Sentiment Analysis of Yelp Data: A Neural Network approach

Kailash Nathan Srinivasan
UMass Amherst
CICS
kailashnatha@cs.umass.edu

Neelesh Kumar Boddu
UMass Amherst
CICS
nboddu@cs.umass.edu

Abstract

Neural networks, especially some variants of LSTM RNNs have recently achieved remarkable performance in the NLP task of sentiment classification, mainly due to their ability to learn sequential and semantic relationships. In this report we explore various neural network approaches to sentiment analysis of Yelp reviews using the 'Yelp Challenge dataset'. Specifically, we aim to use the review text to predict the star rating for the business, which is a quantitative measure of positive or negative sentiment. Here, we conduct a series of experiments beginning with shallow models, then going on to feedforward networks, CNNs and RNNs. We examine the standard naïve 'bag-of-words' representation and go on to employ various word vector embeddings like word2vec, Glove etc. in training the models. We compare the various model performances to gain insight about word-embedding driven deep neural network performance for sentiment classification.

1. Introduction

The objective of our NLP task is to predict sentiment expressed using only the review text. We use the 'star ratings' as a quantitative measure of sentiment and pose the task as a classification task with 'positive' and 'negative' class outputs. First, we start with a naïve 'bag-of-words' representation and employ traditional 'shallow' models such as Naïve Bayes and Random Forest to use as performance baselines. Then we go on to employ 'deep' learning methods, starting with simple feedforward networks, and then focus on working with CNNs and RNNs. We then go on to experiment with word vector embeddings like word2vec and GloVe and finally use them in training the neural network models to learn sequential information and use it to classify overall sentiment. We compare the various model performances to highlight the differences among them as well as the efficacy of unsupervised word-vector learning applied to neural networks for sentiment classification.

2. Related Work

The recent breakthroughs of deep learning have leveraged remarkable advancements for multi-disciplinary AI research problems, especially for NLP and computer vision problems. For example, the deep CNN for image classification [10], image recognition [4]; the RNN for speech recognition [6], sequence to sequence learning [8], machine translation [13] and conversation model [12]. Within NLP, much of the work with deep learning methods has involved learning word vector representations through neural language models [1][2] and performing composition over the learned word vectors for classification [3]. Originally intended for Computer Vision, CNN models have subsequently been shown to be effective for NLP and have achieved excellent results in semantic parsing [15], search query retrieval [16], sentence modeling [17], and other traditional NLP tasks [3].

Yoon Kim [9] showed that a simple CNN with 1 convolution layer with max over time pooling achieves good results for sentence classification. This however is different from document level sentiment classification task which remains non-trivial. [18] is often cited as the first systematic comparison of CNN and RNN on a wide range of representative NLP tasks, which reinforces the idea that while CNN is good at extracting position-invariant features, RNN is good at modeling units in sequence, giving us a basis to explore RNNs for our inherently sequential modeling task.

[19] offers an empirical exploration of recurrent network architectures and shows the efficacy of LSTMs, while comparing with different architectures. This also explores the then recently used Gated Recurrent Unit architectures. This is where we decide to use a model with a simple LSTM architecture and look for further work on GRUs.

In [20] the group led by Chung provide a similar kind of an empirical evaluation of GRU networks on sequence modeling. Their experiments reveal that these units are indeed better than traditional units such as tanh and also comparable to LSTM. The work in [21] explores the use of GRU neural networks for the non-trivial and challenging task of document level sentiment classification. They use similar datasets and their experiments show GRU nets outperform standard RNNs.

3. Approach

We used various baselines such as the Most Frequent Class baseline, Naive Bayes and Random forest to create a benchmark range for our neural network models to hopefully surpass. The neural network models that we employed involve a simple Multi-Layer Perceptron, a Convolutional Neural Network inspired by Yoon Kim [9] and LSTM variants of Recurrent Neural Networks.

