Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.18576/amis/130507

Co-Extraction of Feature Sentiment and Context Terms for Context-Sensitive Feature-Based Sentiment Classification using Attentive-LSTM

S. K .Lavanya [1,*] *and B. Parvathavarthini*[2]

[1] Department of Computer Science and Engineering, Jerusalem College of Engineering, Chennai, 600100., India
[2] Department of Computer Science and Engineering , St.joseph's College of Engineering , Chennai, 600119., India

Received: 3 Dec. 2018, Revised: 12 Feb. 2019, Accepted: 23 Feb. 2019
Published online: 1 Sep. 2019

Abstract: In the field of business intelligence, the context in which customers see, hear and think about a product plays an important role in their call of buying the product. Context-sensitive sentiment classification methods determine the polarity of the sentiment terms by considering the contexts of the target word. Most of the present techniques consider only product-level, user-level contexts for sentiment classification. These contexts are more general and depend on additional features to achieve good performance. Feature-level contexts e.g., car's features include mileage and its context comprises city, highway, short-trips, long-trips and hill station providing fine-grained information needed for sentiment classification. This paper presents a neural network model referred to as Contextual Sentiment LSTM to automatically learn feature-level contexts based on an attention mechanism. Contextual Sentiment LSTM integrates background knowledge about feature, sentiment and context words from knowledge bases with the currently processed review text. A sentence vector generated based on the correlation between feature, opinions and context words in a sentence is classified using a softmax classifier. The proposed model is tested on the benchmark Car dataset and compared its contribution with progressive models like the CMLA, Sentic LSTM, CNN+LP, and RNCRF. Results demonstrate that our model achieves good sensitivity and accuracy when compared to others.

Keywords: Context-aware sentiment analysis, Sentiment Classification, Word embedding, Deep learning

1 Introduction

Online reviews of products and services encompass a wide range of aspects to which sentiments apply. Aggregating sentiments associated with features produces a fine-grained understanding of people's opinion of a particular product. Feature-based sentiment classification is a fundamental task in sentiment analysis that aims at finding the sentiment polarity associated with a specific feature, rather than the whole text [1]. In [2], the authors studied the importance of contextual information in predicting customer behavior for marketing. The field of context-aware sentiment analysis was born. Context-sensitive sentiment classification is a task in context-aware sentiment analysis that ascertains the sentiment polarity of a given target (feature) based on its context. For example, in the sentence ”Backseats are a little less comfortable for my tall teenagers”, ”tall teenagers” are the context words that need to be considered in sentiment classification for the feature ”backseat”. Most of the existing techniques [3] detect contexts only at a product-level (like author, date, or location). A few studies [4] have tried to incorporate manually-constructed contextualized features for sentiment classification. The limitation of these approaches is that they are semi-supervised and cannot deal with a large number of features.
Neural networks take words from the vocabulary of the documents as input and embed them into a lower-dimensional space (word embeddings). Word embeddings, a popular way of representing words in Natural Language processing [5], [6] have become an alternative to traditional human-labeled features. Learning word embeddings is based on the hypothesis that words appearing in similar contexts must have similar meanings [7]. Skip-gram model, one of the most

* Corresponding author e-mail: lavanya@jerusalemengg.ac.in

© 2019 NSP
Natural Sciences Publishing Cor.

popular methods of learning word embeddings, predicts context words appearing in a window around a target word. Most word embedding models rely on the statistics of co-occurrence within a local context window and fail to find context terms outside the context window. In [8], the authors optimized word embeddings to predict dependency context features. This method predicts certain context terms outside the window, based on the dependency relation. Traditional RNNs that use a stochastic gradient descent were unable to process whole input sentences. To circumvent the problem, long short-term memory network was introduced.

LSTM networks are equipped with the memory to master word interactions during the compositional process. The standard LSTM captures information only from the previous words. To address this issue, Bi-directional LSTM networks were proposed to leverage the knowledge accessible on both sides of the target word. In [9] the authors proposed a scheme for dependency parsing based on the bi-directional LSTM, which excels at representing words together with their contexts. LSTM networks struggle to pay attention to relevant information when a long sentence is given as input. To overcome this problem, attention-based LSTM networks were introduced. An attention-based LSTM takes an external memory, and representations of a sequence, as input to produce a probability distribution quantifying the concerns of the target in each position of the input sequence. In [10], the authors proposed an attentive network, the sentic LSTM, which adds common-sense knowledge concepts from the SenticNet through the recall gate to control information flow in the network, based on the target. In [11], the authors proposed a coupled multi-layer attention network for the automatic co-extraction of aspect and opinion terms. They provided an end-to-end solution by providing a couple of attention with a tensor operator in each layer of the network.

The proposed Contextual Sentiment LSTM is inspired by [11], which automatically co-extracts aspect and opinion terms. However, our method differs from [11] in the following ways: 1) Context terms for sentiment classification are added. 2) Word embeddings and dependency-based word embeddings are used as input instead of the dependency relations captured using hierarchical attentions, which takes a long computation time. 3) A set of linguistic rules for extracting context terms is framed. 4) The linguistics rules outlined for context and knowledge bases for feature and sentiment terms control information flow through the network.

The remainder of this paper is organized as follows: Section 2 discusses the deep learning techniques used for sentiment classification. Section 3 presents an overview of the proposed model. Section 4 discusses the network architecture. Section 5 reports the experimental results. Section 6 concludes the work.

