Artificial Intelligence 265 (2018) 1–17

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Memory networks for fine-grained opinion mining

Wenya Wang [a], Sinno Jialin Pan [a,*], Daniel Dahlmeier [b]

[a] Nanyang Technological University, Singapore
[b] SAP Innovation Center Singapore, Singapore

ARTICLE INFO

Article history:
Received 21 November 2017
Received in revised form 28 August 2018
Accepted 17 September 2018
Available online 17 October 2018

Keywords:
Fine-grained opinion mining
Deep learning
Memory networks
Multi-task learning

ABSTRACT

Fine-grained opinion mining has attracted increasing attention recently because of its benefits for providing richer information compared with coarse-grained sentiment analysis. Under this problem, there are several existing works focusing on aspect (or opinion) terms extraction which utilize the syntactic relations among the words given by a dependency parser. These approaches, however, require additional information and highly depend on the quality of the parsing results. As a result, they may perform poorly on user-generated texts, such as product reviews, tweets, etc., whose syntactic structure is not precise. In this work, we offer an end-to-end deep learning model without any preprocessing. The model consists of a memory network that automatically learns the complicated interactions among aspect words and opinion words. Moreover, we extend the network with a multi-task manner to solve a finer-grained opinion mining problem, which is more challenging than the traditional fine-grained opinion mining problem. To be specific, the finer-grained problem involves identification of aspect and opinion terms within each sentence, as well as categorization of the identified terms at the same time. To this end, we develop an end-to-end multi-task memory network, where aspect/opinion terms extraction for a specific category is considered as a task, and all the tasks are learned jointly by exploring commonalities and relationships among them. We demonstrate state-of-the-art performance of our proposed model on several benchmark datasets.

1. Introduction

In fine-grained opinion mining, aspect-based analysis aims to provide fine-grained information via token-level predictions. Under this branch, a number of works have been proposed for aspect/opinion terms extraction [1–3]. Here, an aspect term refers to a word or a phrase describing some feature of an entity, and an opinion term refers to the expression carrying subjective emotions. For example, in the sentence "*The soup is served with nice portion, the service is prompt*", *soup*, *portion* and *service* are aspect terms, while *nice* and *prompt* are opinion terms. Most of the existing works focused on solely aspect terms extraction due to the absence of opinion term annotations in large-scale dataset. However, opinions also play an important role [4] in fine-grained opinion mining in order to achieve structured review summarization as a final goal. At this point, we provide additional opinion term annotations which have been made public[1] and propose to solve both aspect and opinion terms extraction at the same time.

* Corresponding author.
E-mail addresses: wa0001ya@ntu.edu.sg (W. Wang), sinnopan@ntu.edu.sg (S.J. Pan), d.dahlmeier@sap.com (D. Dahlmeier).

[1] https://github.com/happywwy/Coupled-Multi-layer-Attentions.

https://doi.org/10.1016/j.artint.2018.09.002
0004-3702/

In the literature, there exist many lines of works for aspect and/or opinion terms extraction. In [1,5,2] the opinion targets are mined through pre-defined rules based on syntactic or dependency structure of each sentence. In [6,7] extensive feature engineering is applied to build a classifier from annotated corpus to predict a label (aspect, opinion, or others) on each token in each sentence. These two categories of approaches are labor-intensive for constructing rules or features using linguistic and syntactic information. To reduce the engineering effort, deep-learning-based approaches [8,3] are proposed to learn high-level representation for each token, on which a classifier can be trained. Despite some promising results, most deep-learning approaches still require a parser analyzing the syntactic/dependency structure of the sentence to be encoded into the deep models. In this case, the performances might be affected by the quality of the parsing results. There are also recent approaches using convolutional neural networks (CNNs) [9,10] or recurrent neural networks (RNNs) [11]. However, without the syntactic structure, CNN can only learn general contextual interactions within a specified window size without focusing on the desired propagation between aspect terms and opinion terms. It is also challenging to extract the prominent features corresponding to aspects or opinions from convolutional kernels. RNNs are even weaker to capture skip connections among syntactically-related words.

In practice, the dependency structures of many user-generated texts may not be precise with a computational parser, especially in informal texts, which may degrade the performances of existing approaches. Therefore, we propose to use the attention mechanism [12] with tensor operators in a memory network to replace the role of dependency parsers to automatically capture the relations among tokens in each sentence. Specifically, we design a couple of attentions, one for aspects extraction and the other for opinions extraction. They are learned interactively such that label information can be dually propagated among aspect terms and opinion terms by exploiting their relations. Moreover, we use a memory network to explore multiple layers of the coupled attentions in order to extract inconspicuous aspect/opinion terms.

Going further, we extend the extraction task to a finer-grained problem named *category-specific aspect and opinion terms extraction*, where aspect/opinion terms need to be extracted and classified to a category from a pre-defined set, simultaneously. This could provide a more structured opinion outputs and is also benefitial for linking aspect terms and opinion terms through their category information. Consider the previous example. The objective is to extract and classify *soup* and *portion* as aspect terms under the "DRINKS" category, and *service* as an aspect term under the "SERVICE" category, similar for the opinion terms *nice* and *prompt*. To this extent, some previous works focus on categorization of aspect terms, where aspect terms are extracted in advance, and the goal is to classify them into one of the predefined categories [13–16]. The joint task is much more challenging and has rarely been investigated because when specific categories are taken into consideration for terms extraction, training data become extremely sparse, e.g. certain categories may only contain very few reviews or sentences. Moreover, it requires to achieve both extraction and categorization, simultaneously, which significantly increases the difficulty compared with the task of only extracting overall aspect/opinion terms or classifying pre-extracted terms. Although topic models [17,18] can achieve both grouping and extraction at the same time, they mainly focus on grouping, and could only identify general and coarse-grained aspect terms. To solve the problem, we offer an end-to-end deep multi-task learning architecture. Our high-level idea is that we consider terms extraction for each specific category as an individual task, where we can use the proposed memory network aforementioned for co-extracting aspect and opinion terms. The memory networks are then jointly learned in a multi-task learning manner to address the data sparsity issue of each task.

In summary, our contributions are 3-fold: 1) We propose an end-to-end memory network[2] for aspect and opinion terms co-extraction without requiring any syntactic/dependency parsers or linguistic resources to generate additional information as input. Note that preliminary results have been shown in our previous work [19]. 2) We further introduce a finer-grained problem and extend the memory network with a multi-task mechanism to solve it. 3) We conduct extensive experiments on SemEval Challenge benchmark datasets to demonstrate state-of-the-art performance of our proposed approaches.

2. Related work

2.1. Fine-grained sentiment analysis

There have been a number of works proposed for aspect/opinion terms extraction. Hu & Liu [1] proposed to use association rule mining for extracting aspect terms and synonyms/antonyms from WordNet for identifying opinion terms. Qiu *et al.* [2] used a dependency parser to augment a seed collection of aspect and opinion terms through double-propagation, similar for [20,21]. The above methods are unsupervised, but depend on pre-defined rules and linguistic resources. For supervised methods, the task is treated as a sequence labeling problem. Li *et al.* [7] and Jin & Ho [6] implemented CRF and HMM with extensive human-designed features for aspect and opinion terms co-extraction, respectively. Liu *et al.* [22,23] applied a word alignment model in order to capture relations among opinion words, which requires large amount of training data to obtain desired relations. Recently, deep learning methods have been proposed for this task. Liu *et al.* [11] applied recurrent neural network on top of pre-trained word embeddings for aspect extraction. Yin *et al.* [8] proposed an unsupervised embedding method to encode dependency path into a recurrent neural network to learn high-level features for words, which are taken as input features for CRFs for aspect extraction. Wang *et al.* [3] proposed a joint

[2] The code is available at https://github.com/happywwy/Coupled-Multi-layer-Attentions.

model of recursive neural networks and CRFs for aspect and opinion terms co-extraction. The neural network is constructed from the dependency parse tree to capture dual-propagation among aspect and opinion terms. Most existing deep models require a syntactic/denpendency parser and auxiliary linguistic features to boost their extraction accuracy. Recent approaches also applied CNNs for aspect terms extraction [9,10], where multiple convolutional layers are stacked to extract features in some contexts contained in each sliding window. Xu *et al.* [10] applied both general-purpose and domain-specific word embeddings to incorporate more information. He *et al.* [24] proposed unsupervised aspect extraction using attention model, but the focus is more on aspect categorization. Li and Lam [25] proposed to use memory interactions between aspects and opinions. On the other hand, for the task of aspect categorization, most existing methods assume the aspect terms be extracted in advance, and aim to predict their corresponding categories [13–16]. To apply these methods to category-specific aspect/opinion terms extraction, one needs to first identify aspect/opinion terms as a preprocessing step. In such a pipeline solution, error can be propagated across steps. Although topic models or clustering based approaches [26–30,14,31] are able to group potential aspect terms into different clusters or topics (not explicit categories), they still fail to explicitly extract and classify a term into a predefined category.

2.2. Attentions and memory networks

Attentions [32] and memory networks [33] have recently been used for various machine learning tasks, including image generation [34], machine translation [12], sentence summarization [35], document sentiment classification [36], question answering [37], etc. The attention mechanism aims to select and attend to relevant parts of the input which could be thought of as a soft-alignment process. A memory network generally consists of multiple layers of attentions, which has shown superior performance in many NLP tasks [38,39]. In this paper, we aim to develop a memory network to replace the role of a syntactic/dependency parser to capture the relations among words in a sentence for information extraction.

2.3. Deep multi-task learning

Multi-task learning aims to improve generalization for each individual task by exploiting relatedness among different tasks [40]. One common assumption in multi-task learning is that parameters for different tasks lie in a low-dimensional subspace [41,42] which is achieved either by imposing low-rank constraints or matrix factorization. Through factorization, the model of each task becomes a linear combination of a small set of latent tasks. Following this idea, a multi-linear model was proposed in [43] to deal with multi-modal tasks with multiple indexes. This tensor factorization idea also promotes a deep multi-task learning model [36] where the parameters in different layers of a CNN for different tasks form a tensor that could be factorized across tasks. Moreover, many deep learning models have been introduced for multi-task learning [44, 45] with an aim to learn shared hidden features which are regularized from different tasks. Our proposed deep multi-task learning model is specially designed for sentiment analysis. We incorporate additional opinion labels to solve both aspect and opinion terms extraction at the same time by modeling their fine-grained interactions, compared with other general methods.

