SentiMem: Attentive Memory Networks for Sentiment Classification in User Review

Xiaosong Jia[1], Qitian Wu[1], Xiaofeng Gao[1], and Guihai Chen

Shanghai Jiao Tong University, China
{jiaxiaosong,ech0740}.sjtu.edu.cn, {gao-xf, gchen}@cs.sjtu.edu.cn

Abstract. Sentiment analysis for textual contents, as an important research topic of natural language processing, has attracted lots of attentions. However, most existing models only utilize the target text to mine the deep relations from text representation features to sentiment values, ignoring user's historically published texts, which also contain much valuable information. Correspondingly, in this paper we propose SentiMem, a new sentiment analysis framework that incorporates user's historic texts to improve the accuracy of sentiment classification.
In SentiMem, to exploit users' users' interests and preferences such as dog person, football fan, and foodie hidden in the texts, we adopt SenticNet to capture the concept-level semantics; as for users' temperaments like optimism, peevishness, and pessimism, we combine multiple sentiment lexicons with multi-head attention mechanism to extract users' diverse characters. Then, we design two memory networks: *Interests Memory Network* and *Temperaments Memory Network* to store information about users' interests and temperaments respectively. Interests memory is updated in a first-in-first-out way and read by a attention mechanism to match the usrs' most recent interests with the target text. Temperaments memory is updated in a forgetting-and-strengthening manner to match the gradual shift of human's characteristics. Extensive experiments on two real-world datasets show that SentiMem can achieve significant improvement for accuracy over state-of-the-art methods.

Keywords: Sentiment Analysis · Memory Network · Social Network · Natural Language Processing.

1 Introduction

People frequently use texts to express their feelings on online platforms, such as social networks and e-commerce websites. The sentiment analysis, a basic task of nature language processing, is to recognize the sentiment attitudes behind texts and classify them into different categories like positive texts and negative texts. The sentiment scores could play a significant role in many applications and thus there has been an increasing number of researches on sentiment analysis during years.

Recently, deep learning based methods have become mainstream for sentiment analysis and achieved decent performances. There are models based on

CNN/RNN. [25] proposes an aspect sentiment classification model with both word-level and clause-level networks. [18] utilizes commonsense knowledge with an attentive LSTM for targeted aspect-based sentiment analysis. [24] proposes an entirely attention-based methods to explore the deep relationship between words and it outperforms RNN/CNN based models a lot. Up to now, start-of-the-art performances of sentiment analysis tasks are dominated by attention-based pretrained models like BERT [6] and GPT [20].

However, most of the above-mentioned methods **focus on** leveraging textual features from the target text whereas **ignore** the information from text promulgators. Indeed, in the real world individuals possess their own **characteristics** like interests and temperaments, which could greatly influence the sentiment attitudes of users' texts. E.g., in terms of users' **interests**, consider an example where a dog man always post texts about his love of dog. One day, he writes a text *My dog is a very bad guy!*. Even this text includes the word *bad*, we can still recognize its positivity because we know he loves his dog and the word *bad* reflects his cosset. Nevertheless, for sentiment analysis models based on single text, they may tend to give a negative label according to the word *bad*. No matter how powerful the single-text based models are, this kind of mistakes will happen because these models only have that text and can only interpret it in a general way without considering user interests. Additionally, in terms of users' **temperaments**, some people like to use exaggerated words like *great*, *excellent*, and *fantastic* to express their happiness. Then, if a guy posts a twitter *Today is fine.* one day, we can conclude that it may be a bad day for him because he uses the ordinary word *fine* instead of those exaggerated words this time. If we merely feed this text into a model, it would be definitely be classified as a positive one. This overestimation comes from the users' unknown temperaments.

Therefore, an accurate sentiment analysis model should take user' historic texts into consideration. Rare researches pioneered such idea in this field. [23] learns a static embedding for each user and feeds both the user's embedding vector and the target text into the neural network to classify sentiment polarity of thetext. However, this model ignores the order of texts, whose importance has been emphasized by [4]. Actually, since a user's characteristics may change over time, his recent texts may be more representative while a static embedding wrongly treats all texts in the same way. [7] adopts end-to-end memory netowrks [22] to store historic texts' embeedings and reads the memories by the target text. Although it utilizes sequential information, the model is still weak because it only extracts basic semantic information from words by stacking memory networks. It uses pretrained word embeddings such as Word2Vec and Glove containing only general meanings of words. Consider the examples of users' interests and temperaments again: the mistake originates from the abstract and high-level characteristics of human. It is hard to extract them from such a general representations of words without any domain knowledge.

In conclusion, to conduct sentiment classification in historical text sequence, there are several challenges. **Firstly**, same words/sentences may have very different sentiment attitudes for different people because of the diverse and abstract personal characteristics. **Second**, in most situations, we have no user profile information due to some privacy issues and cost of investigations, so we can only deduce one's characteristics from his published texts. **Thirdly**, user's *interests* and *temperaments* may change over time. For example, a person may be pessimistic these days and be optimistic after a few days. Hence the model should dynamically capture user's time-variant characteristics instead of assuming some static features.

