Knowledge-Based Systems 108 (2016) 42–49

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Aspect extraction for opinion mining with a deep convolutional neural network

Soujanya Poria[a], Erik Cambria[b,*], Alexander Gelbukh[c]

[a] Temasek Laboratories, Nanyang Technological University, Singapore
[b] School of Computer Science and Engineering, Nanyang Technological University, Singapore
[c] CIC, Instituto Politécnico Nacional, Mexico

ARTICLE INFO

Article history:
Received 18 November 2015
Revised 2 May 2016
Accepted 8 June 2016
Available online 17 June 2016

Keywords:
Sentiment analysis
Aspect extraction
Opinion mining
CNN
RBM
DNN

ABSTRACT

In this paper, we present the first deep learning approach to aspect extraction in opinion mining. Aspect extraction is a subtask of sentiment analysis that consists in identifying opinion targets in opinionated text, i.e., in detecting the specific aspects of a product or service the opinion holder is either praising or complaining about. We used a 7-layer deep convolutional neural network to tag each word in opinionated sentences as either aspect or non-aspect word. We also developed a set of linguistic patterns for the same purpose and combined them with the neural network. The resulting ensemble classifier, coupled with a word-embedding model for sentiment analysis, allowed our approach to obtain significantly better accuracy than state-of-the-art methods.

1. Introduction

The opportunity to capture the opinion of the general public about social events, political movements, company strategies, marketing campaigns, and product preferences has raised increasing interest of both the scientific community (because of the exciting open challenges) and the business world (because of the remarkable benefits for marketing and financial market prediction). Today, sentiment analysis research has its applications in several different scenarios. There are a good number of companies, both large- and small-scale, that focus on the analysis of opinions and sentiments as part of their mission [1].

Opinion mining techniques can be used for the creation and automated upkeep of review and opinion aggregation websites, in which opinions are continuously gathered from the Web and not restricted to just product reviews, but also to broader topics such as political issues and brand perception. Sentiment analysis also has a great potential as a sub-component technology for other systems. It can enhance the capabilities of customer relationship management and recommendation systems; for example, allowing users to find out which features customers are particularly interested in or to exclude items that have received overtly negative feedback from recommendation lists. Similarly, it can be used in social communication for troll filtering and to enhance anti-spam systems. Business intelligence is also one of the main factors behind corporate interest in the field of sentiment analysis [2].

In opinion mining, different levels of analysis granularity have been proposed, each one having its own advantages and drawbacks [3]. Aspect-based opinion mining [4,5] focuses on the relations between aspects and document polarity. An aspect, also known as an opinion target, is a concept in which the opinion is expressed in the given document. For example, in the sentence, “The screen of my phone is really nice and its resolution is superb” for a phone review contains positive polarity, i.e., the author likes the phone. However, more specifically, the positive opinion is about its *screen* and *resolution*; these concepts are thus called opinion targets, or aspects, of this opinion. The task of identifying the aspects in a given opinionated text is called aspect extraction.

There are two types of aspects defined in aspect-based opinion mining: explicit aspects and implicit aspects. Explicit aspects are words in the opinionated document that explicitly denote the opinion target. For instance, in the above example, the opinion targets *screen* and *resolution* are explicitly mentioned in the text. In contrast, an implicit aspect is a concept that represents the opinion target of an opinionated document but which is not specified explicitly in the text. One can infer that the sentence, “This camera is sleek and very affordable” implicitly contains a positive opinion of the aspects *appearance* and *price* of the entity *camera*. These same aspects would be explicit in an equivalent sentence: “The appearance of this camera is sleek and its price is very affordable.”

[*] Corresponding author. Tel. +65 6790 4328.
E-mail address: cambria@ntu.edu.sg (E. Cambria).

http://dx.doi.org/10.1016/j.knosys.2016.06.009
0950-7051/© 2016 Elsevier B.V. All rights reserved.

Most of the previous works in aspect term extraction have either used conditional random fields (CRFs) [6,7] or linguistic patterns [4,8]. Both of these approaches have their own limitations: CRF is a linear model, so it needs a large number of features to work well; linguistic patterns need to be crafted by hand, and they crucially depend on the grammatical accuracy of the sentences.

In this paper, we overcome both limitations by using a convolutional neural network (CNN), a non-linear supervised classifier that can more easily fit the data. Previously, [9] used such a network to solve a range of tasks (not for aspect extraction), on which it outperformed other state-of-the-art NLP methods. In addition, we use linguistic patterns to further improve the performance of the method, though in this case the above-mentioned issues inherent in linguistic patterns affect the framework.

This paper is the first one to introduce the application of a deep learning approach to the task of aspect extraction. Our experimental results show that a deep CNN is more efficient for aspect extraction than existing approaches. We also introduced specific linguistic patterns and combined a linguistic pattern approach with a deep learning approach for the aspect extraction task.

2. Related work

Aspect extraction from opinions was first studied by Hu and Liu [4]. They introduced the distinction between explicit and implicit aspects. However, the authors only dealt with explicit aspects and used a set of rules based on statistical observations. Hu and Liu's method was later improved by Popescu and Etzioni [10] and by Blair-Goldensohn et al. [11]. Popescu and Etzioni [10] assumed the product class is known in advance. Their algorithm detects whether a noun or noun phrase is a product feature by computing the point-wise mutual information between the noun phrase and the product class.

Scaffidi et al. [12] presented a method that uses language model to identify product features. They assumed that product features are more frequent in product reviews than in a general natural language text. However, their method seems to have low precision since retrieved aspects are affected by noise. Some methods treated the aspect term extraction as sequence labeling and used CRF for that. Such methods have performed very well on the datasets even in cross-domain experiments [6,7].

