A Cognition Based Attention Model for Sentiment Analysis

Yunfei Long[1], Qin Lu[1], Rong Xiang[2], Minglei Li[1] and **Chu-Ren Huang[3]**
[1]Department of Computing, The Hong Kong Polytechnic University
csylong,csluqin,csmli@comp.polyu.edu.hk
[2]Advanced Micro Devices, Inc.(Shanghai)
Rong.Xiang@amd.com
[3]Department of Chinese and Bilingual Studies, The Hong Kong Polytechnic University
churen.huang@polyu.edu.hk

Abstract

Attention models are proposed in sentiment analysis because some words are more important than others. However, most existing methods either use local context based text information or user preference information. In this work, we propose a novel attention model trained by cognition grounded eye-tracking data. A reading prediction model is first built using eye-tracking data as dependent data and other features in the context as independent data. The predicted reading time is then used to build a cognition based attention (CBA) layer for neural sentiment analysis. As a comprehensive model, We can capture attentions of words in sentences as well as sentences in documents. Different attention mechanisms can also be incorporated to capture other aspects of attentions. Evaluations show the CBA based method outperforms the state-of-the-art local context based attention methods significantly. This brings insight to how cognition grounded data can be brought into NLP tasks.

1 Introduction

Sentiment analysis is critical for many applications such as sentimental product recommendation (Dong et al., 2013), public opinion detection (Pang et al., 2008), and human-machine interaction (Clavel and Callejas, 2016), etc.Sentiment analysis has been well-explored (Pang et al., 2002; Vanzo et al., 2014; Tang et al., 2015a; Chen et al., 2016; Maas et al., 2011).Recently, deep learning based methods have further elevated the performance of sentiment analysis without the need for labor intensive feature engineering.

Attention models are incorporated into sentiment analysis because not all words are created equal. Some words are more important than others in conveying the message in a sentence. Similarly, some sentences are more important than others in a document. Although the overall reading time as a cognitive process may reflect the syntax and discourse complexity, reading time of individual words is also an indicator of their semantic importance in text (Roseman, 2001; Demberg and Keller, 2008). Previous attention models are built using information embedded in text including users, products and text in local context for sentiment classification (Tang et al., 2015b; Yang et al., 2016; Chen et al., 2016; Gui et al., 2016). However, attention models using local context based text through distributional similarity lack theoretical foundation to reflect the cognitive basis. But, the key in sentiment analysis lies in its cognitive basis. Thus, we envision that cognition grounded data obtained in text reading should be helpful in building an attention model.

In this paper, we propose a novel cognition based attention(CBA) model for sentiment analysis learned from cognition grounded eye-tracking data. Eye-tracking is the process of measuring either the point of gaze or the motion of an eye relative to the head[1]. In psycho-linguistics experiments, Barrett(2016) shows that readers are less likely to fixate on close-class words that are predictable from context. Readers also fixate longer on words which play significant semantic roles (Demberg and Keller, 2008) in addition to infrequent words, ambiguous words, and morphological complex words (Rayner, 1998). Since reading time can be learned from an eye-tracking dataset, predicted reading time of words in its context can be used as indicators of attention weights.

[1]https://en.wikipedia.org/wiki/Eye-tracking

We first build a regression model to map syntax, and context features of a word to its reading time based on eye-tracking data. We then apply the model to sentiment analysis text to obtain the estimated reading time of words at the sentence level. The estimated reading time can then be used as the attention weights in its context to build the attention layer in a neural network based sentiment analysis model. Evaluation on the four sentiment analysis benchmark datasets (IMDB, Yelp 13, Yelp 14 and IMDB2) show that our proposed model can significantly improve the performance compared to the state-of-the-art attention methods.

To sum up, we have two major contributions: (1) We propose a novel cognition grounded attention model to improve the state-of-the-art neural network based sentiment analysis models by learning attention information from eye-tracking data. This is one of the first attempts to use cognition grounded data in sentiment analysis. The CBA model not only can capture attention of words at the sentence level, it can also be aggregated to work at the document level. (2) Evaluation on several real-world datasets in sentiment analysis shows that our method outperforms other state-of-the-art methods significantly. This work validates the effectiveness of cognition grounded data in building attention models.

2 Related works

The basic task in sentiment analysis can be formulated as a classification problem. Class labels can either be binary (positive/negative) or polarity either as intensity by continuous values or as ratings in certain range such as 0 to 5 or 1 to 10, etc..

