Convolutional Neural Networks for Sentiment Analysis on Amazon Product Reviews

Sijie (Anne) Yu
W266 Fall 2018, Session 4
School of Information
UC Berkeley
syu.anne@berkeley.edu

Guangyu (Gary) Pei
W266 Fall 2018, Session 6
School of Information
UC Berkeley
guangyu.pei@berkeley.edu

Abstract

In this paper, we compare several neural network architectures for analyzing the sentiments of Amazon product reviews (a.k.a. Product Reviews). Representative subset of data are selected from Baby product category collected in 2015. Two types of embeddings are experimented, namely, pre-trained GloVe and self-trained Word2Vec. We perform both binary and multiple sentiment labels classifications using a bag-of-word model (BoW) and proposed multi-resolution convolutional neural networks (MrCNNs). At the end we use commercial-off-the-shelf product (Amazon's Comprehend service) to benchmark models' performance in terms of accuracy.

Our unique contributions include the following. First, we propose a novel MrCNNs model. Second, we perform experiments on a few deep neural networks to demonstrate their performance in product review sentiment analysis. Finally our work focuses on the dataset from single natural language (i.e., English) and from the same business category (e.g. Baby product) to reduce complexity caused by languages and product terms.

1 Introduction

Sentiment is a subjective opinion on an entity affected by human feelings. Therefore sentiment analysis is different from objective analysis in a way that it analyzes emotions. Sentiment analysis has many variations, for example sentiment classification, opinion mining, or sentiment categorization. Sentiment analysis can also be conducted at various levels: phrase or sentence, document and aspect level [Fang and Zhan (2015)]. Some work are essentially a binary classification problem, classifying texts to be positive or negative while others generate fine-grained labels, for example, a 5-star category system. Similar work can also be found from commercial systems, for example, Amazon Comprehend, which classifies one product review into 4 categories: positive, negative, neutral or mixed, along with confidence score.

In this paper, we perform the sentiment analysis on Amazon's Product Reviews to predict customer sentiment regarding one product. The advent of online users has produced a crescent user review growth. For example, "Amazon Elements Baby Wipes, Sensitive, 720 Count Flip-Top Packs", as of now, has 13,532 customer reviews. "Pampers Baby-Dry Disposable Diapers Size 1", has 5,693 customer reviews. Human being seems not be able to process such volume of data in a short online session. But for consumers sentiment is the critical information to guide purchase. Therefore our objective is to create a robust deep learning model, by trained with a large amount of product reviews, can quickly predict a new product's sentiment polarity. Based on literature reviews, we find that even though deep learning has been applied to many sentiment analysis applications but few are relevant to online shopping reviews. This project is targeted to fill this gap with our experimental work.

The rest of the paper is organized as follows. Section 2 and Section 3 provide the overview of the commercially available sentiment analysis products and research literatures in classic convolutional neural network models for the sentence sentiment analysis, respectively. Section 4 describes our proposed model architecture and provides the mathematical definitions of the operations and corresponding key techniques involved

in the computation. Section 5 discusses the experiment setup, datasets, implementation details and iterative improvements from initial design. Section 6 shows experiment results and comparison of different models. In Section 7, we conclude our paper and outline the future work.

2 Commercial Services for Sentiment Analysis

We research both academic papers and commercial systems for product review sentiment analysis. Surprisedly, we find the most decent and relevant work is from industry. We notice that a leap forward in the cloud-based sentiment analysis services releasing in recent years. For example, Google's Cloud Natural language, released in February 2016, analyzes all types of texts including reviews. Amazon released Comprehend in November 2017, which has a feature to to detect product sentiments [Amazon (2017)]. IBM Watson can also detect review sentiment with a higher accuracy as reported in a blog from RStudio with a test to compare these systems accuracies on 498 Twitter comments and tokens with categorization of mood and tense, Amazon gives the a finer grained category label to differentiate neutral from mixed sentiment.

3 Convolutional Neural Networks (CNNs) for Sentence Sentiments

CNNs have been proved to show super performance in terms of accuracy on classifying sentence sentiments, from a basic neural bag-of-words or bag-of-n-grams models to the more structured CNNs [Kim (2014), Kalchbrenner et al. (2014)]. The input vectors could either be pre-trained embeddings, unsupervised trained embeddings, or just as parameters. One CNNs model is designed with multiple stacked CNNs, each of which comprises of one convolutional layer followed by a dynamic pooling layer and a non-linearity function [Kalchbrenner et al. (2014)]. The dynamic pooling layer computes the number of top selected features from sentence length instead of making selection fixed. Another CNNs has multiple channels taking static and dynamic embedding vectors as inputs, which increases the model flexibility for unseen tokens in the unsupervised embedding training [Kim (2014)]. One CNNs model takes not just words but characters features to classify short sentences [Santos and Gatti (2014)]. Its core are two layers of CNNs, extracting word and character features, respectively, with maximum pooling and non-linearity function.