3.1 Convolutional Neural Network

Convolutional Neural Networks (CNNs) are a specialized kind of feed forward neural network where the use of general matrix multiplication operation is replaced by a convolution operation. Furthermore, the CNNs enforce a weight sharing connectivity pattern between neurons of adjacent layers to exploit spatially-local correlation. Our CNN architecture is based on [9], where Yoon Kim suggested the use of a simple 1-layer CNN with 1-dimension convolution where the convolution mask is a vector instead of a matrix. Their work used a $s \times d$ representation of sentence of length s and word features of d dimension (Figure 2). The proposed convolution layer had multiple filter widths and feature maps and was followed by max over time pooling. The idea is to capture the most important feature, i.e. one with the highest value.

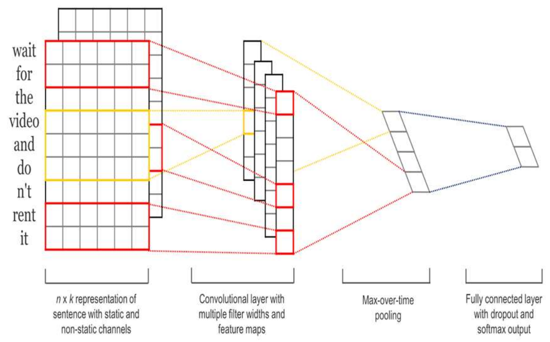
wait
for
the
video
and
do
n't
rent
it
n x k representation of sentence with static and non-static channels
Convolutional layer with multiple filter widths and feature maps
Max-over-time pooling
Fully connected layer with dropout and softmax output

Fig. 3.1.1 Model Architecture

A sentence of length n is represented as

$$x1{:}n = x1 \oplus x2 \oplus \ldots \oplus xn,$$

where $\oplus$ is the concatenation operator. In general, let $xi{:}i+j$ refer to the concatenation of words $x_i, x_{i+1}, \ldots, x_{i+j}$. A convolution operation involves a filter $w \in R$, which is applied to a window of h words to produce a new feature. For example, a feature ci is generated from a window of words by

$$c_i = f(w \cdot x_{i:i+h-1} + b)$$

Here $b \in R$ is a bias term and f is a non-linear function such as the hyperbolic tangent. This filter is applied to each possible window of words in the sentence $\{x_1{:}h, x_2{:}h+1, \ldots, x_n{:}h+1{:}n\}$ to produce a feature map.

$$c = [c1, c2, \ldots, cn-h+1]$$

The model uses multiple filters (with varying window sizes) to obtain multiple features. These features form the penultimate layer and are passed to a fully connected softmax layer whose output is the probability distribution over label. The word-embedding we employed is the Word2Vec model, the same as Yoon Kim's [9] implementation.

For regularization we employ dropout on the penultimate layer with a constraint on l2-norms of the weight vectors [5]. 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 backpropagation. That is, given the penultimate layer. Here, the dropout probability we used was 0.5.

3.2 LSTM Recurrent Neural Networks

Recurrent nets are a type of artificial neural network designed to recognize patterns in sequences of data, such as text, genomes, handwriting, the spoken word, or numerical times series data emanating from sensors, stock markets and government agencies. As such, they are very relevant to our current task of learning labels from a text sequence. They take as their input not just the current input example they see, but also what they have perceived previously in time. They're distinguished from feedforward networks by that feedback loop connected to their past decisions, ingesting their own outputs moment after moment as input. It is often said that recurrent networks have memory (in a sense that each step of the sequence is remembered and builds on what went before, and often, meaning emerges from their order). Adding memory to neural networks has a purpose: there is information in the sequence itself, and recurrent nets use it to perform tasks that feedforward networks can't. That sequential information is preserved in the recurrent network's hidden state, which manages to span many time steps as it cascades forward to affect the processing of each new example. It is finding correlations between events separated by many moments, and these correlations are called "long-term dependencies", because an event downstream in time depends upon, and is a function of, one or more events that came before. One way to think about RNNs is this: they are a way to share weights over time.