In recent years, deep learning based methods have greatly improved the performance of sentiment analysis. Commonly used models include Convolutional Neural Networks (Socher et al., 2011), Recursive Neural Network (Socher et al., 2013), and Recurrent Neural Networks (Irsoy and Cardie, 2014). RNN naturally benefits sentiment classification because of its ability to capture sequential information in text. However, standard RNN suffers from the gradient vanishing problem (Bengio et al., 1994) where gradients may grow or decay exponentially over long sequences. To address this problem, Long-Short Term Memory model (LSTM) is introduced by adding a gated mechanism to keep long term memory. Each LSTM layer is generally followed by mean pooling and then feed into the next layer. Experiments in datasets which contain long documents and sentences demonstrate that the LSTM model outperforms the traditional RNN (Tang et al., 2015a,c).

Not all words contribute equally to the semantics of a sentence (Hahn and Keller, 2016). Attention based neural networks are proposed to highlight their difference in contribution (Yang et al., 2016). In document level sentiment classification, both sentence level attention and document level attention are proposed. In the sentence level attention layer, an attention mechanism identifies words that are important. Those informative words are aggregated as attention weights to form sentence embedding representation. This method is generally called local context attention method. Similarly, some sentences can also be highlighted to indicate their importance in a document.

Apart from local context attention, user/product attentions are also included in deep learning based methods either in a separate network (Gui et al., 2016) or a unified network (Tang et al., 2015c; Gui et al., 2016). Some feature engineering method to some specific datasets can also achieve very good result(Sadeghian and Sharafat, 2015). However, they are not suited for other genre of text as user-product information are not generally available.

Attention models can be built not only from local text or user/product information but also from cognitive grounded data, especially eye-tracking data (Rayner, 1998; Allopenna et al., 1998). Joshi (2014) proposes a novel metric called Sentiment Annotation Complexity for measuring sentiment annotation complexity based on eye-tracking data. Mishra (2014) presents a cognitive study of sentiment detection from the perspective of AI where readers are tested as sentiment readers. Mishra (Mishra et al., 2016b) recently proposes a model in sentiment analysis and sarcasm detection by using eye-tracking data as a feature in addition to text features using Naive-Bayes and SVM classifiers.

In other NLP tasks, Joshi (2013) shows that Word-Sense-Disambiguation can make use of simultaneous eye-tracking. Eye-tracking data are also used to measure the difficulty in translation annotation (Mishra et al., 2013). Barrett (2016) finds that gaze patterns during reading are strongly influenced by the role a word plays in terms of syntax, semantic, and discourse.

Among different available eye-tracking datasets, the Dundee corpus, GECO (the Ghent

Eye-Tracking Corpus), and Mishra et al. (Mishra et al., 2016b) are considered high-quality resources (Kennedy, 2003; Cop et al., 2016; Mishra et al., 2016b). The Dundee corpus contains eye movement data from English and French newspapers (Kennedy, 2003). Measurements were taken while 10 participants read 20 newspaper articles. GECO is an English-Dutch bilingual corpus with eye-tracking data from 17 participants collected from reading the complete novel *The Mysterious Affair at Styles*. The corpus has 4,934 sentences, 774,015 tokens, and 9,876 words. The Mishra(Mishra et al., 2016a) dataset contains 994 text snippets with 383 positive and 611 negative examples from newspaper clippings, sampled from seven native speakers.

To predict reading time using eye-tracking data, Tomanek et al. (2010) proposes a regression model using linguistic features related to syntax and semantics for calibration. Hahn (2016) proposes a novel approach to model both skipping and reading using unsupervised method which combines neural attention with auto-encoding trained on raw text using reinforcement learning.

3 Our proposed CBA model

The basic idea of our method is to add a CBA model into a neural-network based LSTM sentiment classifier. Let D be a collection of documents. A document d_k, $d_k \in D$, has m number of sentences $S_1, S_2, ...S_j, ..., S_m$. A sentence S_j is formed by a sequence of words $S_j = w_1^j w_2^j ... w_{l_j}^j$, where l_j is the length of S_j. The features of a word $w_i \in D$ form a feature vector $\vec{v}^{w_i} = [{F_1}^{w_i}, {F_2}^{w_i} .... {F_n}^{w_i}]$ where n is the feature space size. The purpose of document level sentiment classification is to project a document d_k into the target space of L class labels. Similarly, at the sentence level, the purpose is to project a sentence S_j into the target class space.

To build the CBA model, we need to first build a reading time prediction model for words within each sentence. Reading time is predicted based on word features and text features calibrated by eye-tracking data. Note that reading time from an eye-tracking dataset cannot be used directly because the text of any eye-tracking dataset is too small for sufficient coverage. Consequently, our method has four tasks: (1) to predict the reading time of words using eye-tracking data and $\vec{v}_{w_i}$ as features; (2) to build attention models based on predicted reading time at sentence level and document level; (3) to integrate attentions from other attention models; and (4) to add the attention model into the LSTM based sentiment classifier.