4 MrCNNs Architecture

Our model focuses on review-level sentiment analysis, that is, given a review with multiple short sentences, MrCNNs computes a score s for each sentiment label $\tau \in \top$. Note that, review sentiment orientations can be simply aggregated into an overall sentiment for a product, so to make the model directly useful for cunsumers. Inspired by the multi-resolution recurrent neural networks for dialogue responses [Serban et al. (2016)] and also because of characteristics of review data, which contains one review headline and one review body, we create a multi-resolution convolutional networks (MrCNNs) consisting of a coarser CNNs layer with multiple smaller sized filters to extract the high level headline features and a finer CNNs layer to extract more detailed features from body. That is, the model takes the sequences of words in both headline and body, passes them through two sequences of CNNs layers, where features with increasing levels of complexity are extracted. Selected features are feed into max-pooling layers, respectively, to extract the relatively more important features. Filtered features are concatenated to pass through a fully connected hidden layer, as if combining features from either resolution layer. The outputs are applied with a final output neural network layer to give scores for each label. The main novelty in our network architecture is the inclusion of two convolutional layers, which can handle headline and body sentences of different resolutions separately. That being said, for a review, comprising of body and headline, the coarser and finer CNNs features are computed as the below.

Define a set of tokens x_i, $i \in 1, 2, ..., N$, N is the number of tokens in the review. Define 2 sets of features: $\{C^t_{b,j}, j \in 1, 2, ..., M; C^v_{hl,l}, l \in 1, 2, ..., K\}$, in which, M is the number of features computed from body, K is the number of features computed from headline, $N \leq M$, $N \leq K$. Define 2 sets of filters $\{F^t_b, t \in 1, 2, ...T, F^v_{hl}, v \in 1, 2, ...V\}$, in which , T is the number of a variety size of filters applied to body, and V is the number of a variety size of filters applied to headline. For example, one body feature $C^t_{b,j}$ can be calculated

as,

$$C_{b,j}^{t} = f(w_t \odot x_{j:j+h-1} + b_t) \quad (1)$$

Here, $C_{b,j}^{t}$ is the feature coming from a sequence of tokens, $\{x_j, ..., x_{j+H-1}\}$ for filter t. The convolution operation $\odot$ involves a filter w_t which is applied to a window of size H words, b_t is a bias term and f is a non-linear function, for example, ReLu, to produce a new feature $C_{b,j}^{t}$. Filters are applied to each possible window of body and headline token per review to produce two feature maps. That is, we compute C_{b}^{t} and C_{hl}^{v} from body and headline, respectively, using filter F_{b}^{t} and F_{hl}^{v}.

$$C_{\text{b}}^{t} = \left[C_{b,1}^{t}, C_{b,2}^{t}, ..., C_{b,M}^{t}\right], \quad (2)$$

$$C_{\text{hl}}^{v} = \left[C_{hl,1}^{v}, C_{hl,2}^{v}, ..., C_{hl,K}^{v}\right] \quad (3)$$

We then apply a max-over-time pooling operation over the feature maps and take the maximum value as the feature corresponding to this particular filter. The idea is to capture the most important feature - one with highest value - for each feature map. At the beginning of computation, if headline or body is not long enough to fill in the embedding matrix, we pad missing tokens with a constant tiny score. This pooling scheme naturally deals with padded tiny scores in this case. As we have two sets of features from coarser and finer CNNs layers, we concatenate them into one long feature input. This process is described in the following equation. In particular, one feature, i.e. C_{b}^{t} and C_{hl}^{t}, is extracted from one filter from one body and headline, respectively. The final set of features for one review is calculated as,

$$C_{\text{r}} = [\max(C_{\text{b}}^{1}), \max(C_{\text{b}}^{2}), ..., \max(C_{\text{b}}^{T}), \max(C_{\text{hl}}^{1}, \max(C_{\text{hl}}^{2}), ..., \max(C_{\text{hl}}^{V})] \quad (4)$$

Besides our model uses multiple filters (with varying window sizes) to obtain multiple features. These features form the intermediate layer and are passed to a fully connected softmax hidden layer and then another output layer whose output is the probability distribution over labels. That is,

$$s = \text{softmax}(W_{\text{output}} f(W_{\text{hidden}} \cdot C_{\text{r}} + b_{\text{hidden}}) + b_{\text{output}}). \quad (5)$$

The overall architecture is illustrated in Figure 1.

