Transfer Capsule Network for Aspect Level Sentiment Classification

Zhuang Chen, Tieyun Qian*
School of Computer Science, Wuhan University, China
{zhchen18, qty}@whu.edu.cn

Abstract

Aspect-level sentiment classification aims to determine the sentiment polarity of a sentence towards an aspect. Due to the high cost in annotation, the lack of aspect-level labeled data becomes a major obstacle in this area. On the other hand, document-level labeled data like reviews are easily accessible from online websites. These reviews encode sentiment knowledge in abundant contexts. In this paper, we propose a Transfer Capsule Network (TransCap) model for transferring document-level knowledge to aspect-level sentiment classification. To this end, we first *develop an aspect routing* approach to encapsulate the sentence-level semantic representations into semantic capsules from both aspect-level and document-level data. We then *extend the dynamic routing* approach to adaptively couple the semantic capsules with the class capsules under the transfer learning framework. Experiments on SemEval datasets demonstrate the effectiveness of TransCap.

1 Introduction

Aspect-level sentiment classification (ASC) is a fine-grained subtask in sentiment analysis. Given a sentence and an aspect occurring in the sentence, ASC aims to determine the sentiment polarity of the aspect. Traditional methods mostly use machine learning models with handcrafted features to build sentiment classifiers for ASC tasks (Jiang et al., 2011; Mohammad et al., 2013). Such methods need either laborious feature engineering or massive linguistic resources. With the development of deep learning technique, a number of neural models have been proposed (Wang et al., 2016b; Tang et al., 2016; Chen et al., 2017) for ASC tasks. All these models train classifiers in a supervised manner and require sufficient number of labeled data to get promising results. However, the annotation of opinion targets in ASC is extremely expensive.

The lack of labeled data is a major obstacle in this field. Publicly available datasets for ASC often contain limited number of training samples. On the other hand, document-level labeled data like reviews are easily accessible from online websites such as Yelp and Amazon. Since each review has an accompanying rating score indicating user's overall satisfaction towards an item, such a score can naturally serve as the label of sentiment polarity of the review document.

Intuitively, the document-level data contain useful sentiment knowledge for analysis on aspect-level data since they may share many linguistic and semantic patterns. Unfortunately, for ASC tasks, only one study (He et al., 2018) has taken the utilization of document-level data into account. The PRET+MULT framework proposed in (He et al., 2018) is a successful attempt by adopting pre-training and multi-task learning approaches. However, their model only shares shallow embedding and LSTM layers between ASC and DSC (document-level sentiment classification) tasks. In other words, the document-level knowledge is merely used for improving the word representations in ASC. Consequently, it is unable for PRET+MULT to handle complicated patterns like euphemism and irony which require high-level semantic knowledge from the entire sentence. For example, given a sentence "*The staff should be a bit more friendly*", PRET+MULT will make a wrong prediction (the detail will be given in the analysis part).

In this paper, we propose a novel Transfer Capsule Network (TransCap) model to transfer sentence-level semantic knowledge from DSC to ASC. Our work is inspired by the capsule network (Hinton et al., 2011; Sabour et al., 2017)

*Corresponding author.

Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 547–556
Florence, Italy, July 28 - August 2, 2019.

which uses capsule vectors and the dynamic routing approach to store and cluster features, but we move one step further in that we *develop an aspect routing approach* which can generate sentence-level semantic features shared by ASC and DSC. Moreover, we *extend the dynamic routing approach* by adapting it to the transfer learning framework. We conduct extensive experiments on two SemEval datasets. Results demonstrate that our TransCap model consistently outperforms the state-of-the-art methods.

2 Related Work

Aspect-level Sentiment Classification Traditional methods for sentiment classification (Nakagawa et al., 2010; Jiang et al., 2011; Taboada et al., 2011; Mohammad et al., 2013) mostly use machine learning algorithms to build sentiment classifiers with carefully extracted features, which take massive time and resources to collect. Early studies focus on document-level sentiment classification (DSC) tasks. In recent years, a number of deep learning methods have been proposed for aspect-level sentiment classification (ASC) tasks (Dong et al., 2014; Vo and Zhang, 2015; Tang et al., 2016; Wang et al., 2016a; Ma et al., 2017; Li et al., 2018; Ma et al., 2018; Wang et al., 2018a).

In general, there are three types of neural networks for ASC tasks: LSTM based (Wang et al., 2016b; Ma et al., 2017; Tay et al., 2018), memory based (Tang et al., 2016; Chen et al., 2017; Zhu and Qian, 2018), and hybrid methods (Xue and Li, 2018). For example, Wang et al. (2016b) use attention mechanism to model the inter-dependence between LSTM hidden units and aspects. Tang et al. (2016) utilize memory network to store context words and conduct multi-hop attention to get the sentiment representation towards aspects. Chen et al. (2017) apply recurrent attention to multi-layer memory. Xue and Li (2018) employ CNN and gating mechanism to extract aspect-specific information from contexts.

Although various types of approaches have been proposed, the inherent obstacle, i.e., the lack of labeled data, is still a big challenge for all ASC tasks. Without sufficient labeled data, training procedures in these approaches are likely to converge in a sub-optimal state. We differentiate our work from aforementioned models in that we aim to utilize the abundant labeled DSC data to alleviate the scarcity of labeled data in ASC tasks.

Transfer Learning Transfer learning aims to extract knowledge from one or more source tasks and then apply the knowledge to a target task. It can be categorized into three types based on different situations in the source and target domains/tasks (Pan and Yang, 2010). Our work belongs to "inductive transfer learning (ITL)" type since ASC (target) and DSC (source) in our framework are different but related tasks. In this case, ITL is similar to multi-task learning (MTL) with a slight difference: ITL only aims at achieving high performance in the target task while MTL tries to improve both simultaneously.