2 Related Work

2.1 Feature-based Sentiment Classification

The biggest challenge in feature-based sentiment classification is to find feature-specific sentiment information from an entire sentence. Lexicon-based approaches use a sentiment lexicon or rules governing opinions, to find the sentiment orientation of a feature in a sentence. In [12], product features and opinions were jointly clustered by exploiting common sense reasoning. In [13], the authors explored sentiment features in topical words, which were expanded using word embeddings, taking into consideration the syntactic and semantic relationship between words. In [14], the author proposed a method to learn an adaptive lexicon from a static lexicon, and a mutual information-based approach was used to do so. Meta-features derived from static and adaptive lexicons were used for classification. These approaches were unable to detect sentiments for expressions with novel terms.

2.2 Sentiment Classification with Neural Networks

The success of deep learning in sentence learning has advanced sentiment analysis substantially. Classical models including the convolution neural network, recursive neural network, recurrent neural network, recursive neural tensor network, LSTM and tree LSTMs have been applied to sentiment analysis. In [15], a set of linguistic patterns was combined with a 7-layer deep convolution network to tag each word in sentences as either aspects or non-aspects. In [16], the convolution neural network was used to learn a distributed representation of each review. These representations were used as pre-trained vectors for the recurrent neural network with gated recurrent units to learn the distributed representations of users and products. In [17], a topic-aware convolution neural network and a topic-aware long short-term memory network were used to derive a general sense of words from a large corpus, and a topic-specific sense from a task-specific corpus to address the problem of the multisense nature of words. The authors [11] proposed a model for the extraction of aspect and opinion terms by combining recursive neural networks and conditional random fields. In [24], an attention-based LSTM network was proposed for aspect-based sentiment classification which explores the connection between aspects and other content terms present in the sentence using the attention mechanism. In [18], a bidirectional gated recurrent neural network was used for classification. Initially, the target of the tweets was extracted and the polarity of tweets towards the extracted target was identified. In [19], a multi-domain sentiment analysis approach was proposed to exploit the linguistic overlap between domains in order to infer the polarity for documents belonging to many domains. These methods focus on learning sentiment embeddings based on target word only without considering context words.

2.3 Context-Sensitive Sentiment Classification

A context-aware model named the Contextual Sentiment topic model was proposed in [20] to address the problem of adaptive social emotion classification from the reader's perspective. The emotional distribution of readers was differentiated from that of writers. In [21], the sentiment information of texts was encoded with the contexts of words in sentiment embeddings. A number of neural networks with tailoring loss functions were used to construct sentiment word embeddings. These methods suffer from a lack of supervision in finding context words.

2.4 Incorporating External Knowledge into Neural Networks

In [22], the authors proposed the KBLSTM, in which LSTM learns the word embeddings of concepts in the knowledge base. These embeddings are integrated with a sentential vector to assist machine reading.
Though many approaches have been proposed to improve word embeddings based on existing knowledge bases, an attempt to integrate knowledge from diverse knowledge bases to provide deep language understanding is still under research. In this work, knowledge from ConceptNet, SenticNet and user surveys are integrated with a sentential vector to perform context-sensitive sentiment classification.

3 Context-sensitive Feature-based Sentiment Classification

Contexts are words in terms of which we fully understand the situation of an object. Contexts play a major role in determining a user's decision to buy a product. Whatcar.com is a website that suggests cars based on contexts like the best (and worst) small cars for tall drivers, the top ten cars for tall people 6.3 in India, small cars for six-foot drivers, cars for big and tall UK. Inspired by this, we consider feature –level context for the sentiment classification task shown in Fig 1.
The proposed method consists of the following tasks: Stanford dependency parsing, word embeddings construction, context-feature-sentiment linking, and sentiment classification, as shown in Fig 2.
Stanford dependency parsing produces dependency relations between the words in a sentence. A skip-gram model based on [8] is used to generate target word

I'm 6'3" and fit well, don't have a whole lot of head room. Plenty of interior space and leg room for even the third row passengers. Great car for highway road trips. The gas millage is decent at 23 MPG combined city hwy. The AWD works great in the snow.

Fig. 1: Sample sentences from reviews of a car dataset. The feature terms are highlighted in blue, sentiment terms in yellow and context terms in red.

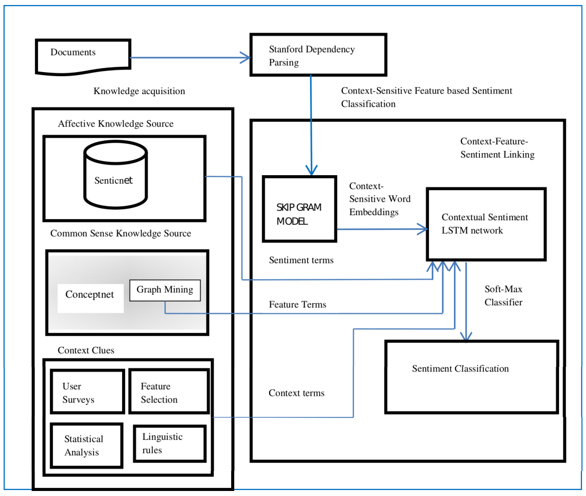


Fig. 2: The architecture of the context-sensitive feature-based sentiment classification

embeddings and context-sensitive word embeddings. It takes words, along with their dependency relations, as input and produces word embeddings and context embeddings as output. The context-feature-sentiment linking phase uses an Attentive LSTM to connect features to context and sentiment words. It learns the importance of features, contexts and sentiment terms from knowledge acquisition components. Attentive sentence vectors obtained from the Attentive LSTM are classified into positive or negative, based on the context, using the softmax classifier.