4.1 Back-propagation and Regularization

Back-propagation is used to adjust parameters from coarser and finer CNNs layers, including a set of convolutional kernels, to intermediate fully-connected hidden and output layer, overfitting occurs when training the large set of parameters. We use $L2$ regularization to combat overfitting and also experiment with increasing the dropout rate. For regularization we employ a constraint on $L2$-norms of the weight and bias [Kim (2014)]. That is, a $L2$-norms of the weight plus bias vectors by rescaling w and b to have $\left\|w^2\right\| + \left\|b^2\right\| = s$ whenever $\left\|w^2\right\| + \left\|b^2\right\| > s$ after a gradient descent step. The larger value of s will tightly constrain the delta values adjusting parameters.

4.2 Various Dropout Rates

Dropout prevents co-adaptation of hidden units by randomly dropping out—i.e., setting to zero—a proportion p of the hidden units during forward back-propagation. However headline contains the more relevant tokens to imply buyer's feeling, for example, "Horrible!", "Great product!". Therefore in the training stage, we use lower dropout rates to keep more headline than body examples.

4.3 Multi-scale Sentiment Classification

We extend our model to predict multi-scale sentiment labels. Amazon Comprehend gives 4, namely, Negative, Neutral, Positive or Mixed. Since we are not sure about Mixed, we combine labels into 3: Negative, Positive and Neutral. The Neutral label includes both Neutral and Mixed cases. The ground truth label is calculated from review star, that is, instead of $\tau \in \{\text{negative, positive}\}$, $\tau \in \{\text{negative, positive, neutral}\}$.

$$\tau = \begin{cases} \text{negative,} & \text{if star} <= 2.0, \\ \text{positive,} & \text{if star} >= 4.0, \\ \text{neutral,} & \text{otherwise} \end{cases} \quad (6)$$

5 Datasets and Experiment Setup

5.1 Amazon Public Customer Reviews Dataset

Amazon's Product Reviews are one of the largest natural language collections consisting of consumer reviews from Amazon's world wide sites.

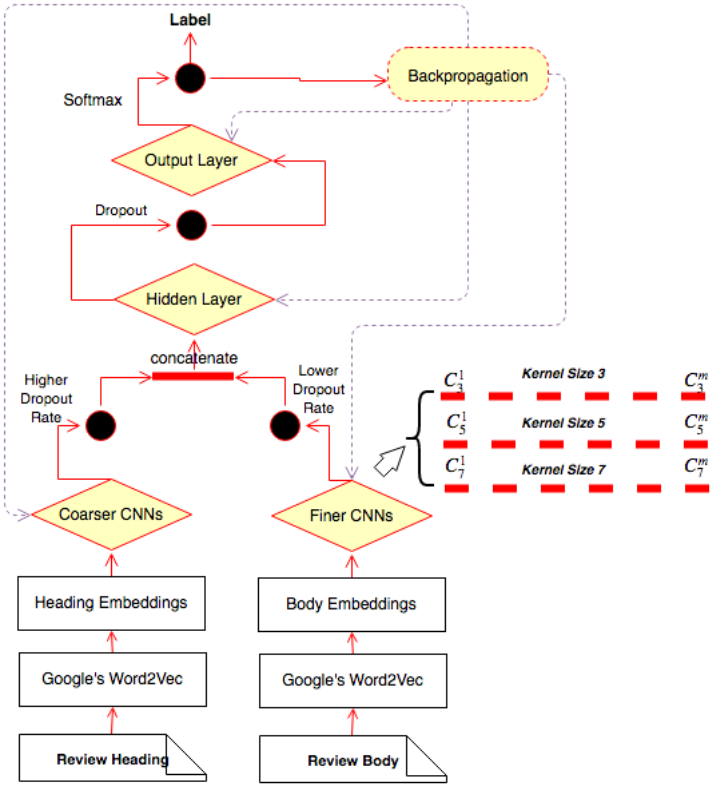

Figure 1: MrCNNs Architecture

It started from 1995 until 2015, collecting a rich dataset consisting of reviews, votes, ratings, and summary. It covers 43 product categories (e.g. Baby, Books) and spans 5 market places (e.g. US, FR). The other important reason to use it is because Amazon makes it free for academic researches. As suggested we create a Hive parquet table to load S3 data, then a query to download them into a cvs disk file [Amazon (2018)]. We extract all reviews in category "Baby" of year 2015, which covers 16,388 products with 39,984 reviews. We select out examples with more than 2 helpful votes and