Several recent attempts have taken ITL or MTL methods for sentiment classification tasks. Dong and de Melo (2018) present a transfer learning framework by utilizing trained models. Xiao et al. (2018) employ capsule network for multi-task learning. Both these methods are designed for document-level text/sentiment classification tasks, and are inappropriate for the fine-grained ASC task in this work. He et al. (2018) propose a multi-task framework to combine ASC with DSC tasks together. This is the closest work to ours. However, the method in (He et al., 2018) is based on an existing AT-LSTM model (Wang et al., 2016b), whereas our framework is a totally new one which employs capsule network with carefully designed strategies for ASC tasks.

3 Our Proposed TransCap Model

In this section, we introduce our Transfer Capsule Network (TransCap) model. TransCap is proposed to conduct aspect-level sentiment classification with the auxiliary knowledge transferred from document-level data. We first present the problem definitions and preliminary. We then illustrate the architecture of TransCap in detail.

3.1 Definitions and Preliminary

Definition 1 $(TransCap)$ Given a source document-level corpus $\mathcal{C}_D$ and the learning task $\mathcal{T}_D$, a target aspect-level corpus $\mathcal{C}_A$ and the learning task $\mathcal{T}_A$, TransCap aims to help improve the learning of the target predictive function $f_A(\cdot)$ in $\mathcal{T}_A$ using the knowledge transferred from $\mathcal{T}_D$.

Definition 2 ($\mathcal{T}_A$ and $\mathcal{T}_D$) Given a sentence $S = \{w_1, ..., w_a, ..., w_L\} \in \mathcal{C}_A$ and an aspect w_a occurring in S, an aspect-level sentiment classification task $\mathcal{T}_A$ aims to determine the sentiment polarity

of S towards w_a. Note there might be multiple aspects in one sentence. Given an opinion sentence (or document) $D \in \mathcal{C}_D$, a document-level sentiment classification task $\mathcal{T}_D$ aims at assigning an overall sentiment polarity for D. Note that $\mathcal{T}_A$ is the main task and $\mathcal{T}_D$ is only for providing auxiliary knowledge in our TransCap model.

Preliminary $(CapsNet)$ Capsule network is first proposed for image classification in computer vision (Hinton et al., 2011; Sabour et al., 2017). Compared with CNN, it replaces the scalar-output feature detectors with vector-output capsules and has the ability to preserve additional information such as position and thickness. The vanilla CapsNet consists of two capsule layers. The primary layer stores low-level image feature maps and the class layer generates the classification probability with each capsule corresponding to one class.

Recently, CapsNet has been applied to several NLP tasks like text classification and relation extraction (Yang et al., 2018b; Gong et al., 2018; Xiao et al., 2018; Zhang et al., 2018; Wang et al., 2018b). CapsNet is able to adaptively decide the information transferred between layers by using dynamic routing. Furthermore, each class in CapsNet has distinctive parameters to aggregate features and an independent probability to be existed. Therefore, CapsNet meets our needs in the transfer learning scenario which includes multiple polarities and tasks. Our TransCap model is the first attempt to exploit the power of CapsNet under the transfer learning framework for ASC tasks.

3.2 An Overview of Architecture

The architecture of TransCap is shown in Figure 1. It consists of four layers: 1) Input layer converts words in a sentence into low-dimensional real-valued vectors, 2) FeatCap layer extracts N-gram features from word vectors and transforms them into feature capsules, 3) SemanCap layer aggregates feature capsules into a set of aspect-related sentence-level semantic capsules, and 4) ClassCap layer generates class capsules which correspond to sentiment polarities in $\mathcal{T}_A$ and $\mathcal{T}_D$, respectively.

Note that $\mathcal{T}_A$ and $\mathcal{T}_D$ tasks share the first three layers, and they separate only in the last ClassCap layer. Since $\mathcal{T}_A$ and $\mathcal{T}_D$ are related tasks both aiming to identify the sentiment polarity, features useful for one task might be useful for the other. We expect the features produced by the shared layers can be improved in a mutual way.

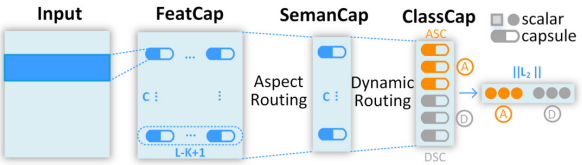

Figure 1: TransCap Architecture.

3.3 Input Layer

The input layer consists of two lookup layers. Let $\mathbf{E}_w \in \mathbb{R}^{d_w \times |V|}$ be the pre-trained word embedding lookup table, where d_w is the dimension of word vectors and $|V|$ is the vocabulary size. The word lookup layer maps the word sequence in $S(D)$ to a list of word vectors $\{\boldsymbol{e_1}, ..., \boldsymbol{e_a}, ..., \boldsymbol{e_L}\} \in \mathbb{R}^{d_w \times L}$.

Following (Gu et al., 2018), we also use another position lookup layer. For $\mathcal{T}_A$, by calculating the absolute distance from every context word w_i to aspect word w_a, we can get an additional position sequence for S. For $\mathcal{T}_D$, the position sequence is a zero sequence since there is no aspect information. Let $\mathbf{E}_l \in \mathbb{R}^{d_l \times |L|}$ be the position embedding lookup table with random initialization, the position lookup layer maps the position sequence to a list of position vectors $\{\boldsymbol{l_1}, ..., \boldsymbol{l_a}, ..., \boldsymbol{l_L}\} \in \mathbb{R}^{d_l \times L}$.