3.1 Linguistic Rules for Context Extraction

The context of a word can be identified from familiar words, called context clues, adjoining it. These clues can be obtained from user surveys, feature selection methods, and statistical analysis techniques. We propose linguistic rules for context extraction based on the patterns of words in a sentence. Let w1, w2, ... wn denote words in a sentence with the Part-of-speech (POS) NN, NNS, RBR, JJ etc and nsubj, advmod, case, amod etc denoting the dependency relation between words.

Rule 1: If {nsubj(w1,w2) $\wedge$
advmod(w1,w3)$\wedge$case(w4,w5) $\wedge$ amod(w4,w6) $\wedge$POS

(w1)=JJ ∧ POS(w2)=NN ∧ POS(w3)=RBR ∧ POS(w4)=NN ∧ POS(w5)=IN ∧POS(w6)=JJ } then mark (w6,w4) as context .
E.g. In ”This car is too small for tall person”, tall person is extracted as context.

Rule 2 : If {nsubj(w1,w2) ∧advmod(w1,w3) ∧ case(w4,w5) ∧ cc(w4,w6)∧ case(w7,w8)∧ amod(w9,w10) ∧ POS (w1)=VBP ∧ POS(w2)=NN ∧ POS(w3)=JJ ∧ POS(w4)=NN ∧ POS(w5)=IN ∧ POS(w6)=CC∧ POS(w7)=NN ∧ POS(w8)=IN ∧ POS(w9)=NN ∧ POS(w10)=JJ } then mark (w4,w7) as context .
E.g. In ”Fuel economy isnt all that great in the city but on the highway its very good”, then city and highway are extracted as context.

Rule 3 : If {nsubj(w1,w2) ∧ advmod(w1,w3) ∧ xcomp(w1,w4) ∧ case(w5,w6)∧ amod(w5,w7)∧ case(w8,w9) ∧ POS (w1)=VBP ∧ POS(w2)=NN ∧ POS(w3)=NN ∧POS(w4)=NN ∧ POS(w5)=JJ ∧ POS(w6)=IN∧ POS(w7)=CC ∧ POS(w8)=NN ∧ POS(w9)=IN ∧ POS(w10)=JJ } then mark (w10,w8) as context .
E.g.In, ”The fuel economy is quite good for such a powerful car”, powerful car is extracted as context .

Rule 4 : If {nsubj(w1,w2) ∧ dobj(w1,w3) ∧ case(w4,w5) ∧ mwe(w4,w6)∧ cc(w4,w7)∧ conj(w4,w8) ∧ case(w8,w9) ∧ amod(w8,w10) ∧ POS (w1)=VBP ∧POS(w2)=NN ∧ POS(w3)=RB ∧ POS(w4)=JJ∧ POS(w5)=NN ∧ POS(w6)=IN ∧ POS(w7)=JJ ∧ POS(w8)=NN ∧ POS(w9)=IN } then mark (w8,w9) as context .
E.g.In, ”Gas mileage is fair, but we knew it wouldn’t be great with the powerful 3.5L V-6”, powerful 3.5L V-6 is extracted as context.

Rule 5 : If {nsubj(w1,w2) ∧ amod(w3,w4) ∧ adv(w5,w6) ∧ adv(w7,w8) ∧ case(w9,w10) ∧ amod(w9,w11)∧POS (w1)=JJ∧POS(w2)=NN∧ POS(w3)=RBR ∧POS(w4)=NN∧ POS(w5)=IN ∧ POS(w6)=JJ } then mark (w4,w6) as context .
E.g.In, ”The car makes a loud whining noise when moving in reverse”, reverse is extracted as context.

These rules are used to identify the context terms present in review sentences. The contexts extracted are stored as vectors serving as prototype vectors for context extraction in the CSLSTM.

4 Network Architecture

4.1 Task Definition

Consider each sentence s_i as a sequence of words i.e, $s_i = \{ w_{i1}, w_{i2}, .. w_{in} \}$. The task of the context-sensitive feature-based sentiment classification can be divided into two subtasks. The first task is to extract all feature terms $F_k = \{ f_{i1}, f_{i2}, \ldots f_{ik} \}$, all sentiment terms $O_i = \{ o_{i1}, o_{i2}, \ldots o_{im} \}$ and all context terms $C_i = \{ c_{i1}, c_{i2}, \ldots c_{il} \}$ appearing in s_i. The second task is to classify the sentiment polarity of each feature term, based on the context terms.

For example, the sentence *“[Back seats] are [little less comfortable] for my [tall teenagers]”* contains a feature word *[Back seats]*, sentiment words *[little less comfortable]* and context words *[tall teenagers]*. The objective is to determine the sentiment and context terms, based on the feature, and to classify the sentence considering the variation in the polarity of the sentiment terms based on the context terms. The desired output for [Back seats] is [‘general’: negative; ‘with the context: [tall teenagers]’: positive].