3.1 Modeling of reading time

To learn the reading time of words in a sentence, our method is based on regression analysis using eye-tracking data as dependent variables and context information in $\vec{v}_{w \in S_j}$ as independent variables. In the eye-tracking process, a number of different time measures such as *first fixation duration*, *gaze duration*, and *total reading time*. In this work, we only use *the total reading time*.

Since a document set is always available for sentiment analysis, we use features extracted from these documents to train the regression model. We select features based on the works from Demberg(2008) and Tomanek (2010) to include word features such as word length and POS tags as well as context level syntax and semantic features such as the total number of dominated nodes in a dependency parsing three, the maximum dependency distance, semantic category etc..

Given a word w in a sentence S_j, $w \in S_j$, and its feature vector $\vec{v}_{w \in S_j} = [F_1^w, F_2^w, ..., F_n^w]$ where n is the dimension size in feature space, the regression model on eye-tracking data is a mapping function g between reading time $t_{w \in S_j}$ and $\vec{v}_{w \in S_j}$ as defined below:

$$t_{w \in S_j} = g(\alpha_1 F_1^w + \alpha_2 F_2^w + ... + \alpha_n F_n^w + b), \quad (1)$$

where $t_{w \in S_j}$ is the predicted reading time for w, α_i is the weight of feature F_i^w, and b is a constant. Note that the set of $\alpha_i (i = 1...n)$ forms the weight vector $\vec{\alpha}_w$ for $t_{w \in S_j}$. When $\vec{v}_{w \in S_j}$ takes scalar values, g can be an identity function and thus this model becomes a typical linear regression model. When $t_{w \in S_j}$ takes discrete values, g can be a logistic function and this model becomes a typical logistic regression model.

we set g to be the identity function. The objective function then becomes:

$$\min_{\vec{\alpha}} \sum_{a_i \in \vec{\alpha}}^{n} ||t_{w \in S_j} - y_{w \in S_j}||_2^2 + \lambda R(\vec{\alpha}), \quad (2)$$

where $y_{w \in S_j}$ is the true eye-tracking values of reading time, $R(\vec{\alpha})$ is the regularization of $\vec{\alpha}$, and λ is the regularization weight. When $\lambda = 0$, the

model degrades to a linear regression function. In this work, we evaluate the use of both the linear regression model and the Ridge regression model.

3.2 Building the attention based model

Once we have predicted reading time for words used in sentences, the attention model can be built with two components. The first component works at the sentence level to give different words different emphasis in a sentence. The second component works at the document level to give different sentences different emphasis in a document.

For a sentence $S_j = w_1 w_2 ... w_i ... w_{l_j}$ with length l_j, each word w_i in S_j has a corresponding reading time t_{w_i}. Let t_{S_j} denote the total reading time of S_j. Then,

$$t_{S_j} = \sum_{i=1, w_i \in S_j}^{l_j} t_{w_i}. \tag{3}$$

For sentence level attention, the CBA weight for w_i in S_j, denoted as $A_{S_j:w_i}$, can be defined as:

$$A_{S_j:w_i} = \frac{t_{w_i}}{t_{S_j}}. \tag{4}$$

This sentence level attention model defined above gives more weights to words that have longer reading time relative to the total reading time of the sentence.

Let a document d_k, $d_k \in D$, be formed by a set of sentences $S_j = w_1 w_2 ... w_i ... w_{l_j}$. Now the CBA weight for a sentence S_j in d_k is defined as:

$$A_{d_k:S_j} = \frac{t_{s_j}}{\sum_{i=1}^{m} t_{S_i}}. \tag{5}$$

This aggregated document level attention model gives more weights to the sentences that have longer reading time relative to the total reading time of the document. Let $\vec{A}_{d_k}$ denote the document level attention weight vector. The size of $\vec{A}_{d_k}$ should be m, the number of sentences in d_k.

Let $\vec{S}_j$ denote the embedding of S_j in N dimensional space, where $S_j \in d_k$. Then, the set of sentence representations for d_k should be a matrix of size $m \times N$, denoted by $\hat{S}_{d_k}$. After the inclusion of the attention model, $\hat{S}_{d_k}$ should be:

$$\hat{S}_{d_k} = \vec{A}_{d_k} \vec{S}_j^T. \tag{6}$$

Let $\vec{d}_k$ denote the document embedding of d_k. Since $\vec{d}_k$ is an N dimensional vector, $\vec{d}_k$ can now be defined by the adjusted attention model as

$$(\vec{d}_k)_i = \sum_{j=1}^{m} (\hat{S}_{d_k})_{i,j}. \tag{7}$$

3.3 Incorporation of other attention models

Since document embedding representation allows the combined use of multiple attention mechanisms, it is to our advantage to incorporate different attention mechanisms which may help to capture different aspects of attentions. Generally speaking, different attention mechanisms can be incorporated either serially or in parallel.