To these ends, in this paper, we propose SentiMem, a new framework that models user's characteristics of *interests* and *temperaments* to help with the sentiment analysis task of target text. Specifically, we use a pre-trained model to extract general **semantic** embedding of texts first. To obtain users' **interests** from texts, we combines SenticNet [3], a knowledge base includes concepts of words with dimension reduction technique [3] to deal with the high-dimensional and sparse matrix of knowledge base. With domain knowledge and concepts, SentiMem could process words with specific semantic easier to extract users' interests. As for users' **temperaments**, we let scores from multiple sentiment lexicons of a word as its embedding. By using sentiment polarity of words explicitly and directly, SentiMem could better extract users' temperaments from texts in terms of writing style and intensities of common used words. Then, we design two memory networks, a kind of neural network structure proposed by [22] which has shown quite promising performances for many sequential tasks, to store users' characteristics. To capture the temporal dynamics in the historic sequence of user's publish texts. The updating method occurs sequentially for user's historically published texts so that it could capture user's time-variant characteristics. The reading and updating of the two memory networks are also based on an attention mechanism to dynamically weigh the influence of those significant and relevant historic texts. Additionally, we apply mutlihead attention mechanism to capture the very diverse characteristics of human in different embedding space [24]. For Interests Memory Network, the updating is in a first-in-first-out manner to store users' most recent interests. For Temperaments Memory Network, SentiMem learns a global table to represent the common features of human's temperaments and updates the memory network in a forgetting-and-strengthening manner to match the gradual shift of human's temperaments. To verify the model, we conduct experiments on two real-world datasets from Amazon reviews and compare with several state-of-the-art methods. The results show that SentiMem outperforms other competitors. Also, we do some ablation tests and verify the effectiveness of two memory networks for modeling the temporal dynamics in user's characteristics.

Our main contributions are summarized as follows:

– **New aspect:** We take a fresh view and combine users' historic texts with SenticNet and sentiment lexicons to capture users' interests and temperaments hidden behind texts.
– **Methodology:** We propose a new sentiment analysis framework with two memory networks, which can collaboratively capture temporal dynamics in user's interests and temperaments.

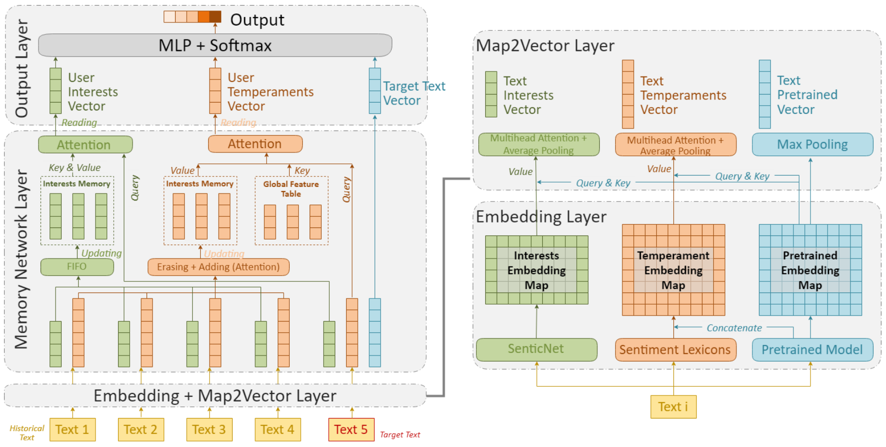


Fig. 1: Model framework of SentiMem with four layers. Embedding Layer and Map2Vector Layer are used to produce three vectors for individual texts, while Memory Network Layer and Output Layer are for the sequential texts.

– **Experiment:** We conduct experiments on two Amazon reviews datasets to show superiority of proposed method as well as effectiveness of sequential components.

In the following section, we will give a formal formulation of problem. Then we will introduce the proposed SentiMem model in a layer-by-layer way. Fig. 1 shows the overall architecture of the model.

2 Problem Formulation

We denote a text $\boldsymbol{x}$ of length m as a sequence of m words, i.e. $\boldsymbol{x} = \{\boldsymbol{w_1}, \boldsymbol{w_2}, ..., \boldsymbol{w_m}\}$ where $\boldsymbol{w_j}$ is the j^{th} word. Then we define the problem as: Given a sequence of n texts in time order posted by user u: $X^u = \{\boldsymbol{x_1}, \boldsymbol{x_2}, ..., \boldsymbol{x_n}\}$ where $\boldsymbol{x_j}$ is the j^{th} text the user posted, we want to predict the n^{th} text's label y_n^u where $y_n^u \in \{1, 2, ..., K\}$ indicates the sentiment attitude of text with 1 as lowest and K as highest. In this paper, we call $\boldsymbol{x_n}$ as the **target text** and other texts of user u as **historic texts**.

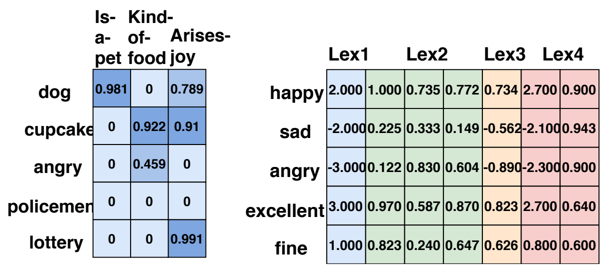
3 SentiMem Framework

In this section, we will introduce the proposed SentiMem model in a layer-by-layer way. Fig. 1 shows the overall architecture of the model.

3.1 Embedding Layer

In the Embedding Layer, we aims to obtain several embeddings of texts in a word-level manner, i.e. there would be a vector for each word. Thus there would be a matrix for each text.