The neural architecture consists of a sequential encoder and a couple of attention as shown in Fig 3. Given a sentence s=$\{w_1, w_2, \ldots, w_N\}$, a skip-gram model generates word embeddings $\{v_{w1}, v_{w2}, \ldots\ldots, v_{wN}\}$ and dependency-based context embeddings $\{c_{w1}, c_{w2}, \ldots, c_{wN}\}$ for input words. The sequence encoder transforms the word and context embeddings into a sequence of hidden outputs. A set of attentions is built on top of the hidden outputs: one for feature term extraction, another for sentiment term extraction and the last for context term extraction. Each attention takes as input the hidden outputs found at the position of feature, sentiment or context expressions and computes an attention vector over these words. The attention score is calculated by finding the similarity between the attention vector and vectors from knowledge bases, using a bilinear operator. Based on the attention score, important words are extracted and the sentence vector is computed by

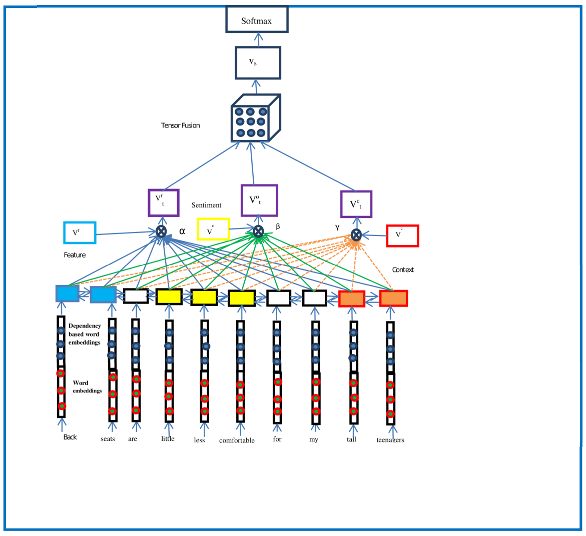
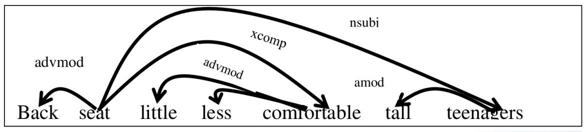


Fig. 3: Attentive neural architecture for context-sensitive sentiment classification



WORD	CONTEXTS
Seat	Back/advmod, comfortable/xcomp, teenager/nsubj
Back	seat/advmod^{-1}
comfortable	little/advmod, less/advmod, seat/xcomp^{-1}
little	comfortable/ advmod^{-1}
less	comfortable/ advmod^{-1}
tall	teenager/ amod^{-1}
teenagers	tall/amod, seat/nsubj^{-1}

Fig. 4: Dependency-based context extraction

aggregating the vectors of these words. The sentence vector is then fed into soft-max classifier that classifies the sentence into positive or negative , considering the context term.

4.2 Dependency-based Skip-Gram Model

In the standard Skip-gram model, the contexts of a word w are the words surrounding it in the local window. Using a context window of size k, the k words before and after w are the contexts produced. This model does not include contexts that are outside the window, including accidental contexts which may result in certain words being plotted as neighbors in the embedding space. In [8], the authors generalized the skip-gram model by replacing the bag-of-word contexts with arbitrary contexts obtained by considering dependency between words as shown in Fig 4. Dependency-based contexts yield functional similarities between words, which is essential for sentiment classification. For each word, context embeddings are created in addition to word embeddings. Each word w_1 of a sentence is mapped into a continuous word embedding, v_{w1} ε V^N and a context embedding, c_{w1} ε C^N . A word vector w = [v_{w1} , c_{w1}] ε R^N is built for each word by concatenating both word embeddings and context embeddings.

4.3 Proposed Contextual Sentiment LSTM Network

The word vector w, built by concatenating word embeddings and context embeddings, is given as input to the bi-directional LSTM. The LSTM cell consists of the input, output, and forget gate, which control the information flow at the current timestamp. The LSTM cell can be computed as shown in Eqs.(1-6)

$$vi_t = \sigma(W_i[h_{t-1}; w_t] + b_i) \quad (1)$$

$$f_t = \sigma(W_f[h_{t-1}; w_t] + b_f) \quad (2)$$

$$o_t = \sigma(W_o[h_{t-1}; w_t] + b_o) \quad (3)$$

$$\tilde{c}_t = \tanh(W_c[h_{t-1}; w_t] + b_c) \quad (4)$$

$$c_t = f_t * c_{t-1} + i_t * \tilde{c}_t \quad (5)$$

$$h_t = o_t * \tanh(c_t) \quad (6)$$

Where i_t, f_t and o_t are the input, forget and output gates, W_i, W_f and W_o are the weight matrices, b_i, b_f and b_o are the bias for the input, forget and output gates respectively, w_t is the word vector for the current term. The sequence of hidden outputs produced is denoted by H = $\{h_1, h_2,...,h_N\}$, where each hidden element is a concatenation of both the forward and backward hidden outputs.