In principle, any number of attention models can be included. As an an example to illustrate the capability of our proposed method, we choose one state-of-the-art local attention model(shorthanded as LA). The model is a semantic-based local attention model proposed by Yang (2016) and also used by Chen (2016). For inclusion serially, the attention weight is formulated as follows:

$$A^s{}_{S_j:}w_i = LA_{S_j:w_i} * A_{S_j:w_i}, \tag{8}$$

where $LA_{s_j:w_i}$ the sentence level attention model by the local attention model. To incorporate LA in parallel mode, the attention weight can be formulated by:

$$A^p{}_{S_j:}w_i = LA_{S_j:}w_i + A_{S_j:}w_i. \tag{9}$$

Similar methods can be used at document level.

3.4 General sentiment analysis model

We take the neural network based LSTM sentiment classifier (Gers, 2001) to be applied in both the sentence level and the document level because of its excellent performance on long sentences (Tang et al., 2015a). The basic LSTM model has five internal vectors for a node i including an input gate $\vec{i}_i$, a forget gate $\vec{f}_i$, an output gate $\vec{o}_i$, a candidate memory cell $\vec{c}'_i$, and a memory cell $\vec{c}_i$, and $\vec{i}_i \vec{f}_i$ and $\vec{o}_i$ are used to indicate which values will be updated, forget or for keeping in the LSTM model. $\vec{c}'_i$ and $\vec{c}_i$ are used to keep the candidate features and the actual accepted features, respectively.

At the sentence level, each word w_i in a sentence S_j is represented by its word embedding $\vec{w}_i$ in the N dimensional space. The LSTM cell state $\vec{c}_i$ and the hidden state $\vec{h}_{S_j:w_i}$ can be updated in two steps. In the first step, the previous hidden

state $\vec{h}_{S_j:w_{i-1}}$ uses a hyperbolic function to form $\vec{c}_i'$ as defined below.

$$\vec{c}_i' = tanh(\hat{W}_c * [\vec{h}_{S_j:w_{i-1}} * \vec{w}_i] + \hat{b}), \quad (10)$$

where $\hat{W}_c$ is a parameter matrix, $\vec{h}_{S_j:w_{i-1}}$ is the previous hidden state and $\vec{w}_i$ is the word vector. $\hat{b}$ is the regularization parameter matrix. In the second step, $\vec{c}_i$ is updated by $\vec{c}_i'$ and its previous state $\vec{c}_{i-1}$ to form $\vec{c}_i$ according to the below formula:

$$\vec{c}_i = \vec{f}_i \odot \vec{c}_{i-1} + \vec{i}_i \odot \vec{c}_i'. \quad (11)$$

The hidden state of w_i can be obtained by

$$\vec{h}_{S_j:w_i} = \vec{o}_i tanh(\vec{f}_i \odot \vec{c}_i). \quad (12)$$

The forget gate $\vec{f}_i$ is designed to keep the long term memory. A series of hidden states $\vec{h}_1\vec{h}_2...\vec{h}_i$ can serve as input to the attention layer to obtain sentence representation $\vec{S}_j$. In the document level, similar method is used to get the sentence matrix $\hat{S}$ in the document level LSTM layer to obtain the final document representation $\vec{d}_k$.

In our work, the final document representation $\vec{d}_k$ encodes both the sentence level information and the document level information. In the LSTM model, we use a hidden layer to project the final document vector $\vec{d}_k^f$ through a hyperbolic function.

$$\vec{d}_k^f = tanh(\hat{W}_h\vec{d}_k + \hat{b}_h), \quad (13)$$

where $\hat{W}_h$ is the hidden layer weight matrix and $\hat{b}_h$ is the regularization matrix.

Finally, sentiment prediction for any label $l \epsilon L$ obtained by the softmax function defined below:

$$P(y = l|\vec{d}_k^f) = \frac{e^{\vec{d}_k^{fT}\vec{W}_l}}{\sum_{l=1}^{L} e^{\vec{d}_k^{fT}\vec{W}_l}} \quad (14)$$

where $\vec{W}_l$ is the softmax weight for each label.

4 Performance evaluation

Our proposed CBA for sentiment classification is evaluated on four document sets: The first three datasets IMDB, Yelp 13, and Yelp14 which are review texts including user/product information developed by Tang (2015a). The last dataset IMDB2 is a plain text by Maas (2011). All four datasets are tokenized through the Stanford NLP tool (Manning et al., 2014).

Table 1 list the statistics of the datasets including number of classes, number of documents, and average length of sentence. We split train/development/test set in the rate of 8:1:1. The best configuration of the development dataset is used in the test set to obtain the final result.