3. Problem statement and motivation

We denote a sentence by a sequence of tokens $\mathbf{s}_i = \{w_{i1}, w_{i2}, ..., w_{in_i}\}$ and represent it as a $D \times n_i$ matrix $\mathbf{X}_i = [\mathbf{x}_{i1}, ..., \mathbf{x}_{in_i}]$, where $\mathbf{x}_{ij} \in \mathbb{R}^D$ is a feature vector for the j-th token of the sentence. For fine-grained aspect and opinion terms extraction, the expected output is a sequence of token-level labels $\mathbf{y_i} = (y_{i1}, y_{i2}, ..., y_{in_i})$, where each $y_{ij} \in \{\text{BA, IA, BP, IP, O}\}$ that represents beginning of an aspect, inside of an aspect, beginning of an opinion, inside of an opinion or none of the above. A subsequence of labels started with "BA" and followed by "IA" indicates a multi-word aspect term, similar for opinion terms. For the finer-grained terms extraction, we consider the category information, and let $\mathcal{C} = \{1, 2, ..., C\}$ denote a predefined set of C categories, where $c \in \mathcal{C}$ is an entity/attribute type, e.g., "DRINK#QUALITY" is a category in the restaurant domain. A superscript c denotes the category-related variable. We denote $\mathbf{y}_i^c \in \mathbb{R}^{n_i}$, where $y_{ij}^c \in \{\text{BA}_c, \text{IA}_c, \text{BP}_c, \text{IP}_c, \text{O}_c\}$ is the label of the j-th token. Here, BA_c and IA_c refer to *beginning of aspect* and *inside of aspect*, respectively, of **category** c, similar for BP_c, IP_c and O_c. In the following, we use j to denote the index of a token in a sentence, c to denote the association with category c and for simplifying notations, we omit the sentence index i if the context is clear.

As discussed in the previous sections, to fully exploit the syntactic relations among different tokens in a sentence, most existing methods applied a computational parser to analyze the syntactic/dependency structure of each sentence in advance and use the relations between aspects and opinions to double propagate the information. One major limitation is that the generated relations are deterministic and fail to handle uncertainty underlying the data. It is even worse when grammar and syntactic errors commonly exist in user-generated texts, in which case the outputs of a dependency parser may not be precise, and thus degrades the performance. To avoid this, we develop a memory network with coupled attentions to automatically learn the relations between aspect terms and opinion terms without any linguistic knowledge. In the sequel, we name our proposed memory network with the coupled attentions for fine-grained sentiment analysis as MNCA.

To further explore category information for each aspect/opinion term, one straightforward solution is to apply the extraction model to identify general aspect/opinion terms first, and then post-classify them into different categories using an additional classifier. However, this pipeline approach may suffer from error propagation from the extraction phase to the

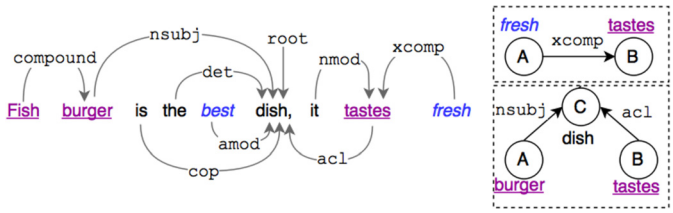

Fig. 1. A dependency example and illustration for the functionalities of MNCA.

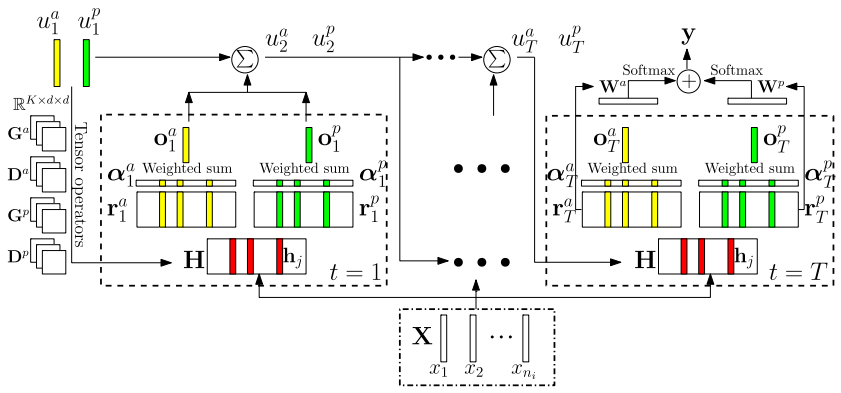

Fig. 2. Architecture of MNCA for aspect and opinion terms co-extraction.

classification phase. An alternative solution is to train an extraction model for each category c independently, and then combine the results of all the extraction models to generate the final prediction. However, in this way, for each fine-grained category, aspect and opinion terms become extremely sparse for training, which makes it difficult to learn a precise model for each category. To address the above issues, we propose to model the problem in a multi-task learning manner, where aspect/opinion terms extraction for each category is considered as an individual task, and an end-to-end deep learning architecture is developed to jointly learn the tasks by exploiting their commonalities and similarities. We name our proposed multi-task model as Multi-task Memory Networks (MTMN).

Note that MNCA is the basic component of MTMN. In the following, we first present MNCA for aspect and opinion terms extraction in Section 4. We then present the detailed architecture of MTMN for category-specific aspect and opinion terms co-extraction in Section 5.

4. Memory network with coupled attentions

In order to model the interactions between aspect terms and opinion terms automatically, we propose a novel memory network, MNCA, that consists of the following features:

- For each sentence, we construct a pair of attentions: an aspect attention for aspect terms extraction and an opinion attention for opinion terms extraction. Each of them aims to learn a general prototype vector, a token-level feature vector and a token-level attention score for each word in the sentence. The feature vector and attention score measure the extent of correlation between each input token and the prototype through a tensor operator, where a token with a higher score indicates a higher chance of being an aspect or opinion.
- To capture direct relations between aspect and opinion terms, e.g., the $A \xrightarrow{xcomp} B$ relation shown in Fig. 1, the aspect and opinion attentions are coupled in learning such that the learning of each attention is affected by the other. This helps to double-propagate information between them.
- To further capture indirect relations among aspect and opinion terms, e.g., the $A \xrightarrow{nsubj} C \xleftarrow{acl} B$ relation shown in Fig. 1, we construct a memory network with multiple layers to update the learned prototype vectors, feature vectors, and attention scores to better propagate label information for aspect and opinion terms co-extraction.

The overall architecture of MNCA is shown in Fig. 2, where each block shows the computation of a single layer with the shared input $\mathbf{X}$ and four 3-dimensional tensors $\{\mathbf{G}^a, \mathbf{D}^a, \mathbf{G}^p, \mathbf{D}^p\}$. Next, we illustrate each component in details.

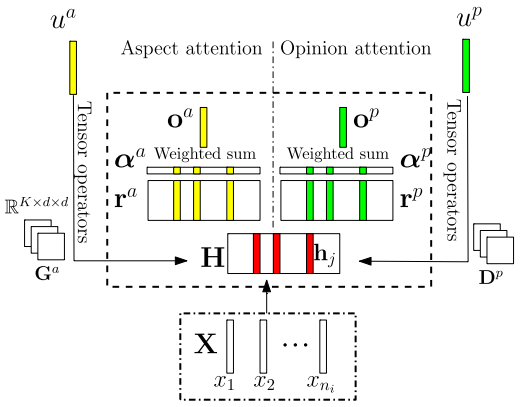

Fig. 3. Independent attentions with tensor operator.

4.1. Attention with tensor operator

A basic unit of MNCA is a pair of attentions: aspect attention and opinion attention. Different from traditional attentions which are used for generating a weighted sum of the input to represent the sentence-level information, we use attentions to identify the possibility of each token being an aspect or opinion term. As shown in Fig. 3, given a sentence with pre-trained word embeddings $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_{n_i}]$, we first apply Gated Recurrent Unit (GRU) [46] to obtain a memory matrix $\mathbf{H} = [\mathbf{h}_1, ..., \mathbf{h}_{n_i}]$, where $\mathbf{h}_j \in \mathbb{R}^d$ is a feature vector for j-th token considering its context.

In the aspect attention, we first generate a prototype vector $\mathbf{u}^a$ which can be viewed as a general feature representation for aspect terms. This aspect prototype aims to guide the model to attend to the most relevant tokens (most likely aspect words).[3] Given $\mathbf{u}^a$ and $\mathbf{H}$, the model scans the input sequence and computes an attention vector $\mathbf{r}^a_j$ and an attention score $\boldsymbol{\alpha}^a_j$ for the j-th token. To obtain $\mathbf{r}^a_j$, we first compute a composition vector $\boldsymbol{\beta}^a_j \in \mathbb{R}^K$ that encodes the extent of correlations between $\mathbf{h}_j$ and the prototype vector $\mathbf{u}^a$ through a tensor operator:

$$\boldsymbol{\beta}^a_j = \tanh(\mathbf{h}_j^\top \mathbf{G}^a \mathbf{u}^a), \tag{1}$$

where $\mathbf{G}^a \in \mathbb{R}^{K\times d\times d}$ is a 3-dimensional tensor. Motivated by [47], a tensor operator could be viewed as multiple bilinear matrices that model more complicated compositions between 2 units. Here, $\mathbf{G}^a$ could be decomposed into K slices, where each slice $\mathbf{G}^a_k \in \mathbb{R}^{d\times d}$ is a bilinear term that interacts with 2 vectors and captures one type of composition, e.g., a specific syntactic relation. Hence $\mathbf{h}_j^\top \mathbf{G}^a \mathbf{u}^a \in \mathbb{R}^K$ inherits K different kinds of compositions between $\mathbf{h}_j$ and $\mathbf{u}^a$ that indicates complicated correlations between each input token and the aspect prototype. Then $\mathbf{r}^a_j$ is obtained from $\boldsymbol{\beta}^a_i$ via a GRU network:

$$\mathbf{r}^a_j = (1 - z^a_j) \odot \mathbf{r}^a_{j-1} + z^a_j \odot \widetilde{\mathbf{r}}^a_j, \tag{2}$$

where

$$\begin{aligned} g^a_j &= \sigma(\mathbf{W}^a_g \mathbf{r}^a_{j-1} + \mathbf{U}^a_g \boldsymbol{\beta}^a_j), \\ z^a_j &= \sigma(\mathbf{W}^a_z \mathbf{r}^a_{j-1} + \mathbf{U}^a_z \boldsymbol{\beta}^a_j), \\ \widetilde{\mathbf{r}}^a_j &= \tanh(\mathbf{W}^a_r (g^a_j \odot \mathbf{r}^a_{j-1}) + \mathbf{U}^a_r \boldsymbol{\beta}^a_j). \end{aligned}$$

This step helps to encode sequential context information into the attention vector $\mathbf{r}^a_j \in \mathbb{R}^K$. Indeed, many aspect terms consist of multiple tokens, and exploiting context information is helpful for making predictions. For simplicity, we use $\mathbf{r}^a_j = \mathrm{GRU}(\boldsymbol{\beta}^a_j, \Theta^a)$ where $\Theta^a = \{\mathbf{W}^a_g, \mathbf{U}^a_g, \mathbf{W}^a_z, \mathbf{U}^a_z, \mathbf{W}^a_r, \mathbf{U}^a_r\}$ to denote (2).