4.3.1 Word Attention

The basic unit of the contextual sentiment LSTM network is a set of attention: feature, sentiment, and context as shown in Fig 3. In the feature attention, a prototype vector is used which guides the network to attend to the most important words. ConceptNet is a common sense knowledge base popularly used to discover features based on relationships between concepts like synonyms, functionalities, and hierarchies. ConceptNetNumberbatch[25]represents words and phrases as a set of semantic vectors used to compare word meanings numerically. These vectors are considered prototype vectors V^f for feature attention. The vector representation of a feature is computed as in Eq.(7):

$$v_t^f = \sum_k \alpha_k h_{t_k} \quad (7)$$

Where α is the attention vector distributed over the feature terms in the sentence. The attention phase takes the hidden output and prototype vector as input and produces an attention vector and an attention score for the current term, as shown in Eqs.(8,9):

$$e_{ft} = \tanh(v^{f^T} W_f h_t) \quad (8)$$

$$\alpha = softmax(e_{ft}) \quad (9)$$

The attention weight α and attention score e_{ft} determine how important the current word in the sentence is, with respect to the prototype vector. Similarly, sentiment and context attentions are carried out. For sentiment attention, the prototype vector is taken from SenticNet, an affective knowledge base used for polarity detection. Senticnet4 [26] uses conceptual primitives to generalize concepts such as nouns and verbs and detects sentiments even for multiword expressions that do not convey sentiment explicitly. These concepts are represented as vectors and given as input to the attentive LSTM network, as stated in [10]. The vector representation of sentiment is computed as given by Eq.(10):

$$v_t{}^o = \sum_m \beta_m h_{t_m} \tag{10}$$

Where β is the attention vector distributed over the sentiment terms in the sentence. Given the hidden output and prototype vector V^s , the attention phase produces an attention vector and an attention score for the current term, as shown in Eqs.(11,12) :

$$e_{ot} = \tanh(v^{o^T} W_o h_t) \tag{11}$$

$$\beta = softmax(e_{ot}) \tag{12}$$

The attention weight β and attention score e_{ot} determine how important the current word in the sentence is, with respect to the prototype vector. For context attention, context clues are collected from various user surveys, feature selection methods and statistical methods. Vectors are built from these clues and used as a prototype vector V^c for attention in the LSTM network. The vector representation of the context is computed as in Eq.(13):

$$v_t{}^c = \sum_l \gamma_l h_{t_l} \tag{13}$$

Where γ is the attention vector distributed over the context terms in the sentence. The attention phase produces an attention vector and an attention score for the current term, as shown in Eqs.(14,15):

$$e_{ct} = \tanh(v^{c^T} W_c h_t) \tag{14}$$

$$\gamma = softmax(e_{ct}) \tag{15}$$

The attention weight γ and attention score e_{ct} determine how important the current word in the sentence is, with respect to the prototype vector.

4.3.2 Sentence Representations

Word-level attentions result in attention vectors which provide weights to the terms in a sentence, corresponding to the prototype vectors in knowledge bases. A higher weight indicates the importance of a particular term in providing context to an input sentence. To obtain the feature representation of a sentence, attention vectors are fused using a neural tensor network, as advanced by [27]. Initially, a composition vector that encodes the correlation between feature and sentiment attention vectors is computed using a tensor, as given by Eq.(16):

$$v_{fo} = g(\begin{bmatrix} v_t{}^f \\ v_t{}^o \end{bmatrix}^T T^{[1:N]} \begin{bmatrix} v_t{}^f \\ v_t{}^o \end{bmatrix} + W \begin{bmatrix} v_t{}^f \\ v_t{}^o \end{bmatrix}) \tag{16}$$

Where W is the weight matrix and $T^{[1:N]} \varepsilon R^{K*M*N}$ a tensor that defines the multiple bilinear compositions between feature and sentiment attention vectors. The dependency relation is captured as a composition between the attention vectors. The next step is to find the correlation between the context attention vector and the composition vector, computed as given by Eq.(17):

$$v_s = g(\begin{bmatrix} v_t{}^c \\ v_{fo} \end{bmatrix}^T T^{[1:N]} \begin{bmatrix} v_t{}^c \\ v_{fo} \end{bmatrix} + W \begin{bmatrix} v_t{}^c \\ v_{fo} \end{bmatrix}) \tag{17}$$

Where v_s is the sentence vector that summarizes all the important information in a sentence.

4.3.3 Sentence Classification

The sentence vector v_s is classified into one of the gold-standard polarity classes, r, based on the probability, as given by Eq.18:

$$p_s{}^r = softmax(W^r v_s + b_s{}^r) \tag{18}$$

Where W^r and $b_s{}^r$ are the parameters that map the vector to the polarity label. The negative log-likelihood function is used to minimize the training loss of the classifier, as given by Eq.19:

$$L_s = -\sum \log p^r{}_s \tag{19}$$

5 Experiments

5.1 Datasets and System Settings:

The proposed model is evaluated on the Benchmark Car dataset [28], which comprises the date, author's name, and review sentences, alongside sentences describing the particular author's favorites with respect to a particular model. There were approximately 1030772 sentences, with 206154 sentences containing context terms, of which 173170 are classified under the positive class and the remaining 32984 under the negative class for training the network. A skip-gram model proposed by [8] was used for producing dependency-based word embeddings. Initially, sentences are tokenized and tokens that occur less than 100 times filtered. The size of the hidden layer and the dimension of word embeddings and context embeddings are set to 300. Table 1 shows the difference between context embeddings produced for three feature terms of the Car dataset by the original word2vec

Table 1: Comparison of word2vec and dependency-based word embeddings.