Data	#class	#doc	#user	#pro	#len*[2]
IMDB	10	84,919	1,310	1,635	24.56
Yelp14	5	231,163	4,818	4,194	17.25
Yelp13	5	78,966	1,631	1,631	17.37
IMDB2	2	50,000	N/A	N/A	20.10

Table 1: Statistics of three benchmark datasets

Two commonly used performance evaluation metrics are used. The first one is accuracy and the second one is rooted mean square error (RMSE)[3]. Let GR_i be the golden sentiment ratings, PR_i be the predicted sentiment rating, and T be the number of documents where $GR_i = PR_i$. Accuracy is then defined by

$$Accuracy = \frac{T}{N}, \quad (15)$$

and RMSE is defined by

$$RMSE = \sqrt{\sum_{i=1}^{N}(GR_i - PR_i)^2 * \frac{1}{N}}. \quad (16)$$

We train the skip-gram word embedding (Mikolov et al., 2013) on each dataset separately to initialize the word vectors. All embedding sizes on the model are set to 200, a commonly used size.

Three sets of experiments are conducted. The first is on the selection of the regression model for reading time prediction. The second set of experiments compares our proposed CBA with another sentiment analysis method which use text only. The third set of experiments evaluates the effectiveness of combining different attention models.

4.1 Reading time prediction

The training for the regression model for reading time prediction using eye-tracking data requires the learning from text and context features as discussed in Section 3.1. We compare our regression model with more complex deep learning based regression models in each of the three eye-tracking datasets.[4]

[3]Normally accuracy is a problematic measure in highly unbalanced data sets. But in In IMDB, the largest class only takes less than 20% of all instances out of classes. The most imbalanced data are Yelp 13 whose largest class is 41% among 5 classes and second largest is about 30%. IMDB has a 50/50 split for 2-classes.

[4]Mishra et.al (Mishra et al., 2016a) only provides fixation time. So, fixation time is used when training by this set of eye-tracking data.

We take the first 90% of sentences as training data and the rest 10% as test data. The configuration that performs the best is selected and predicated on the document sentiment analysis dataset to obtain estimated reading time. Ideally, an eye-tracking corpus built from on-line reviews is more suitable for our experiments. But, we can only work with what is available.

In addition to the linear regression model(LL) and the Ridge regression model(RR), we also choose the Recurrent Neural Network (RNN) model and the Long Short Time Memory (LSTM) model for regression learning. For both models, there are two versions. The basic version inputs the extracted feature sets as word representation, labeled as RNN-1 and LSTM-1, respective. The second version takes word embedding (Pennington et al., 2014) as the initial word representation input, labeled as RNN-2 and LSTM-2, respectively. The RMSE results are listed in Table 2.

	GECO	DUNDEE	Mishra
LR	72.47	73.52	87.25
RR	69.47	70.52	84.22
RNN-1	75.47	83.52	96.23
LSTM-1	79.47	84.52	114.25
RNN-2	79.57	86.47	101.25
LSTM-2	83.88	95.88	122.27

Table 2: RMSE for reading time prediction(Unit:Milliseconds)

Note that Ridge Regression(RR) has the best performance on all the three datasets because regularization in RR reduces over-fitting problem.In three eye tracking datasets, the RR can achieve coefficient of determination[5] of 0.32, 0.30 and 0.27 in three eye tracking datasets. The features, their types and the corresponding coefficients in RR are shown in Table 3.

The more complicated deep learning models suffer from serious over-fitting problem. And the result of Deep learning model with word embedding initialization partly supports the fact that the reading time are more depend on the micro level syntax and semantic feature for the word, such as number of letters in word and complexity score of the word instead of the deep level context features.

4.2 Comparison of different sentiment classification methods

Because the features used in our model are all text based, we compare CBA with two groups of baseline methods which also only use review text for learning. Group 1 methods include commonly known linguistic and context features for SVM classifiers. Group 2 includes recent sentiment classification algorithms which are top performers using review text for training including one method that uses local attention model. Below is the list of Group1 methods.

Feature_Name	Type	Cofficient
Number of letters	Num	22.441
Start with capital letter	Bool	1.910
Capital letters only	Bool	161.580
Have alphanumeric letters	Bool	6.020
Is punctuation	Bool	-8.930
Is abbreviation	Bool	10.551
Is entity-critical word	Bool	7.612
Number of dominated nodes	Num	0.980
Max dependency distance	Num	1.982
Inverse document frequency	Num	-9.291
Number of senses in wordnet	Num	7.494
Complexity score	Num	57.240
Constant	Num	239.910

Table 3: Major features used by the Ridge Regression Model

- **Majority** — A simple majority based classifier based on sentence labels.
- **Trigram** — A SVM classifier using unigrams/bigrams/trigram as features.
- **Text feature** — A SVM classifier using word level and context level features, such as n-gram and sentiment lexicons.
- **AvgWordvec** — A SVM classifier that takes the average of word embeddings in Word2Vec as document embedding.

