis et al., 2004; Yu et al., 2014a) address the data sparsity issue with two steps: compression and decompression. The compression step learns a low-dimensional label embedding that is projected from original and high-dimensional label space. When data instances are classified to these label embeddings, they will be projected back to the high-dimensional label space, which is the decompression step. Recent works came up with different compression or decompression techniques, e.g., SLEEC (Bhatia et al., 2015).

Deep learning models: FastText (Joulin et al., 2016) uses averaged word embeddings to classify documents, which is computationally efficient but ignores word order. Various CNNs inspired by Kim (2014) explored MTC with dynamic pooling, such as Bow-CNN (Johnson and

Zhang, 2014) and XML-CNN (Liu et al., 2017).

Linear classifiers: PD-Sparse (Yen et al., 2016) introduces a Fully-Corrective Block-Coordinate Frank-Wolfe algorithm to address data sparsity.

4.2 Question and Answering

State-of-the-art approaches to QA fall into two categories: IR-based and knowledge-based QA.

IR-based QA firstly preprocesses the question and employ information retrieval techniques to retrieve a list of relevant passages to questions. Next, reading comprehension techniques are adopted to extract answers within the span of retrieved text. For answer extraction, early methods manually designed patterns to get them (Pasca). A recent popular trend is neural answer extraction. Various neural network models are employed to represent questions (Severyn and Moschitti, 2015; Wang and Nyberg, 2015). Since the attention mechanism naturally explores relevancy, it has been widely used in QA models to relate the question to candidate answers (Tan et al., 2016; Santos et al., 2016; Sha et al., 2018). Moreover, some researchers leveraged external large-scale knowledge bases to assist answer selection (Savenkov and Agichtein, 2017; Shen et al., 2018; Deng et al., 2018).

Knowledge-based QA conducts semantic parsing on questions and transforms parsing results into logical forms. Those forms are adopted to match answers from structured knowledge bases (Yao and Van Durme, 2014; Yih et al., 2015; Bordes et al., 2015; Yin et al., 2016; Hao et al., 2017). Recent developments focused on modeling the interaction between question and answer pairs: Tensor layers (Qiu and Huang, 2015; Wan et al., 2016) and holographic composition (Tay et al., 2017) have pushed the state-of-the-art.

4.3 Capsule Networks

Capsule networks were initially proposed by Hinton (Hinton et al., 2011) to improve representations learned by neural networks against vanilla CNNs. Subsequently, Sabour et al. (2017) replaced the scalar-output feature detectors of CNNs with vector-output capsules and max-pooling with routing-by-agreement.

Hinton et al. (2018) then proposed a new iterative routing procedure between capsule layers based on the EM algorithm, which achieves better accuracy on the smallNORB dataset. Zhang et al. (2018a) applied capsule networks to relation extraction in a multi-instance multi-label learning framework. Xiao et al. (2018) explored capsule networks for multi-task learning.

Xia et al. (2018) studied the zero-shot intent detection problem with capsule networks, which aims to detect emerging user intents in an unsupervised manner. Zhao et al. (2018) investigated capsule networks with dynamic routing for text classification, and transferred knowledge from the single-label to multi-label cases. Cho et al. (2019) studied capsule networks with determinantal point processes for extractive multi-document summarization.

Our work is different from our predecessors in the following aspects: (i) we evaluate the performance of routing processes at instance level, and introduce an adaptive optimizer to enhance the reliability of routing processes; (ii) we present capsule compression and partial routing to achieve better scalability of capsule networks on datasets with a large output space.

5 Conclusion

Making computers perform more like humans is a major issue in NLP and machine learning. This not only includes making them perform on similar levels (Hassan et al., 2018), but also requests them to be robust to adversarial examples (Eger et al., 2019) and generalize from few data points (Rücklé et al., 2019). In this work, we have addressed the latter issue.

In particular, we extended existing capsule networks into a new framework with advantages concerning scalability, reliability and generalizability. Our experimental results have demonstrated its effectiveness on two NLP tasks: multi-label text classification and question answering.

Through our modifications and enhancements, we hope to have made capsule networks more suitable to large-scale problems and, hence, more mature for real-world applications. In the future, we plan to apply capsule networks to even more challenging NLP problems such as language modeling and text generation.

6 Acknowledgments

We thank the anonymous reviewers for their comments, which greatly improved the final version of the paper. This work has been supported by the German Research Foundation as part of the Research Training Group Adaptive Preparation of In-

formation from Heterogeneous Sources (AIPHES) at the Technische Universität Darmstadt under grant No. GRK 1994/1.