Topic modeling has been widely used as a basis to perform extraction and grouping of aspects [13,14]. Two models were considered: pLSA [15] and LDA [16]. Both models introduce a latent variable "topic" between the observable variables "document" and "word" to analyze the semantic topic distribution of documents. In topic models, each document is represented as a random mixture over latent topics, where each topic is characterized by a distribution over words. Such methods have been gaining popularity in social media analysis like emerging political topic detection in Twitter [17]. The LDA model defines a Dirichlet probabilistic generative process for document-topic distribution; in each document, a latent aspect is chosen according to a multinomial distribution, controlled by a Dirichlet prior α. Then, given an aspect, a word is extracted according to another multinomial distribution, controlled by another Dirichlet prior β. Among existing works employing these models are the extraction of global aspects (such as the brand of a product) and local aspects (such as the property of a product [18]), the extraction of key phrases [19], the rating of multi-aspects [20], and the summarization of aspects and sentiments [21]. [22] employed the maximum entropy method to train a switch variable based on POS tags of words and used it to separate aspect and sentiment words.

Mcauliffe and Blei [23] added user feedback to LDA as a response-variable related to each document. Lu and Zhai [24] proposed a semi-supervised model. DF-LDA [25] also represents a semi-supervised model, which allows the user to set must-link and cannot-link constraints. A must-link constraint means that two terms must be in the same topic, while a cannot-link constraint means that two terms cannot be in the same topic. Poria et al. [26] integrated common-sense computing [27] in the calculation of word distributions in the LDA algorithm, thus enabling the shift from syntax to semantics in aspect-based sentiment analysis.

Wang et al. [28] proposed two semi-supervised models for product aspect extraction based on the use of seeding aspects. In the category of supervised methods, [29] employed seed words to guide topic models to learn topics of specific interest to a user, while [20] and [30] employed seeding words to extract related product aspects from product reviews.

On the other hand, recent approaches using deep CNNs [9,31] showed significant performance improvement over the state-of-the-art methods on a range of natural language processing (NLP) tasks. Collobert et al. [9] fed word embeddings into a CNN to solve standard NLP problems such as named entity recognition (NER), part-of-speech (POS) tagging and semantic role labeling.

3. Some background on deep CNN

A deep neural network (DNN) can be viewed as a composite of simple, unsupervised models such as restricted Boltzmann machines (RBMs), where each RBM's hidden layer serves as the visible layer for the next RBM. An RBM is a bipartite graph comprising of two layers of neurons: a visible and a hidden layer; connections between neurons in the same layer are not allowed.

To train such a multi-layer system, one needs to compute the gradient of the total energy function E with respect to weights in all the layers. To learn these weights and maximize the global energy function, the approximate maximum likelihood contrastive divergence approach can be used. This method employs each training sample to initialize the visible layer. Next, it uses the Gibbs sampling algorithm to update the hidden layer and then reconstruct the visible layer consecutively, until convergence occurs [32]. As an example, consider a logistic regression model to learn the binary hidden neurons. Each visible neuron is assumed to be a sample from a normal distribution [33]. The continuous state $\hat{h}_j$ of the hidden neuron j, with bias b_j, is a weighted sum over all continuous visible neurons $\boldsymbol{v}$:

$$\hat{h}_j = b_j + \sum_i v_i w_{ij}, \tag{1}$$

where w_{ij} is the weight of connection from the visible neuron v_i to the hidden neuron j. The binary state h_j of the hidden neuron can be defined by a sigmoid activation function:

$$h_j = \frac{1}{1 + e^{-\hat{h}_j}}. \tag{2}$$

Similarly, at the next iteration, the continuous state of each visible neuron v_i is reconstructed. Here, we determine the state of the visible neuron i, with bias c_i, as a random sample from the normal distribution where the mean is a weighted sum over all binary hidden neurons:

$$v_i = c_i + \sum_j h_i w_{ij}, \tag{3}$$

where w_{ij} is the weight of connection from the visible neuron i to the hidden one j. This continuous state is a random sample from a normal distribution $\mathcal{N}(v_i, \sigma)$, where σ is the variance of all visible neurons. Unlike hidden neurons, in a Gaussian RBM the visible ones can take continuous values.

Then, the weights are updated as the difference between the original data $\boldsymbol{v}_{data}$ and reconstructed visible layer $\boldsymbol{v}_{recon}$:

$$\triangle w_{ij} = \alpha(\langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{recon}), \tag{4}$$

where α is the learning rate and $\langle v_i h_j \rangle$ is the expected frequency with which the visible neuron i and the hidden neuron j are active together, when the visible vectors are sampled from the training set and the hidden neurons are calculated according to (1)–(3), after some k iterations.

Finally, the energy of a DNN can be determined from the final layer (the one before the output layer) as:

$$\mathrm{E} = -\sum_{i,j} v_i h_j w_{ij}. \tag{5}$$

To extend the deep neural network to a deep CNN, one simply partitions the hidden layer into Z groups. Each of the Z groups is associated with an $n_x \times n_y$ filter, where n_x is the height of the kernel and n_y is the width of the kernel. Assume that the input has dimensions $\mathrm{L_x} \times \mathrm{L_y}$, which in our case is given by L_x words in the sentence and L_y features, such as word embedding, of each word. Then the convolution will result in a hidden layer of Z groups, each of dimension $(\mathrm{L_x} - n_x + 1) \times (\mathrm{L_y} - n_y + 1)$.