Here is a list of Group 2 methods:

- **SSWE** (Tang et al., 2014) — A SVM classifier using sentiment specific word embedding.
- **RNTN+RNN** (Socher et al., 2013) — A Recursive Neural Tensor Network(RNTN) to represent sentences and trained using RNN.
- **Paragraph vector** (Le and Mikolov, 2014) — A SVM classifier using document embedding as features.
- **LSTM+LA** (Chen et al., 2016) — State-of-the-art LSTM using local context as attention mechanism in both sentence level and document level.

[5] $https://en.wikipedia.org/wiki/Coefficient_of_determination$

		IMDB		Yelp13		Yelp14	
		ACC	RMSE	ACC	RMSE	ACC	RMSE
General baseline (Group 1)	Majority	0.196	2.495	0.411	1.060	0.392	1.097
	Trigram	0.399	1.783	0.569	0.814	0.577	0.804
	TextFeature	0.402	1.793	0.556	0.845	0.572	0.801
	AveWord2vec	0.304	1.985	0.526	0.898	0.531	0.893
Recently developed methods (Group 2)	SSWE+SVM	0.312	1.973	0.549	0.849	0.557	0.851
	Paragraph Vector	0.314	1.814	0.554	0.832	0.564	0.802
	RNTN+RNN	0.401	1.764	0.574	0.804	0.582	0.821
	CLSTM	0.421	1.549	0.592	0.769	0.594	0.766
	B-CLSTM	0.462	1.453	0.619	0.705	0.592	0.741
	LSTM	0.443	1.465	0.627	0.701	0.637	0.686
	LSTM+LA	0.487	1.381	0.631	0.706	0.631	0.715
CBA based models	LSTM+CBA^M	0.447	1.495	0.610	0.746	0.613	0.768
	LSTM+CBA^D	0.468	1.419	0.623	0.706	0.628	0.702
	LSTM+CBA^G	**0.489**	**1.365**	**0.638**	**0.697**	**0.641**	**0.678**

Table 4: Evaluation on sentiment classification using review text for training

- **CLSTM** (Xu et al., 2016) — Cached LSTM to capture the overall semantic information in long text. The two variations include regular **CLSTM** and bi-directional **B-CLSTM**.
- **LSTM+UPA** (Chen et al., 2016) — State-of-the-art LSTM including LA as well as user/product as attention mechanism at both sentence level and document level.

Our proposed CBA model has several variations as explained below.

- **LSTM+CBA** — The LSTM classifier using only CBA model at sentence level and document level. Based on the three eye-tracking datasets(GECO, DUNDEE and Mishra's) for reading time prediction, we label the same model by different training data as **LSTM+CBA**^{G},**LSTM+CBA**^{D} and **LSTM+CBA**^{M}.
- **LSTM+CBA+LA**^{G} — The LSTM based classifier using both the CBA model and the local text context based attention model(LA) (Chen et al., 2016). Since combining method can either be serial or in parallel, there are actually two corresponding variations: LSTM+CBA+LA$_s^G$ and LSTM+CBA+LA$_p^G$.
- **LSTM+CBA+UPA**^{G} — The same framework to **LSTM+CBA+LA**^{G} with additional user/product attention. The two corresponding variations are LSTM+CBA+UPA$_s^G$ and LSTM+CBA+UPA$_p^G$.

Table 4 shows the performance of the three groups using review text without user/product information on only the first three datasets methods in Group 1 and Group 2 do not have evaluations on IMDB2. Among all the reference methods that do not use any attention mechanism including all methods in Group 1 and Group 2(except LSTM+LA), LSTM is the best performer. LSTM+LA (2016), which is the state-of-the-art method, uses local attention mechanism to improve performance significantly. Among our CBA based variations, using the GECO dataset gives the best result outperforming LSTM+LA in all three datasets. LSTM+CBA^G has significant improvement over LSTM+LA with p values of $p < 0.016$ on IMDB, $p < 0.0019$ on Yelp 13, and $p < 0.00023$ on Yelp 14. LSTM+CBA^G has the best result compared to the other two variations because GECO has larger participant size. Its text genre is also closer to the review datasets for sentiment analysis.

In the third set of experiment, we compare our LSTM+CBA model with the combination of other attention models including the LA model and the UPA model as shown in Table 5. In the second set of experiment, since the GECO dataset gives the best performance, Table 5 shows the performance of LSTM+CBA using only the GECO dataset including LSTM+CBA^G, LSTM+CBA+LA$_s^G$, LSTM+CBA+LA$_p^G$, LSTM+CBA+UPA$_s^G$,and LSTM+CBA+UPA$_p^G$. Note that UPA is build based on user/product information. So it works