We harness a encoder structure as a pretrained model to obtain basic semantic representations of texts. The pretrained model should keep the shape



	Is-a-pet	Kind-of-food	Arises-joy
dog	0.981	0	0.789
cupcake	0	0.922	0.91
angry	0	0.459	0
policemen	0	0	0
lottery	0	0	0.991

	Lex1	Lex2			Lex3	Lex4	
happy	2.000	1.000	0.735	0.772	0.734	2.700	0.900
sad	-2.000	0.225	0.333	0.149	-0.562	-2.100	0.943
angry	-3.000	0.122	0.830	0.604	-0.890	-2.300	0.900
excellent	3.000	0.970	0.587	0.870	0.823	2.700	0.640
fine	1.000	0.823	0.240	0.647	0.626	0.800	0.600

Fig. 2: Illustration about how to obtain Interests Embedding Map and Temperaments Embedding Map in Embedding Layer. (Left) Each row represents a word and the scores on each column represent how much one word is related to this feature in the SenticNet. (Right) Each row represents a word and the scores on each column represent the scores on different sentiment lexicons. Note that there may be multiple scores in one lexicon like valence, arousal, and dominance in [19].

of sequences (like LSTM/GRU/BiGRU) to maintain semantics of each word. Formally, for a input text $\boldsymbol{x} = \{\boldsymbol{w_1}, \boldsymbol{w_2}, ..., \boldsymbol{w_m}\}$, after the pretrained model, we will get a sequence of dense representation of the text $\boldsymbol{x^p} = \{\boldsymbol{w_1^p}, \boldsymbol{w_2^p}, ..., \boldsymbol{w_n^p}\}$ where $\boldsymbol{w_k^p}$ represents the denser and more abstractive representation of word w_k. In this paper, we call the representations for a text after the pretrained model as **Pretrained Embedding Map** (as shown in Fig. 1).

To embed a single word to an interests vector, we use SenticNet [3], a knowledge base that contains 100,000 concepts with a huge set of affection properties. As we can see in Fig. 2 (Left), features of a word including the interests and knowledge information are scored between 0 and 1. For example, *dog* is scored highly at *IsA-pet* and *Arises-joy* which means we can capture word-level interests information from SenticNet. Nevertheless, SenticNet is a high-dimensional and sparse embedding which makes it hard to use directly. Followed by [2, 18], we adopt a dimension reduction technique to map the resuls of SenticNet to a relatively low-dimensional(100) continuous space with little loss of the semantic and relativeness. For a text $\boldsymbol{x} = \{\boldsymbol{w_1}, ..., \boldsymbol{w_m}\}$, we can get its **Interests Embedding Map** (as shwon in Fig. 1) $\boldsymbol{x^i} = \{\boldsymbol{w_1^i}, ..., \boldsymbol{w_n^i}\}$ where $\boldsymbol{w_k^i}$ is word w_k's SenticNet embedding.

To obtain a single word's temperaments vector, we utilize sentiment lexicons, a traditional sentiment analysis tool, which has been shown powerful when coupled with word embeddings [21]. Specifically, as shown in Fig. 2 (right), we concatenate scores of a word obtained from multiple open-source sentiment lexicons with the word's pretrianed embedding to get the *temperaments vector* of this word. The reason why using concatenation is that sentimental scores are just sentiment polarity and using them singly would lose too much semantic information. With the concatenation of scores and pretrained embeddings, it can better capture the temperaments characteristic of a word. Finally, we can get **Temperaments Embedding Map** (as shown in Fig. 1) of each text as $\boldsymbol{x^t} = \{\boldsymbol{w_1^t}, \boldsymbol{w_2^t}, ..., \boldsymbol{w_n^t}\}$ where $\boldsymbol{w_k^t}$ is the concatenation mentioned before.

In conclusion, by embed words in different ways, they are represented from a different perspective. Specifically, pretrained embedding is based on Glove/Word2Vec and basic sequential models(LSTM/GRU/BiGRU) which are generely used in natural language processing area to extract semantics of texts. As for SenticNet [3], it can extract concepts of words(as shown in Fig. 2 Left) so that we can capture users' interests and knowledge from texts. [21] has shown that with multiple open-source sentiment lexicons, the words can be represented in a more diverse way. With the sentiment scores, valence, arousal, and dominance [19], the users' temperaments behind the words can be better represented.

3.2 Map2Vector Layer

In the Map2Vector layer, the matrices(Pretrained Embedding Map, Interests Embedding Map, Temperaments Embedding Map) obtained from embedding layer are represented in a dense form - document-level vectors. It has been shown to be effective [17, 7] for downstream tasks for several reasons like reducing the model complexity, representing in a more abstract and global way, and lower computational demands.

Target Text Vector For Pretrained Embedding Map to merge the pretrained embedding map into one vector for target text (shown by **Target Text Vector** in the Output Layer, Fig. 1), we adopt the max-pooling to its pretrained embedding map, which is a widely used technique. Specifically, we take the maximum among all words for each dimension.

Text Interests Vector To capture the interests and knowledge in one text of user, Map2Vector layer extracts **Text Interests Vector** from Interests Embedding Map. For example, if a user posted *This dog is so cute.*, the corresponding Text Interests Vector should indicate she is a dog-person. Then, when dealing with his/her new text about dogs, the model should tend to classify it as a positive text.

We use multi-head attention mechanism, a powerful method which lots of state-of-the-art NLP model based on like Bert [6] and GPT [20].The basic equations are shown in (1). Considering it is not our contributions, please see [24] for technical details. The intuition behind using mutli-head attention here is: multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions(like kernels in the CNN) [24]. In our scene, considering that human have complex and diverse characteristics, with multiple head, SentiMem could capture writers' interests and temperaments from different perspective of texts.