References

Rahul Agrawal, Archit Gupta, Yashoteja Prabhu, and Manik Varma. 2013. Multi-label learning with millions of labels: Recommending advertiser bid phrases for web pages. In *Proceedings of the 22nd international conference on World Wide Web*, pages 13–24. ACM.

Krishnakumar Balasubramanian and Guy Lebanon. 2012. The landmark selection method for multiple output prediction. *arXiv preprint arXiv:1206.6479*.

Kush Bhatia, Himanshu Jain, Purushottam Kar, Manik Varma, and Prateek Jain. 2015. Sparse local embeddings for extreme multi-label classification. In *Advances in Neural Information Processing Systems*, pages 730–738.

Wei Bi and James Kwok. 2013. Efficient multi-label classification with many labels. In *International Conference on Machine Learning*, pages 405–413.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075*.

Yao-Nan Chen and Hsuan-Tien Lin. 2012. Feature-aware label space dimension reduction for multi-label classification. In *Advances in Neural Information Processing Systems*, pages 1529–1537.

Sangwoo Cho, Logan Lebanoff, Hassan Foroosh, and Fei Liu. 2019. Improving the similarity measure of determinantal point processes for extractive multi-document summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Moustapha M Cisse, Nicolas Usunier, Thierry Artieres, and Patrick Gallinari. 2013. Robust bloom filters for large multilabel classification tasks. In *Advances in Neural Information Processing Systems*, pages 1851–1859.

Dorin Comaniciu and Peter Meer. 2002. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 24(5):603–619.

Yang Deng, Ying Shen, Min Yang, Yaliang Li, Nan Du, Wei Fan, and Kai Lei. 2018. Knowledge as a bridge: Improving cross-domain answer selection with external knowledge. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3295–3305.

Steffen Eger, Gözde Gül Şahin, Andreas Rücklé, Ji-Ung Lee, Claudia Schulz, Mohsen Mesgar, Krishnkant Swarnkar, Edwin Simpson, and Iryna Gurevych. 2019. Text processing like humans do: Visually attacking and shielding nlp systems. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*.

C-S Ferng and H-T Lin. 2011. Multi-label classification with error-correcting codes. In *Asian Conference on Machine Learning*, pages 281–295.

Yanchao Hao, Yuanzhe Zhang, Kang Liu, Shizhu He, Zhanyi Liu, Hua Wu, and Jun Zhao. 2017. An end-to-end model for question answering over knowledge base with cross-attention combining global knowledge. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 221–231.

Hany Hassan, Anthony Aue, Chang Chen, Vishal Chowdhary, Jonathan Clark, Christian Federmann, Xuedong Huang, Marcin Junczys-Dowmunt, William Lewis, Mu Li, Shujie Liu, Tie-Yan Liu, Renqian Luo, Arul Menezes, Tao Qin, Frank Seide, Xu Tan, Fei Tian, Lijun Wu, Shuangzhi Wu, Yingce Xia, Dongdong Zhang, Zhirui Zhang, and Ming Zhou. 2018. Achieving human parity on automatic chinese to english news translation. *CoRR*, abs/1803.05567.

Geoffrey E Hinton, Alex Krizhevsky, and Sida D Wang. 2011. Transforming auto-encoders. In *International Conference on Artificial Neural Networks*, pages 44–51. Springer.

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

Daniel J Hsu, Sham M Kakade, John Langford, and Tong Zhang. 2009. Multi-label prediction via compressed sensing. In *Advances in neural information processing systems*, pages 772–780.

Shuiwang Ji, Lei Tang, Shipeng Yu, and Jieping Ye. 2008. Extracting shared subspace for multi-label classification. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 381–389. ACM.

Rie Johnson and Tong Zhang. 2014. Effective use of word order for text categorization with convolutional neural networks. *arXiv preprint arXiv:1412.1058*.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Ashish Kapoor, Raajay Viswanathan, and Prateek Jain. 2012. Multilabel classification using bayesian compressed sensing. In *Advances in Neural Information Processing Systems*, pages 2645–2653.

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

B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum. 2015. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338.

David D Lewis, Yiming Yang, Tony G Rose, and Fan Li. 2004. Rcv1: A new benchmark collection for text categorization research. *Journal of machine learning research*, 5(Apr):361–397.

Jingzhou Liu, Wei-Cheng Chang, Yuexin Wu, and Yiming Yang. 2017. Deep learning for extreme multi-label text classification. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 115–124. ACM.

Eneldo Loza Mencia and Johannes Fürnkranz. 2008. Efficient pairwise multilabel classification for large-scale problems in the legal domain. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 50–65. Springer.