Target	Word2vec	Dependency	W5	W10
comfort	comforts solace comforting reassurance warmth spaciousness Comfort convenience comfy comforted	Comfort Performance quality Luxury quietness handling Roominess superior Superb Legroom	Comfort quietness quality thoughtfulness Luxury sophistication Roominess drivability performance appointments	Comfort Performance quality Luxury quietness handling Roominess superior Superb Legroom
Mileage	Mileage mpg mileage_ reimbursement ##mpg MPG gas_guzzler exhaust_ emissions fuel Honda_Insight Toyota_Prius_ hybrid	Mileage milage Mpg Advertised MPG 27 Avg Highway City guzzler estimated	Mileage milage Mpg Guzzler wow Advertised economy 28 Avg hog Consumption	Mileage milage Mpg Advertised MPG 27 Avg Highway City guzzler estimated
Interior	interior exteriors décor decor upholstery exterior furnishings Interiors cabinetry luxurious	Interior Exterior Rich Attractive Appealing Ergonomic Metalic Cabin Striking sleek	Interior inside streamlined aerodynamic impeccable interior/ exterior ergonomic build timeless sleek	Interior Exterior Rich Attractive Appealing Ergonomic Metalic Cabin Striking sleek

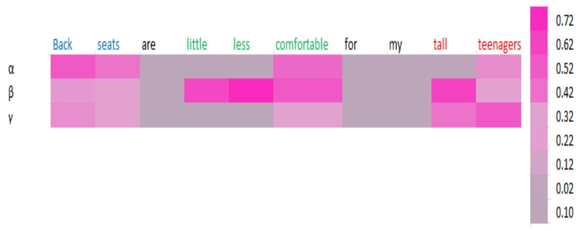


Fig. 5: Visualization of attention weights (α, β and γ) for different words in a sentence

Table 2: Comparison of Predictions between CSLSTM and CMLA

Prediction with CSLSTM	Prediction with CMLA[11]
Gas mileage is great on the highway (about 29 mpg) and is horrible in the city	Gas mileage is great on the highway (about 29 mpg) and ishorrible in the city
Air conditioner works fine on first trip	Air conditioner works fine on first trip
We are tall people (5'9 6'3") and can fit very comfortably in the front and second row seats.	We are tall people (5'9 6'3") and can fit very comfortably in the front and second row seats.
The shift changes in automatic mode are quite jerky.	The shift changes in automatic mode are quite jerky.
This car is too small for tall person	This car is too small for tall person

implementation and the modified word2vec [8] model. It also shows the difference in results, based on window size. A context window of size 5 may miss certain important contexts for the feature term *comfort* (opinion terms like *superior, superb*) while including accidental ones (like *thoughtfulness*). A window of size 10 captures better contexts than windows of sizes 5.

Context embeddings produced by the modified model capture different syntactic information. A word vector is built by concatenating word embeddings and dependency-based word embeddings produced by the Omer Levy model and given as input to the attentive CSLSTM network. The CSLSTM network combines its hidden state vector with ConceptNetNumberbatch (concept embeddings) to produce the final feature vector. Similarly, AffectiveSpaceembeddings and context embeddings are combined with hidden vectors to produce sentiment and context vectors.

5.2 Visualization of Attention

The CSLSTM network calculates the attention score to indicate the correlation between each token and its prototype vector. The higher the attention scores, the greater the correlation with feature, sentiment or context prototype. The visualization of attention scores for different words in a sentence is shown in Fig 5. Attention scores, represented in the y-axis, have a different range of values for diverse parameters like feature, sentiment or context terms. Tokens in blue correspond to feature, green to sentiment and red to context terms. It can be observed that blue tokens have large scores for the feature, green for sentiment and red for context terms. All other irrelevant tokens have lower scores. The results show that the CSLSTM model attends to all the important terms of interest. The CSLSTM model extracts all the targets based on the semantic relationships between them. To demonstrate the efficiency of the CSLSTM model, the results of a few sentences from the dataset are compared with the CMLA model, as shown in Table 2. The CMLA model was proposed to extract aspects and opinion terms in a sentence. The left column shows the prediction of the proposed model and the right column shows the prediction of the CMLA. The CMLA model does not

consider aspect terms (blue) and context terms (red) separately. It also fails to extract some opinion terms, like comfortably (green), as shown in the first sentence.
To evaluate the results of the CSLSTM model, a few baseline models are considered:
• RNCRF: It is a joint model of the CRF and the recursive neural network, proposed for extracting both aspects and opinion terms. It outperforms hand-crafted CRF models.
• CMLA: It is a model proposed for co-extracting aspect and opinion terms by modeling the relationship among tokens automatically.
• SenticLSTM: In this model, the encoder is replaced with a knowledge-embedded LSTM which filters information that does not match with the concepts in the knowledge base in order to extract aspect terms.
A comparison of the results in terms of F1 scores for feature, sentiment and context extraction is shown in Table 3. The original results of the baseline models for aspect, sentiment, and context extraction are reported. The SenticLSTM and CNN+LP models are used for aspect extraction. The SenticLSTM shows significant improvement in aspect extraction when compared to the RNCRF and CMLA because of the supervision by the knowledge base. The CNN+LP model performs better than the SenticLSTM as it incorporates linguistic patterns to extract valid aspects missed by the CNN.