The learned weights of these kernels are shared among all hidden neurons in a particular group. The energy function of the layer l is now a sum over the energy of individual blocks:

$$\mathrm{E}^l = -\sum_{z=1}^{\mathrm{Z}} \sum_{i,j}^{\substack{(\mathrm{L_x}-n_x+1),\\(\mathrm{L_y}-n_y+1)}} \sum_{r,s}^{n_x,n_y} v_{i+r-1,j+s-1} h_{ij}^z w_{rs}^l. \tag{6}$$

4. Training CNN for sequential data

We used a special training algorithm suitable for sequential data, proposed by Collobert et al. [9]. We will summarize it here, mainly following [34].

The algorithm trains the neural network by back-propagation in order to maximize the likelihood over training sentences. Consider the network parameter θ. We say that h_y is the output score for the likelihood of an input x to have the tag y. Then, the probability to assign the label y to x is calculated as

$$p(y|x,\theta) = \frac{\exp(h_y)}{\sum_j \exp(h_j)}. \tag{7}$$

Define the logadd operation as

$$\operatorname*{logadd}_i h_i = \log \sum_i \exp h_i, \tag{8}$$

then for a training example, the log-likelihood becomes

$$\log p(y|x,\theta) = h_y - \operatorname*{logadd}_i h_i. \tag{9}$$

In aspect term extraction, the terms can be organized as chunks and are also often surrounded by opinion terms. Hence, it is important to consider sentence structure on a whole in order to obtain additional clues. Let it be given that there are T tokens in a sentence and y is the tag sequence while $h_{t,\,i}$ is the network score for the t-th tag having i-th tag. We introduce $A_{i,\,j}$ transition score from moving tag i to tag j. Then, the score tag for the sentence s to have the tag path y is defined by

$$s(x,y,\theta) = \sum_{t=1}^{T} (h_{t,y_t} + A_{y_{t-1},y_t}). \tag{10}$$

This formula represents the tag path probability over all possible paths. Now, from (8) we can write the log-likelihood

$$\log p(y|x,\theta) = s(x,y,\theta) - \operatorname*{logadd}_{p,j} s(x,j,\theta). \tag{11}$$

The number of tag paths has exponential growth. However, using dynamic programming techniques, one can compute in polynomial time the score for all paths that end in a given tag [9]. Let y_t^k denote all paths that end with the tag k at the token t. Then, using recursion, we obtain

$$\delta_t(k) = \operatorname*{logadd}_{p \in y_t^k} s(x,p,\theta) = h_{t,k} + \operatorname*{logadd}_j \delta_{t-1}(j) + A_{j,k}. \tag{12}$$

For the sake of brevity, we shall not delve into details of the recursive procedure, which can be found in [9]. The next equation gives the log-add for all the paths to the token T:

$$\operatorname*{logadd}_{p,y} s(x,y,\theta) = \operatorname*{logadd}_i \delta_T(i). \tag{13}$$

Using these equations, we can maximize the likelihood of (11) over all training pairs. For inference, we need to find the best tag path using the Viterbi algorithm; e.g., we need to find the best tag path that minimizes the sentence score (10).

5. Our network architecture

The features of an aspect term depend on its surrounding words. Thus, we used a window of 5 words around each word in a sentence, i.e., ± 2 words. We formed the local features of that window and considered them to be features of the middle word. Then, the feature vector was fed to a CNN.

The network contained one input layer, two convolution layers, two max-pool layers, and a fully connected layer with softmax output. The first convolution layer consisted of 100 feature maps with filter size 2. The second convolution layer had 50 feature maps with filter size 3. The stride in each convolution layer is 1 as we wanted to tag each word. A max-pooling layer followed each convolution layer. The pool size we use in the max-pool layers was 2. We used regularization with dropout on the penultimate layer with a constraint on L2-norms of the weight vectors, with 30 epochs. The output of each convolution layer was computed using a non-linear function; in our case we used the hyperbolic tangent.

As features, we used word embeddings trained on two different corpora. We also used some additional features and rules to boost the accuracy; see Section 7. The CNN produces local features around each word in a sentence and then combines these features into a global feature vector. Since the kernel size for the two convolution layers was different, the dimensionality $L_x \times L_y$ mentioned in Section 3 was 3 × 300 and 2 × 300, respectively. The input layer was 65 × 300, where 65 was the maximum number of words in a sentence, and 300 the dimensionality of the word embeddings used, per each word.

The process was performed for each word in a sentence. Unlike traditional max-likelihood leaning scheme, we trained the system using propagation after convolving all tokens in the sentence. Namely, we stored the weights, biases, and features for each token after convolution and only back-propagated the error in order to correct them once all tokens were processed using the training scheme from in Section 4.

If a training instance s had n words, then we represented the input vector for that instance as $s_{1:n} = s_1 \bigoplus s_2 \bigoplus \ldots \bigoplus s_n$. Here, $s_i \in \mathbb{R}^k$ is a k-dimensional feature vector for the word s_i. We found that this network architecture produced good results on both of our benchmark datasets. Adding extra layers or changing the pooling size and window size did not contribute to the accuracy much but only increased computational cost.

6. Datasets used

In this section, we present the data used in our experiments.

6.1. Word embeddings

Word embeddings are distributed representations of text, which encode semantic and syntactic properties of words. Usually they

Table 1
Characteristics of the dataset developed by Qiu et al. [37] and comparison of our approach with the state of the art on it. Popescu stands for [10] and Prof-dep for [37]; P stands for precision and R for recall.