Specifically, for a text $\boldsymbol{x}$ with its pretrained embedding map $\boldsymbol{x^p}$ and interests embedding map $\boldsymbol{x^i}$, we let $\boldsymbol{x^p} = Q = K$ and $\boldsymbol{x^i} = V$. Setting pretrained embedding as key and query is because pretrained model's embedding includes general and position information of words and it can give relatively precise weights. Setting

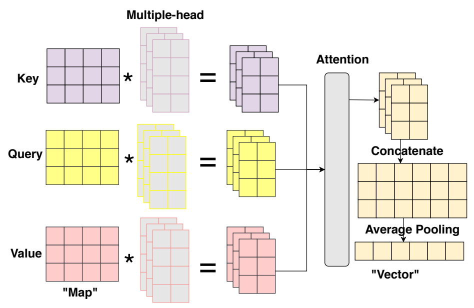


Fig. 3: Illustration for Multihead Attention+Average Pooling in Map2Vector layer. It takes Pretrained Embedding Map as query and key and Interests/Temperaments Embedding Map as value to obtain Text Interests/Temperaments Vector respectively.

interests embedding as value is because SenticNet model's embedding includes only words' specific interests knowledge and this information is what we want to extract. Finally, after taking average pooling over all the words' interests embedding in MultiHead, we can get **Text Interests Vector** (as shown in Fig. 1). Fig. 3 illustrates how this layer works.

$$\text{Attention}(Q,K,V) = \text{softmax}(QK^T)V \tag{1}$$

Text Temperaments Vector Similarly, we use multi-head attention to obtain **Text Temperaments Vector** from Temperaments Embeeding Map, which can reflect user's temperaments hidden in the text. For example, if a user often posts texts to express his sadness, the temperaments vectors should contain the semantic information that he is an pessimistic person. Then, the model will be inclined to classify his/her new text as a positive one.

Formally, for a text $\boldsymbol{x}$ with its pretrained embedding map $\boldsymbol{x^p}$ and temperaments embedding map $\boldsymbol{x^t}$, we let $\boldsymbol{x^p} = Q = K$ and $\boldsymbol{x^t} = V$ as show in (1). The intuition behind it is also the same as Text Interests Vector. After taking average pooling over all the vector's in MutliHead, we can finally get **Text Temperaments Vector** (as shown in Fig. 1).

In conclusion, in the Map2Vector layer, we use mutli-head attention mechanism to extract three kinds of vectors(Pretrained, Temperaments, Interests) of each text from embedding maps. These vectors represent the basic semantic, temperaments, and interests features of each text respectively.

3.3 Memory Network Layer

In the Memory Network layer, we use memory network to store users' temporal interests and temperaments dynamically by writing operation and obtain how both of them influence the target text by reading operation. Additionally, by attention mechanism, SentiMem weighs the influences of those significant and relevant historic texts and characteristics.

Interests Memory Network The *Interests Memory Network* (IMN) aims to store users' interests and knowledge in sequential order and help to capture users' temporal dynamics of interests in the sequence.

Formally, we have text interests vectors (from Map2Vector layer) of n texts posted by one user $X^{\text{inter}} = \{\boldsymbol{x_1}^{\text{inter}}, ..., \boldsymbol{x_n}^{\text{inter}}\}$ and we need to get **User Interests Vector** for target text. We have a initially uniformly randomized memory matrix $M_u^i = \{\boldsymbol{m_1^u}, ..., \boldsymbol{m_K^u}\}$ where K is the number of memory slots and $\boldsymbol{m_j^u}$ indicates the j^{th} memory vectors of user u which has the same dimension as text interests vector.

We **update** user's IMN so that it contains the information of historic texts. Intuitively, what a user recently has done have larger influence on their sentiments. So we adopt a first-in-first-out strategy for the updating of IMN, i.e. IMN only stores the most recent K interests vectors of a user. Formally, if IMN is $M_u^i = \{\boldsymbol{x_{j-1}^{\text{inter}}}, \boldsymbol{x_{j-2}^{\text{inter}}}, ...\boldsymbol{x_{j-K}^{\text{inter}}}\}$, after updating with j^{th} historic text's interest vector $\boldsymbol{x_j}^{\text{interests}}$, then $M_u^i = \{\boldsymbol{x_j^{\text{inter}}}, \boldsymbol{x_{j-1}^{\text{inter}}}, ...\boldsymbol{x_{j-K+1}^{\text{inter}}}\}$.

We **read** IMN with target text's interest vector to obtain User Interests Vector as shown in Fig. 1. Here we use attention mechanism and the intuition behind is that different aspects of interests are contained in different historic texts and they should have different levels of influences on the target text. For $M_u^i = \{\boldsymbol{m_1^u}, ..., \boldsymbol{m_K^u}\}$ and target text's interest vector $\boldsymbol{x_n}^{\text{interests}}$, weights z_u^j can be computed by (2). With weights z_u^j for user u's j^{th} memory vectors in IMN, we can get **User Interests Vector $\boldsymbol{u^i}$** as shown in (2).

$$w_j^u = \boldsymbol{m_j^{uT}} \cdot \boldsymbol{x_n}^{\text{inter}}, \qquad z_j^u = \frac{\exp(w_j^u)}{\sum\limits_{i=1}^{K} \exp(w_i^u)}, \qquad \boldsymbol{u^i} = \sum_{i=1}^{K} z_i^u \boldsymbol{m_i^u} \tag{2}$$

Temperaments Memory Network The *Temperaments Memory Network* (TMN) aims to model users' temperaments such as optimism, peevishness, and pessimism. Temperaments is more abstract than interests and knowledges since the former one is more about the internal characteristics of human and the latter one is more external form. To be specific, in SentiMem, the temperaments embedding map only is derived from sentiment scores while the interests embedding map comes from concepts of SenticNet. So we propose a more complicated memory network to extract users' temperaments. Though Temperaments of human are abstract, there are also some common features among them. Inspired by latent factor models in recommender systems [15], we use **a global latent vector table** to represent the common features of human's temperaments and a memory matrix for each user to represent this user's temporal personal dispositions for the common features.