Table 3: Comparison of F1 scores for feature, sentiment and context extraction

Model	Feature	Sentiment	Context
RNCRF[23]	67.06	66.90	-
CMLA[11]	70.73	73.68	-
Sentic LSTM[10]	78.18	-	-
CNN+LP [15]	85.70	-	-
CSLSTM	86.02	84.80	81.82

The RNCRF and CMLA models extract both aspects and opinion terms. The CMLA, however, outperforms the RNCRF without pre-extracted syntactic relations, as it uses multi-layer attentions with tensors to exploit the interaction between aspects and opinions. The proposed CSLSTM model outperforms baseline models as context embeddings inject semantic relations between terms which prove the usefulness of the model. To show the strength of the CSLSTM, sensitivity studies are conducted with different word embedding dimensions. From Fig 6, it can be seen that the predictions for feature, sentiment and context terms are relatively stable with different word-embedding dimensions. The highest scores for all the three parameters are achieved at the dimension 200.

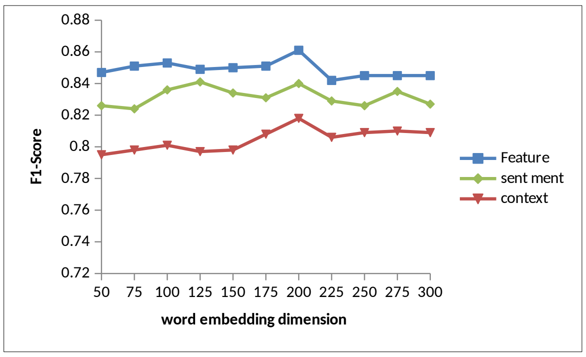


Fig. 6: Sensitivity of the dataset with the word-embedding dimension

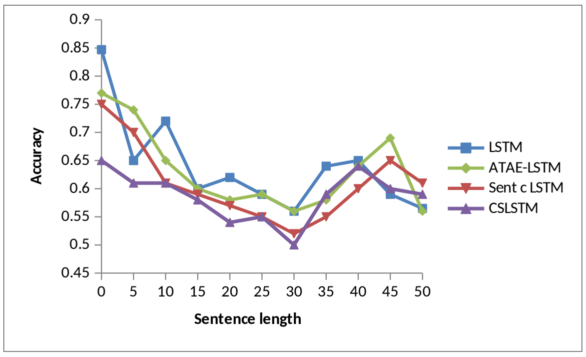


Fig. 7: Accuracy of feature-based sentiment classification

5.3 Sentiment Classification

Given a set of extracted feature, sentiment and context terms, the task is to determine the rating of each feature with respect to a particular context. Fig 7 illustrates the comparative results of sentiment classification. The standard LSTM model does not find aspect-specific polarity since no aspect-specific information is fed into the network. The attentive LSTM with aspect embedding performs better than the standard LSTM as it attends to important words in a sentence, based on aspect-specific information. The sentic LSTM produces good results compared to the standard LSTM but performs at a slightly lower level than the CSLSTM, as concepts from the knowledge base may occasionally mislead the network to attend to terms unrelated to aspects. The CSLSTM network achieves the best performance of all, as its results are based on the generation of correct attention scores using context-sensitive word embeddings.

6 Conclusion

In this paper, an attentive neural network architecture for context-sensitive feature-based sentiment classification has been proposed. Different from existing methods, the proposed architecture incorporates dependency relation to differentiate words used for expressing sentiment towards different features. The embeddings of the word are combined with dependency-based embeddings to get context-sensitive word embeddings. These context-sensitive word embeddings are good at capturing semantics. To capture the diverse sentiment information present in a sentence, the contextual sentiment LSTM network integrates knowledge from three different knowledge bases with context-sensitive word embeddings. The feature, sentiment and context attentions done at word level helps to select among different knowledge base concepts available for each word in a sentence. The sentence vector obtained by encoding feature, sentiment and context information is classified using a softmax classifier. Experimental results show that the classifiers incorporating context-sensitive information outperform the state-of-the-art models in terms of classification accuracy. In addition, different from the earlier models which depend on a single knowledge base for training the classifier, the integration of knowledge from diverse knowledge bases increases the sensitivity of the classifier.
In the future, the following research directions can be considered :(1) The mapping of text present in review sentences to the knowledge base can be extended to consider the relationship between entities in order to capture more contextual information. (2) The model can be extended to capture user-specific contextual information for providing personalized solutions. (3) An investigation about the appending of contrastive word embeddings with embeddings of the word for contrastive opinion summarization tasks can be attempted.