Dataset (domain)	# Reviews	# Sentences	# Aspects	Popescu		Prop-dep		CNN+LP (our)	
				P	R	P	R	P	R
Canon	45	597	79	87%	74%	90%	81%	**92.59%**	**85.02%**
Nikon	34	346	96	**86%**	80%	81%	84%	82.65%	**87.23%**
DVD	41	546	67	89%	80%	87%	81%	**90.29%**	**84.30%**
Mp3	95	1716	57	89%	74%	90%	86%	**92.75%**	**86.05%**
Cellphone	99	740	49	90%	78%	92%	86%	**92.67%**	**88.32%**

Table 2
SemEval data used for evaluation.

Domain	Training	Test	Total
Laptop	3041	800	3841
Restaurant	3045	800	3845
Total	6086	1600	7686

are dense, low-dimensional vectors. In this section, we describe two word embedding datasets that we used in our experiments.

6.1.1. Google embeddings

Mikolov et al. [35] presented two different neural network models for creating word embeddings. The models were log-linear in nature, trained on large corpora. One of them is a bag-of-words based model called CBOW; it uses word context in order to obtain the word embeddings. The other one is called skip-gram model; it predicts the word embeddings of surrounding words given the current word. Those authors made a dataset called `word2vec` publicly available. These 300-dimensional vectors were trained on a 100-billion-word corpus from Google News using the CBOW architecture.

6.1.2. Our Amazon embeddings

We trained the CBOW architecture proposed by Mikolov et al. [35] on a large Amazon product review dataset developed by McAuley and Leskovec [36]. This dataset consists of 34,686,770 reviews (4.7 billion words) of 2,441,053 Amazon products from June 1995 to March 2013. We kept the word embeddings 300-dimensional. The model is available at http://sentic.net/AmazonWE.zip.

Due to the nature of the text used to train this model, this includes opinionated/affective information, which is not present in ordinary texts such as the Google News corpus.

6.2. Evaluation corpora

For training and evaluation of the proposed approach, we used two corpora:

- Aspect-based sentiment analysis dataset developed by Qiu et al. [37]; see Table 1, and
- SemEval 2014 dataset.[1] The dataset consists of training and test sets from two domains, Laptop and Restaurant; see Table 2.

The annotations in both corpora were encoded according to IOB2, a widely used coding scheme for representing sequences. In this encoding, the first word of each chunk starts with a “B-Type” tag, “I-Type” is the continuation of the chunk and “O” is used to tag a word which is out of the chunk. In our case, we are interested to determine whether a word or chunk is an aspect, so we only have “B–A”, “I–A” and “O” tags for the words. Here is an example of IOB2 tags:

also/O excellent/O operating/B-A system/I-A ,/O size/B-A and/O weight/B-A for/O optimal/O mobility/B-A excellent/O durability/B-A of/O the/O battery/B-A the/O functions/O provided/O by/O the/O trackpad/B-A is/O unmatched/O by/O any/O other/O brand/O

7. Features and rules used

Here we present the features, the representation of the text, and linguistic rules used in our experiments.

7.1. Features

We used the following the features:

- **Word embeddings** We used the word embeddings described in Section 6.1 as features for the network. This way, each word was encoded as 300-dimensional vector, which was fed to the network.
- **Part of speech tags** Most of the aspect terms are either nouns or noun chunk. This justifies the importance of POS features. We used the POS tag of the word as its additional feature. We used 6 basic parts of speech (noun, verb, adjective, adverb, preposition, conjunction) encoded as a 6-dimensional binary vector. We used Stanford Tagger as a POS tagger.
  These two features vectors were concatenated and fed to CNN. So, for each word the final feature vector is 306 dimensional.

7.2. Linguistic patterns

In some of our experiments, we used a set of linguistic patterns (LPs) that leverage on SenticNet [38] and its extensions [39,40], a concept-level knowledge base for sentiment analysis built by means of sentic computing [41]. The five LPs used are listed below.

Rule 1 Let a noun *h* be a subject of a word *t*, which has an adverbial or adjective modifier present in a large sentiment lexicon, SenticNet. Then mark *h* as an aspect.

Rule 2 Except when the sentence has an auxiliary verb, such as *is, was, would, should, could*, etc., we apply:

  Rule 2.1 If the verb *t* is modified by an adjective or adverb or is in adverbial clause modifier relation with another token, then mark *h* as an aspect. E.g., in “The battery lasts little”, *battery* is the subject of *lasts*, which is modified by an adjective modifier *little*, so *battery* is marked as an aspect.

  Rule 2.2 If *t* has a direct object, a noun *n*, not found in SenticNet, then mark *n* an aspect, as, e.g., in “I like the lens of this camera”.

Rule 3 If a noun *h* is a complement of a couplar verb, then mark *h* as an explicit aspect. E.g., in “The camera is nice”, *camera* is marked as an aspect.

[1] http://alt.qcri.org/semeval2014/task4/index.php?id=data-and-tools

Table 3
Random features vs. Google embeddings vs. Amazon embeddings on the SemEval 2014 dataset.

Domain	Feature	F-Score
Laptop	Random	71.21%
Laptop	Google embeddings	77.32%
Laptop	Amazon embeddings	**80.68%**
Restaurant	Random	77.05%
Restaurant	Google embeddings	83.50%
Restaurant	Amazon embeddings	**85.70%**

Table 4
Feature analysis for the CNN classifier.

Domain	Features	Recall	Precision	F-Score
Laptop	WE	75.20%	86.05%	80.68%
Laptop	WE+POS	**76.31%**	**86.46%**	**81.06%**
Restaurant	WE	84.11%	87.35%	85.70%
Restaurant	WE+POS	**85.01%**	**87.42%**	**86.20%**

Rule 4 If a term marked as an aspect by the CNN or the other rules is in a noun-noun compound relationship with another word, then instead form one aspect term composed of both of them. E.g., if in “battery life”, “battery” or “life” is marked as an aspect, then the whole expression is marked as an aspect.