Like most neural networks, recurrent nets are old. By the early 1990s, the vanishing gradient problem emerged as a major obstacle to recurrent net performance. In the mid-90s, a variation of recurrent net with so-called Long Short-Term Memory units, or LSTMs, was proposed by the German researchers Sepp Hochreiter and Juergen Schmidhuber [11] as a solution to the vanishing gradient problem. LSTMs help preserve the error that can be backpropagated through time and layers. By maintaining a more constant error, they allow recurrent nets to continue to learn over many time steps (over 1000), thereby opening a channel to link causes and effects remotely.

LSTMs contain information outside the normal flow of the recurrent network in a gated cell. Information can be stored in, written to, or read from a cell, much like data in a computer's memory. The cell makes decisions about what to store, and when to allow reads, writes and erasures, via gates that open and close. Unlike the digital storage on computers, however, these gates are analog, implemented with element-wise multiplication by sigmoids, which are all in the range of 0-1. Analog has the advantage over digital of being differentiable, and therefore suitable for backpropagation. Those gates act on the signals they receive, and similar to the neural network's nodes, they block or pass on information based on its strength and import, which they filter with their own sets of weights. Those weights, like the weights that modulate input and hidden states, are adjusted via the recurrent networks learning process. That is, the cells learn when to allow data to enter, leave or be deleted through the iterative process of making guesses, backpropagating error, and adjusting weights via gradient descent.

The diagram below illustrates how data flows through a memory cell and is controlled by its gates.

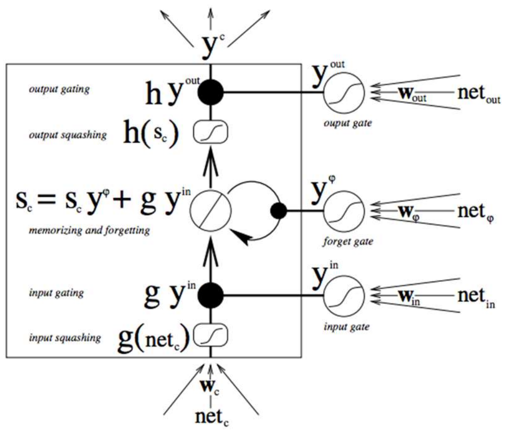

Fig 3.2.1 RNN memory cell data flow

It should be noted that while feedforward networks map one input to one output, recurrent nets can map one to many, as above (one image to many words in a caption), many to many (translation), or many to one (sentiment classification). Text (like our review text here) contains recurrent themes at varying intervals. LSTM can capture this.

There's another variant of LSTMs which are the Gated Recurrent Units or GRUs. A gated recurrent unit (GRU) is basically an LSTM without an output gate, which therefore fully writes the contents from its memory cell to the larger net at each time step.

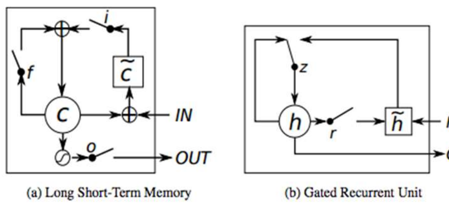

Fig 3.2.2 LSTM and GRU cells

In our implementation, we used both the LSTM model variants (with and without GRUs). In both models, we use 200-dimensional pre-trained GloVe vector weights trained on 27 billion instances of twitter data to create the word embeddings to be fed into the network. As a lot of opinions and sentiments are expressed on Twitter, we thought this would serve our purpose well. In essence, both our LSTM models take the same GloVe word vector as input. This guides our specification of the embedding layer parameters.

The actual model specifications of the two LSTM models we employed are as follows:

3.2(a). LSTM model 1

This network involved using one hidden LSTM layer that takes the 200-dimensional input from the embedding layer as described above and outputs a 100-dimensional output, which is regularized with a dropout probability of 0.2. The final layer outputs the class label (0 or 1) with sigmoid activation. The loss function we used here is binary cross entropy, given the binary class partition and the optimizer used is Adam, which fits in nicely into our paradigm. The computation is done in batches, with a batch size of 256 (high, because we have the GPU at our disposal) and is done for 10 epochs.