An attention score $\boldsymbol{\alpha}^a_j$ for token w_j is then computed as

$$\boldsymbol{\alpha}^a_j = \frac{\exp(\mathbf{e}^a_j)}{\sum_k \exp(\mathbf{e}^a_k)}, \tag{3}$$

where $\boldsymbol{\alpha}^a_j$ denotes the j-th element of the vector $\boldsymbol{\alpha}^a$, similar for $\mathbf{e}_j$. Here $\mathbf{e}^a_j = \langle \mathbf{v}^a, \mathbf{r}^a_j \rangle$. Since $\mathbf{r}^a_j$ is a correlation feature vector, $\mathbf{v}^a \in \mathbb{R}^K$ can be deemed as a weight vector that weighs each feature accordingly. Therefore, $\boldsymbol{\alpha}^a_j$ becomes the normalized score, where a higher score indicates a higher correlation with the prototype, and a higher chance of being attended.

[3] We randomly initialize $\mathbf{u}^a$ from a uniform distribution: $\mathbf{u}^a \sim U[-0.2, 0.2] \in \mathbb{R}^d$, which is then trained and updated iteratively.

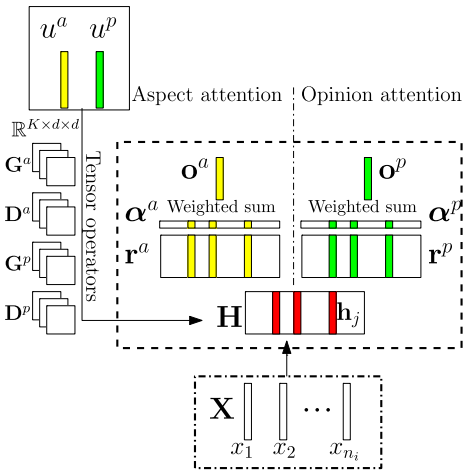

Fig. 4. Coupled attentions with tensor operator.

The procedure for opinion attention is similar. In the subsequent sections, we use a superscript p to denote the opinion attention.

4.2. Coupled attentions for dual propagation

As aforementioned, a crucial research issue for co-extraction of aspect and opinion terms is how to fully exploit the relations between aspect terms and opinion terms such that the information can be propagated to each other to assist final predictions. However, learning of the aspect attention and the opinion attentions independently as described in the previous section fails to utilize their relations. Here, we propose to couple the learning of the two attentions such that information of each attention can be dually propagated to the other. As shown in Fig. 4, instead of a single attention, the prototype to be fed into each attention module becomes a pair of vectors $\{\mathbf{u}^a, \mathbf{u}^p\}$, and the tensor operator in (1) becomes a set of tensors $\{\mathbf{G}^a, \mathbf{D}^a, \mathbf{G}^p, \mathbf{D}^p\}$. The composition vectors $\boldsymbol{\beta}_j^a$ and $\boldsymbol{\beta}_j^p$ are computed as follows,

$$\boldsymbol{\beta}_j^a = \tanh([\mathbf{h}_j^\top \mathbf{G}^a \mathbf{u}^a : \mathbf{h}_j^\top \mathbf{D}^a \mathbf{u}^p]), \text{ and } \boldsymbol{\beta}_j^p = \tanh([\mathbf{h}_j^\top \mathbf{G}^p \mathbf{u}^a : \mathbf{h}_j^\top \mathbf{D}^p \mathbf{u}^p]), \tag{4}$$

where [:] denotes concatenation of two vectors. Intuitively, $\mathbf{G}^a$ or $\mathbf{D}^p$ is to capture the K syntactic relations within aspect terms or opinion terms themselves, while $\mathbf{G}^p$ and $\mathbf{D}^a$ are to capture syntactic relations between aspect terms and opinion terms for dual propagation. Note that $\boldsymbol{\beta}_j^a$ and $\boldsymbol{\beta}_j^p$, both of which are of $2K$ dimensions, go through the same procedure as (2) and (3) to produce $\mathbf{r}_j^a, \mathbf{r}_j^p \in \mathbb{R}^{2K}$ as the hidden representations for $\mathbf{h}_j$ w.r.t. the aspect attention and the opinion attention, respectively.

4.3. The overall memory network

A single layer with the coupled attentions is able to capture the direct relations between aspect terms and opinion terms, but fails to exploit the indirect relations among them, such as the $A \xrightarrow{nsubj} C \xleftarrow{acl} B$ relation shown in Fig. 1. To address this issue, we integrate the coupled attentions into a memory network such that the information learned from the attentions could be updated and used for better extraction. The memory network consists of multiple layers of coupled attentions. For each layer $t+1$ as shown in Fig. 2, the prototype vectors $\mathbf{u}_{t+1}^a$ and $\mathbf{u}_{t+1}^p$ are updated based on the prototype vectors in the previous layer $\mathbf{u}_t^a$ and $\mathbf{u}_t^p$ to incorporate more feasible representations for aspect terms or opinion terms through

$$\mathbf{u}_{t+1}^a = \tanh(\mathbf{Q}^a \mathbf{u}_t^a) + \mathbf{o}_t^a, \text{ and } \mathbf{u}_{t+1}^p = \tanh(\mathbf{Q}^p \mathbf{u}_t^p) + \mathbf{o}_t^p, \tag{5}$$

where $\mathbf{Q}^a, \mathbf{Q}^p \in \mathbb{R}^{d\times d}$ are recurrent transformation matrices to be learned, and $\mathbf{o}_t^a$, $\mathbf{o}_t^p$ are accumulated vectors computed via

$$\mathbf{o}_t^a = \sum_j \boldsymbol{\alpha}_t^a \mathbf{h}_j, \text{ and } \mathbf{o}_t^p = \sum_j \boldsymbol{\alpha}_t^p \mathbf{h}_j. \tag{6}$$

Intuitively, $\mathbf{o}_t^a$ and $\mathbf{o}_t^p$ are dominated by the input feature vectors $\{\mathbf{h}_j\}$'s with higher attention scores. Therefore, $\mathbf{o}_t^a$ and $\mathbf{o}_t^p$ tend to approach to the attended feature vectors of aspect or opinion words. In this way, $\mathbf{u}_{t+1}^a$ (or $\mathbf{u}_{t+1}^p$) incorporates the most probable aspect (or opinion) terms, which in turn will be used to interact with $\{\mathbf{h}_j\}$'s at layer $t+1$ to learn more

precise token representations and attention scores, and sentence representations for selecting other non-obvious target tokens. At the last layer T, after generating all the $\{\mathbf{r}^a_{T,j}\}$'s and $\{\mathbf{r}^p_{T,j}\}$'s, we compute two 3-dimensional label vectors $\mathbf{y}^a_j$ and $\mathbf{y}^p_j$ as follows,

$$\mathbf{y}^a_j = \text{softmax}(\mathbf{W}^a \mathbf{r}^a_{T,j}), \text{ and } \mathbf{y}^p_j = \text{softmax}(\mathbf{W}^p \mathbf{r}^p_{T,j}), \tag{7}$$

where $\mathbf{W}^a, \mathbf{W}^p \in \mathbb{R}^{3\times 2K}$ are transformation matrices for the predictions on aspects and opinions, respectively, and $\mathbf{y}^a_j$ denotes the probabilities of $\mathbf{h}_j$ being BA, IA and O, while $\mathbf{y}^p_j$ denotes the probabilities of $\mathbf{h}_j$ being BP, IP and O. For training, we define the loss function as follows,

$$\mathcal{L} = \sum_{j=1}^{n_i} \sum_{m\in\{a,p\}} \ell\left(\hat{\mathbf{y}}^m_j, \mathbf{y}^m_j\right), \tag{8}$$

where $\ell(\cdot)$ is the cross-entropy loss, and $\hat{\mathbf{y}}^m_j \in \mathbb{R}^3$ is a one-hot vector representing the ground-truth label for the j-th token w.r.t. aspect or opinion. For testing or making predictions, the final label for each token j is produced by comparing the values in $\mathbf{y}^a_j$ and $\mathbf{y}^p_i$. If both of them are O, then the label is O. If only one of them is O, we pick the other one as the label. Otherwise, the label is the one with the largest value.

4.4. Discussion

From the formulation, the proposed memory network is able to attend to relevant words that are highly interactive given the prototypes. This is achieved by tensor interactions, e.g., $\mathbf{h}_j \top \mathbf{G}^a \mathbf{u}^a_t$ between jth word and the aspect prototype. By updating the prototype vector $\mathbf{u}^a_{t+1}$ with extracted information from the tth layer, we obtain

$$\mathbf{u}^a_{t+1} = \tanh(\mathbf{Q}^a \mathbf{u}^a_t) + \sum_j \alpha^a_t \mathbf{h}_j, \tag{9}$$

where highly interactive $\mathbf{h}_j$ contributes more to the prototype updates. Since the final feature representation $\mathbf{r}^a_{T,j}$ for each word is generated from the above tensor interactions, it transforms the normal feature space $\mathbf{h}_j$ to interaction space $\mathbf{r}_{T,j}$, compared to simple RNNs that only computes $\mathbf{h}_j$.

Compared with recursive neural network [3], where the final feature representation for each word is generated from the composition with the child nodes in a dependency tree, our memory network avoids the construction of dependency trees and is not prune to parsing errors. For example, if we denote the final feature for jth word as $\mathbf{h}'_j$ for the recursive neural network, according to [3], $\mathbf{h}'_j = f(\mathbf{W}_v \cdot \mathbf{x}_j + \mathbf{b} + \sum_{k\in\mathcal{K}_j} \mathbf{W}_{r_{jk}} \cdot \mathbf{h}_k)$. Here $\mathcal{K}_j$ denotes the set of children for node j and $\mathbf{W}_{r_{jk}}$ represents the transformation matrix for each dependency relation r_{jk} between jth node and its child. In this case, an incorrect relation parsing will lead to different $\mathbf{W}_{r_{jk}}$ or $\mathbf{h}_k$, resulting in possibly erroneous hidden representations. Our memory network, on the other hand, does not require pre-defined composition nodes. The attention mechanism in the previous layer will automatically select relevant words to make interactions, according to (9).