Rule 5 The above rules 1–4 improve recall by discovering more aspect terms. However, to improve precision, we apply some heuristics: e.g., we remove stop-words such as *of, the, a*, etc., even if they were marked as aspect terms by the CNN or the other rules.

We used the Stanford parser to determine syntactic relations in the sentences.

We combined the LPs with the CNN as follows: both LPs and CNN-based classifier are run on the text; then all terms marked by any of the two classifiers are reported as aspect terms, except for those unmarked by the last rule.

8. Experimental results

Table 1 shows that our approach outperforms the state-of-the-art methods by Popescu and Etzioni [10] and Dependency Based Propagation [37] by 5%–10%, respectively. The paired *t*-tests show that all our improvements were statistically significant at the confidence level of 95%.

Table 4 shows the accuracy of our aspect term extraction framework in laptop and restaurant domains. The framework gave better accuracy on restaurant domain reviews, because of the lower variety of aspect available terms than in laptop domain. However, in both cases recall was lower than precision. Table 4 shows improvement in terms of both precision and recall when the POS feature is used.

Pre-trained word embeddings performed better than randomized features (each word’s vector initialized randomly); see Table 3. Amazon embeddings performed better than Google `word2vec` embeddings. This supports our claim that the former contains opinion-specific information, which helped it to outperform the accuracy of Google embeddings trained on more formal texts—the Google news corpus.

Because of this, in the sequel we only show the performance using Amazon embeddings, which we denote simply as WE (word embeddings).

In both domains, CNN suffered from low recall, i.e., it missed some valid aspect terms. Linguistic analysis of the syntactic structure of the sentences substantially helped to overcome some drawbacks of machine learning-based analysis. Our experiments showed good improvement in both precision and recall when the linguistic patterns (Section 7.2) were used together with CNN; see Table 5.

Table 5
Impact of linguistic patterns on the SemEval 2014 dataset.

Domain	Classifiers	Recall	Precision	F-Score
Laptop	LP	62.39%	57.20%	59.68%
Laptop	CNN	76.31%	86.46%	81.06%
Laptop	CNN+LP	**78.35%**	**86.72%**	**82.32%**
Restaurant	LP	65.41%	60.50%	62.86%
Restaurant	CNN	85.01%	87.42%	86.20%
Restaurant	CNN+LP	**86.10%**	**88.27%**	**87.17%**

Table 6
Comparison with the state of the art. ZW stands for [7]; LP stands for linguistic patterns.

Domain	Framework	Recall	Precision	F-Score
Laptop	ZW	66.51%	84.80%	74.55%
Laptop	CNN+LP (our)	**78.35%**	**86.72%**	**82.32%**
Restaurant	ZW	82.72%	85.35%	84.01%
Restaurant	CNN+LP (our)	**86.10%**	**88.27%**	**87.17%**

Table 7
Impact of the POS feature on the dataset by Qiu et al. [37].

Domain	Classifiers	Precision	Recall	F-Score
Canon	WE	82.74%	75.15%	78.76%
Canon	WE+POS	**85.42%**	**77.21%**	**81.10%**
Nikon	WE	73.19%	79.27%	76.10%
Nikon	WE+POS	**77.65%**	**82.30%**	**79.90%**
DVD	WE	84.41%	77.26%	80.67%
DVD	WE+POS	**85.48%**	**79.25%**	**82.24%**
Mp3	WE	87.35%	81.23%	84.17%
Mp3	WE+POS	**89.40%**	**83.77%**	**86.49%**
Cellphone	WE	86.01%	81.32%	83.59%
Cellphone	WE+POS	**90.15%**	**83.47%**	**86.68%**

As to the linguistic patterns, the removal of stop-words, Rule 1, and Rule 3 were most beneficial. Fig. 1 shows a visualization for Table 5.

Table 6 and Fig. 2 shows the comparison between the proposed method and the state of the art on the SemEval dataset.

One can see that about 36.55% aspect terms present in the laptop domain corpus are phrase and restaurant corpus consists of 24.56% aspect terms. The performance of detecting aspect phrases are lower than single word aspect tokens in both domains. This shows that the sequential tagging is indeed a tough task to do. Lack of sufficient training data for aspect phrases is also one of the reasons to get lower accuracy in this case. In particular, we got 79.20% and 83.55% F-score to detect aspect phrases in laptop and restaurant domain respectively. We observed some cases where only 1 term in an aspect phrase is detected as aspect term. In those cases Rule 4 of the linguistic patterns helped to correctly detect the aspect phrases. We also carried out experiments on the aspect dataset originally developed by Qiu et al. [37]. This is to date the largest comprehensive aspect-based sentiment analysis dataset. Table 1, left part, shows the details of this dataset.

The best accuracy on this dataset was obtained when word embedding features were used together with the POS features. This shows that while the word embedding features are most useful, the POS feature also plays a major role in aspect extraction (Table 7).