$$\frac{\text{votes}_{\text{helpful}}}{\text{votes}_{\text{total}}} \geq 0.25.$$

.

5.2 Natural Language Processing For Reviews

Review texts are beyond natural languages, because they contain numbers, symbols, expressions, emoji's, and unique syntax to express feelings. Before feeding into embedding layer, we use a preprocess to clean and chunk raw texts into tokens. In particular, the following preprocess is performed:

1. sentences are tokenized and lowercased, remove punctuation characters in between. For example, "Changing Pad/Changing Table Pad" becomes "changing, pad, table".

2. normalize numbers to replace all digits with 0 (e.g. 1345 to be 0000).

3. keep "?" and "!" at the end of sentence because they usually indicate strong emotion.

5.3 Statistics of Raw Data

We split original dataset into a training dataset of 9,137 reviews and a test dataset of 1,381 reviews, analyze the length of body and headline tokens to guess the hyper-parameter initial values. Table 1 shows the statistics for the datasets after tokenization, where `Max` is maximum number of tokens per body and headline; `Avg` is the average number of tokens per body and headline; `Min` is the minimum number of tokens per body and headline and `Vocab` is vocabulary size for all bodies

Review	Max	Avg	Min	Vocab
Body	346	60.9	10	20,429
Headline	24	5.5	2	8,561

Table 1: Summary statistics of datasets

and headlines. The main observation is that headline contains much less tokens than body.

5.4 Pre-trained Word Vectors

Initializing word vectors with a classic unsupervised neural language model is a popular method to improve performance in the absence of a large supervised training set [Iyyer et al. (2014)]. We use the publicly available GloVe vectors that were trained on Twits, Wikipedia, crawled web pages by Stanford NLP lab [Pennington et al. (2014)]. Because of resource limitation we use one model with dimensionality of 100 and size of 6B (i.e. 6B tokens, 400k vocabulary). Words not present in pre-trained vocabulary are filled with a tiny constant score.

5.5 Self-Trained Word2Vec

We notice more than 30% tokens are missing from pre-trained vocabulary, so we decide to train our own embedding scores. Recent work has show that accuracy can be improved by conducting unsupervised pre-training for word embeddings. In our project we perform unsupervised learning of word embeddings using the word2vec algorithm, which implements the continuous bag-of-words and skip-gram architectures to comput vectors [Mikolov et al. (2013)]. We download all Baby product reviews in 2015, i.e. 39,985 views, build a unique vocabulary corpus for our model. We use the `gensim.models.word2vec.Word2Vec` implementation, which is a highly optimized C set of routines for Word2Vec algorithms [Řehůřek (2018)]. To reduce running time we use the embedding dimension of 50, with a token size of $38,724$, trained on $73,670$ sentences. The context window size is configured as 7. The trained embedding size is $21,434 \times 50$. The vocab is selected from tokens with occurrence of more than 2. We integrate both embedding models into MrCNNs to compare the overall performance. As a result, the Word2Vec has a slight boost on the F1 score.

Hyper-parameter	Value
Word Embeddings Dimension	50
Word Context Window	7
Finer CNNs Filter Sizes	3,4,5,6
Number of Finer CNNs Filters	104
Coarser CNNs Filter Sizes	3,4,5
Number of Coarser CNNs Filters	10
Finer Layer Dropout Rate	0.5
Coarser Layer Dropout Rate	0.2
L2 Regularization Lambda	2.5
Batch Size	38
Hidden Layer Dimension	110
Hidden Layer Dropout Rate	0.2

Table 2: MrCNNs Hyper-parameters

5.6 Baseline: BoW Neural Networks

We implement a neural bag-of-words classifier as it is the simplest neural model to establish an effective baseline [Kunz (2018)]. It takes its name from the bag-of-word assumption common to linear models, in which the weights for each input word are summed to make a prediction. For the neural version, instead sum the vector representations of each word, we add feed-forward (hidden) layers to make a deep network. We make the embedding layer with dimension of 50, with a hidden layer of dimension of 25 and Adagrad (Adaptive Subgradient Methods) as the optimizer.