The final representation of each word w_i is calculated as $\boldsymbol{x_i} = (\boldsymbol{e_i} \oplus \boldsymbol{l_i}) \in \mathbb{R}^{d_h}$ where $\oplus$ denotes concatenation and $d_h = d_w + d_l$. The sentence $S(D)$ is transformed into a sentence embedding $\mathbf{X} = \{\boldsymbol{x_1}, ..., \boldsymbol{x_L}\} \in \mathbb{R}^{d_h \times L}$.

3.4 Feature Capsule Layer

This layer is used to extract n-gram features from sentence embedding $\mathbf{X}$. N-gram features contain raw and local semantic meaning in a fixed window. We apply multiple convolution operations to the i-th n-gram in $\mathbf{X}$ and get its feature vector $\boldsymbol{r_i}$:

$$\boldsymbol{r_i} = \mathbf{X}_{i:i+K} * \mathbf{F} + b, \tag{1}$$

where $\mathbf{F} \in \mathbb{R}^{d_p \times (d_h \times K)}$ is the kernel group, $(d_h \times K)$ is the size of one convolutional kernel, K is the n-gram size and d_p is the dimension of one feature capsule. After sliding $\mathbf{F}$ in $\mathbf{X}$, we get a set of feature capsules $\boldsymbol{r} \in \mathbb{R}^{d_p \times (L-K+1)}$ encapsulating n-gram features extracted from the whole sentence $S(D)$.

Since one kernel group $\mathbf{F}$ corresponds to one category of semantic meaning, we repeat the above procedure C times with different kernel groups, and get multiple channels of feature capsules representing C categories of semantic meaning. The final output of feature capsule layer is arranged as $\mathbf{R} \in \mathbb{R}^{C \times d_p \times (L-K+1)}$:

$$\mathbf{R} = [\boldsymbol{r_1}, \boldsymbol{r_2}, ..., \boldsymbol{r_C}] \tag{2}$$

3.5 Semantic Capsule Layer

Aspect Routing Approach The sentence or document in two corpora $\mathcal{C}_\mathcal{A}$ and $\mathcal{C}_\mathcal{D}$ differs in whether an aspect term occurs in the sentence/document. The $\mathcal{T}_D$ task does not contain aspects. Meanwhile, it is crucial for the $\mathcal{T}_A$ task to determine the relation between contexts and aspects. Especially when a sentence contains two opposite sentiment polarities, different contexts must be separated for different aspects. For example, given a sentence *"Great food but the service is dreadful !"*, the context word "dreadful" should be strengthened for the aspect "service" and be weakened for the aspect "food".

To this end, we propose a novel *aspect routing* approach to compute the aspect weight for the context words of K-size window in $\mathcal{T}_A$. Formally, we apply a fusing convolution operation to the sentence embedding $\mathbf{X}$ with a kernel $\mathbf{F}_a \in \mathbb{R}^{d_h \times K}$, and we get the aspect routing weight a_i:

$$a_i = sigmoid(\mathbf{X}_{i:i+K} * \mathbf{F}_a + \mathbf{T}_a \boldsymbol{e_a} + b_a), \quad (3)$$

where $\boldsymbol{e_a}$ is the aspect embedding (or average embedding in the case of multi-word aspect), $\mathbf{T}_a \in \mathbb{R}^{1 \times d_w}$ is a transfer matrix to map $\boldsymbol{e_a}$ to a scalar value, and b_a is bias. The generated routing weight $a_i \in [0, 1]$ fuses aspect information with respect to its context. It controls how much information in the current context can be transmitted to the next layer. If a_i is zero, the feature capsule would be totally blocked.

A minor challenge is that, for a $\mathcal{T}_D$ task, there is actually no aspect in the document and we need to distinguish two types of sources from $\mathcal{C}_A$ and $\mathcal{C}_D$. Hence we design a piecewise function g_i for calculating the aspect routing weight g_i for an arbitrary feature vector $\boldsymbol{r_i}$ from $\mathbf{X}$ as:

$$g_i = \begin{cases} a_i & \mathrm{X} \in \mathcal{C}_A \\ 1.0 & \mathrm{X} \in \mathcal{C}_D \end{cases} \quad (4)$$

After sliding in $\mathbf{X}$, we can get $\boldsymbol{g} \in \mathbb{R}^{1 \times (L-K+1)}$ for the whole sentence $S(D)$. Since we have C channels of feature capsules, we repeat the above procedure C times to get the entire aspect routing weights $\mathbf{G} \in \mathbb{R}^{C \times 1 \times (L-K+1)}$ as:

$$\mathbf{G} = [\boldsymbol{g_1}, \boldsymbol{g_2}, ..., \boldsymbol{g_C}], \quad (5)$$

Finally, the feature capsules are routed using these weights:

$$\mathbf{P} = \mathbf{R} \odot \mathbf{G}, \quad (6)$$

where $\mathbf{P} \in \mathbb{R}^{C \times d_p \times (L-K+1)}$ are the aspect-customized feature capsules, and $\odot$ denotes element-wise multiplication (with broadcasting).