As on the SemEval dataset, linguistic patterns together with CNN increased the overall accuracy. However, linguistic patterns have performed much better on this dataset than on the SemEval dataset. This supports the observation made previously [37] that on this dataset linguistic patterns are more useful. One of the

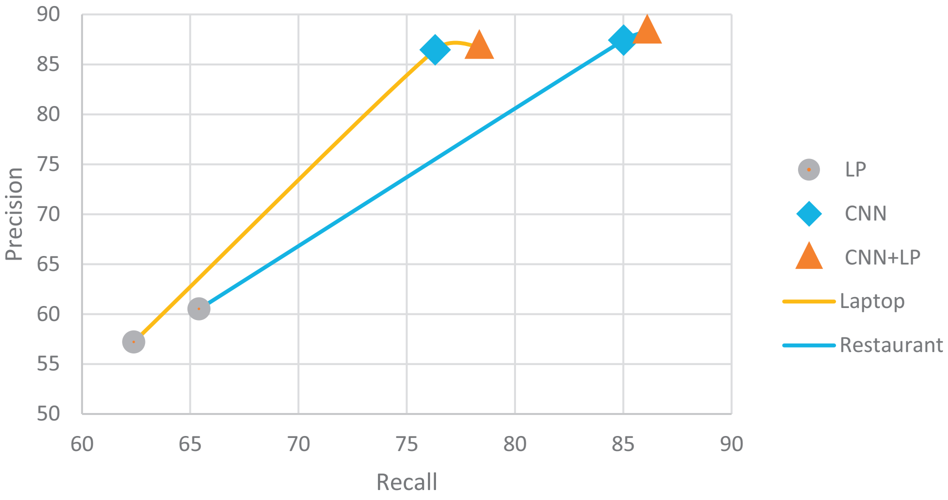

Fig. 1. Comparison of the performance of CNN, CNN-LP and LP.

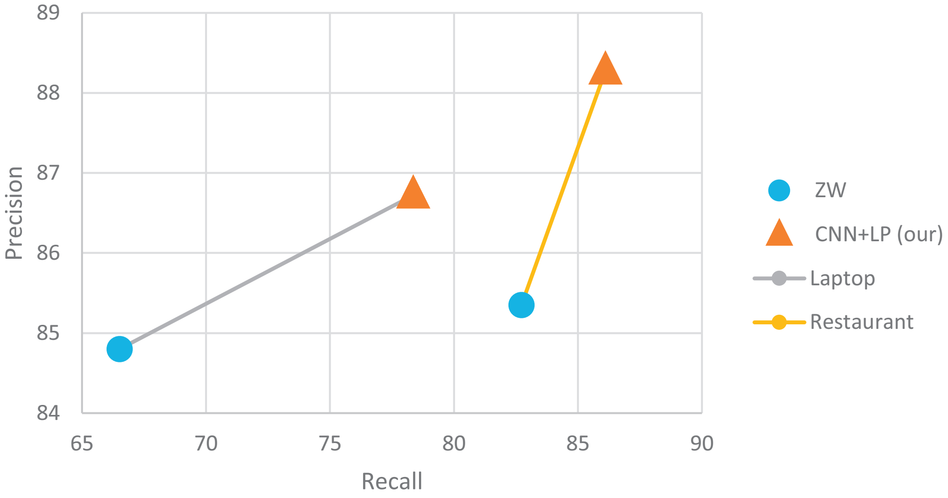

Fig. 2. Comparison of the performance with the state of the art.

Table 8
Impact of linguistic patterns on the dataset by Qiu et al. [37].

Domain	Classifiers	Precision	Recall	F-Score
Canon	CNN	85.42%	77.21%	81.10%
Canon	CNN+LP	**92.59%**	**85.02%**	**88.64%**
Nikon	CNN	77.65%	82.30%	79.90%
Nikon	CNN+LP	**82.65%**	**87.23%**	**84.87%**
DVD	CNN	85.48%	79.25%	82.24%
DVD	CNN+LP	**90.29%**	**84.30%**	**87.19%**
Mp3	CNN	89.40%	83.77%	86.49%
Mp3	CNN+LP	**92.75%**	**86.05%**	**89.27%**
Cellphone	CNN	90.15%	83.47%	86.68%
Cellphone	CNN+LP	**92.67%**	**88.32%**	**90.44%**

possible reasons for this is that most of the sentences in this dataset are grammatically correct and contain only one aspect term. Here we combined the linguistic patterns and a CNN to achieve even better results than the approach of by Qiu et al. [37] based only on linguistic patterns. Our experimental results showed that this ensemble algorithm (CNN+LP) can better understand the semantics of the text than [37]'s pure LP-based algorithm, and thus extracts more salient aspect terms. Table 8 and Fig. 3 shows the performance and comparisons of different frameworks.

Fig. 4 compares the proposed method with the state of the art.

We believe that there are two key reasons for our framework to outperform state-of-the-art approaches. First, a deep CNN, which is non-linear in nature, better fits the data than linear models such as CRF. Second, the pre-trained word embedding features help our framework to outperform state-of-the-art methods that do not use word embeddings. The main advantage of our framework is that it does not need any feature engineering. This minimizes development cost and time.

9. Conclusion

We have introduced the first deep learning-based approach to aspect extraction. As expected, this approach gave a significant improvement in performance over state-of-the-art approaches. We proposed a specific deep CNN architecture that comprises seven layers: the input layer, consisting of word embedding features for each word in the sentence; two convolution layers, each followed by a max-pooling layer; a fully connected layer; and, finally, the output layer, which contained one neuron per each word.

We also developed a set of heuristic linguistic patterns and integrated them with the deep learning classifier. In the future, we plan to extend and refine these patterns.

