Product Recommendation using Neural Network based Opinion Mining Framework

Raheesa K K[1] Vijesh K[2]
[1,2]Department of Computer Science & Engineering
[1,2]Cochin Engineering College Valanchery, India

Abstract— The opinion mining models are applied to extract customer interest on the products. The product recommendation applications are built to analyze the product reviews submitted by the customers. Product reviews are valuable for upcoming buyers in helping them make decisions. To this end, different opinion mining techniques are applied judging a review sentence's orientation is one of their key challenges. The deep learning has emerged as an effective means for solving sentiment classification problems. A neural network intrinsically learns a useful representation automatically without human efforts. The success of deep learning highly relies on the availability of large-scale training data. The deep learning framework is constructed for product review sentiment classification which employs prevalently available ratings as weak supervision signals. The framework consists of two steps: learning a high level representation captures the general sentiment distribution of sentences through rating information and adding a classification layer on top of the embedding layer and use labeled sentences for supervised fine-tuning. Two kinds of low level network structure are explored for modeling review sentences, namely, convolution feature extractors and long short-term memory. The recommendation process increases the accuracy level with minimum computational overhead.

Keywords: Deep Learning, Opinion Mining, Sentiment Classification, Neural Networks and Product Recommendation

I. INTRODUCTION

Online user-generated reviews are of great practical use. They have become an inevitable part of decision making process of consumers on product purchases, hotel bookings, etc. They collectively form a low-cost and efficient feedback channel, which helps businesses to keep track of their reputations and to improve the quality of their products and services [1]. As a matter of fact, online reviews are constantly growing in quantity, while varying largely in content quality. To support users in digesting the huge amount of raw review data, many sentiment analysis techniques have been developed for past years.

The sentiments and opinions can be analyzed at different levels of granularity. We call the sentiment expressed in a whole piece of text, e.g., review document or sentence, overall sentiment. The task of analyzing overall sentiments of texts is typically formulated as classification problem, e.g., classifying a review document into positive or negative sentiment. Then, a variety of machine learning methods trained using different types of indicators have been employed for overall sentiment analysis. Analyzing the overall sentiment expressed in a whole piece of text alone, does not discover what specifically people like or dislike in the text. In reality, the finegrained sentiments may very well tip the balance in purchase decisions. For example, savvy consumers nowadays are no longer satisfied with just overall sentiment/rating given to a product in a review; They are often eager to see why it receives that rating positive or negative attributes contribute to the particular rating of the product.

In aspect level sentiment analysis where an aspect means a unique semantic facet of an entity commented on in text documents, and is typically represented as a high-level hidden cluster of semantically related keywords. Aspect-based sentiment analysis generally consists of two major tasks, one is to detect hidden semantic aspect from given texts, the other is to identify finegrained sentiments expressed towards the aspects. Probabilistic topic models, which are typically built on a basic latent Dirichlet allocation (LDA) model have been used for aspect-based sentiment analysis, where the semantic aspect can be naturally formulated as one type of latent topics.

The probabilistic joint topic-sentiment models are unsupervised or weakly/partially supervised, meaning that they primarily model user-generated text content, and have not considered overall ratings or labels of the text documents in their frameworks. As a result, though they may capture the hidden thematic structure of text data, the models cannot directly predict the overall sentiments or ratings of text documents, instead, they only rely on document-specific sentiment distribution to approximate the overall sentiments of documents.

II. RELATED WORK

Feature and aspect extraction is a core component of aspect-based opinion mining systems. Zhang and Liu provide a broad overview of the tasks and the current state-of-the-art techniques. Feature identification has been explored in many studies. Most methods focus on explicit features, including unsupervised methods that utilize association rules, dependency relations, or statistical associations between features and opinion words, and supervised ones that treat it as a sequence labeling problem and apply Hidden Markov Model (HMM) or Conditional Random Fields (CRF) to it. A few methods have been proposed to identify implicit features, e.g., using co-occurrence associations between implicit and explicit features, or leveraging lexical relations of words in dictionaries. Many of these techniques require seed terms, handcrafted rules/patterns, or other annotation efforts.

The features have been extracted beforehand or can be extracted from semi-structured Pros and Cons reviews. Methods including similarity matching, topic modeling, Expectation-Maximization (EM) based semi supervised learning and synonym clustering have been explored in this context. To extract features and aspects at the same time, topic model-based approaches have been explored by a large number of studies in recent years. Standard topic modeling methods such as pLSA and LDA are extended to suit the peculiarities of the problem, e.g., capturing local topics corresponding to ratable aspects, jointly extracting both topic/aspect and sentiment, incorporating prior knowledge to generate coherent aspects, etc.