5. Multi-task memory network

In this section, we further extend MNCA to deal with category-specific aspect and opinion terms extraction by integrating the multi-task learning strategy. The proposed multi-task memory network consists of four main components: 1) **Category-specific MNCA** to co-extract aspect and opinion terms for each category, 2) **Shared Tensor Decomposition** to model the commonalities of syntactic relations among different categories by sharing the tensor parameters, 3) **Context-aware Multi-task Feature Learning** to jointly learn features among categories through constructing context-aware task similarity matrices, and 4) **Auxiliary Task** to create an auxiliary task to predict overall sentence-level category labels to assist token-level prediction tasks. In the following section, we present the four components of MTMN in detail.

5.1. Category-specific MNCA

We use MNCA as the base classifier in MTMN for aspect and opinion terms co-extraction for each category c. As described in Section 4, we apply the procedure of MNCA for each category c by denoting each variable with the subscript c:

$$\boldsymbol{\beta}^a_{c[j]} = \tanh([\mathbf{h}^\top_j \mathbf{G}^a_c \mathbf{u}^a_c : \mathbf{h}^\top_j \mathbf{D}^a_c \mathbf{u}^p_c]), \text{ and } \boldsymbol{\beta}^p_{c[j]} = \tanh([\mathbf{h}^\top_j \mathbf{G}^p_c \mathbf{u}^a_c : \mathbf{h}^\top_j \mathbf{D}^p_c \mathbf{u}^p_c]), \tag{10}$$

where $\mathbf{G}^a_c, \mathbf{G}^p_c, \mathbf{D}^a_c, \mathbf{D}^p_c \in \mathbb{R}^{K\times d\times d}$. We then obtain $\mathbf{r}^a_{c[j]}$ and $\mathbf{r}^p_{c[j]}$ as the hidden representations for $\mathbf{h}_j$ w.r.t. aspect and opinion of category c, respectively. Normalized attention scores for $\mathbf{h}_j$ for each category c are computed as

$$\boldsymbol{\alpha}^a_{c[j]} = \frac{\exp(\mathbf{e}^a_{c[j]})}{\sum_k \exp(\mathbf{e}^a_{c[k]})}, \text{ and } \boldsymbol{\alpha}^p_{c[j]} = \frac{\exp(\mathbf{e}^p_{c[j]})}{\sum_k \exp(\mathbf{e}^p_{c[k]})}. \tag{11}$$

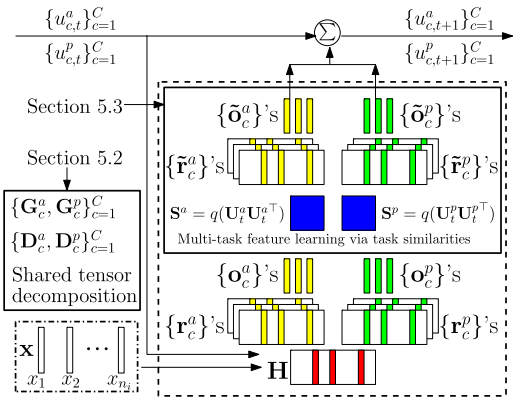

Fig. 5. The architecture of each non-output layer used in MTMN.

The overall representations of the sentence for category c in terms of aspects and opinions, denoted by $\mathbf{o}_c^a$ and $\mathbf{o}_c^p$, respectively, are then computed using (6), which will be further used to produce the prototype vectors $\mathbf{u}_{c,t+1}^a$, $\mathbf{u}_{c,t+1}^p$ in the next layer using (5). At the last layer T, after generating all $\{\mathbf{r}_{c[j]}^a\}$'s and $\{\mathbf{r}_{c[j]}^p\}$'s, for each category c, we compute two 3-dimensional label vectors $\mathbf{y}_{c[j]}^a$ and $\mathbf{y}_{c[j]}^p$ as follows,[4]

$$\mathbf{y}_{c[j]}^a = \text{softmax}(\mathbf{W}^a \mathbf{r}_{c[j]}^a), \text{ and } \mathbf{y}_{c[j]}^p = \text{softmax}(\mathbf{W}^p \mathbf{r}_{c[j]}^p). \tag{12}$$

For training, we define the loss function as follows,

$$\mathcal{L}_{\text{tok}} = \sum_c \sum_{j=1}^{n_i} \sum_{m\in\{a,p\}} \ell\left(\hat{\mathbf{y}}_{c[j]}^m, \mathbf{y}_{c[j]}^m\right), \tag{13}$$

where $\ell(\cdot)$ is the cross-entropy loss. For testing, we generate a label for each token j as follows. We first produce a label $\mathbf{y}_{c[j]}$ for category c on the j-th token by comparing the largest value in $\mathbf{y}_{c[j]}^a$ and $\mathbf{y}_{c[i]}^p$ using the same method as MNCA. We then generate the final label on the j-th token by integrating $\mathbf{y}_{c[j]}$'s across all the categories, which is similar to multi-label classification, because some word might belong to multiple categories.

If we directly apply the above formulation to extract aspect terms and opinion terms for each category independently, the result is not satisfactory as will be shown in the experiments. This is because in our proposed finer-grained extraction problem, training data for each specific category becomes too sparse to learn precise predictive models if extractions for different categories are considered independently. In the following sections, we introduce how to incorporate multi-task learning techniques and MNCA into a unified memory network to make aspect and opinion terms co-extraction effective.

5.2. Shared tensor decomposition

As described in the previous section, for each category c, there are four tensor operators $\mathbf{G}_c^a$, $\mathbf{G}_c^p$, $\mathbf{D}_c^a$, and $\mathbf{D}_c^p$ to model the complex token interactions, each of which is in $\mathbb{R}^{K\times d\times d}$. When the number of categories increases, the parameter size may be very large. As a result, available training data may be too sparse to estimate the parameters precisely. Therefore, instead of learning the tensors for each category independently, we assume that interactive relations among tokens are similar across categories. Therefore, we propose to learn a low-rank shared information among the tensors through collective tensor factorization as shown in the architecture of each non-output layer in Fig. 5. Specifically, let $\mathbf{G}^a \in \mathbb{R}^{C\times K\times d\times d}$ be the concatenation of all the $\{\mathbf{G}_c^a\}$'s, and denote by $\mathbf{G}_k^a = \mathbf{G}^a{}_{[\cdot,k,\cdot,\cdot]} \in \mathbb{R}^{C\times d\times d}$ the collection of k-th bi-linear interaction matrices across C tasks for the aspect attention. The same also applies to $\mathbf{G}^p$ and $\mathbf{G}_k^p$ for the opinion attention. Factorization is performed on each $\mathbf{G}_k^a$ and $\mathbf{G}_k^p$, respectively, via

$$\mathbf{G}_{k[c,\cdot,\cdot]}^a = \mathbf{Z}_{k[c,\cdot]}^a \mathcal{G}_k^a, \text{ and } \mathbf{G}_{k[c,\cdot,\cdot]}^p = \mathbf{Z}_{k[c,\cdot]}^p \mathcal{G}_k^p, \tag{14}$$

where $\mathcal{G}_k^a, \mathcal{G}_k^p \in \mathbb{R}^{m\times d\times d}$ are shared factors among all the tasks with $m < C$, while $\mathbf{Z}_k^a, \mathbf{Z}_k^p \in \mathbb{R}^{C\times m}$ with each row $\mathbf{Z}_{k[c,\cdot]}^a$ and $\mathbf{Z}_{k[c,\cdot]}^p$ being specific factors for category c. The shared factors can be considered as m latent basis interactions, where the original k-th bi-linear relation matrix $\mathbf{G}_{k[c,\cdot,\cdot]}^a$ (or $\mathbf{G}_{k[c,\cdot,\cdot]}^p$) for c is the linear combination of the latent basis interactions. The same approach also applies to the tensors $\{\mathbf{D}_c^a\}$'s and $\{\mathbf{D}_c^p\}$'s. In this way, we reduce the parameter dimensions by enforcing sharing within a small number of latent interactions.

[4] We omit the subscript T for the ease of notation.

5.3. Context-aware multi-task feature learning

Besides jointly decomposing tensors of syntactic relations across categories, in this section, we further exploit similarities between categories or tasks[5] to learn more powerful features for each token and each sentence. Consider the following motivating example, "FOOD#PRICE" is more similar to "DRINK#PRICE" than "SERVICE#GENERAL" because the first two categories may share some common aspect/opinion terms, such as *expensive*. Therefore, by representing each task in a form of distributed vector, we can directly compute their similarities to facilitate knowledge sharing. Based on this motivation, we aim to update features $\tilde{\mathbf{r}}_c^a$ (or $\tilde{\mathbf{r}}_c^p$) from $\mathbf{r}_c^a$ (or $\mathbf{r}_c^p$) by integrating task relatedness. Specifically, at a layer t, suppose that $\mathbf{u}_{c,t}^a$, and $\mathbf{u}_{c,t}^p$ are the updated prototype vectors passed from the previous layer. These two prototype vectors can be used to represent task c, because $\mathbf{u}_{c,t}^a$ and $\mathbf{u}_{c,t}^p$ are learned interactively with the category-specific sentence representations $\mathbf{o}_c^a$'s and $\mathbf{o}_c^p$'s of the previous $t-1$ layers, respectively. Let $\mathbf{U}^a, \mathbf{U}^p \in \mathbb{R}^{d\times C}$ denote the matrices consisting of $\mathbf{u}_c^a$ and $\mathbf{u}_c^p$ as a column vector, respectively, then the task similarity matrices, $\mathbf{S}^a$ and $\mathbf{S}^p$, in terms of aspects and opinions can be computed as follows,

$$\mathbf{S}^a = q(\mathbf{U}^{a\top}\mathbf{U}^a), \text{ and } \mathbf{S}^p = q(\mathbf{U}^{p\top}\mathbf{U}^p), \tag{15}$$

where $q(\cdot)$ is the softmax function carried in a column-wise manner so that the similarity scores between a task and all the tasks sum up to 1. The similarity matrices $\mathbf{S}^a$ and $\mathbf{S}^p$ are then used to refine feature representation of each token for each task by incorporating feature representations from related tasks:

$$\tilde{\mathbf{r}}_{c,[j]}^a = \sum_{c'=1}^{C} \mathbf{S}_{cc'}^a \mathbf{r}_{c',[j]}^a, \text{ and } \tilde{\mathbf{r}}_{c[j]}^p = \sum_{c'=1}^{C} \mathbf{S}_{cc'}^p \mathbf{r}_{c'[j]}^p, \tag{16}$$

where $\mathbf{r}_{c',[j]}^a$ and $\mathbf{r}_{c',[j]}^p$ denote the j-th column of the matrix $\mathbf{r}_{c'}^a$ and $\mathbf{r}_{c'}^p$, respectively. Similarly, we refine feature representation of each sentence for each task as follows,

$$\tilde{\mathbf{o}}_c^a = \sum_{c'=1}^{C} \mathbf{S}_{cc'}^a \mathbf{o}_{c'}^a, \text{ and } \tilde{\mathbf{o}}_c^p = \sum_{c'=1}^{C} \mathbf{S}_{cc'}^p \mathbf{o}_{c'}^p. \tag{17}$$

Regarding the update of the prototype vectors, we replace $\mathbf{o}_c^a$ and $\mathbf{o}_c^p$ by $\tilde{\mathbf{o}}_c^a$ and $\tilde{\mathbf{o}}_c^p$, respectively. This context-aware multi-task architecture is also shown in Fig. 5. Note that the feature sharing among different tasks is context-aware because $\mathbf{U}^a$ and $\mathbf{U}^p$ are category representations depending on each sentence. This means that different sentences might indicate different task similarities. For example, when *cheap* is presented, it might increase the similarity between "FOOD#PRICES" and "RESTAURANT#PRICES". As a result, $\tilde{\mathbf{r}}_{c[j]}^a$ for task c could incorporate more information from task c' if c' has higher similarity score indicated by $\mathbf{S}_{cc'}^a$.