Formally, we have text temperaments vectors of n texts posted by one user $X^{\text{temper}} = \{\boldsymbol{x_1}^{\text{temper}}, ..., \boldsymbol{x_n}^{\text{temper}}\}$ and we aim to get *User Temperaments Vector* for the target text. We maintain a temperaments memory matrix for each user $M_u^t = \{\boldsymbol{m_1} ..., \boldsymbol{m_K}\}$ where K is the number of memory slots. Also we have a global temperaments feature table $F = \{\boldsymbol{f_1}, ..., \boldsymbol{f_K}\}$ where $\boldsymbol{f_j}$ represents one common feature of human's temperaments and is a parameter vector with the same dimension as the memory vector.

Reading of TMN resorts to attention mechanism as well. For the target text's temperaments vector x_n^{temper}, weights z_j^u can be computed by (3) and **User Temperaments Vector** $\boldsymbol{u^t}$ can be given by weighed sum of memory matrix as in (3).The intuition behind this reading method is that using target text's temperaments vector as query and global temperaments feature vectors as key to capture which temperaments features this text is more about. Then using this user's own temperaments disposition vectors as value to obtain user's specific temperaments about target text.

$$w_j^u = \boldsymbol{f_j}^{\boldsymbol{T}} \cdot \boldsymbol{x_n}^{\text{temper}}, \qquad z_j^u = \frac{\exp(w_j^u)}{\sum\limits_{i=1}^{K} \exp(w_i^u)}, \qquad \boldsymbol{u^t} = \sum_{i=1}^{K} z_i^u \boldsymbol{m_i^u} \tag{3}$$

For **updating** of TMN, as mentioned in Sec. 1, a user's temperaments may shift over time in a slight and slow manner. To model this nature, we need to update a user's personal dispositions after each posting. However, we should not use first-in-first-out strategy again because people's temperaments usually would not change a lot in a short time.

Here we adopt the *Earsing-Adding* updating method. Formally, for a historic text's temperaments vector $\boldsymbol{x^{temper}}$, we first obtain a erasing vector $\boldsymbol{e}$ with a linear layer and then we *erase* each of the memory vector $\boldsymbol{m_j}$ as (4).

$$\boldsymbol{e} = \delta(\boldsymbol{W}^{\text{erase}} \boldsymbol{x^{temper}} + \boldsymbol{b}^{\text{erase}}), \qquad \boldsymbol{m_j} \leftarrow \boldsymbol{m_j} \odot (\boldsymbol{1} - z_j^u \boldsymbol{e}), \tag{4}$$

where $j \in \{1, 2, ..., K\}$, δ is a non-linear function, $\boldsymbol{1}$ is a vector of all 1 and $\boldsymbol{z_j^u}$ is the weight.

After erasing, we *add* new contents to each of the memory vector $\boldsymbol{m_j}$ with adding vector $\boldsymbol{a}$ obtained from the linear layer. (5) shows the details.

$$\boldsymbol{a} = \delta(\boldsymbol{W}^{\text{add}} \boldsymbol{x^{temper}} + \boldsymbol{b}^{\text{add}}), \qquad \boldsymbol{m_j} \leftarrow \boldsymbol{m_j} + z_j^u \boldsymbol{a} \tag{5}$$

The intuition behind the updating equations is that: the magnitude of updating for each disposition vector $\boldsymbol{m_j}$ is decided by how much this text has shown about it. With two additional linear layers for forgetting and strengthening, the model can update user's temperaments matrix more flexibly.

3.4 Output Layer

In the Output Layer, we concatenate **Target Text Vector**, **User Interests Vector** and **User Temperaments Vector** together as one vector and feed it into a fully-connected layer followed by a Softmax layer. We use the label with the highest probability as the prediction result. The loss function is the weighted

Table 1: Statistics of datasets.

Datasets	#Users	#Reviews	#Reviews/Users
Cell Phones and Accessories	27874	190821	6.85
Movies and TV	123952	1691396	13.65

cross-entropy loss defined as

$$L = -\sum_{i=1}^{M}\sum_{c=1}^{C} w_c y_i^c \log(p(\hat{y_i^c}|x)) + \frac{l}{2}\|\theta\|_2^2, \tag{6}$$

where M is the number of train data, C is the number of class, l is the weight of L2 regularization and w_c is the weight of class c. We set w_c as the reciprocal of class c's ratio to deal with the imbalanced-class problem.

4 Experiments

4.1 Experiment Setup

Datasets We conduct experiments on Amazon review dataset [10] which consists of product reviews and metadata from Amazon including 142.8 million reviews spanning 1996 - 2014. We choose two product categories: *Cell Phones and Accessories*(5-core), *Movies and TV*(5-core). The statistics of our dataset are shown in Tab. 1. For each user, we first sort their reviews in time order. For each of reviews, we let it to be the target text, its label as the target and all the reviews posted before it as historic texts. We randomly select 70% reviews (with its historic texts) as the training set and the rest as the test set. Note that: due to the privacy problem, we fail to obtain pure-text datasets like twitter which should include a sequence of texts from the same user, which is most suitable for our scenarios since it would contain texts highly correlated to users' interests and temperaments. As a result, we turn to the Amazon dataset for its consecutive reviews from the same user. Though the scenarios may not be most suitable, we still achieve improvements over other state-of-the-art methods.

Metric We use the same metric and labels processing method as [17]. Specifically, we use accuracy of prediction for labels as the metric. We evaluate the model on both 2-class task which divide labels into 2 kinds (positive and negative) and 5-class task which divide labels into 5 kinds (1-5). Since original labels in Amazon dataset are 5-class, to get 2-class labels, we consider labels between 1 and 4 as negative and labels 5 as positive. The reason why taking 5 as the threshold is that the number of labels 5 in dataset are nearly as much as the number of 1-4, which makes class balanced.