3.2(b). LSTM model with GRUs

This network involved using 2 hidden GRU layers. The first GRU takes the 200-dimensional input from the embedding layer and outputs a 200-dimensional vector forward as sequences, regularized by dropout probability of 0.2. The second GRU propagates the 200-dimensional vector to the final layer. The final layer uses the sigmoid activation and outputs the class label. The loss function, optimizer, batch size and the number of epochs are the same as LSTM model 1.

4 Experiments

4.1 Dataset

The dataset we use is the Yelp 2017 challenge dataset, introduced in the 10th round of Yelp Challenge, comprises user reviews about local businesses in 11 cities across 4

countries with star rating from 1 to 5. The large-scale dataset comprises 4.7M reviews and 947 K tips by 1M users for 144K businesses [7]. Yelp 2017 challenge dataset has been updated compared to datasets in previous rounds, such as Yelp 2015 challenge dataset
.

4.2 Data visualization and exploration:

To get an insight on the length of each review, we temporarily created a new column in yelp called text length. This column stores the number of words in each review. Using a box-plot to represent the distribution of review text lengths against star ratings, we observed the 1-star and 2-star ratings have much longer text, but there are many outliers.

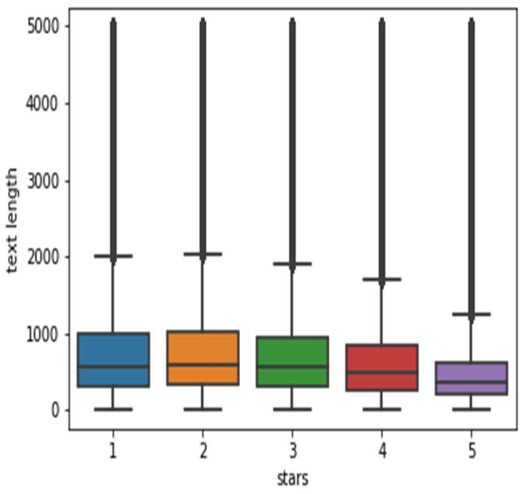

Fig 4.2.1 Box plot of length of reviews

4.3 Data preprocessing:

The visualized statistics help us understand the data better, guiding us in preprocessing. We filtered out 3-star ratings as they don't contribute to the polarity of the review in the context of our classification task, which is modeled as binary classification into 'positive' or 'negative' sentiment classes. We then found that the data contained non-English reviews as well, so we had to filter them out as our resources (nltk etc.) are based on English. The filtration process introduced some inconsistencies in the pandas data frame we were using, as part of the implementation of the langdetect module. So, we further cleaned the data, removing rows with 'nan' and irrelevant values. Finally, we used this curated version to create a toy dataset which we used to perform the experiments as described in the following section.

The data is skewed towards the 4 and 5-star ratings and this could affect our results. So, we balanced our dataset in such a way that all of our 4 categories have even distribution among 100000 reviews. We have used both balanced and unbalanced data on our models.

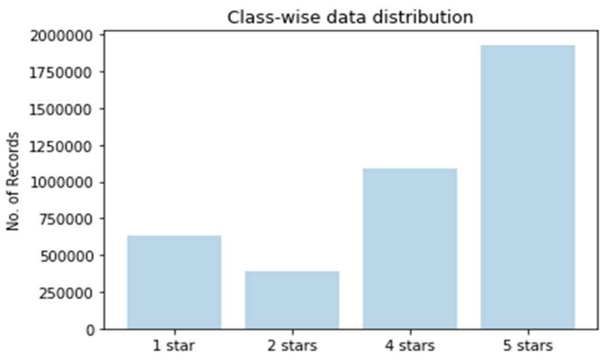

Fig 4.3.1 Class-wise distribution of the data

4.4 Baseline Models:

The effectiveness of applying machine learning techniques in sentiment classification of product or movie reviews can be demonstrated using traditional approaches such as representing text using bag-of-words model and different methods such as Naive Bayes, maximum entropy classification, Random Forest and Perceptron algorithms.