Semantic Capsule Generation The above generated $\mathbf{P}$ are transformed from the n-gram feature capsules. Though encoding aspect-related information, $\mathbf{P}$ are still local features without a sentence-level view. Moreover, the large number of capsules in $\mathbf{P}$ may prevent the next layer from learning robust representations. Hence we adopt the element-wise maximum function (Lai et al., 2015) in $\mathbf{P}$ to aggregate all feature capsules in same channel horizontally.

$$\mathbf{U} = \max_{t=1}^{C \times d_p} \mathbf{P}_t, \quad (7)$$

where $\mathbf{U} \in \mathbb{R}^{C \times d_p}$ are the generated semantic capsules. Eq. 7 condenses all local features in each channel and thus we can obtain more precise and global semantic representations from subtle expressions, e.g., an euphemistic sentence. Finally, we want the length of each semantic capsule $\boldsymbol{u_i}$ to represent the probability that $\boldsymbol{u_i}$'s semantic meaning is present in the current input, so we use a nonlinear "*squash*" function (Sabour et al., 2017) to limit its length in [0,1] as

$$\boldsymbol{u_i} \leftarrow \frac{\|\boldsymbol{u_i}\|^2}{1 + \|\boldsymbol{u_i}\|^2} \frac{\boldsymbol{u_i}}{\|\boldsymbol{u_i}\|} \quad (8)$$

3.6 Class Capsule Layer

In the original capsule network, there is only one classification task and it uses class capsules to denote classes and their lengths as classification probabilities. However, there are two different tasks in our problem, and it is necessary to discern sentiment polarities (classes) in these tasks. To achieve this, we introduce two types of class capsules into TransCap, with six capsules in total. Such a structure makes it possible for our model to train $\mathcal{T}_A$ and $\mathcal{T}_D$ in a unified framework.

Given input data from two tasks in turn, the first three layers share most parameters (except those in Eq. 3) to jointly train $\mathcal{T}_D$ and $\mathcal{T}_A$, so that knowledge from document-level data can be successfully transferred into aspect-level task. In the last layer, each class capsule is used for calculating the classification probability of each class in $\mathcal{T}_D$ and $\mathcal{T}_A$ separately. Hence each class capsule should have its own routing weights to adaptively aggregate semantic capsules from the previous layer. Below we give the detail.

A semantic capsule i generates a "prediction vector" $\hat{\boldsymbol{u}}_{j|i}$ towards a class capsule j as:

$$\hat{\boldsymbol{u}}_{j|i} = \mathbf{W}_{ij}\, \boldsymbol{u_i}, \quad (9)$$

where $\mathbf{W}_{ij} \in \mathbb{R}^{d_c \times d_p}$ is a weight matrix, d_p and

d_c are the dimensions of semantic capsule i and class capsule j, $\boldsymbol{u_i}$ is the vector representation of semantic capsule i. All "prediction vectors" generated by semantic capsules are summed up with weights c_{ij} to obtain the vector representation $\boldsymbol{s_j}$ of class capsule j:

$$\boldsymbol{s_j} = \sum_i c_{ij} \hat{\boldsymbol{u}}_{\boldsymbol{j|i}}, \tag{10}$$

where c_{ij} is a coupling coefficient defined by a "routing softmax":

$$c_{ij} = \frac{exp(b_{ij})}{\sum_k exp(b_{ik})}, \tag{11}$$

where each b_{ij} is the log prior probability that a semantic capsule i should pass to a class capsule j. It is computed using a *dynamic routing* approach which will be presented later.

After that, we again apply the non-linear "*squash*" function (Sabour et al., 2017) to $\boldsymbol{s_j}$ in Eq. 10 to get a final representation $\boldsymbol{v_j}$ for class capsule j.

$$\boldsymbol{v_j} = squash(\boldsymbol{s_j}), \tag{12}$$

where the length of $\boldsymbol{v_j}$ is limited in [0,1] to represent the active probability of class capsule j.

Dynamic Routing Approach The logit b_{ij} in Eq. 11 determines the intensity of the connection between the semantic capsule i and the class capsule j. It is initialized with 0 and is updated with an agreement coefficient a_{ij}.

$$a_{ij} = \hat{\boldsymbol{u}}_{\boldsymbol{j|i}} \cdot \boldsymbol{v_j} \tag{13}$$

This agreement coefficient is added to the initial logit b_{ij} before computing the new values for all coupling coefficients c_{ij} linking semantic capsules to class capsules.

$$b_{ij} \leftarrow b_{ij} + a_{ij} \tag{14}$$

The dynamic routing procedure can be summarized as (Eq. 11→ 10→ 12→ 13→ 14). The procedure can be repeated for r iterations.

3.7 Margin Loss

The length of a class capsule is used to represent the probability of the sentiment polarity. The capsule length of the active class should be larger than others. Hence we adopt a separate margin loss $\mathcal{L}_j$ for each class capsule j in each task:

$$\begin{aligned}\mathcal{L}_j &= Y_j max(0, m^+ - \|\boldsymbol{v_j}\|)^2 \\ &+ \lambda(1 - Y_j) max(0, \|\boldsymbol{v_j}\| - m^-)^2,\end{aligned} \tag{15}$$

where Y_j=1 if the sentiment polarity is present in class capsule j, and we simply set m^+=0.9, m^-=0.1, λ=0.5 following those in (Sabour et al., 2017). The loss for a single task is $\mathcal{L}_T = \sum_{j=1}^{J} \mathcal{L}_j$, where T is either A or D, denoting the loss $\mathcal{L}_A$ and $\mathcal{L}_D$ for task $\mathcal{T}_A$ and $\mathcal{T}_D$, respectively. The final loss $\mathcal{L}$ for our TransCap model is the linear combination of two losses on single tasks.

$$\mathcal{L} = \mathcal{L}_A + \gamma \mathcal{L}_D \tag{16}$$

where $\gamma \in$[0,1] is a hyper-parameter controlling the weight of $\mathcal{T}_D$. When training converges, the class capsule with the largest active probability in a task is chosen as the prediction of sentiment polarities.