References

[1] E. Cambria , D. Das , S. Bandyopadhyay and A. Feraco. *A practical guide to sentiment analysis*, 1st ed., Springer International Publishing, (2017).
[2] C. Palmisano , A. Tuzhilin and M. Gorgoglione. Using context to improve predictive modeling of customers in personalization applications. *IEEE T knowl. Data Eng,* 20, 1535-1549, (2008).
[3] Y. Ren , Y. Zhang , M. Zhang and D. Ji. *Context-Sensitive Twitter Sentiment Classification Using Neural Network.* in Proc. AAAI , 215-221, (2016).
[4] Z. Teng , D.T. Vo and Y. Zhang. *Context-sensitive lexicon features for neural sentiment analysis.* in Proc. ACL-EMNLP ,1629-1638, (2016).
[5] T. Mikolov , K. Chen , G. Corrado , J. Dean. *Efficient estimation of word representations in vector space.* arXiv preprint arXiv:1301.3781, (2013).
[6] Y. Li and T. Yang. *Word Embedding for Understanding Natural Language: A Survey.* in Guide to Big Data Applications, Springer International Publishing, 83-104, (2018).
[7] S. Lai , K. Liu , S. He and J. Zhao. How to generate a good word embedding. *IEEE Intell. Syst.* 31, 5-14, (2016).
[8] O. Levy and Y. Goldberg. *Dependency-based word embeddings.* in Proc. ACL Annual Meeting, 302-308, (2014).
[9] E. Kiperwasser and Y. Goldberg. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transactions of the ACL.* 4 , 313-327, (2016).
[10] Y. Ma , H. Peng and E. Cambria. *Targeted aspect-based sentiment analysis via embedding commonsense knowledge into an attentive LSTM.* in Proc. AAAI , 5876-5883, (2018).
[11] W. Wang , S.J. Pan , D. Dahlmeier and X. Xiao. *Coupled Multi-Layer Attentions for Co-Extraction of Aspect and Opinion Terms.* in Proc. AAAI, 3316-3322 , (2018).
[12] S.K. Lavanya and B.P. Varthini. Jointly Clustering Product Features and Opinions by Exploiting Common Sense Reasoning. *J Comput Theor Nanosci.* 14 , 4480-4487, (2017).
[13] C. Liao , C. Feng , S. Yang and H. Huang. Topic-related Chinese message sentiment analysis. *Neurocomputing.* 210 ,237-246, (2016).
[14] A.S. Hosseini. Sentence-level emotion mining based on combination of adaptive Meta-level features and sentence syntactic features. *Eng Appl Artif Intell.* 65, 361-374 , (2017).
[15] S. Poria , E. Cambria and A. Gelbukh. Aspect extraction for opinion mining with a deep convolutional neural network. *Knowl Based Syst.* 108 , 42-49, (2016).
[16] T. Chen , R. Xu , Y. He , Y. Xia and X. Wang. Learning user and product distributed representations using a sequence model for sentiment analysis. *IEEE Comput. Intell. Mag.* 11 ,34-44 , (2016).
[17] R. Zhao and K. Mao. Topic-aware deep compositional models for sentence classification. *IEEE T Audio Speech.* 25, 248-260 , (2017).
[18] M. Jabreel , F. Hassan and A. Moreno. *Target-Dependent Sentiment Analysis of Tweets Using Bidirectional Gated Recurrent Neural Networks.* in Advances in Hybridization of Intelligent Methods: Models, Systems and Applications , Springer International Publishing, 39-55, (2018).
[19] M. Dragoni and G. Petrucci. A neural word embeddings approach for multi-domain sentiment analysis. *IEEE T Affect Comput.* 8 , 457-470 , (2017).
[20] Y. Rao. Contextual sentiment topic model for adaptive social emotion classification. *IEEE Intell Syst.* 31 , 41-47, (2016).
[21] D. Tang , F. Wei , B. Qin , N. Yang , T. Liu and M. Zhou. Sentiment embeddings with applications to sentiment analysis. *IEEE T knowl. Data Eng.* 28 , 496-509, (2016).
[22] B. Yang and T. Mitchell. *Leveraging knowledge bases in lstms for improving machine reading.* in Proc. ACL Annual Meeting ,1436-1446 , (2017).
[23] W. Wang , S. J. Pan , D. Dahlmeier , X. Xiao. *Recursive neural conditional random fields for aspect-based sentiment analysis.* in Proc. ACL-EMNLP, 616-626 , (2016).
[24] Y. Wang , M. Huang and L. Zhao. *Attention-based lstm for aspect-level sentiment classification.* in Proc. ACL-EMNLP, 606-615, (2016).
[25] R. Speer and J. Chin. *An ensemble method to produce high-quality word embeddings.* arXiv preprint arXiv:1604.01692.

[26] E. Cambria , S. Poria , R. Bajpai and B. Schuller. *SenticNet 4: A semantic resource for sentiment analysis based on conceptual primitives.* in Proc. ACL, 2666-2677 , (2016).
[27] R. Socher R, A. Perelygin , J. Wu , J. Chuang , C. D. Manning , A. Ng , C. Potts. *Recursive deep models for semantic compositionality over a sentiment treebank.* in Proc. ACL-EMNLP , 1631-1642 , (2013).
[28] K. Ganesan and C. Zhai. Opinion-based entity ranking. *Inf Retr.* 15 , 116-150 , (2012).

S. K. Lavanya obtained her Bachelor's degree and Master's degree in Computer Science and Engineering from Anna University, Chennai. Currently she is working as Assistant Professor in the Department of Computer Science and Engineering at Jerusalem college of Engineering, Chennai. Her research interests focus on Machine learning, Artificial Intelligence, Deep learning and Text Mining. She has presented various papers in National and International Conferences

B. Parvathavarthini received her Ph.D in the Faculty of Information and Communication Engineering, Anna University Chennai, India in 2008. She received her M.Sc and M.Phil degree in 1988 and 1989 respectively. She received M.B.A and M.E (Computer Science and Engineering) degree in 1998 and 2001 respectively. Presently she is working as Professor and Head in Computer Application Department at St. Joseph's College of Engineering, Chennai, India. Her fields of interests are Computer network security, Computational algorithm and Image processing etc. She has published research papers in national/International conferences and 21 in International/national Journals. She is a life member of ISTE, CSI and IACSIT.