The clustering method utilizes the mutual reinforcement associations between features and opinion words. It employs standard clustering algorithms such as k-means to iteratively group feature words and opinion words separately. The similarity between two feature words is determined by a linear combination of their intrasimilarity and inter-similarity. Intra-similarity is the traditional similarity, and inter-similarity is calculated based on the degree of association between feature words and opinion words. To calculate the inter-similarity, a feature word is represented as a vector where each element is the co-occurrence frequency between that word and opinion words in sentences. Then the similarity between two items is calculated by cosine similarity of two vectors. In each iteration, the clustering results of one type of data items are used to update the pairwise similarity of the other type of items [4]. After clustering, the strongest links between features and opinion words form the aspect groups. Mauge et al. first trained a maximum-entropy classifier to predict the probability that two feature expressions are synonyms, then construct a graph based on the prediction results and employ greedy agglomerative clustering to partition the graph to clusters. Bancken et al. used k-medoids clustering algorithm with a WordNet-based similarity metric to cluster semantically similar aspect mentions.

These clustering methods take two steps. In the first step, features are extracted based on association rules or dependency patterns and in the second step features are grouped into aspects using clustering algorithms. In contrast, our method extracts features and groups them at the same time. Moreover, most of these methods extract and group only explicit features, while our method deals with both explicit and implicit features. The method also handles implicit features, but their similarity measure largely depends on co-occurrence between features and opinion words may not be efficient in identifying features that are semantically similar but rarely co-occur in reviews.

III. Weakly-supervised Product Review Sentiment Analysis

People are getting used to consuming online and writing comments about their purchase experiences on merchant/review Websites. These opinionated contents are valuable resources both to future customers for decision-making and to merchants for improving their products and/or service. The volume of reviews grows rapidly, people have to face a severe information overload problem. To alleviate this problem, many opinion mining techniques have been proposed, e.g. opinion summarization opinion polling [5] and comparative analysis. The key challenge is how to accurately predict the sentiment orientation of review sentences.

The sentiment classification methods generally fall into two categories: (1) lexicon-based methods and (2) machine learning methods. Lexicon-based methods typically take the tack of first constructing a sentiment lexicon of opinion words and then design classification rules based on appeared opinion words and prior syntactic knowledge. Despite effectiveness, this kind of methods requires substantial efforts in lexicon construction and rule design. Furthermore, lexicon-based methods cannot well handle implicit opinions, i.e. objective statements such as “I bought the mattress a week ago, and a valley appeared today”. As pointed out in [9], this is also an important form of opinions. Factual information is usually more helpful than subjective feelings. Lexicon-based methods can only deal with implicit opinions in an ad-hoc way [11].

The first machine learning based sentiment classification work applied popular machine learning algorithms such as Naïve Bayes to the problem. After that, most research in this direction revolved around feature engineering for better classification performance. Different kinds of features have been explored, e.g. n-grams, Part-of-speech (POS) information and syntactic relations, etc. Feature engineering also costs a lot of human efforts, and a feature set suitable for one domain may not generate good performance for other domains.

The deep learning has emerged as an effective means for solving sentiment classification problems [10]. A deep neural network intrinsically learns a high level representation of the data, thus avoiding laborious work such as feature engineering. A second advantage is that deep models have exponentially stronger expressive power than shallow models. However, the success of deep learning heavily relies on the availability of large-scale training data. Labeling a large number of sentences is very laborious.

Most merchant/review Websites allow customers to summarize their opinions by an overall rating score. Ratings reflect the overall sentiment of customer reviews and have already been exploited for sentiment analysis [3]. Nevertheless, review ratings are not reliable labels for the constituent sentences, e.g. a 5-stars review can contain negative sentences and we may also see positive words occasionally in 1-star reviews. The treating binarized ratings as sentiment labels could confuse a sentiment classifier for review sentences.

The deep learning framework is build for review sentence sentiment classification. The framework treats review ratings as weak labels to train deep neural networks. For example, with 5-stars scale we can deem ratings above/below 3-stars as positive/ negative weak labels respectively. The framework generally consists of two steps. In the first step, rather than predicting sentiment labels directly, we try to learn and embedding space reflects the general sentiment distribution of sentences, from a large number of weakly labeled sentences [2]. That is, we force sentences with the same weak labels to be near each other, while sentences with different weak labels are kept away from one another. To reduce the impact of sentences with rating-inconsistent orientation, we propose to penalize the relative distances among sentences in the embedding space through a ranking loss. In the second step, a classification layer is added on top of the embedding layer, and we use labeled sentences to fine-tune the deep network.

The framework is dubbed Weakly-supervised Deep Embedding (WDE). Regarding network structure, two popular schemes are adopted to learn to extract fixed-length feature vectors from review sentences, namely, convolution feature extractors and Long Short-Term Memory (LSTM) [8]. With a slight abuse of concept, we will refer to the former model as Convolution Neural Network based WDE (WDE-CNN); the latter one is called LSTM based WDE (WDE-LSTM). We then compute high level features synthesizing the extracted features, as well as the contextual aspect information of the product. The aspect input represents prior knowledge regarding the sentence's orientation.