4 Experiments

4.1 Datasets and Settings

Datasets for $\mathcal{T}_A$ We evaluate TransCap on two aspect-level datasets from SemEval2014 Task 4 (Pontiki et al., 2014). The datasets contain reviews from *Restaurant* and *Laptop* domains respectively with 3-way sentiment polarity labels: *positive*, *neutral* and *negative* [1]. Both datasets have a fixed training/test split. We further randomly sample 20% training data as the development set, and use the remaining 80% for training.

Datasets for $\mathcal{T}_D$ We use three document-level datasets to transfer knowledge: *Yelp*, *Amazon* and *Twitter*. All the documents (reviews) in Yelp Review (Zhang et al., 2015) and Amazon Electronics (McAuley et al., 2015) datasets have accompanying five-star ratings (1..5). We consider reviews with a score <3 as *negative*, =3 as *neutral* and >3 as *positive*. The *Twitter* dataset is collected from SemEval 2013 to 2017, where the original tweets are already labeled with 3-way polarities. Each dataset for $\mathcal{T}_D$ contains 30,000 samples with balanced class labels. All samples in these datasets are used for auxiliary training. We do not report performance for the $\mathcal{T}_D$ task since it is not our focus.

Also note that the first two datasets in $\mathcal{T}_D$ are of the same topics as those in $\mathcal{T}_A$, while the topics in *Twitter* are more general and less relevant to our main task $\mathcal{T}_A$. There are two combinations for TransCap: {Y,A} denotes {*Res.*+*Yelp*, *Lap.*+*Amazon*}, {T,T} denotes {*Res.*+*Twitter*, *Lap.*+*Twitter*}. By doing so, we wish to investigate how our proposed model performs on various types of auxiliary information. The statistics for these datasets are summarized in Table 1.

[1] We remove samples with *conflict* polarities following previous studies (Tang et al., 2016; Chen et al., 2017; He et al., 2018).

Task	Dataset	Type	Pos.	Neu.	Neg.
$\mathcal{T}_A$	Restaurant	train	2164	633	805
		test	728	196	196
	Laptop	train	987	460	866
		test	341	169	128
$\mathcal{T}_D$	Yelp	train	10k	10k	10k
	Amazon	train	10k	10k	10k
	Twitter	train	10k	10k	10k

Table 1: The statistics for datasets.

Settings We use Glove vectors with 840B tokens (Pennington et al., 2014) as the pre-trained word embeddings. r=3 following Sabour et al. (2017). The rest of hyperparameters are tuned on the development set. We set d_w=300, d_l=100, K=3, C=16, d_p=16, d_c=24. γ={0.7, 0.8, 0.8, 0.3} for the {R,Y}, {R,T}, {L,A}, {L,T} dataset combinations, respectively. We use Adam optimizer (Kingma and Ba, 2014) with learning rate 0.001 and batch size 128. We train all models for 50 epochs with early-stopping, i.e., stop training if the performance on the development set does not improve among 5 epochs. The averaged accuracy (Acc.) and Macro-F1 (F1) scores are reported over 5 runs with random initialization on the same split of evaluation datasets [2].

Compared Methods To demonstrate the superiority of our TransCap for ASC tasks, we compare it with followings baselines: ATAE-LSTM (Wang et al., 2016b), IAN (Ma et al., 2017), AF-LSTM(CONV) (Tay et al., 2018), AF-LSTM(CORR) (Tay et al., 2018), PBAN (Gu et al., 2018), MemNN (Tang et al., 2016), RAM (Chen et al., 2017), CEA (Yang et al., 2018a), DAuM (Zhu and Qian, 2018), IARM (Majumder et al., 2018), PRET+MULT (He et al., 2018) and GCAE (Xue and Li, 2018). Most of them are the latest methods published in 2018. The rest are frequently-used classical models.

4.2 Main Results

The comparison results for all models are shown in Table 2. For clarity, we classify the models into four categories: the first is the LSTM-based methods (from M1 to M5), the second is the memory-based ones (from M6 to M10), the third is the hybrid ones (M11 and M12), and the last three lines (M13 to M15) are the variants of our model, where TransCap{S} denotes the one with $\mathcal{T}_A$ task only, TransCap{Y,A} and TransCap{T,T} utilize the knowledge from different sources in $\mathcal{T}_D$.

[2]Our code and data are available at https://github.com/NLPWM-WHU/TransCap.

	Model	Restaurant		Laptop	
		Acc.	F1	Acc.	F1
M1	ATAE-LSTM	78.38	66.36	69.12	63.24
M2	IAN	78.71	67.71	69.56	63.72
M3	AF-LSTM(CONV)	76.46	65.54	69.97	63.70
M4	AF-LSTM(CORR)	75.96	64.41	69.78	63.38
M5	PBAN	78.62	67.45	71.98	66.91
M6	MemNN	77.69	67.53	68.86	62.60
M7	RAM	78.41	68.52	<u>72.16</u>	66.97
M8	CEA	78.44	66.78	70.52	64.52
M9	DAuM	77.91	66.47	70.36	65.86
M10	IARM	77.73	66.66	68.63	63.30
M11	PRET+MULT	<u>78.73</u>	<u>68.63</u>	71.91	<u>68.79</u>
M12	GCAE	76.09	63.29	68.72	63.32
M13	TransCap{S}	78.84	69.70	72.65	68.77
M14	TransCap{Y,A}	**79.55**	**71.41**	73.51	69.81
M15	TransCap{T,T}	79.29	70.85	**73.87**	**70.10**

Table 2: Comparison of different methods. Best scores are in bold, and the second best ones (except those in our variants) are underlined.