5.4. Auxiliary task

As MTMN could produce sentence-level feature representations, to better address the data sparsity issue, we propose to use additional global information on categories in the sentence level. Consider the following motivating example, if we know the sentence "*The soup is served with nice portion, the service is prompt*" belongs to the categories "DRINKS#STYLE_OPTIONS" and "SERVICE#GENERAL", we can infer that some words in the sentence should belong to one of these two categories. To make use of this information, we construct an auxiliary task to predict the categories of a sentence. From training data, sentence-level labels can be automatically obtained by integrating tokens' labels. Therefore, besides the token loss in (8) for our target token-level prediction task, we also define the sentence loss for the auxiliary task. Note that the learning of the target task (terms extraction) and auxiliary task (multi-label classification on sentences) are not independent. On one hand, the global sentence information helps the attentions to select category-relevant tokens. On the other hand, if the attentions are able to attend to target terms, the output context representation will filter out irrelevant noise, which helps making a prediction on the overall sentence.

To be specific, as shown in Fig. 6, for category c, we define $\tilde{\mathbf{o}}_c = [\tilde{\mathbf{o}}_c^a : \tilde{\mathbf{o}}_c^p] \in \mathbb{R}^{2d}$ the final representation for the sentence, and generate the output using the softmax function,

$$\boldsymbol{l}_c = \text{softmax}(\mathbf{W}_c\tilde{\mathbf{o}}_c), \tag{18}$$

where $\mathbf{W}_c \in \mathbb{R}^{2\times 2d}$, and $\boldsymbol{l}_c \in \mathbb{R}^2$ indicates the probability of the sentence belonging to category c or not. The loss of the auxiliary task is defined as $\mathcal{L}_{\text{sen}} = \sum_c \ell(\hat{\boldsymbol{l}}_c, \boldsymbol{l}_c)$, where $\ell(\cdot)$ is the cross-entropy loss, and $\hat{\boldsymbol{l}}_c \in \{0,1\}^2$ is the ground truth using one-hot encoding indicating whether category c is presented for the sentence. By incorporating the loss of the auxiliary task, the final objective for MTMN is written as $\mathcal{L} = \mathcal{L}_{\text{sen}} + \mathcal{L}_{\text{tok}}$, where $\mathcal{L}_{\text{tok}}$ is defined in (8).

[5] We interchangeably use the terms "task" and "category" in the rest of this paper.

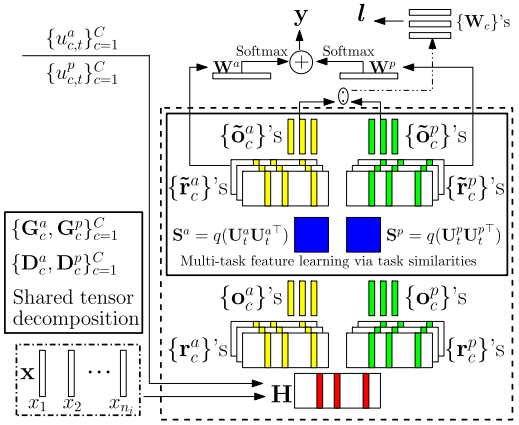

Fig. 6. The architecture of the output layer used in MTMN.

Table 1
Dataset description.

Dataset	Description	Training		Test		Total	
		text	tuple	text	tuple	text	tuple
S1	SemEval-15 Restaurant	1,315	1,654	685	845	2,000	2,499
S2	SemEval-16 Restaurant	2,000	2,507	676	859	2,676	3,366
S3	SemEval-14 Laptop	3,045	1,974	800	545	3,845	2,519
S4	SemEval-14 Restaurant	3,041	–	800	–	3,841	–

6. Experiments

6.1. Datasets & experimental setup

The experiments are conducted on four benchmark datasets from subtask 1 in SemEval Challenge 2015 task 12 [48], SemEval Challenge 2016 task 5 [49], laptop and restaurant dataset in SemEval Challenge 2014 task 4 [50], which are denoted by S1, S2, S3 and S4 respectively. Note that S1 and S2 are both reviews in restaurant domain. We use term-level aspect-opinion annotations provided by [19] for S1, S3 and S4, and manually annotate opinion terms for S2. To facilitate our experiment, we additionally annotate category labels on target terms for S3, while the aspect term categories for S1 and S2 are provided by SemEval. The statistics of each dataset is shown in Table 1, where *text* and *tuple* represent the number of sentences and the number of tuples consisting of an aspect term and its corresponding category label, respectively. Each sentence may contain multiple aspect terms with more than one categories. The aspect categories are shown in Table 2.[6] For S1 and S2, an aspect category is defined as the combination of an entity and an attribute, e.g., "FOOD#PRICES". There are in total 12 categories. For S3, an aspect category is an entity.

Follow [19], we first obtain word embeddings by applying *word2vec*[7] on Yelp Challenge dataset[8] consisting of 2.2M restaurant reviews with 54K vocabulary size and electronic domain in Amazon reviews [51] containing 1M reviews with 590K vocabulary size for restaurant and laptop datasets, respectively. We set the dimension of word embeddings to be 150 and the dimension after GRU transformation to be 50. We use two layers of memory network for experiments. For each layer, the number of bi-linear interactions for the 3-dimensional tensors is 20 ($K = 20$), and tensor factorization for MTMN operates with $m = 5$ for S1 and S2, and $m = 8$ for S3. We apply partial dropout at 0.5 to chosen parameters (non-recurrent parameters of GRU) to avoid overfitting. For MNCA, we fix the learning rate to be 0.07 for S1, S2, S4, and 0.1 for S3. For MTMN, the training is carried with *rmsprop* with the initial value at 0.001 and decayed with rate 0.9. The trade-off parameter λ is set to be 1.0. All the hyper-parameters are chosen according to cross-validation.

[6] We filter out some categories with very few target terms and remove the corresponding sentences.
[7] https://radimrehurek.com/gensim/models/word2vec.html.
[8] http://www.yelp.com/dataset_challenge.

Table 2
Aspect Categories for two domains.

Restaurant		**Laptop**
Entity Labels	Attribute Labels	Entity Labels
1. RESTAURANT 2. FOOD 3. DRINKS 4. AMBIENCE 5. SERVICE 6. LOCATION	A. GENERAL B. PRICES C. QUALITY D. STYLE_OPTIONS E. MISCELLANEOUS	1. LAPTOP 2. DISPLAY 3. KEYBOARD 4. MOUSE 5. BATTERY 6. GRAPHICS 7. HARD_DISC 8. MULTIMEDIA_DEVICES 9. SOFTWARE 10. OS 11. SUPPORT 12. COMPANY

Table 3
Comparison results in F_1 scores. AS (OS) refers to aspect (opinion) terms extraction.

Model	S1		S2		S3		S4	
	AS	OP	AS	OP	AS	OP	AS	OP
EliXa	70.04	–	–	–	–	–	–	–
NLANG	67.11	–	72.34	–	–	–	–	–
DLIREC	–	–	–	–	73.78	–	84.01	–
IHS_RD	63.12	–	–	–	74.55	–	79.62	–
LSTM	64.30	66.43	68.43	72.04	72.73	74.98	81.15	80.22
WDEmb	69.12	–	–	–	74.68	–	84.31	–
WDEmb*	69.73	–	–	–	75.16	–	84.97	–
RNCRF(r)	64.16	64.10	67.49	71.08	75.57	74.40	80.47	79.51
RNCRF	67.06	66.90	69.09	75.79	76.83	76.76	84.05	80.93
RNCRF*	67.74	67.62	69.74	76.15	**78.42**	79.44	84.93	**84.11**
MNCA	**70.73**	**73.68**	**75.21**	**77.90**	77.80	**80.17**	**85.29**	83.18

6.2. Experimental results

6.2.1. Aspect and opinion terms extraction

In this section, we first present the experimental results of the proposed MNCA model for aspect terms and opinion terms extraction without considering categorization, and show the state-of-the-art performances compared with various baseline models listed in the following:

EliXa, NLANG, DLIREC, IHS_RD: the top performing systems for S1, S2, S3, S4 in corresponding SemEval Challenges.

LSTM: an LSTM network built on top of word embeddings proposed by [11]. The settings are the same as [3].

WDEmb: the model proposed by [8] using word and dependency path embeddings combined with linear context embedding features, dependency context embedding features as CRF input.[9]

RNCRF: the joint model with CRF and recursive neural network proposed by [3], which has been shown to outperform CRFs with hand-crafted features.

RNCRF(r): Change the pre-generated parsed trees by injecting some random noise on dependency relations (each relation has 25% probability of being replaced by a random relation).

WDEmb*, RNCRF*: the corresponding models with additional human-engineered linguistic features.