Marius Pasca. *Open-Domain Question Answering from Large Text Collections*, volume 29.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Yashoteja Prabhu and Manik Varma. 2014. Fastxml: A fast, accurate and stable tree-classifier for extreme multi-label learning. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 263–272. ACM.

Xipeng Qiu and Xuanjing Huang. 2015. Convolutional neural tensor network architecture for community-based question answering. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Andreas Rücklé, Nafise Sadat Moosavi, and Iryna Gurevych. 2019. Coala: A neural coverage-based approach for long answer selection with small data. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19)*.

Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. 2017. Dynamic routing between capsules. In *Advances in Neural Information Processing Systems*, pages 3856–3866.

Cicero dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. 2016. Attentive pooling networks. *arXiv preprint arXiv:1602.03609*.

Denis Savenkov and Eugene Agichtein. 2017. Evinets: Neural networks for combining evidence signals for factoid question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 299–304.

Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, pages 373–382. ACM.

Lei Sha, Xiaodong Zhang, Feng Qian, Baobao Chang, and Zhifang Sui. 2018. A multi-view fusion neural network for answer selection. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Ying Shen, Yang Deng, Min Yang, Yaliang Li, Nan Du, Wei Fan, and Kai Lei. 2018. Knowledge-aware attentive neural network for ranking question answer pairs. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 901–904. ACM.

Ming Tan, Cicero Dos Santos, Bing Xiang, and Bowen Zhou. 2016. Improved representation learning for question answer matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 464–473.

Yi Tay, Minh C Phan, Luu Anh Tuan, and Siu Cheung Hui. 2017. Learning to rank question answer pairs with holographic dual lstm architecture. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 695–704. ACM.

Shengxian Wan, Yanyan Lan, Jiafeng Guo, Jun Xu, Liang Pang, and Xueqi Cheng. 2016. A deep architecture for semantic matching with multiple positional sentence representations. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Matt P Wand and M Chris Jones. 1994. *Kernel smoothing*. Chapman and Hall/CRC.

Di Wang and Eric Nyberg. 2015. A long short-term memory model for answer sentence selection in question answering. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 707–712.

Mengqiu Wang, Noah A Smith, and Teruko Mitamura. 2007. What is the jeopardy model? a quasi-synchronous grammar for qa. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

Mingxuan Wang, Jun Xie, Zhixing Tan, Jinsong Su, et al. 2018. Towards linear time neural machine translation with capsule networks. *arXiv preprint arXiv:1811.00287*.

Jason Weston, Ameesh Makadia, and Hector Yee. 2013. Label partitioning for sublinear ranking. In *International Conference on Machine Learning*, pages 181–189.

Congying Xia, Chenwei Zhang, Xiaohui Yan, Yi Chang, and Philip S Yu. 2018. Zero-shot user intent detection via capsule neural networks. *arXiv preprint arXiv:1809.00385*.

Liqiang Xiao, Honglun Zhang, Wenqing Chen, Yongkun Wang, and Yaohui Jin. 2018. Mcapsnet: Capsule network for text with multi-task learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4565–4574.

Xuchen Yao and Benjamin Van Durme. 2014. Information extraction over structured data: Question answering with freebase. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 956–966.

Ian En-Hsu Yen, Xiangru Huang, Pradeep Ravikumar, Kai Zhong, and Inderjit Dhillon. 2016. Pd-sparse: A primal and dual sparse approach to extreme multiclass and multilabel classification. In *International Conference on Machine Learning*, pages 3069–3077.

Scott Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base.

Wenpeng Yin, Mo Yu, Bing Xiang, Bowen Zhou, and Hinrich Schütze. 2016. Simple question answering by attentive convolutional neural network. *arXiv preprint arXiv:1606.03391*.

Hsiang-Fu Yu, Prateek Jain, Purushottam Kar, and Inderjit Dhillon. 2014a. Large-scale multi-label learning with missing labels. In *International conference on machine learning*, pages 593–601.

Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. 2014b. Deep learning for answer sentence selection. *arXiv preprint arXiv:1412.1632*.

Ningyu Zhang, Shumin Deng, Zhanling Sun, Xi Chen, Wei Zhang, and Huajun Chen. 2018a. Attention-based capsule network with dynamic routing for relation extraction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 986–992.

Suofei Zhang, Wei Zhao, Xiaofu Wu, and Quan Zhou. 2018b. Fast dynamic routing based on weighted kernel density estimation. *arXiv preprint arXiv:1805.10807*.

Wei Zhao, Jianbo Ye, Min Yang, Zeyang Lei, Suofei Zhang, and Zhou Zhao. 2018. Investigating capsule networks with dynamic routing for text classification. In *Proceedings of the 2018 conference on empirical methods in natural language processing (EMNLP)*, pages 3110–3119.