It is clear that our TransCap model consistently outperforms all baselines on both datasets. The hybrid model PRET+MULT, which is a multi-task learning based model, also has the overall better performance than other baselines. Both these demonstrate that the aspect-level sentiment classification task $\mathcal{T}_A$ can benefit a lot by transferring knowledge from the auxiliary task $\mathcal{T}_D$. PRET+MULT is inferior to our model. The reason is that it only shares low-level features and transfers limited knowledge between tasks.

We also find that two multi-task variants of our model, TransCap{Y,A} and TransCap{T,T}, achieve similar performance. {Y,A} provides knowledge from relevant domains, but their labels are not very accurate since they may contain a lot of noises. Though the knowledge in {T,T} are from tweets of mixed and less relevant topics, their labels are manually-annotated and thus are quite reliable. Overall, given the sufficient number of training samples in the auxiliary task $\mathcal{T}_D$, the performance of $\mathcal{T}_A$ tasks can be significantly enhanced over its single task counterpart TransCap{S}.

Among LSTM-based models, PBAN and IAN achieve higher performance than others since they use the bi-directional attention mechanism. RAM is better than other memory-based models because it utilizes a non-linear combination for attention results in different hops. GCAE performs the worst among all baselines, as its simple CNN-based model can not capture the long-term dependencies between context words.

5 Analysis

5.1 Ablation Study

To investigate the effects of different components in our model, we conduct the following ablation study on TransCap. (i)"- A": We remove the aspect routing approach, and set same weights 1.0 for all feature capsules. (ii)"- S": We remove semantic capsules, and pass weighted feature capsules directly to class capsules. (iii)"- D": We remove the dynamic routing approach, i.e., a semantic capsule would be coupled to all class capsules with equal probabilities.

Results for the ablation study are shown in Table 3, where "Ori." denotes results for the original TransCap model, and "-*" for those removing the corresponding components.

	Restaurant				Laptop			
	{Y,A}		{T,T}		{Y,A}		{T,T}	
	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1
Ori.	79.55	71.41	79.29	70.85	73.51	69.81	73.87	70.10
- A.	3.75↓	**6.49**↓	**2.63**↓	**3.95**↓	2.98↓	5.34↓	**3.34**↓	**3.80**↓
- S.	**4.01**↓	5.14↓	1.45↓	2.08↓	2.35↓	3.64↓	2.40↓	2.15↓
- D.	2.80↓	4.06↓	0.54↓	1.01↓	**3.29**↓	**6.03**↓	1.14↓	1.75↓

Table 3: Ablation study for TransCap. ↓ denotes the drop of performance. The worst scores are in bold.

As expected, results for the simplified models all drop a lot. This clearly demonstrates the effectiveness of these components. Specifically, TransCap-A performs the worst, since it cannot generate aspect-related feature capsules after removing aspect routing from TransCap. Dynamic routing is critical as it helps TransCap to reduce the interference between $\mathcal{T}_A$ and $\mathcal{T}_D$. The drop of performance of TransCap-S also shows that semantic capsules are important for building robust and precise connections between features and polarities.

5.2 Parameter Analysis

Influence of Auxiliary Corpus Size To show the influence of DSC task on our major ASC task, we vary the size of auxiliary document-level corpus $\mathcal{C}_D$ and observe the performance changes in $\mathcal{T}_A$. We use a percentage $\in [0, 1]$ to control the ratio of $\mathcal{C}_D$ and present results in Figure 2.

As can be seen, all curves in Figure 2 tend to rise with the increasing amount of document-level knowledge. This shows the effectiveness of our model by transferring knowledge from document-level data. At the initial stages where only 20% or 40% of $\mathcal{C}_D$ are introduced, we find small decreases of performance. The reason may be that when the auxiliary document-level corpus $\mathcal{C}_D$ is small, the model in $\mathcal{T}_D$ has not been well trained. Hence it provides limited transferable knowledge to train the shared input, feature capsule and semantic capsule layers. Consequently, ASC task $\mathcal{T}_A$ gets misleading information from these layers and then performs worse. After getting sufficient document-level data, $\mathcal{T}_D$ becomes robust and stable, and $\mathcal{T}_A$ also improves its performance.

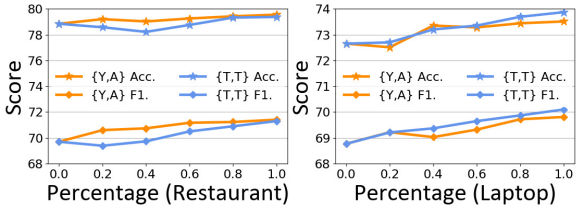

Figure 2: Influence of $\mathcal{C}_D$ size.

Effects of Balance Factor γ The balance factor γ determines how important the DSC task $\mathcal{T}_D$ is in the model. To evaluate its effects, we vary γ in range [0,1] and present results in Figure 3.

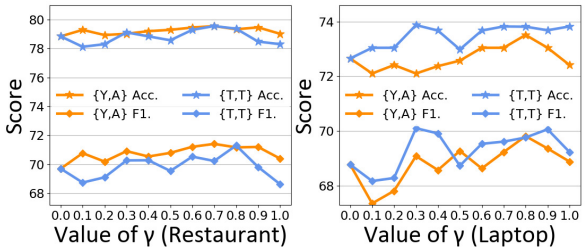

Figure 3: Effects of γ.