The comparison results in terms of F_1 scores are shown in Table 3. We report results for both aspect terms extraction (AS) and opinion terms extraction (OP) for all the four datasets. To make fair comparisons, we use the same corpus as in LSTM, RNCRF, RNCRF* for training word embeddings, and same training set with both aspect and opinion labels. Among deep-learning-based models, the models that combine neural network with CRF (i.e., WDEmb and RNCRF) perform better than LSTM because of the incorporation of dependency structure. However, the dependency information is not perfect. To support the claim that dependency-based models are prune to parsing errors, we simulate a poor parsing result by replacing around 25% dependency relations used for RNCRF with a random relation and denote the setting by RNCRF(r). As shown in Table 3, the results of RNCRF with noisy dependency trees are much worse. It is clear that MNCA achieves the state-of-the-art results for most of the time without any pre-extracted linguistic/syntactic information. Specifically, MNCA outperforms WDEmb by 1.61%, 3.12% and 0.98% respectively for S1, S3 and S4, and RNCRF by 3.67%, 6.12%, 0.97% and 1.24%, on S1, S2, S3 and S4, respectively, for aspect extraction. Even compared with the deep models with additional hand-crafted features, i.e., WDEmb* and RNCRF*, MNCA still gets improvements for most of the time. Moreover, the improvements over RNCRF and RNCRF* are all significant[10] ($p < 0.01$), except for the aspects extraction on S1 and S2 over RNCRF*. Note that besides linguistic features, WDEmb* and RNCRF* also require dependency parsers to perform the task. Therefore, MNCA is more effective and simpler to implement.

[9] We report the original result from [8] as the source code is not available.

[10] We divide the training data into 5-fold. Each time we leave one fold out and use the rest for training the model to be evaluated on the test dataset. We list 5 different results for each model and use t-test with the null hypothesis being "The baseline model and the proposed model have the same performance". With $p < 0.01$, we reject the null hypothesis.

Table 4
Comparisons under varying layers and different settings.

		S1		S2		S3		S4	
		AS	OP	AS	OP	AS	OP	AS	OP
Layer	1	69.27	69.56	75.10	**78.94**	77.28	78.12	84.90	81.85
	2	**70.73**	**73.68**	**75.21**	77.90	**77.80**	**80.17**	**85.29**	**83.18**
	3	69.78	71.95	73.82	76.80	77.24	79.29	84.41	82.38
Setting	ASL	69.53	–	73.87	–	76.45	–	84.38	–
	ASL+OPL	69.49	72.73	74.09	77.03	77.05	79.66	84.14	82.10
	MNCA	**70.73**	**73.68**	**75.21**	**77.90**	**77.80**	**80.17**	**85.29**	**83.18**

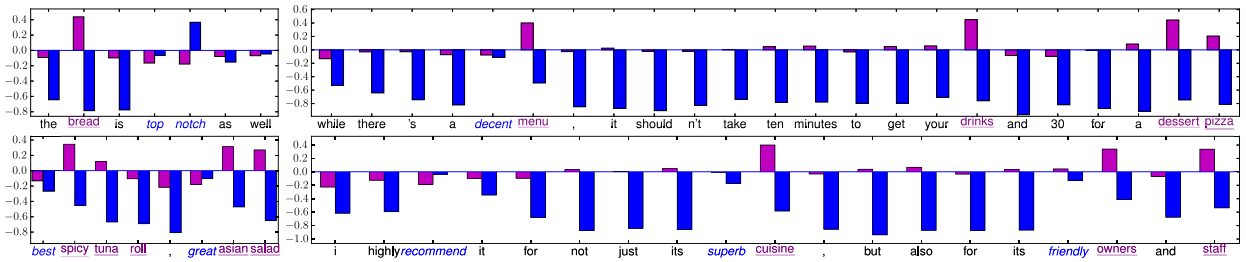

Fig. 7. Visualization of attention weights for different tokens within a sequence. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

To show the effect of the number of layers, we present experimental results varying the number of layers in Table 4. The best results are obtained with 2 layers. With only one layer, the results for aspect extraction are 1.46%, 0.11%, 0.52% and 0.39% inferior than the best scores on S1, S2, S3 and S4, respectively, but they are still comparable with other baselines shown in Table 3. Similar observations can be found for the results with 3 layers. This shows that MNCA with 2 layers is enough to exploit most of the relations among input tokens.

We also conducted experiments to explicitly show the advantage of coupling the learning of aspect and opinion attentions. The second part in Table 4 specifies different settings of the model. ASL refers to the multi-layer network with only aspect attention and is trained with aspect labels only. We can see that even without opinion labels, the network still proves comparable and even superior than deep models without linguistic features for aspect terms extraction shown in Table 3. This shows that multi-layer attentions with tensors is advantageous for exploiting interactions. ASL+OPL in Table 4 trains the aspect attention and opinion attention independently using (1) where each attention predicts one of the three labels. The results of ASL+OPL in terms of aspect extraction are similar to ASL, which shows that the additional opinion labels have little effect on aspect extraction if they are not interactively trained. By coupling the aspect and opinion attentions, MNCA achieves the best performance. To provide more complete analysis, we conduct a cost-benefit analysis for the labeling effort. Specifically, we create two different settings: 1) use 2,000 training sentences from S4 with only aspect labels, 2) select 1,200 training sentences from S4 with both aspect and opinion labels. We try to make the labeling effort comparable in these two settings and use test data from S4 for evaluation. The results for aspect extractions are 73.87% for setting 1 and 72.56% for setting 2. The performances do not vary greatly, but we benefit from providing the solutions for 2 tasks (both aspect and opinion terms extraction) at the same time and utilize their interactions to help predictions.

As a core component, an attention computes a score for each token to indicate its correlation with the corresponding prototype. We visualize the actual attention scores for the tokens of 4 sentences in Fig. 7. The y-axis represents the scores before normalization which can be positive or negative, but only the magnitude matters. Higher scores mean larger correlations with the aspect/opinion prototype. As the aspect and opinion attention have different sets of parameters, the scores can correspond to different ranges of the values. Tokens in purple (blue) are the ground-truth aspect (opinion) terms. Obviously, purple tokens correspond to large scores for aspect extraction (purple bars with large values), and blue tokens correspond to large scores for opinion extraction (blue bars with large values). All the other non-relevant terms have lower scores. This is aligned with our intuition that attentions could select the inputs of interest.

As mentioned previously, MNCA is able to extract target terms without any dependency parser, and hence does not depend on the quality of the parsing results. To show that, we pick a few example reviews from the test datasets as presented in Table 5. The left and right column show the prediction results from the proposed model and RNCRF [3], respectively, where predicted opinions are made *italic*, and aspects are "quoted". These reviews use informal texts that may not be parsed correctly. Hence, RNCRF fails to extract some of the targets, which can be successfully identified by MNCA.

To show the robustness of MNCA, we provide two sensitivity studies on word embedding dimensions and the number of different interactions within a 3-dimensional tensor on S4 in part [a] and [b] of Fig. 8. From the plot, we can see that the performances for both aspect and opinion terms extraction are relatively stable when varying word embedding dimensions, with the highest scores achieved at 200. For the number of tensor interactions, the model attains the best performance at 20 for aspect extraction and 10 for opinion extraction.

Table 5
Prediction comparison between MNCA and RNCRF.

Prediction with MNCA	Prediction with RNCRF
also *stunning* "colors" and *speedy*	also *stunning colors* and speedy
Only 2 "usb ports" ... seems kind of *limited*	Only 2 "usb ports" ... seems kind of limited
strong "build" though which really adds to its "durability"	*strong* "build" though which really adds to its durability
Save room for "deserts" – they're to *die for*	Save room for "deserts" – they're to die for
You must try "Odessa stew" or "Rabbit stew"; "salads" – all *good*	You must try "Odessa stew or Rabbit stew"; salads – all good

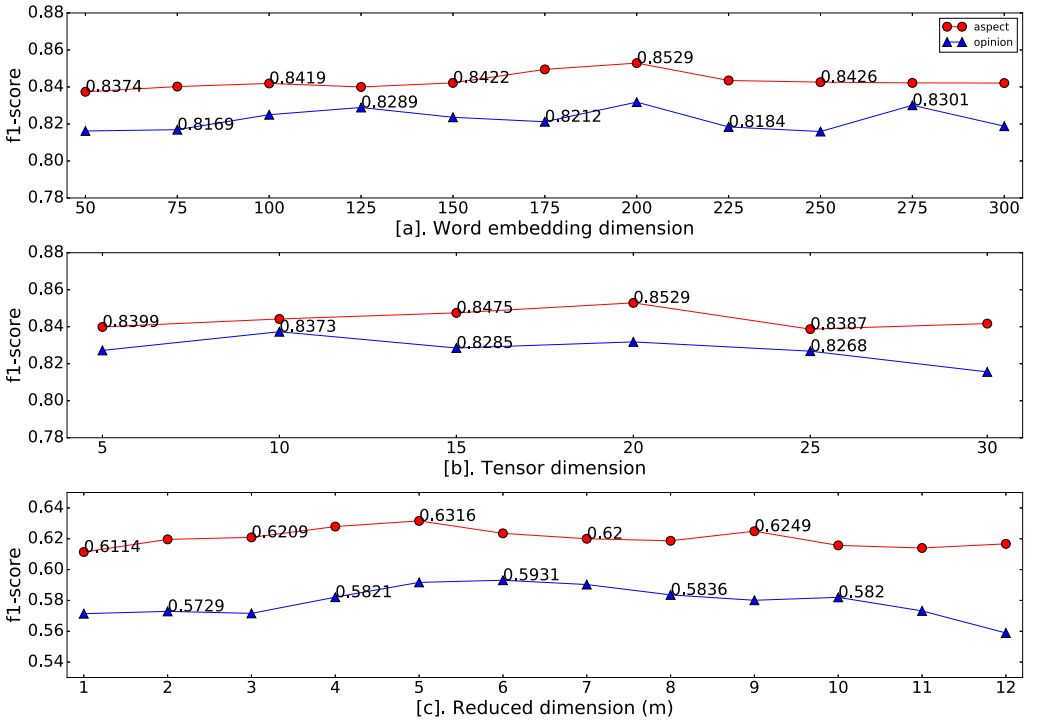

Fig. 8. Sensitivity studies on the datasets S1 and S4.

6.2.2. Category-specific aspect and opinion terms extraction

In this section, we conduct experiments to verify the effectiveness of MTMN for category-specific aspect and opinion terms extraction. The results are shown for the first three datasets S1, S2, and S3, because we do not manually label the category information for S4. Moreover, the sentences in S3 whose categories are rarely seen are removed to facilitate our experiment. We conduct comparisons with the following baseline models:

NLANG: The best system for both SemEval-15 and SemEval-16 for the proposed task.

IHS-RD, XRCE: The second best systems for SemEval-15 and SemEval-16, respectively.

RNCRF+: We modify RNCRF [3], which is for aspect-opinion terms extraction, by defining finer-grained categories as labels. This means directly apply RNCRF to a multi-class classification problem with $C \times 5$ classes, as there are 5 classes for each category and there are C categories.