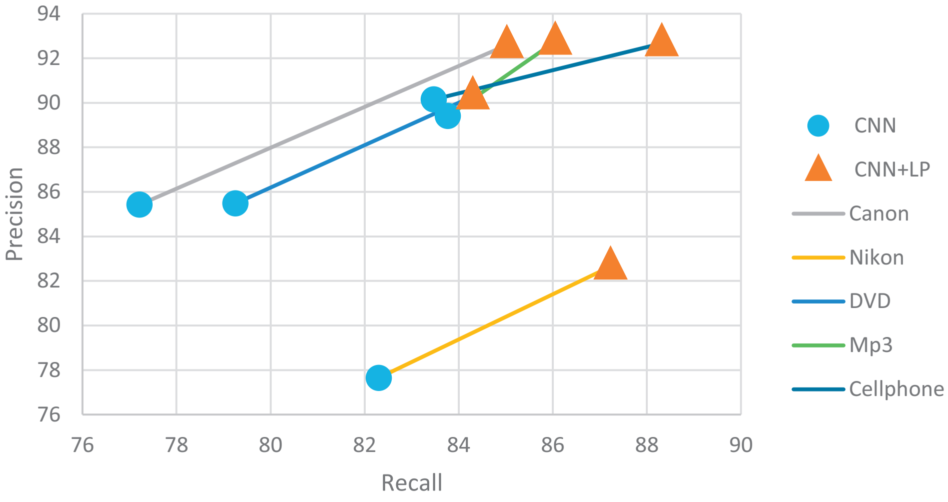

Fig. 3. Comparison of the performance of CNN, CNN-LP and LP.

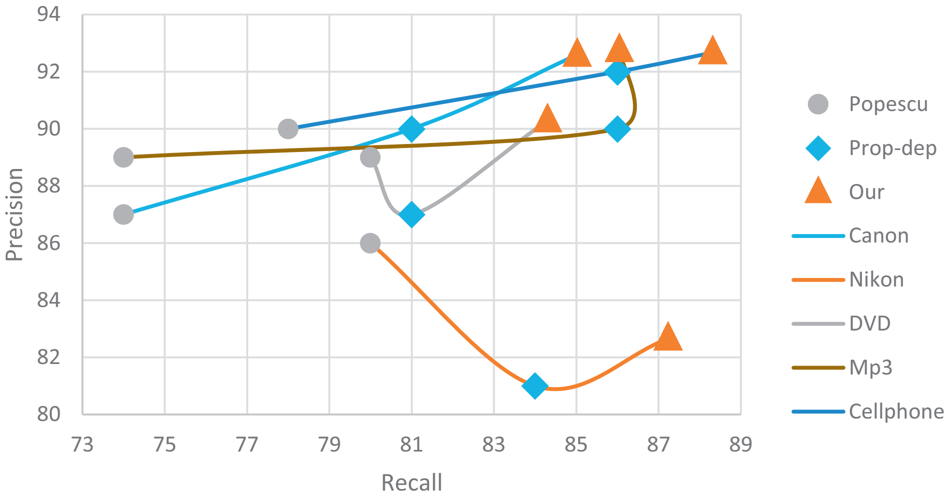

Fig. 4. Comparison of the performance with the state of the art on Bing Liu dataset.

References

[1] E. Cambria, Affective computing and sentiment analysis, IEEE Intell. Syst. 31 (2) (2016) 102–107.

[2] E. Cambria, H. Wang, B. White, Guest editorial: Big social data analysis, Knowl.-Based Syst. 69 (2014a) 1–2.

[3] E. Cambria, B. Schuller, B. Liu, H. Wang, C. Havasi, Statistical approaches to concept-level sentiment analysis, IEEE Intell. Syst. 28 (3) (2013) 6–9.

[4] M. Hu, B. Liu, Mining and summarizing customer reviews, in: Proceedings of ACM SIGKDD Conference on Knowledge Discovery & Data Mining. Seattle, 2004, pp. 168–177.

[5] X. Ding, B. Liu, P.S. Yu, A holistic lexicon-based approach to opinion mining, in: Proceedings of First ACM International Conference on Web Search and Data Mining (WSDM-2008), Stanford University, Stanford, California, USA, 2008, pp. 231–240.

[6] N. Jakob, I. Gurevych, Extracting opinion targets in a single- and cross-domain setting with conditional random fields, in: Proceedings of EMNLP-2010, ACL, 2010, pp. 1035–1045.

[7] T. Zhiqiang, W. Wenting, DLIREC: Aspect term extraction and term polarity classification system, in: Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014), 2014, pp. 235–240.

[8] S. Poria, E. Cambria, A. Gelbukh, F. Bisio, A. Hussain, Sentiment data flow analysis by means of dynamic linguistic patterns, IEEE Comput. Intell. Mag. 10 (4) (2015a) 26–36.

[9] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, P. Kuksa, Natural language processing (almost) from scratch, J. Mach. Learn. Res. 12 (2011) 2493–2537.

[10] A.-M. Popescu, O. Etzioni, Extracting product features and opinions from reviews, in: Proceedings of EMNLP-2005, 2005, pp. 3–28.

[11] S. Blair-Goldensohn, K. Hannan, R. McDonald, T. Neylon, G.A. Reis, J. Reynar, Building a sentiment summarizer for local service reviews, in: Proceedings of WWW-2008 workshop on NLP in the Information Explosion Era, 2008, pp. 14–23.

[12] C. Scaffidi, K. Bierhoff, E. Chang, M. Felker, H. Ng, C. Jin, Red Opal: Product-feature scoring from reviews, in: Proceedings of the 8th ACM Conference on Electronic Commerce, ACM, 2007, pp. 182–191.

[13] Y. Hu, J. Boyd-Graber, B. Satinoff, A. Smith, Interactive topic modeling, Mach. Learn. 95 (3) (2014) 423–469.

[14] Z. Chen, B. Liu, Mining topics in documents: standing on the shoulders of big data, in: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2014, pp. 1116–1125.

[15] T. Hofmann, Probabilistic latent semantic indexing, in: Proceedings of 22nd ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, 1999, pp. 50–57.