IV. Neural Network based Opinion Mining Framework

The classic deep learning methods take an "unsupervised training then supervised fine-tuning" scheme, where restricted Boltzmann machines (RBM) or auto-encoders are used to pre-train network parameters from large quantities of unlabeled data. This works well when the data distribution is correlated with label prediction. Nevertheless, in sentiment analysis the word co-occurrence information is usually not well correlated with sentiment prediction [7], which motivates us to exploit large-scale rating data for training deep sentiment classifiers.

The ratings are noisy labels for review sentences and would mislead classifier training if directly used in supervised training. In this paper, we adopt a simple rule to assign weak labels to sentences with 5-stars rating scale:
`(s) = pos; if s is in a 4 or 5-stars review neg; if s is in a 1 or 2-stars review ; (1)

where `(s) denotes the weak sentiment label of sentence s. Note we follow previous works on aspect level sentiment analysis only consider positive and negative sentiment labels. The reason is that when commenting on various aspects of a product, people hardly express neutral opinions. We can see the noise level is moderate but not ignorable.

The WDE model uses large quantities of weakly labeled sentences to train a good embedding space so that a linear classifier would suffice to accurately make sentiment predictions [6]. Here good embedding means in the space sentences with the same sentiment labels are close to one another, while those with different labels are kept away from each other. In the following, we first present the network architecture and explain the specific design choices for WDE-CNN and WDE-LSTM. Then we discuss how to train it with large-scale rating data, followed by supervised fine-tuning on labeled sentences.

A. General Network Architecture

The general architecture of the neural network designed for WDE. At the first layer, the network takes a review sentence as input and extracts a fixed-length low-level feature vector from the sentence. Unlike many traditional methods for sentiment analysis, no feature engineering is required and the extractor is learned automatically. Specific implementation of the extractor will be discussed in the following for WDE-CNN and WDE-LSTM respectively. The low-level feature vector is then passed through a hidden layer, adding sufficient nonlinearity, and the output is used to compute the embedding representation of the sentence. The embedding representation also takes the sentence's aspect contextual information into consideration. An aspect is a topic on which customers can comment with respect to a sort of entities. For instance, battery life is an aspect for cell phones. We use a learnable context vector to represent an aspect. The motivation for incorporating aspect information as the context of a sentence is that similar comments in different contexts could be of opposite orientations, e.g. "the screen is big" vs. "the size is big".

In the weakly-supervised training phase, the goal is to learn an embedding space which can properly reflect data's semantic distribution. Hence, the network used in this phase contains only the layers up to the embedding layer. The final classification layer is added in the following supervised training phase, in order to learn the final prediction model.

B. Network Architecture of WDE-CNN

The network architecture of WDE-CNN is a variant of the CNNs described. In what follows, we use upper case bold letters such as W to denote matrices and lower case bold letters such as x to denote column vectors. The i-th element in vector x is denoted by x(i). This pooling scheme keeps the most important indicator of a feature and naturally leads to a fixed-length vector output v at the max pooling layer. A filter with window size h is intrinsically a feature extractor which performs "feature selection" from the h-gram features of a sentence. When the input h-gram matches its w, we will obtain a high feature value, indicating this h-gram activates the feature. This resembles the traditional feature selection in sentiment classification, but is done automatically by the network. Since traditional machine learning based methods often exploit unigrams, bigrams and trigrams.

C. Network Architecture of WDE-LSTM

The convolution filters in WDE-CNN can only capture input patterns from a text window, e.g., of 3 words. It is hard for a CNN model to explicitly capture long-term dependencies in sequential data. Hence, we propose another instantiation of WDE by Long short-term memory (LSTM). LSTM is a popular technique for recurrent neural networks (RNNs). An RNN progressively updates its hidden state given the current input and the previous hidden state from the last time step, making it a natural choice for modeling sequential data such as natural language sentences.

LSTM has recently attracted much attention due to its ability to learn long-term dependencies by using a gating mechanism. A LSTM maintains a structure called memory cell which can be viewed as a continuous analogy of a memory circuit. The memory cell controls the read, write and reset operations of its internal state through output, input and forget gates respectively, allowing the gradient information to be back-propagated through many time steps.

D. Embedding Training with Ratings

The advantages of triplet-based training over pair-based training via a toy example. We use circles and triangles to represent sentences in P and N respectively. Black nodes denote wrong-labeled sentences. Since the majority of sentences are with correct labels, they would gather together in the training process. Wrong-labeled sentences would go towards the wrong clusters, but with slower speeds. In both training methods, undesirable moves could happen when wrong-labeled sentences are sampled. For clarity, we just show three such cases that are representative for respective methods.