MNCA+: Similar to RNCRF+, we modify MNCA [19] by defining finer-grained categories as labels. Note that the proposed MTMN can be reduced to this baseline model by only using a single dual propagation memory introduced in Section 5.1 with $C \times 5$ classes.

MNCA++: MNCA is used to extract all the aspect and opinion terms first, and then a category classification layer is added to classify the extracted terms.

We report the results from top performing systems in the Challenges for S1 and S2. There are no reported results for S3 as the original task is different from ours. Note that original task for S1 and S2 includes two slots: slot 1 for sentence-level aspect category prediction and slot 2 for aspect terms extraction. Moreover, SemEval also evaluated on the pairing of slot 1 and slot 2 by joining them as an additional task that corresponds to the problem we study. However, most of the reported models did not provide feasible methods for the joint prediction of aspect terms and corresponding categories. Instead, they trained the model for slot 2 first and then combined with slot 1. This may fail to capture the relations between target terms and their categories. In order to show the advantage of our model, we modify the existing state-of-the-art deep models for aspect/opinion term extraction to fit our problem settings. Since RNCRF and MNCA both exploit the correlations between

Table 6
Comparison results in terms of F_1 scores. ASC (OPC) refers to category-specific aspect (opinion) terms extraction. AS (OS) refers to aspect (opinion) terms extraction.

Model	S1				S2				S3			
	ASC	**OPC**	AS	OP	**ASC**	**OPC**	AS	OP	**ASC**	**OPC**	AS	OP
NLANG	42.90	–	67.11	–	52.61	–	72.34	–	–	–	–	–
IHS_RD	42.72	–	63.12	–	–	–	–	–	–	–	–	–
XRCE	–	–	–	–	48.89	–	61.98	–	–	–	–	–
RNCRF+	54.00	47.86	67.74	67.62	56.04	51.09	69.74	76.15	54.05	58.90	71.87	76.62
MNCA+	57.35	55.70	70.73	73.68	57.83	56.04	75.21	77.90	55.71	62.40	72.42	76.98
MNCA++	53.46	53.94	70.73	73.68	54.05	54.34	75.21	77.90	54.31	62.95	72.42	76.98
MTMN	**63.16**	**59.17**	71.31	72.23	**65.34**	**61.44**	73.26	76.10	**57.06**	**63.53**	69.14	75.76

Table 7
Comparison results with reductions.

Different Components	S1				S2			
	ASC (tt)	OPC (tt)	ASC (cv)	OPC (cv)	ASC (tt)	OPC (tt)	ASC (cv)	OPC (cv)
MTMN (C1+C2+C3)	**63.16** (1.02)	**59.17** (0.27)	63.32	62.98	**65.34** (1.20)	**61.44** (0.97)	63.65	64.93
C1+C3	61.95 (0.25)	58.57 (0.57)	61.68	59.74	63.30 (0.65)	59.16 (1.24)	61.03	57.17
C2+C3	61.67 (0.67)	55.89 (1.05)	62.25	56.55	60.86 (0.33)	58.68 (0.31)	60.65	59.69
C2+C3*	61.30 (0.59)	55.30 (1.18)	61.64	58.63	62.68 (1.57)	58.93 (0.51)	62.30	60.83
C1+C2	60.67 (0.46)	56.97 (0.78)	62.48	57.62	61.29 (0.50)	58.16 (1.09)	60.43	58.37
C3	60.18 (0.74)	57.03 (0.76)	61.19	58.73	60.57 (0.61)	57.36 (1.19)	61.06	60.04

aspect terms and opinion terms, which have been shown to be effective for extraction task, a simple idea is to increase the number of classes to incorporate different categories, e.g., BA becomes {BA$_c$}'s for different category c. By increasing the number of classes, the only change to the original model is the dimension of classification matrix. As a result, the modified model should be able to capture the correlations between target terms based on their categories. On the other hand, we also construct another baseline model (denoted by MNCA++) based on MNCA by separating the task into 2 steps. The first step is the same as MNCA for extracting target terms. Then the second step performs category prediction only on the extracted terms.

The comparison results are shown in Table 6. It can be seen that MTMN achieves the state-of-the-art performances in category-specific aspect and opinion terms extraction (ASC and OPC). And there is a large gap between the results of MTMN and the other baseline models on S1 and S2. This is because RNCRF+ and MNCA+ can only propagate information between target terms within each category, but fail to explore the relations and commonalities among different categories. The other model MNCA++ performs even poorer, because the training is separated into different stages, similar to the top systems in SemEval Challenges. This separation results in the failure of propagating information from category prediction to target term extraction. The result proves the effectiveness of MTMN for learning shared information among different tasks, as well as the addition of global information to assist extraction. The improvement for S3 is not significant, which might indicate that the category correlations are not obvious in laptop domain, as can be seen in Table 2. Only entity labels make different categories distinct from each other.

Moreover, we also report the results on target terms extraction (AS and OP) by accumulating the aspect/opinion terms that are assigned at least one category by MTMN. It can be seen that MTMN still achieves comparable performances even if the data becomes sparser when adding the category information. On the contrary, the results for RNCRF+, MNCA+ and MNCA++ are obtained using the original models that ignore category labels, which are much easier to obtain high performances.

As have been discussed in the previous sections, the multi-task memory network explores the commonalities and relations among tasks through both tensor sharing and feature sharing, as well as enhances prediction results by incorporating auxiliary labels. To test the effect of each component, we conduct comparison experiments for different combinations of these components as shown in Table 7, where C1, C2 and C3 represents separate component for multi-task tensor sharing, context-aware feature sharing and auxiliary task, respectively.

Note that for C2+C3*, we use the same tensor across all the tasks. We report both the testing results with standard variation as well as the validation results. Clearly, each component is helpful for final predictions if we compare the results between (C1+C2+C3) with (C1+C2), (C2+C3) or (C1+C3). Furthermore, tensor sharing is more beneficial than feature sharing most of the time by comparing (C1+C3) with (C2+C3). This might indicate that the commonalities in terms of token interactions are more obvious for different tasks. Moreover, either independent tensors (C2+C3) or the same tensor (C2+C3*) across tasks does not perform well. This indicates the importance to explore both the uniqueness and commonality of all the tasks, which are preserved in our proposed model.

To show the robustness of our model, we test it with different dimensions of factorization (m). The results on S1 are shown in part [c] of Fig. 8. We also provide some examples in Table 8 to show what the attentions learn for different categories. The second and third columns show the extracted aspect and opinion terms with corresponding normalized

Table 8
Examples of attention scores for different categories.

Sentence	Aspect (score)	Opinion (score)
Excellent **food** though the **interior** could use some help.	**1**: food (0.77); **2**: interior (0.79)	**1**: Excellent (0.56)
The **sashimi** is always *fresh* and the **rolls** are *innovative* and delicious.	**1**: sashimi (0.41), rolls (0.51) **3**: rolls (0.49)	**1**: fresh (0.52) **3**: innovative (0.38)
The **food** was *good*, the **place** was *clean* and *affordable*.	**1**: food (0.70) **2**: place (0.81)	**1**: good (0.66); **2**: clean (0.46) **4**: affordable (0.34)
Overall, *decent* **food** at a *good* price, with *friendly* **people**.	**1**: food (0.70) **6**: people (0.53)	**1**: decent (0.31); **5**: good (0.28) **6**: friendly (0.38)
The **wine list** was *expensive*, though the **staff** are not *knowledgeable*.	**6**: staff (0.51) **7**: wine (0.32) list (0.31)	**6**: knowledgeable (0.49) **7**: extensive (0.64)
The **martinis** are *amazing* and very *fairly priced*.	**8**: martinis (0.42) **9**: martinis (0.53)	**8**: fairly (0.21) priced (0.69) **9**: amazing (0.36)
Abulous **food**, if the **front of house staff** don't put you off.	**1**: food (0.82); **6**: front (0.26) of (0.10) house (0.23) staff (0.21)	**1**: Abulous (0.46)
The **food** was *great* and *tasty*, but the **sitting space** was too *small*.	**1**: food (0.59) **2**: sitting (0.35) space (0.27)	**1**: great (0.36), tasty (0.35) **2**: small (0.32)
I have never been so *disgusted* by **food** and **service**.	**1**: food (0.86); **6**: service (0.66)	**1, 6**: disgusted (0.50, 0.76)
Despite the *confusing* **mirrors** this will be my *go-to* for **Japanese food**.	**1**: Japanese (0.32) food (0.30) **2**: mirrors (0.35)	**1**: go-to (0.50) **2**: confusing (0.39)
Service *ok*, but *unfriendly filthy* **bathroom**.	**2**: bathroom (0.79) **6**: service (0.52)	**2**: unfriendly (0.33), filthy (0.41) **6**: ok (0.31)

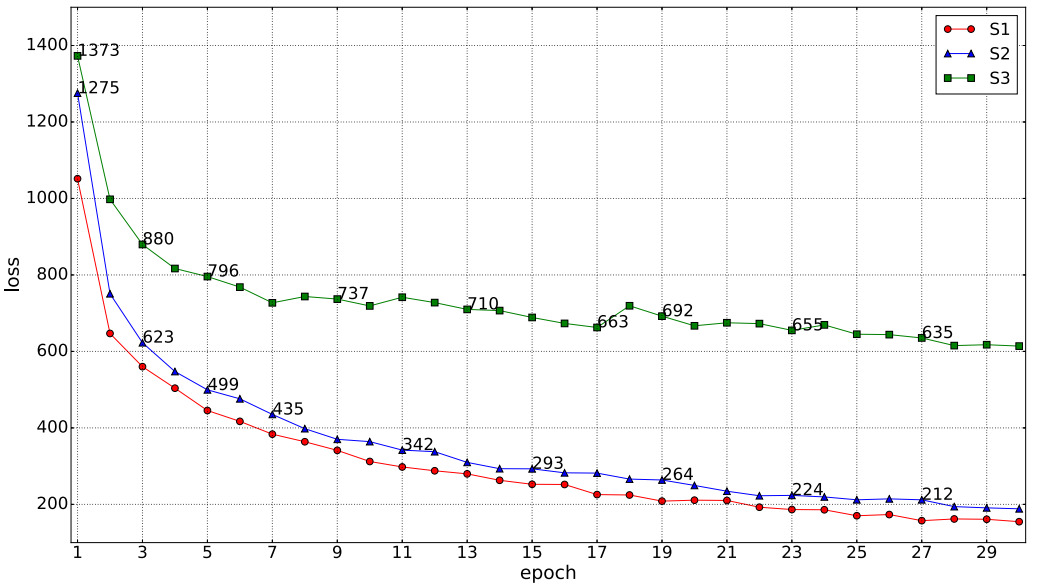

Fig. 9. Convergence analysis on S1, S2 and S3.

attention scores, respectively. The **bold numbers** denote category indexes. As shown in these cases, MTMN is able to attend to target terms for different categories. In some cases, MTMN could identify multiple categories for specific terms.