The key observation from Figure 3 is that there are Turning Points (denoted as TP) for both two datasets: TP≈0.7 for *Restaurant* and TP≈0.3 for *Laptop*. The curves have an overall upward trend when $\gamma <$ TP, but become flat or downward once $\gamma >$ TP. This phenomenon can be explained with multi-task learning mechanism. In upward part, lots of useful sentiment knowledge is transferred from document-level data to aspect-level data, thus the performance of $\mathcal{T}_A$ gets improved. Once the weight for $\mathcal{T}_D$ exceeds TP, $\mathcal{T}_D$ begins to dominate the whole TransCap model while $\mathcal{T}_A$ gradually loses the mastership and performs worse.

5.3 Case Study

To have a close look, we further select three samples from different datasets for a case study.

Part 1 We first illustrate what kind of knowledge TransCap will transfer. Below is an example from

Laptop where the target is enclosed in [] with a subscript denoting its true polarity:

1.*"It has so much more speed and the [screen]$_{pos}$ is very sharp."*

Humans can easily identify the *positive* polarity towards aspect [screen]. However, the single-task variant TransCap{S} and most baselines give a false *negative* prediction. This is because "sharp" is a multi-polarity word in the training set as the following two examples show:

2.*"Once open, the [leading edge]$_{neg}$ is razor sharp."*

3.*"[Graphics]$_{pos}$ are clean and sharp, internet interfaces are seamless."*

The training set in *Laptop* contains only 8 samples including "sharp" with 5 of them are labeled as *negative*. It is hard for single-task models to learn a correct meaning for "sharp" with several contradictory samples. Hence they simply consider it as a *negative* token due to the superiority of this polarity and make false predictions. However, for TransCap{Y,A}, the auxiliary *Amazon* dataset contains 294 samples where "sharp" co-occurs with lots of different contexts. With the help of sufficient training samples, three shared layers have learned to recognize the true polarity of "sharp" with respect to its contexts, thus the class capsule layer in TransCap{Y,A} finally makes a correct prediction.

Part 2 This part aims to visualize the decision-making process of TransCap with an example from *Restaurant* dataset:

4.*"Great [food]$_{pos}$ but the [service]$_{neg}$ is dreadful !"*.

The coupling coefficients $c_{ij} \in [0,1]$ for this example are visualized in Figure 4, which presents the c_{ij} between each pair of (semantic capsule, class capsule) after dynamic routing with respect to different aspects. Note that the sum of c_{ij} in every *column* (not row as that in the attention mechanism) is 1.0.

When the input aspect is [service] (the upper part in Figure 4), the detailed decision-making process is as follow. Firstly, several semantic capsules such as 4 and 8 have already captured corresponding sentence-level semantic meaning from the review's content. Secondly, by calculating the coupling coefficient c_{ij} after dynamic routing, these semantic capsules are highly coupled with the *negative* class capsule, and thus this *negative* capsule gets a higher active probability than other class capsules. As a result, TransCap makes the *negative* prediction for the aspect [service]. Similarly, when the input aspect is [food] (the lower part in Figure 4), the *positive* class capsule gets a high active probability and TransCap then makes a correct prediction for this aspect.

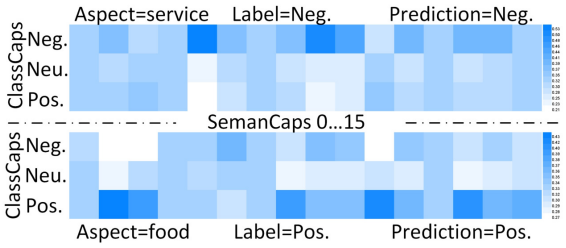

Figure 4: Visualization of coupling coefficients c_{ij} after dynamic routing.

Part 3 In last part, we present an example from *Restaurant* to show the advantage of TransCap over PRET+MULT (He et al., 2018):

5.*"The [staff]$_{neg}$ should be a bit more friendly."*

This is an euphemistic *negative* review towards the aspect [staff] though each word in the sentence itself does not convey a *negative* sentiment. PRET+MULT generates features and transfers knowledge only at the word level. Although embedding for each word is enhanced by the auxiliary document-level data, PRET+MULT can not recognize the overall *negative* sentiment behind each word and makes a false *positive* prediction due to the word "friendly". In contrast, TransCap generates sentence-level semantic capsules containing overall semantic meanings of the sentence, and shares these sentence-level features between ASC and DSC tasks. Both these help TransCap make a correct decision.

6 Conclusion

In this paper, we present a novel transfer capsule network (TransCap) model for aspect-level sentiment classification. In order to solve the problem of lacking aspect-level labeled data, we wish to utilize the abundant document-level labeled data. We develop a transfer learning framework to transfer knowledge from the document-level task to the aspect-level task. We implement it with a carefully designed capsule network, which mainly consists of the aspect routing and dynamic routing approaches. Experiments on two SemEval datasets demonstrate that TransCap outperforms the state-of-the-art baselines by a large margin.

Acknowledgments

The work described in this paper is supported by the NSFC projects (61572376, 91646206), and the 111 project (B07037).

References

Peng Chen, Zhongqian Sun, Lidong Bing, and Wei Yang. 2017. Recurrent attention network on memory for aspect sentiment analysis. In *Conference on Empirical Methods in Natural Language Processing (EMNLP 2017)*.

Li Dong, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, and Ke Xu. 2014. Adaptive recursive neural network for target-dependent twitter sentiment classification. In *Annual Meeting of the Association for Computational Linguistics (ACL 2014)*.

Xin Dong and Gerard de Melo. 2018. A helping hand: Transfer learning for deep sentiment analysis. In *Annual Meeting of the Association for Computational Linguistics (ACL 2018)*.

Jingjing Gong, Xipeng Qiu, Shaojing Wang, and Xuanjing Huang. 2018. Information aggregation via dynamic routing for sequence encoding. In *Conference on Computational Linguistics (COLING 2018)*.