5.7 Hyper-parameters and Bayesian Optimization

Hyper-parameters are chosen by running Bayesian optimization on development set of size 1,890 reviews. Bayesian optimization is a random search function to find the global optimal values for target function. Here we use accuracy as the target, find the optimal set of values for hyper-parameters. The implementation comes from the scikit-optimize package [scikit (2016)]. We do not otherwise perform any dataset specific tuning other than early stopping on dev sets. Training is done through stochastic gradient descent over shuffled mini-batches with the Adam update rule [Kim (2014)]. The other hyper-parameters are decided by try and error, for example, number of epochs and embedding dimension. In Table 2, we list the selected hyper-parameters values for binary classification case in our project.

Be noted, Number of Finer CNNs Filters is the number of features selected from body per review

Model	F1 Score
BoW Multiple Labels	0.804
Comprehend Multiple Labels	0.830
MrCNNs Binary Labels GloVe	0.833
MrCNNs Binary Labels Word2Vec	0.864
MrCNNs Multiple Labels GloVe	0.788
MrCNNs Multiple Labels Word2Vec	0.812

Table 3: Accuracy of different models for binary and fined grained (3-class) predictions using product reviews.

and Number of Coarser CNNs Filters is the number of features selected from headline per review. The former is much larger than the latter since it contains richer information.

6 Test Results

We program to connect to Amazon Comprehend service then get sentiments for our test review dataset. We train BoW neural with embeddings as parameters. We then integrate MrCNNs model with both pre-trained GloVe embeddings and Word2Vec embeddings trained from Baby product review corpus. Results of our models against other methods are listed in Table 3.

In table above, we can note that,

- Even the baseline BoW neural networks performs surprisingly well, and all neural models are actually not remarkably better from one of the other.
- Pre-trained embeddings perform actually worser than project specific embeddings. One reason could be there are too many unique tokens found from reviews while unseen from a generic corpus. For example, emoji, misspelled words (i.e. thisties, sevral), self-created words (i.e. 1m54, rubix), and product names (i.e. munchik).
- Binary classification case generally has higher F1 score than multiple label classification case, across all models.

7 Conclusions and Future Work

In this paper, we proposed a new algorithm MrCNNs to predict product review sentiment. We compared MrCNNs with a classic neural network model using bag of words as well as the state of art commercial-off-the-shelf solution from Amazon Comprehend service. Our test results showed that input embeddings can play several percentage performance impact on accuracy. The performance of MrCNNs is comparable with Amazon Comprehend service.

Our extensive experiments provide several insights for our future work. To improve input layer, pre-trained embeddings can be taken as initial inputs, then trained with other hyper-parameters. In another words, one can use both static and dynamically trained embeddings. Another improvement could be making the neural model deeper with one additional local hidden layer to combine results from each sentences. In this way, one can treat sentence as feature rather than token.

In summary, proposed novel MrCNNs model offers flexible multi-resolution architecture for the product review sentiment analysis and provides comparable performance with respect to Amazon Comprehend solution.

References

Amazon. 2017. Amazon comprehend.

Amazon. 2018. Amazon Customer Reviews Dataset.

X. Fang and J. Zhan. 2015. Sentiment analysis using product review data. *Journal of Big Data*, 2: 5(2196-1115).

Mohit Iyyer, Jordan Boyd-Graber, Leonardo Claudino, Richard Socher, and Hal Daumé III. 2014. A neural network for factoid question answering over paragraphs. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 633–644. Association for Computational Linguistics.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *CoRR*, abs/1404.2188.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751. Association for Computational Linguistics.

James Kunz. 2018. Neural bag-of-words model.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language*

Processing (EMNLP), pages 1532–1543. Association for Computational Linguistics.

CíCero Nogueira Dos Santos and Maíra A. De C. Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *COLING*.

scikit. 2016. Bayesian optimization with skopt.

Iulian V. Serban, Tim Klinger, Gerald Tesauro, Kartik Talamadupula, Bowen Zhou, Yoshua Bengio, and Aaron Courville. 2016. Multiresolution recurrent neural networks: An application to dialogue response generation. *arXiv e-prints*, abs/1606.00776.

Radim Řehůřek. 2018. models.word2vec – word2vec embeddings.