As being discussed, the tensor decomposition of MTMN deals with data sparsity. Compared with separate tensor for each task, our decomposition exploits sharing among tensor parameters and reduce the complexity. Specifically in the experiments, the tensor for each task has size $20 \times 50 \times 50$. With 12 tasks in total, the resulting parameter size becomes $12 \times 20 \times 50 \times 50$ when there is no sharing. In MTMN, some sharing among the tensors reduces the parameter size to $20 \times (5 \times 50 \times 50 + 20 \times 5)$, which is much smaller than no sharing. Under such condition, our model is able to converge in less than 30 epochs. We plot the exact total loss according to the objective function for all the training sentences in S1, S2 and S3. As shown in Fig. 9, we observe considerable loss decreasing for the first 5 epochs, followed by slower speed for the following 10 epochs until convergence.

7. Conclusion

In this work, we develop a memory network with coupled attentions for fine-grained sentiment analysis. We then extend the memory network in a multi-task learning manner to solve a finer-grained opinion mining problem, which involves the predictions of both aspect/opinion terms and their corresponding aspect categories. In the end, we demonstrate the effectiveness of our proposed models on several benchmark datasets compared with state-of-the-art baseline methods.

Acknowledgements

This work is partially supported by the NTU Singapore Nanyang Assistant Professorship (NAP) grant M4081532.020, Singapore MOE AcRF Tier-2 grant MOE2016-T2-2-060, and Singapore MOE AcRF Tier-1 grant 2016-T1-001-159.

References

[1] M. Hu, B. Liu, Mining and summarizing customer reviews, in: KDD, 2004, pp. 168–177.
[2] G. Qiu, B. Liu, J. Bu, C. Chen, Opinion word expansion and target extraction through double propagation, Comput. Linguist. 37 (1) (2011) 9–27.
[3] W. Wang, S.J. Pan, D. Dahlmeier, X. Xiao, Recursive neural conditional random fields for aspect-based sentiment analysis, in: EMNLP, 2016.
[4] B. Liu, Sentiment Analysis – Mining Opinions, Sentiments, and Emotions, 2015.
[5] M. Hu, B. Liu, Mining opinion features in customer reviews, in: AAAI, 2004, pp. 755–760.
[6] W. Jin, H.H. Ho, A novel lexicalized hmm-based learning framework for web opinion mining, in: ICML, 2009, pp. 465–472.
[7] F. Li, C. Han, M. Huang, X. Zhu, Y.-J. Xia, S. Zhang, H. Yu, Structure-aware review mining and summarization, in: COLING, 2010, pp. 653–661.
[8] Y. Yin, F. Wei, L. Dong, K. Xu, M. Zhang, M. Zhou, Unsupervised word and dependency path embeddings for aspect term extraction, in: IJCAI, 2016.
[9] S. Poria, E. Cambria, A. Gelbukh, Aspect extraction for opinion mining with a deep convolutional neural network, Knowl.-Based Syst. 108 (C) (2016) 42–49.
[10] H. Xu, B. Liu, L. Shu, P.S. Yu, Double embeddings and cnn-based sequence labeling for aspect extraction, in: ACL, 2018.
[11] P. Liu, S. Joty, H. Meng, Fine-grained opinion mining with recurrent neural networks and word embeddings, in: EMNLP, 2015, pp. 1433–1443.
[12] D. Bahdanau, K. Cho, Y. Bengio, Neural machine translation by jointly learning to align and translate, CoRR abs/1409.0473 (2014).
[13] G. Carenini, R.T. Ng, E. Zwart, K-Cap, 2005, pp. 11–18.
[14] J. Yu, Z.-J. Zha, M. Wang, T.-S. Chua, Aspect ranking: Identifying important product aspects from online consumer reviews, in: ACL, 2011.
[15] Z. Zhai, B. Liu, H. Xu, P. Jia, Grouping product features using semi-supervised learning with soft-constraints, in: COLING, 2010, pp. 1272–1280.
[16] Z. Zhai, B. Liu, H. Xu, P. Jia, Clustering product features for opinion mining, in: WSDM, 2011, pp. 347–354.
[17] H. Guo, H. Zhu, Z. Guo, X. Zhang, Z. Su, Product feature categorization with multilevel latent semantic association, in: CIKM, 2009, pp. 1087–1096.
[18] I. Titov, R. McDonald, Modeling online reviews with multi-grain topic models, in: www, 2008, pp. 111–120.
[19] W. Wang, S.J. Pan, D. Dahlmeier, X. Xiao, Coupled multi-layer tensor network for co-extraction of aspect and opinion terms, in: AAAI, 2017.
[20] A.-M. Popescu, O. Etzioni, Extracting product features and opinions from reviews, in: EMNLP, 2005, pp. 339–346.
[21] Y. Wu, Q. Zhang, X. Huang, L. Wu, Phrase dependency parsing for opinion mining, in: EMNLP, 2009, pp. 1533–1541.
[22] K. Liu, L. Xu, J. Zhao, Opinion target extraction using word-based translation model, in: EMNLP-CoNLL, 2012, pp. 1346–1356.
[23] K. Liu, L. Xu, Y. Liu, J. Zhao, Opinion target extraction using partially-supervised word alignment model, in: IJCAI, 2013, pp. 2134–2140.
[24] R. He, W.S. Lee, H.T. Ng, D. Dahlmeier, An unsupervised neural attention model for aspect extraction, in: ACL, 2017, pp. 388–397.
[25] X. Li, W. Lam, Deep multi-task learning for aspect term extraction with memory interaction, in: EMNLP, 2017, pp. 2886–2892.
[26] I. Titov, R.T. McDonald, A joint model of text and aspect ratings for sentiment summarization, in: ACL, 2008, pp. 308–316.
[27] Y. Lu, C. Zhai, N. Sundaresan, Rated aspect summarization of short comments, in: WWW, 2009, pp. 131–140.
[28] W.X. Zhao, J. Jiang, H. Yan, X. Li, Jointly modeling aspects and opinions with a maxent-lda hybrid, in: EMNLP, 2010, pp. 56–65.
[29] Z. Chen, A. Mukherjee, B. Liu, Aspect extraction with automated prior knowledge learning, in: ACL, 2014, pp. 347–358.
[30] Q. Su, X. Xu, H. Guo, Z. Guo, X. Wu, X. Zhang, B. Swen, Z. Su, Hidden sentiment association in chinese web opinion mining, in: WWW, 2008, pp. 959–968.
[31] L. Chen, J. Martineau, D. Cheng, A.P. Sheth, Clustering for simultaneous extraction of aspects and features from reviews, in: NAACL-HLT, 2016, pp. 789–799.
[32] V. Mnih, N. Heess, A. Graves, K. Kavukcuoglu, Recurrent models of visual attention, in: NIPS, 2014, pp. 2204–2212.
[33] J. Weston, S. Chopra, A. Bordes, Memory networks, in: ICLR, 2015.
[34] K. Gregor, I. Danihelka, A. Graves, D.J. Rezende, D. Wierstra DRAW, A recurrent neural network for image generation, in: ICML, 2015, pp. 1462–1471.
[35] A.M. Rush, S. Chopra, J. Weston, A neural attention model for abstractive sentence summarization, in: EMNLP, 2015, pp. 379–389.
[36] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, E. Hovy, Hierarchical attention networks for document classification, in: NAACL, 2016, pp. 1480–1489.
[37] K.M. Hermann, T. Kočiský, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, P. Blunsom, Teaching machines to read and comprehend, in: NIPS, 2015.
[38] A. Kumar, O. Irsoy, P. Ondruska, M. Iyyer, I.G. James Bradbury, V. Zhong, R. Paulus, R. Socher, Ask me anything: dynamic memory networks for natural language processing, in: ICML, 2016.
[39] S. Sukhbaatar, A. Szlam, J. Weston, R. Fergus, End-to-end memory networks, in: NIPS, 2015, pp. 2440–2448.
[40] R. Caruana, Multitask learning, Mach. Learn. 28 (1) (1997) 41–75.
[41] A. Argyriou, T. Evgeniou, M. Pontil, Convex multi-task feature learning, Mach. Learn. 73 (3) (2008) 243–272.
[42] A. Kumar, H.D. III, Learning task grouping and overlap in multi-task learning, in: ICML, 2012.
[43] B. Romera-Paredes, H. Aung, N. Bianchi-Berthouze, M. Pontil, Multilinear multitask learning, in: ICML (3), in: J. Mach. Learn. Res. Workshop Conf. Proc., vol. 28, 2013, pp. 1444–1452.
[44] X. Liu, J. Gao, X. He, L. Deng, K. Duh, Y.-Y. Wang, Representation learning using multi-task deep neural networks for semantic classification and information retrieval, in: NAACL, 2015, pp. 912–921.
[45] I. Misra, A. Shrivastava, A. Gupta, M. Hebert, Cross-stitch networks for multi-task learning.
[46] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning phrase representations using rnn encoder-decoder for statistical machine translation, in: EMNLP, 2014, pp. 1724–1734.
[47] R. Socher, A. Perelygin, J. Wu, J. Chuang, C.D. Manning, A.Y. Ng, C. Potts, Recursive deep models for semantic compositionality over a sentiment treebank, in: EMNLP, 2013, pp. 1631–1642.
[48] M. Pontiki, D. Galanis, H. Papageorgiou, S. Manandhar, I. Androutsopoulos, SemEval-2015 task 12: aspect based sentiment analysis, in: SemEval 2015, 2015, pp. 486–495.

[49] M. Pontiki, D. Galanis, H. Papageorgiou, I. Androutsopoulos, S. Manandhar, M. AL-Smadi, M. Al-Ayyoub, Y. Zhao, B. Qin, O.D. Clercq, V. Hoste, M. Apidianaki, X. Tannier, N. Loukachevitch, E. Kotelnikov, N. Bel, S.M. Jiménez-Zafra, G. Eryiğit, SemEval-2016 task 5: aspect based sentiment analysis, in: SemEval 2016, 2016.
[50] M. Pontiki, D. Galanis, J. Pavlopoulos, H. Papageorgiou, I. Androutsopoulos, S. Manandhar, Semeval-2014 task 4: aspect based sentiment analysis, in: SemEval, 2014, pp. 27–35.
[51] J. McAuley, C. Targett, Q. Shi, A. van den Hengel, Image-based recommendations on styles and substitutes, in: SIGIR, 2015, pp. 43–52.