Embedding training is done by taking the derivative of Lweak respect to all the parameters under the Embedding Layer. Hyperbolic tangent is employed as the activation function for all the layers. The detailed derivation of the back-propagation procedure for WDE can be found in the appendix. We do SGD over sampled sentence triplets with Ada-Grad update rule. Each mini-batch consists of 1024 sentence triplets. To cope with the overfitting problem, we do early stopping in the training process according to the network's performance on the validation set. We devise a measure for validation. Intuitively, a good embedding space will keep intra-class instances near each other and separate instances from different classes, providing a proper prior model for supervised training. Hence, we propose to assess the intra-class affinity and inter-class separability of the embedding on the validation set as an indicator of the model's generalization ability.

The WDE-LSTM can naturally capture long-term dependencies while WDE-CNN is limited by the window sizes of convolution filters. Nevertheless, WDECNN is more efficient than WDE-LSTM. With the above hyper parameter setting, WDE-CNN needs about half an hour for processing 1M triplets on a Nvidia GTX 980ti GPU, while for WDE-LSTM the corresponding time is about 5 hours. The low efficiency of WDE-LSTM stems from two facts: (1) there are much more parameters in LSTM layer than in CNN layer; (2) when computing derivatives for a LSTM cell's weights, each word position must be considered, while for a convolutional filter, we only need to care about the word position which wins the max operation. Moreover, the parallelization can only be done sentence-wise in WDE-LSTM, while for WDE-CNN we can do word-level parallelization.

E. Supervised Fine-tuning

The classification layer is added on the top further train the network using labeled sentences. The classification layer simply performs standard affine transformation of the embedding layer output y and then applies a softmax activation function the result for label prediction. In this work, we focus on binary sentiment prediction since we only consider sentences which comment on specific aspects of an entity. This kind of sentences hardly contain neutral sentences. Nevertheless, WDE could also be adapted to multi-class prediction problems. For binary prediction, the classification layer is equivalent to a logistic regression model. We train the network using standard SGD, since AdaGrad can easily "forget" the prior model learned in the first phase. The mini-batch size is set to 64. A similar early stopping strategy is adopted as in the embedding training phase.

V. CONCLUSION

The product reviews are analyzed with the deep learning framework named Weakly-supervised Deep Embedding for review sentence sentiment classification. WDE trains deep neural networks by exploiting rating information of reviews which is prevalently available on many merchant/review Websites. The training is a 2-step procedure: first we learn and embedding space which tries to capture the sentiment distribution of sentences by penalizing relative distances among sentences according to weak labels inferred from ratings; then a softmax classifier is added on top of the embedding layer and we fine-tune the network by labeled data. WDE is effective and outperforms baseline methods.

REFERENCES

[1] Zhen Hai, Gao Cong, Kuiyu Chang, Peng Cheng and Chunyan Miao, "Analyzing Sentiments in One Go: A Supervised Joint Topic Modeling Approach", IEEE Transactions On Knowledge And Data Engineering, Vol. 29, No. 6, June 2017.
[2] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. IEEE TPAMI, 35(8):1798–1828, 2013.
[3] L. Qu, R. Gemulla, and G. Weikum. A weakly supervised model for sentence-level semantic orientation analysis with multiple experts. In EMNLP-CoNLL, pages 149–159, 2012.
[4] L. Chen, J. Martineau, D. Cheng, and A. Sheth. Clustering for simultaneous extraction of aspects and features from reviews. In NAACL-HLT, pages 789–799, 2016.
[5] J. Zhu, H. Wang, M. Zhu, B. K. Tsou, and M. Ma. Aspect-based opinion polling from customer reviews. IEEE Transactions on Affective Computing, 2(1):37–49, 2011.
[6] Wei Zhao, Ziyu Guan_ , Long Chen, Xiaofei He, Fellow, IAPR, Deng Cai, Beidou Wang and Quan Wang, "Weakly-supervised Deep Embedding for Product Review Sentiment Analysis", IEEE Transactions on Knowledge and Data Engineering, January 2018
[7] L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts. Learning word vectors for sentiment analysis. In ACL, pages 142–150, 2011.
[8] K. Greff, R. K. Srivastava, J. Koutn´ık, B. R. Steunebrink, and J. Schmidhuber. Lstm: A search space odyssey. arXiv preprint arXiv:1503.04069, 2015.
[9] R. Feldman. Techniques and applications for sentiment analysis. Communications of the ACM, 56(4):82–89, 2013.
[10] X. Glorot, A. Bordes, and Y. Bengio. Domain adaptation for large-scale sentiment classification: A deep learning approach. In ICML, pages 513–520, 2011.
[11] L. Zhang and B. Liu. Identifying noun product features that imply opinions. In ACL, pages 575–580, 2011.