	IMDB		Yelp13		Yelp14		IMDB2	
	ACC	RMSE	ACC	RMSE	ACC	RMSE	ACC	RMSE
LSTM+LA	0.487	1.381	0.631	0.706	0.631	0.715	0.885	0.337
LSTM+CBA^G	0.489	1.365	0.638	0.697	0.641	0.678	0.894	0.332
LSTM+CBA+LA$_s^G$	0.488	1.369	0.633	0.706	0.643	0.672	0.898	0.328
LSTM+CBA+LA$_p^G$	0.492	1.362	0.639	0.696	0.639	0.675	**0.901**	**0.322**
LSTM+UPA	**0.533**	1.281	0.650	0.692	0.667	0.654	N/A	N/A
LSTM+CBA+UPA$_s^G$	0.523	**1.277**	0.654	0.693	0.664	0.645	N/A	N/A
LSTM+CBA+UPA$_p^G$	0.521	1.278	**0.655**	**0.685**	**0.668**	**0.644**	N/A	N/A

Table 5: Evaluation on sentiment classification on using dual attention

only if user/product information is available. Such data is provided in the first three sets of data.

Table 5 shows that among all three single attention models, UPA outperforms both LA and CBA in the first three datasets. This is easier to understand as UPA already included LA and it has more explicit information from users and products for its attention model compared to CBA which needs to learn hidden attention information. The combined method of CBA with UPA can still further improve performance. When CBA+UPA are combined in parallel, it has the best performance for both Yelp13 and Yelp14 (with p value of 0.027 and 0.032 respectively compare to LSTM+UPA). In the IMDB dataset, however, UPA has the best performance. This may be because user/product information is more effective in movie review IMDB dataset which is more subjective.

However, the UPA model works only if user and product information is available. Thus for IMDB2 where user/product information is not available, only CBA and LA models work and the combined use of CBA+LA gives the best performance.

4.3 Case study

A random sentence sample *'The Shelton hotel is lucky to receive 2stars from me considering ...'* is taken from the Yelp13 dataset to demonstrate the difference in the two attention mechanisms, i.e. local text(LA), and cognition-based(CBA). Figure 1 shows visually the difference in attention weights of the two models.

The attention weights of words in the LA model does not change much. CBA, on the other hand, gives higher weights to the sentiment linked word *2stars* and the verb *receive*. This two words do play significant roles as an indirect object and a main verb, respectively. This case shows that CBA does a better job in capturing micro level information in the sentence level. This support the experimental results in Table 4 and Table 5.

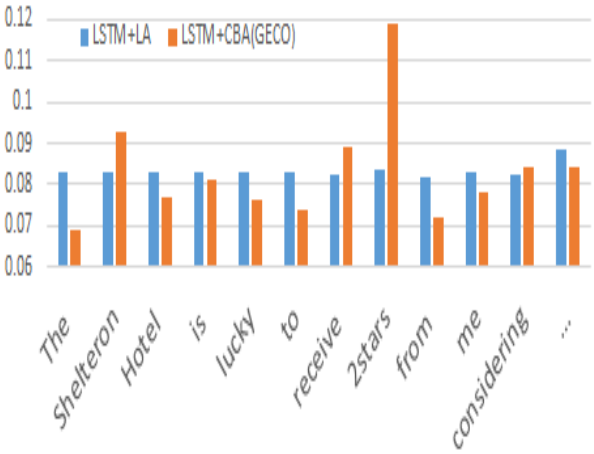

Figure 1: Case Study on attention weights

5 Conclusion and future works

In this paper, we propose a novel cognition based attention model to improve the state-of-the-art neural sentiment analysis model through cognition grounded eye-tracking data. A simple and effective regression model is used to predict reading time using both eye-tracking data and local text features. The predicted reading time is then used to build an attention layer in neural sentiment analysis models. The attention model considers both reading time and other syntactic and context features. It works in both the sentence level and the document level sentiment analysis.

Evaluation on benchmarking datasets validates the effectiveness of our method in sentiment analysis as our method clearly outperforms other state-of-the-art methods that use local context information to build their attention models. Our CBA mechanism can also be combined with other attention mechanisms to provide room for further

improvement. Future work includes using other eye-tracking information such as saccade and fixation. The incorporation of other information such as user-product information can also be explored.

Acknowledgments

The work is partially supported by the research grants from Hong Kong Polytechnic University (PolyU RTVU) and GRF grant(CERG PolyU 15211/14E).

References

Paul D Allopenna, James S Magnuson, and Michael K Tanenhaus. 1998. Tracking the time course of spoken word recognition using eye movements: Evidence for continuous mapping models. *Journal of memory and language*, 38(4):419–439.

Maria Barrett, Joachim Bingel, Frank Keller, and Anders Søgaard. 2016. Weakly supervised part-of-speech tagging using eye-tracking data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics: Short Papers*, volume 579, page 584.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166.

Huimin Chen, Maosong Sun, Cunchao Tu, Yankai Lin, and Zhiyuan Liu. 2016. Neural sentiment classification with user and product attention. EMNLP.