[16] D.M. Blei, A.Y. Ng, M.I. Jordan, Latent Dirichlet allocation, J. Mach. Learn. Res. 3 (2003) 993–1022.

[17] S. Rill, D. Reinel, J. Scheidt, R. Zicari, Politwi: Early detection of emerging political topics on twitter and the impact on concept-level sentiment analysis, Knowl.-Based Syst. 69 (2014) 14–23.

[18] I. Titov, R. McDonald, Modeling online reviews with multi-grain topic models, in: Proceedings of 17th Conference on World Wide Web, ACM, 2008, pp. 111–120.

[19] S. Branavan, H. Chen, J. Eisenstein, R. Barzilay, Learning document-level semantic properties from free-text annotations, J. Artif. Intell. Res. 34 (2) (2009) 569.

[20] H. Wang, Y. Lu, C. Zhai, Latent aspect rating analysis on review text data: a rating regression approach, in: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2010, pp. 783–792.

[21] Y. Lu, C. Zhai, N. Sundaresan, Rated aspect summarization of short comments, in: Proceedings of 18th World Wide Web Conference, ACM, 2009, pp. 131–140.

[22] W.X. Zhao, J. Jiang, H. Yan, X. Li, Jointly modeling aspects and opinions with a MaxEnt-LDA hybrid, in: Proceedings of the 2010 Conference on Empirical

Methods in Natural Language Processing, Association for Computational Linguistics, 2010, pp. 56–65.

[23] J.D. Mcauliffe, D.M. Blei, Supervised topic models, in: Advances in Neural Information Processing Systems, 2008, pp. 121–128.

[24] Y. Lu, C. Zhai, Opinion integration through semi-supervised topic modeling, in: Proceedings of the 17th International Conference on World Wide Web, ACM, 2008, pp. 121–130.

[25] D. Andrzejewski, X. Zhu, M. Craven, Incorporating domain knowledge into topic modeling via Dirichlet forest priors, in: Proceedings of the 26th Annual International Conference on Machine Learning, ACM, 2009, pp. 25–32.

[26] S. Poria, I. Chaturvedi, E. Cambria, F. Bisio, Sentic LDA: Improving on LDA with semantic similarity for aspect-based sentiment analysis, IJCNN, 2016.

[27] E. Cambria, A. Hussain, C. Havasi, C. Eckl, Common sense computing: From the society of mind to digital intuition and beyond, in: J. Fierrez, J. Ortega, A. Esposito, A. Drygajlo, M. Faundez-Zanuy (Eds.), Biometric ID Management and Multimodal Communication, Lecture Notes in Computer Science, 5707, Springer, Berlin Heidelberg, 2009, pp. 252–259.

[28] T. Wang, Y. Cai, H.-f. Leung, R.Y. Lau, Q. Li, H. Min, Product aspect extraction supervised with online domain knowledge, Knowl.-Based Syst. 71 (2014) 86–100.

[29] J. Jagarlamudi, H. Daumé III, R. Udupa, Incorporating lexical priors into topic models, in: Proceedings of the 13th EACL Conference, Association for Computational Linguistics, 2012, pp. 204–213.

[30] A. Mukherjee, B. Liu, Aspect extraction through semi-supervised modeling, in: Proceedings of 50th Annual Meeting of the ACL: Long Papers, Volume 1, ACL, 2012, pp. 339–348.

[31] S. Poria, E. Cambria, A. Gelbukh, Deep convolutional neural network textual features and multiple kernel learning for utterance-level multimodal sentiment analysis, in: EMNLP, 2015b, pp. 2539–2544.

[32] G.E. Hinton, Training products of experts by minimizing contrastive divergence, Neural Comput. 14 (8) (2002) 1771–1800.

[33] G.W. Taylor, G.E. Hinton, S.T. Roweis, Modeling human motion using binary latent variables, in: Advances in Neural Information Processing Systems, 19, MIT Press, 2007, pp. 1345–1352.

[34] E.R. Fonseca, J.L.G. Rosa, A two-step convolutional neural network approach for semantic role labeling, in: The 2013 International Joint Conference on Neural Networks (IJCNN), IEEE, Dallas, TX, 2013, pp. 1–7.

[35] T. Mikolov, W.-t. Yih, G. Zweig, Linguistic regularities in continuous space word representations., in: HLT-NAACL, 2013, pp. 746–751.

[36] J. McAuley, J. Leskovec, Hidden factors and hidden topics: Understanding rating dimensions with review text, in: Proceedings of RecSys'13. Hong Kong, China, 2013.

[37] G. Qiu, B. Liu, J. Bu, C. Chen, Opinion word expansion and target extraction through double propagation, Comput. Linguist. 37 (1) (2011) 9–27.

[38] E. Cambria, D. Olsher, D. Rajagopal, SenticNet 3: a common and common-sense knowledge base for cognition-driven sentiment analysis, in: AAAI. Quebec City, 2014b, pp. 1515–1521.

[39] S. Poria, A. Gelbukh, E. Cambria, P. Yang, A. Hussain, T. Durrani, Merging senticnet and wordnet-affect emotion lists for sentiment analysis, in: Signal Processing (ICSP), 2012 IEEE 11th International Conference on, vol. 2, IEEE, 2012a, pp. 1251–1255.

[40] S. Poria, A. Gelbukh, E. Cambria, D. Das, S. Bandyopadhyay, Enriching SenticNet polarity scores through semi-supervised fuzzy clustering, in: IEEE ICDM. Brussels, 2012b, pp. 709–716.

[41] E. Cambria, A. Hussain, Sentic Computing: A Common-Sense-Based Framework for Concept-Level Sentiment Analysis, Springer, Cham, Switzerland, 2015.