Baseline Methods As mentioned before, this model is a kind of generalized enhancement technique which needs a pretrained model. So pretrained models could be baselines and the goal of this model is to improve the performance of pretrained models. We conduct experiments with the following three pretrained models:

– **LSTM** [11]: Long short-term memory (LSTM) is an artificial recurrent neural network (RNN) architecture which is composed of a cell, an input gate, an output gate and a forget gate. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell. It is widely used in NLP field.
– **GRU** [5]: Gated recurrent units (GRU) is like LSTM with forget gate but has fewer parameters than LSTM, as it lacks an output gate. GRUs have been shown to exhibit better performance on some tasks.
– **BiGRU** [13]: Bidirectional gated recurrent units(BiGRU) connect hidden layers of two GRU with opposite directions to the same output. It is especially useful when the context of the input is needed.

Considering there have been works about utilizing users' historic texts, some of these models could also be baselines. As mentioned in Sec. 1, the proposed model aims to **only use texts** as inputs since in many scenarios other types of information is unavailable. So to fairly compare the proposed model with other baseline models, the baselines should process texts relatively independently so that we can only keep the texts processing parts of them without modifying them a lot. After we surveyed papers about utilizing user' historic texts, we choose the following two representative and fair-to-compare-with models and modify them to satisfy our task:

– **UPNN** [23] uses CNN+average-pooing to obtain document-level vectors of texts and proposes to learn a static embedding vector for each user and each product from users' reviews and information about purchased products respectively. Here we only keep the part about processing users' reviews.
– **UPDMN** [7] uses LSTM to extract document-level vectors of texts and uses end-to-end memory netowrks [22] to store and read embeddings of users' reviews and purchased production. Similar to UPNN, we only keep the memory network for processing users' reviews.

Preprocessing and Word Embedding We use public NLP tool torchtext to prepossess all texts. We use GloVe (dimension 300)as word embedding for pretrained model. Four open-source sentiment lexicons are used to build temperaments embedding of words. All lexicons have scores for words and we scale them all to range $[0, 1]$. Some lexicons also include detailed sentiment information such as valence, arousal and dominance. The Lexicons are as follows: AFINN-111 [9], MaxDiff Twitter Sentiment Lexicon [14], The NRC Valence, Arousal, and Dominance Lexicon [19], and VADER-Sentiment-Analysis [12]. For temperaments embedding (dimension 300+6) from lexicons and interests embedding (dimension 100) from SenticNet, there are missing features for some words. For these missed words, we set its corresponding embedding all 0 to make it neutral.

Table 2: 2-class task and 5-class task prediction accuracy on the Amazon Review Dataset including two categories: *Cell Phones and Accessories*, *Movies and TV*. The symbol **+X** means SentiMem improves the performance of pretrained models by X.

Cell Phones and Accessories	**2-class**	**5-class**
UPNN	81.7	70.5
UPDMN	82.1	71.0
LSTM	81.5	70.2
SentiMem+LSTM	82.6 (**+1.1**)	71.3 (**+1.1**)
GRU	81.5	70.4
SentiMem+GRU	82.5 (**+1.0**)	71.5 (**+1.1**)
BiGRU	82.0	70.8
SentiMem+BiGRU	83.0 (**+1.0**)	72.0 (**+1.2**)

Movies and TV	**2-class**	**5-class**
UPNN	76.8	67.3
UPDMN	77.2	67.5
LSTM	76.4	66.8
SentiMem+LSTM	77.6 (**+1.2**)	68.0 (**+1.2**)
GRU	76.6	66.9
SentiMem+GRU	77.7 (**+1.1**)	68.1 (**+1.2**)
BiGRU	77.1	67.2
SentiMem+BiGRU	78.5 (**+1.4**)	68.5 (**+1.3**)

4.2 Experimental Results

We denote our model as *SentiMem* and SentiMem+X means that we use X as pretrained model. We conduct experiments on two categories of Amazon Review Dataset categories: *Cell Phones and Accessories*, Movies and TV. The results are shown in Table 2. We can find that:

– SentiMem model can always improve performances of pretrain models by about 1% with the additional information from historic texts, sentiments lexicons and SenticNet. There is no significant difference among the improvements for different pretrianed models and different tasks. SentiMem outperforms UPNN and UPDMN as well.
– All the models works worse on Movies and TV dataset than on Cell Phones and Accessories. It is because reviews about cell phones are more objective and easier to classify the sentiment attitudes since they are usually just about whether or not people are satisfied with products. Reviews about movies usually contains more complicated feelings of people, which makes it much harder for models to find the main sentiment attitude of reviews. However, SentiMem can improve the performances of pretrained models slightly more on Movies and TV dataset.

4.3 Ablation Test

To evaluate the effects of each component of SentiMem, we do ablation test by removing user interests vector and user temperaments vector respectively. The results are shown in Fig. 4. We can see that:

– Both user interests vector and user temperaments vector can improve the performance of pretrained models. It shows the effectiveness of capturing