The classification algorithm will need some sort of feature vector in order to perform the classification task. The simplest way to convert a corpus to a vector format is the bag-of-words approach, where each unique word in a text will be represented by one number. Transforming this to a sparse matrix. Our first experiments are done on a smaller subset of Yelp dataset having 100000 training samples and 10000 testing samples.

We used bag of words representation on this data sample and applied Multinomial Naive Bayes Classifier from scikit learn. Using the same bag of words, we then used Random Forest Classifier with the hyperparameter as number of trees set to 100.

Finally, we implemented a multi-layer perceptron (MLP) algorithm that trains using Backpropagation. It supports multi-class classification by applying SoftMax as the output function. The perceptron had the learning rate of 1e-5 and one hidden layer as hyperparameters.

4.5 Word2Vec

Word2vec is a group of Deep Learning models developed by Google with the aim of capturing the context of words while at the same time proposing a very efficient way of preprocessing raw text data. The powerful concept behind word2vec is that word vectors that are close to each other in the vector space represent words that are not only of the same meaning but of the same context as well. Each word in the corpus is being assigned a unique vector in the vector space. This model takes as input a large corpus of

documents like tweets or news articles and generates a vector space of typically several hundred dimensions. These models are shallow, two-layer neural networks that are trained to reconstruct linguistic contexts of words.We have used word vectors with deep learning via word2vec's "skip-gram and CBOW models", using hierarchical softmax.We used the following parameters to implement it.
Downsampling rate: 1e-3 Context window: 10
Number of features: 128 Number of threads in parallel: 4

4.6 CNN

We used the CNN before mentioned to train on our polarised dataset in TensorFlow. The first layers embeds words into low-dimensional vectors. The next layer performs convolutions over the embedded word vectors using multiple filter sizes. For example, sliding over 3, 4 or 5 words at a time. Next, we max-pool the result of the convolutional layer into a long feature vector, add dropout regularization, and classify the result using SoftMax layer. We used 128 filters of varying filter size and an embedding dimension of 128.We have used Adam optimizer and set the dropout probability to 0.5.

4.7 GloVe

GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space. The approach is described in [14]. For our task, we used 200-dimensional pretrained word vector weights trained on 27 billion twitter records.

4.8 RNNs

We first implemented the LSTM model 1 hoping to surpass the performance of the CNN model. As expected, it did surpass the CNN performance, and by a lot. In fact, it reached accuracy of 90% within 10 epochs, with a score of 0.23.

We then implemented the GRU RNN model. Although the expectation was for this model to perform better than the CNN, we didn't have particular expectations in terms of how it stacked up against the LSTM model 1. As it turned out, the model breached 90% accuracy within 10 epochs, with a score of 0.32.

Results

Accuracies obtained in various models		
Model	Balanced data	Unbalanced data
Baseline	20	41
Naïve Bayes	55	58
Random Forest	59	61
MultiLayer Perceptron	63	67
CNN	70	76
LSTM RNN	92	90
GRU RNN	93	91

The results of the models were largely within the scope of our expectations, given the context of our sentiment classification task. The baseline models performed as expected, not being able to do much with the naive bag of words representation. Then the first 'deep' neural network model, the multilayer perceptron showed glimpses of the power of neural network models, although it was neither too deep nor too 'intelligent'.

The results got interesting from the CNN model, where although it seemingly performed a tad bit better than the MLP, the inherent multi-sentence structure hindered the performance and as several others who implemented Yoon Kim's model noted, couldn't translate the sentence classification results to a paragraph sequence classification paradigm.