Chloé Clavel and Zoraida Callejas. 2016. Sentiment analysis: from opinion mining to human-agent interaction. *IEEE Transactions on affective computing*, 7(1):74–93.

Uschi Cop, Nicolas Dirix, Denis Drieghe, and Wouter Duyck. 2016. Presenting geco: An eyetracking corpus of monolingual and bilingual sentence reading. *Behavior research methods*, pages 1–14.

Vera Demberg and Frank Keller. 2008. Data from eye-tracking corpora as evidence for theories of syntactic processing complexity. *Cognition*, 109(2):193–210.

Ruihai Dong, Michael P O'Mahony, Markus Schaal, Kevin McCarthy, and Barry Smyth. 2013. Sentimental product recommendation. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 411–414. ACM.

Felix Gers. 2001. *Long short-term memory in recurrent neural networks*. Ph.D. thesis, Universität Hannover.

Lin Gui, Ruifeng Xu, Yulan He, Qin Lu, and Zhongyu Wei. 2016. Intersubjectivity and sentiment: from language to knowledge. In *Proceedings of International Joint Conference on Artificial Intelligence*, pages 2789–2795.

Michael Hahn and Frank Keller. 2016. Modeling human reading with neural attention. *arXiv preprint arXiv:1608.05604*.

Ozan Irsoy and Claire Cardie. 2014. Opinion mining with deep recurrent neural networks. In *EMNLP*, pages 720–728.

Aditya Joshi, Abhijit Mishra, Nivvedan Senthamilselvan, and Pushpak Bhattacharyya. 2014. Measuring sentiment annotation complexity of text. In *ACL (2)*, pages 36–41.

Salil Joshi, Diptesh Kanojia, and Pushpak Bhattacharyya. 2013. More than meets the eye: Study of human cognition in sense annotation. In *HLT-NAACL*, pages 733–738.

Michael Hahn Frank Keller. 2016. Modeling human reading with neural attention. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, volume 85, page 95.

A Kennedy. 2003. The dundee corpus [cd-rom]. *Psychology Department, University of Dundee*.

Quoc V Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *ICML*, volume 14, pages 1188–1196.

Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 142–150. Association for Computational Linguistics.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *ACL (System Demonstrations)*, pages 55–60.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Abhijit Mishra, Pushpak Bhattacharyya, Michael Carl, and IBC CRITT. 2013. Automatically predicting sentence translation difficulty. In *ACL (2)*, pages 346–351.

Abhijit Mishra, Aditya Joshi, and Pushpak Bhattacharyya. 2014. A cognitive study of subjectivity extraction in sentiment annotation. *ACL 2014*, page 142.

Abhijit Mishra, Diptesh Kanojia, and Pushpak Bhattacharyya. 2016a. Predicting readers' sarcasm understandability by modeling gaze behavior. In *AAAI*, pages 3747–3753.

Abhijit Mishra, Diptesh Kanojia, Seema Nagar, Kuntal Dey, and Pushpak Bhattacharyya. 2016b. Leveraging cognitive features for sentiment analysis. *CoNLL 2016*, page 156.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics.

Bo Pang, Lillian Lee, et al. 2008. Opinion mining and sentiment analysis. *Foundations and Trends® in Information Retrieval*, 2(1–2):1–135.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

Keith Rayner. 1998. Eye movements in reading and information processing: 20 years of research. *Psychological bulletin*, 124(3):372.

Ira J Roseman. 2001. A model of appraisal in the emotion system: Integrating theory, research, and applications.

Amir Sadeghian and Ali Reza Sharafat. 2015. Bag of words meets bags of popcorn.

Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 151–161. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642. Citeseer.

Duyu Tang, Bing Qin, and Ting Liu. 2015a. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1422–1432.

Duyu Tang, Bing Qin, and Ting Liu. 2015b. Learning semantic representations of users and products for document level sentiment classification. In *Proc. ACL*.

Duyu Tang, Bing Qin, and Ting Liu. 2015c. 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)*, pages 1014–1023, Beijing, China. Association for Computational Linguistics.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning sentiment-specific word embedding for twitter sentiment classification. In *ACL (1)*, pages 1555–1565.

Katrin Tomanek, Udo Hahn, Steffen Lohmann, and Jürgen Ziegler. 2010. A cognitive cost model of annotations based on eye-tracking data. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1158–1167. Association for Computational Linguistics.

Andrea Vanzo, Danilo Croce, and Roberto Basili. 2014. A context-based model for sentiment analysis in twitter. In *COLING*, pages 2345–2354.

Jiacheng Xu, Danlu Chen, Xipeng Qiu, and Xuangjing Huang. 2016. Cached long short-term memory neural networks for document-level sentiment classification. *arXiv preprint arXiv:1610.04989*.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.