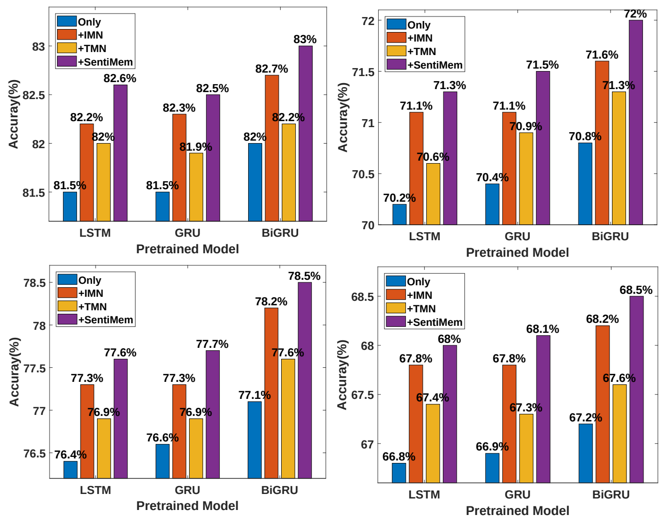


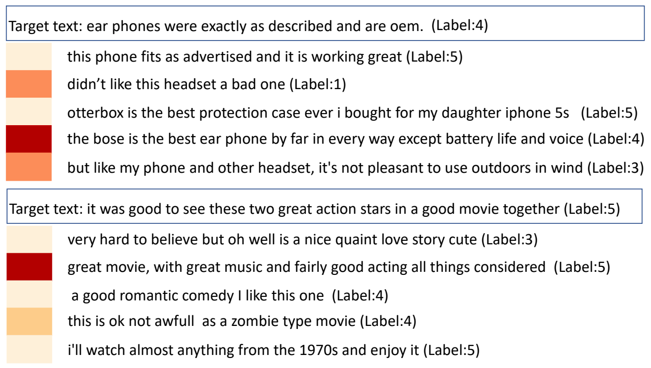
Fig. 4: Ablation Test of SentiMem Model on (a) Cell Phones and Accessories(2-class) (b) Cell Phones and Accessories(5-class) (c) Movies and TV(2-class) (d) Movies and TV(5-class) **Only** means only pretrained model. **+IMN** means adding User Interests Vector. **+TMN** means adding User Temperaments Vector. **+SentiMem** means adding both.

both users' interests and temperaments by multi-head attention, SenticNet/sentiment lexicons, and IMN/TMN.

- User interests vector can improve more than user temperaments vector does. It may be because SenticNet embedding is dimension 100 while sentiment lexicons embedding is dimension 6. Carrying more information, SenticNet can enhance the performances better.
- Adding both of them can achieve best improvements but the improvements are smaller than the sum of two individual improvements. It may come from the fact that they both utilize historic texts and there is overlapped information.

4.4 Attention Visualization in Interests Memory Network

As shown in Fig. 5, we visualize the attention weights when reading from Interests Memory Network(IMN). What below target text is the 5 texts stored in IMN when reading. Deeper color on the right of the 5 texts means higher attention weight. Note that the texts shown is a part of the origin reviews, which we choose to show the main idea of reviews. In (a), the target texts is about ear phones. The 4^{th} text has the highest weight since it is still about ear phones. The 2^{th} and 5^{th} are lighter because they are about headset which is similar to ear phone. We can know from historic texts that this user is a headset fan and is quite picky. IMN can capture this and help classify the sentiment attitudes of this user's review about headsets. In (b), we can see that 2^{th} text has the highest



Target text: ear phones were exactly as described and are oem. (Label:4)

this phone fits as advertised and it is working great (Label:5)

didn't like this headset a bad one (Label:1)

otterbox is the best protection case ever i bought for my daughter iphone 5s (Label:5)

the bose is the best ear phone by far in every way except battery life and voice (Label:4)

but like my phone and other headset, it's not pleasant to use outdoors in wind (Label:3)

Target text: it was good to see these two great action stars in a good movie together (Label:5)

very hard to believe but oh well is a nice quaint love story cute (Label:3)

great movie, with great music and fairly good acting all things considered (Label:5)

a good romantic comedy I like this one (Label:4)

this is ok not awfull as a zombie type movie (Label:4)

i'll watch almost anything from the 1970s and enjoy it (Label:5)

Fig. 5: Attention Visualization in Interests Memory Network(IMN). (a) Cell Phones and Accessories (b) Movies and TV. What below target text is the 5 texts stored in IMN when reading. Deeper color on the right of texts means higher attention weight.

weight since it is also about action movie just the same as the target text. Other texts are less concerned. IMN can capture this user's love for action movie and improve the prediction performance.

5 Related Works

5.1 Attention Mechanism

The attention mechanism has show its effectiveness in natural language processing (NLP) fields. The intuition behind it is that: let the neural network learn to pay attentions to the important parts of the word sequence like human. For example, Bert [6] and GPT [20] have achieved great improvements on multiple tasks of NLP.

5.2 Memory Network

The recurrent neural network is widely used for sequential data (in this paper, for the historic texts). However, the behavior that merging historic states into one vector has two drawbacks: (1) it loses much information and weaken the correlation between the significant records and target records (2) it is difficult to understand and explain the results with such a single vector [22]. With the better ability to store and utilize sequential data, memory network has been proposed and gained much attention [8, 26].

5.3 Sentiment Analysis

Sentiment analysis is a computational science about people's opinions/opinions/emotions toward some kind of entities. Recently, with the power of neural network, this field has developed quickly. For example, GRU, BiGRU, and transformer can all be used for sentiment classification task. As for the state-of-art works: [16] proposes a hierarchical query driven attention mechanism for financial sentiment analysis. [27] jointly extracting target-polarity pairs. [17] proposes a hierarchical end-to-end model for jointly learning text summarization and sentiment classification. [1] proposes an aspect sentiment model for aspect-based sentiment analysis focused on micro reviews. [18] utilized commonsense knowledge with an attentive LSTM for targeted aspect-based sentiment analysis. As for existing work about modeling users: [23] propose to learn a static vector to represent each user. Then, [7] utilizes a end-to-end memory network to store historic texts' vectors and read .