Shuqin Gu, Lipeng Zhang, Yuexian Hou, and Yin Song. 2018. A position-aware bidirectional attention network for aspect-level sentiment analysis. In *Conference on Computational Linguistics (COLING 2018)*.

Ruidan He, Wee Sun Lee, Hwee Tou Ng, and Daniel Dahlmeier. 2018. Exploiting document knowledge for aspect-level sentiment classification. In *Annual Meeting of the Association for Computational Linguistics (ACL 2018)*.

Geoffrey E. Hinton, Alex Krizhevsky, and Sida D. Wang. 2011. Transforming auto-encoders. In *International Conference on Artificial Neural Networks (ICANN 2011)*.

Long Jiang, Mo Yu, Ming Zhou, Xiaohua Liu, and Tiejun Zhao. 2011. Target-dependent twitter sentiment classification. In *Annual Meeting of the Association for Computational Linguistics (ACL 2011)*.

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

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *AAAI Conference on Artificial Intelligence (AAAI 2015)*.

Xin Li, Lidong Bing, Wai Lam, and Bei Shi. 2018. Transformation networks for target-oriented sentiment classification. In *Annual Meeting of the Association for Computational Linguistics (ACL 2018)*.

Dehong Ma, Sujian Li, Xiaodong Zhang, and Houfeng Wang. 2017. Interactive attention networks for aspect-level sentiment classification. In *International Joint Conference on Artificial Intelligence (IJCAI 2017)*.

Yukun Ma, Haiyun Peng, and Erik Cambria. 2018. Targeted aspect-based sentiment analysis via embedding commonsense knowledge into an attentive LSTM. In *AAAI Conference on Artificial Intelligence (AAAI 2018)*.

Navonil Majumder, Soujanya Poria, Alexander F. Gelbukh, Md. Shad Akhtar, Erik Cambria, and Asif Ekbal. 2018. IARM: Inter-aspect relation modeling with memory networks in aspect-based sentiment analysis. In *Conference on Empirical Methods in Natural Language Processing (EMNLP 2018)*.

Julian J. McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. 2015. Image-based recommendations on styles and substitutes. In *Conference on Research and Development in Information Retrieval (SIGIR 2015)*.

Saif Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu. 2013. NRC-Canada: Building the state-of-the-art in sentiment analysis of tweets. In *International Workshop on Semantic Evaluation, (SemEval@NAACL-HLT 2013)*.

Tetsuji Nakagawa, Kentaro Inui, and Sadao Kurohashi. 2010. Dependency tree-based sentiment classification using CRFs with hidden variables. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL 2010)*.

Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*.

Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. 2014. SemEval-2014 task 4: Aspect based sentiment analysis. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*.

Sara Sabour, Nicholas Frosst, and Geoffrey E. Hinton. 2017. Dynamic routing between capsules. In *Conference on Neural Information Processing Systems (NIPS 2017)*.

Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberly Voll, and Manfred Stede. 2011. Lexicon-based methods for sentiment analysis. *Computational Linguistics*.

Duyu Tang, Bing Qin, and Ting Liu. 2016. Aspect level sentiment classification with deep memory network. In *Conference on Empirical Methods in Natural Language Processing (EMNLP 2016)*.

Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2018. Learning to attend via word-aspect associative fusion for aspect-based sentiment analysis. In *AAAI Conference on Artificial Intelligence (AAAI 2018)*.

Duy-Tin Vo and Yue Zhang. 2015. Target dependent twitter sentiment classification with rich automatic features. In *International Joint Conferences on Artificial Intelligence (IJCAI 2015)*.

Shuai Wang, Sahisnu Mazumder, Bing Liu, Mianwei Zhou, and Yi Chang. 2018a. Target-sensitive memory networks for aspect sentiment classification. In *Annual Meeting of the Association for Computational Linguistics (ACL 2018)*.

Wenya Wang, Sinno Jialin Pan, Daniel Dahlmeier, and Xiaokui Xiao. 2016a. Recursive neural conditional random fields for aspect-based sentiment analysis. In *Conference on Empirical Methods in Natural Language Processing (EMNLP 2016)*.

Yequan Wang, Minlie Huang, Xiaoyan Zhu, and Li Zhao. 2016b. Attention-based LSTM for aspect-level sentiment classification. In *Conference on Empirical Methods in Natural Language Processing (EMNLP 2016)*.

Yequan Wang, Aixin Sun, Jialong Han, Ying Liu, and Xiaoyan Zhu. 2018b. Sentiment analysis by capsules. In *Conference on World Wide Web (WWW 2018)*.

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

Wei Xue and Tao Li. 2018. Aspect based sentiment analysis with gated convolutional networks. In *Annual Meeting of the Association for Computational Linguistics (ACL 2018)*.

Jun Yang, Runqi Yang, Chongjun Wang, and Junyuan Xie. 2018a. Multi-entity aspect-based sentiment analysis with context, entity and aspect memory. In *AAAI Conference on Artificial Intelligence (AAAI 2018)*.

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

Ningyu Zhang, Shumin Deng, Zhanlin Sun, Xi Chen, Wei Zhang, and Huajun Chen. 2018. Attention-based capsule networks with dynamic routing for relation extraction. In *Conference on Empirical Methods in Natural Language Processing (EMNLP 2018)*.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Conference on Neural Information Processing Systems (NIPS 2015)*.

Peisong Zhu and Tieyun Qian. 2018. Enhanced aspect level sentiment classification with auxiliary memory. In *Conference on Computational Linguistics (COLING 2018)*.