The results from the RNN models both reached and surpassed our expectations, surprising us. They reached expectations in that we expected them to leverage the GloVe embeddings and their inherent ability to 'memorize' and classify sequences, giving them an edge over the CNN model. They surpassed expectations in that although we expected them to perform 'better' qualitatively, breaching the 90% mark was an impressive result, all the while taking advantage of GPU-accelerated training to keep the training times under sane limits given our time constraints. However, the fact that there was no major difference in the performance of the LSTM and GRU variants remains an interesting result.

5. Conclusion

In the present work we have described a series of experiments with neural networks used for sentiment classification of multi-sentence texts. Despite little tuning of hyperparameters, a simple CNN with one layer of convolution performs remarkably well compared to shallow and simpler model baselines. Our results add to the well-established evidence that unsupervised pre-training of word vectors is an important ingredient in deep learning for NLP. Our key take-away that the use of GloVe in our RNN models seems to have helped a lot, reinforces this idea. Dropout was a good mode of regularization and consistently added 2%–4% relative performance. The fact that we managed to achieve impressive results while keeping the networks relatively shallow, i.e, not using lots of hidden layers and convoluted architectures goes to showcase the power of LSTM RNNs, especially when complemented with a good word embedding. The task could be modeled as multi-class when the classes are categorical and have importance of their own. The networks then would have to be adjusted accordingly to account for the multi-class categorical case. It is therefore not very hard to see that tweaking the architectures further and perhaps preprocessing the data and extracting features would improve performance even more, giving us hope in conquering the non-trivial task of sentiment classification of reviews.

References

[1] Bengio, R. Ducharme, P. Vincent. 2003. Neural Probabilistic Language Model. Journal of Machine Learning Research 3:1137–1155

[2] Mikolov, I. Sutskever, K. Chen, G. Corrado, J. Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In Proceedings of NIPS 2013.

[3] Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, P. Kuksa. 2011. Natural Language Processing (Almost) from Scratch. Journal of Machine Learning Research 12:2493–2537

[4] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. CoRR, abs/1512.03385, 2015.

[5] Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, R. Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors.

[6] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed,N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath,et al. Deep neural networks for acoustic modeling in speech recognition.

[7] Yelp Challenge Dataset, 2017.

[8] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In Advances in neural information processing systems, pages 3104–3112, 2014.

[9] Y.Kim. Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882, 2014.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems, pages 1097–1105, 2012.

[11] S. Hochreiter and J. Schmidhuber. 1997. Long Short-Term Memory. In Neural Computation, 9(8):1735–1780.

[12] O. Vinyals and Q. Le. A neural conversational model. arXiv preprint arXiv:1506.05869, 2015.

[13] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey,et al. Google’s neural machine translation system: Bridging the gap between human and machine translation. arXiv preprint arXiv:1609.08144, 2016.

[14] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation.

[15] Yih, K. Toutanova, J. Platt, C. Meek. 2011. Learning Discriminative Projections for Text Similarity Measures. Proceedings of the Fifteenth Conference on Computational Natural Language Learning, 247–256.

[16] Shen, X. He, J. Gao, L. Deng, G. Mesnil. 2014. Learning Semantic Representations Using Convolutional Neural Networks for Web Search. In Proceedings of WWW 2014.

[17] Kalchbrenner, E. Grefenstette, P. Blunsom. 2014. A Convolutional Neural Network for Modelling Sentences. In Proceedings of ACL 2014.

[18] Yin, Wenpeng, et al. "Comparative Study of CNN and RNN for Natural Language Processing." *arXiv preprint arXiv:1702.01923* (2017).

[19] Jozefowicz, Rafal, Wojciech Zaremba, and Ilya Sutskever. "An empirical exploration of recurrent network architectures." *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*. 2015.

[20] Chung, Junyoung, et al. “Empirical evaluation of gated recurrent neural networks on sequence modeling.” arXiv preprint arXiv:1412.3555 (2014).

[21] Tang, Duyu, Bing Qin, and Ting Liu. "Document Modeling with Gated Recurrent Neural Network for Sentiment Classification." *EMNLP*. 2015.