6 Conclusion

In this paper, we propose SentiMem to model user's characteristics including interests and temperaments, which can be helpful for sentiment classification of target text, by making use of user's historically published texts, sentiment lexicons, and SenticNet. We propose two kinds of memory networks (IMN and TMN) and utilize multi-head attention mechanism to capture the temporal dynamics of user's characteristics hidden in the historic texts. Experiments on two Amazon Review datasets show that SentiMem can explicitly improve the performances of sentiment classification over baseline methods.

References

1. Amplayo, R.K., Hwang, S.: Aspect sentiment model for micro reviews. In: 2017 IEEE International Conference on Data Mining (ICDM). pp. 727–732 (Nov 2017). https://doi.org/10.1109/ICDM.2017.83
2. Cambria, E., Fu, J., Bisio, F., Poria, S.: Affectivespace 2: Enabling affective intuition for concept-level sentiment analysis. In: Association for the Advancement of Artificial Intelligence (AAAI). pp. 508–514 (2015)
3. Cambria, E., Poria, S., Hazarika, D., Kwok, K.: Senticnet 5: discovering conceptual primitives for sentiment analysis by means of context embeddings. In: Association for the Advancement of Artificial Intelligence (AAAI) (2018)
4. Chen, X., Xu, H., Zhang, Y., Tang, J., Cao, Y., Qin, Z., Zha, H.: Sequential recommendation with user memory networks. In: ACM International Conference on Web Search and Data Mining (WSDM). pp. 108–116 (2018)
5. Cho, K., van Merrienboer, B., Bahdanau, D., Bengio, Y.: On the properties of neural machine translation: Encoder–decoder approaches. In: Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation. pp. 103–111 (2014)
6. Devlin, J., Chang, M., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. CoRR **abs/1810.04805** (2018), http://arxiv.org/abs/1810.04805
7. Dou, Z.Y.: Capturing user and product information for document level sentiment analysis with deep memory network. In: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing. pp. 521–526 (2017)
8. Graves, A., Wayne, G., Reynolds, M., Harley, T., Danihelka, I., Grabska-Barwińska, A., Colmenarejo, S.G., Grefenstette, E., Ramalho, T., Agapiou, J., et al.: Hybrid computing using a neural network with dynamic external memory. Nature p. 471 (2016)
9. Hansen, L.K., Arvidsson, A., Nielsen, F.Å., Colleoni, E., Etter, M.: Good friends, bad news-affect and virality in twitter. In: Future information technology, pp. 34–43. Springer (2011)

10. He, R., McAuley, J.: Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In: International Conference on World Wide Web (WWW). pp. 507–517 (2016)
11. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)
12. Hutto, C.J., Gilbert, E.: Vader: A parsimonious rule-based model for sentiment analysis of social media text. In: Association for the Advancement of Artificial Intelligence (AAAI) (2014)
13. Jagannatha, A.N., Yu, H.: Bidirectional rnn for medical event detection in electronic health records. In: Proceedings of NAACL-HLT. pp. 473–482 (2016)
14. Kiritchenko, S., Zhu, X., Mohammad, S.M.: Sentiment analysis of short informal texts. Journal of Artificial Intelligence Research (JAIR) **50**, 723–762 (2014)
15. Koren, Y., Bell, R.M., Volinsky, C.: Matrix factorization techniques for recommender systems. IEEE Computer **42**(8), 30–37 (2009)
16. Luo, L., Ao, X., Pan, F., Wang, J., Zhao, T., Yu, N., He, Q.: Beyond polarity: Interpretable financial sentiment analysis with hierarchical query-driven attention. In: International Joint Conferences on Artificial Intelligence (IJCAI). pp. 4244–4250 (2018)
17. Ma, S., Sun, X., Lin, J., Ren, X.: A hierarchical end-to-end model for jointly improving text summarization and sentiment classification. In: International Joint Conference on Artificial Intelligence (IJCAI). pp. 4251–4257. AAAI Press (2018)
18. Ma, Y., Peng, H., Cambria, E.: Targeted aspect-based sentiment analysis via embedding commonsense knowledge into an attentive lstm. In: Association for the Advancement of Artificial Intelligence (AAAI) (2018)
19. Mohammad, S.M.: Obtaining reliable human ratings of valence, arousal, and dominance for 20,000 english words. In: Association for Computational Linguistics (ACL). Melbourne, Australia (2018)
20. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I.: Language models are unsupervised multitask learners. OpenAI Blog **1**, 8 (2019)
21. Shin, B., Lee, T., Choi, J.D.: Lexicon integrated CNN models with attention for sentiment analysis. In: Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis (WASSA). pp. 149–158 (2017)
22. Sukhbaatar, S., Weston, J., Fergus, R., et al.: End-to-end memory networks. In: Advances in Neural Information Processing Systems (NeurIPS). pp. 2440–2448 (2015)
23. Tang, D., Qin, B., Liu, T.: Learning semantic representations of users and products for document level sentiment classification. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers). vol. 1, pp. 1014–1023 (2015)
24. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Advances in Neural Information Processing Systems (NIPS). pp. 5998–6008 (2017)
25. Wang, J., Li, J., Li, S., Kang, Y., Zhang, M., Si, L., Zhou, G.: Aspect sentiment classification with both word-level and clause-level attention networks. In: International Joint Conferences on Artificial Intelligence (IJCAI). pp. 4439–4445 (2018)
26. Weston, J., Chopra, S., Bordes, A.: Memory networks. Computing Research Repository (CoRR) (2014)
27. Zeng, Z., Song, R., Lin, P., Sakai, T.: Attitude detection for one-round conversation: Jointly extracting target-polarity pairs. In: International Conference on Web Search and Data Mining (WSDM). pp. 285–293. ACM (2019)