Deep Learning MonkeyBusiness A3

Michael Rabadi
New York University
michael.rabadi@nyu.edu

Sam Royston
New York University
sfr265@nyu.edu

David Garwin
New York University
drg314@nyu.edu

1 Introduction

For this assignment we were provided a dataset of yelp reviews labeled with 1 - 5 star ratings. We created a temporal convolutional neural network on vectorized business reviews provided by Yelp. We utilized a customized version of word2vec and created a 212k word database of vectors.

2 Data Augmentation, Training, and Testing

2.1 Unsupervised

We initially experimented with the GloVe dataset, as well as [1], however we later decided to use a customized version of word2vec. We adapted the word2vec framework with custom c code to preprocess and augment training set and also pulled from two large databases that were similarly processed to create a large word corpus on which word2vec was trained. Preprocessing steps included removing all punctuation and we converted everything into lower case. We combined the skipgram and bag-of-words models in word2vec to convert words into 200 dimensional vectors. We trained word2vec on the entire yelp review dataset, the default encyclopedia exerpt for word2vec, and an unlabeled dataset of movie reviews. The dictionary was biased to weight the words from the yelp dataset more heavily than the other sources. The other sources added more words, giving us a total of over 212k word dictionary to improve generalization. Notably, this dictionary included typos, but, empirically, the cosine distance between words with typos was very close to the correctly spelled words.

2.2 supervised

We created a "dictionary" that mapped words to vectors via the unsupervised framework described above. Reviews were shuffled, then truncated to 100 words and had all punctuation removed and were put into lowercase. Finally, each word was converted into a 200 dimensional vector. We added borders of zeros to the beginning and end of these 100-word truncated reviews. These reviews were passed to the temporal convolutional neural network (architecture described below). We witheld 10% of the reviews for testing.

3 Architecture

In the original model, *averaged* word embeddings were fed into a 3-layer neural network, which represents significant information loss in the form of word frequencies and word order.

In our second model, we opted to for a 4-layer architecture Which uses temporal convolutions to derive some meaning from the word order, in addition to the relative word "meanings" supplied by word2vec. For reviews that were less that 100 words, the remainder are zeroed out. Therefore, our input size per review was 200×100, with the convolution occurring over 7 words for the first convolutional layer and then 3 words for the second convolution layer. Our complete architecture consisted of two temporal convolution layers, each with temporal max pooling, followed by two linear layers, with 50% dropout after the first linear layer. We then took a log soft maximum likelihood in order to predict the rating for the review.

4 Techniques

We utilized a number of different techniques, including drop out, clipping techniques, and the customized word2vec code described above. Notably, we experimented with a number of clipping parameters. Our best construction was achieved by clipping all reviews to 100 words. We also tested with 50 words, which had comparable performance while training, but we were concerned about the algorithms ability to generalize to the test dataset. We tested various batch sizes and learning rates and found that a learning rate of 0.1 witha moment of 0.1 for stochastic gradient descent was ideal. We tried various batch sizes, but batch size did not seem to have an effect - we trained our final model with a batch size of 128. We also implemented a learning rate decay of 0.001.

5 Training Procedure

We first shuffled the reviews evenly so the network was *not* trained in blocks. We then withheld 10% of the reviews for testing during cross validation. We trained on the remaining 90% of reviews and cross validated with the holdout set. We then reshuffled all reviews and repeated the procedure for 2 epochs.

6 Performance

Our learner correctly predicts a review's rating 50.62% of the time in our test set after training for 8 epochs. While this is a seemingly low accuracy rating, we think this does not fully capture the success of our model. For example, one would hope that our model does not typically mistake 1 star reviews for star reviews, and would make the majority of its mistakes by only missing a the rating by a single star.

Given that our measure of success offers no consolation for incorrect, but plausible, predictions. In the future we would like to measure the distribution of guesses for a given label. We could expect a gaussian distribution centered at the correct answer, where the variance will be a good measure of how gracefully it is failing.

7 Failed Models

Most of the models that failed did so because of the specific hyperparameters that were used. We experimented with a 3 layer network that we could not get to perform above chance. Interestingly, our current model performed at chance when we trained with GloVe, which indicates that our customized word2vec dictionary significantly boosted performance (see discussion below). We also experimented with different sized review lengths. 100 words seemed to work well, but 50 words also had high performance and benefitted from slightly faster training.

8 Discussion

We trained a temporal convolutional neural network to predict the ratings of the Yelp business reviews dataset. We found that hyperparameter tuning was essential for high success. A 3 layer convolutional net had high performance when combined with customized

word2vec vector mapping. The success of the word2vec mapping probably had to do with the fact that our customized corpus was trained with all of the training data, a large database of movie reviews, and exerpt form an encyclopedia. Thus, our word2vec code was trained with well over 300 million words. Because of this, we expect our word embeddings to be a strong point in our model.

Our changes to the model architecture itself are in part designed to make use of word order, so we ran a few tests where the meaning of a sentence was changed by swapping words. Initial observations indicate that word order is being taken into account, but at a much lower priority than which words are present. For example, the short sentences

the food was not terrible, it was amazing

the food was not amazing, it was terrible

both receive a score of 1, which is not ideal, but in the following case

anything but, the food was awful

the food was anything but awful

The classifier gives *"the food was anything but awful"* a slightly higher score of 2 stars, even though the only difference is word order. It should be noted that short phrases like this are not what our learner was designed for, but nonetheless this offers some hope that the nuances of word order are being learned.

A main concnern that we have regarding testing has to do with the distributions of the reviews. It has been shown that high prediction performance can only be guaranteed when the discrepancy between the training and testing distributions is small [3]. Interestingly, you can approximate the test distribution and bias the model appropriately in an online setting if we are given the test labels after each sample, or if we are given the entire unlabelled test set. Unfortunately, those were not provided in this assignment. A natural extension would be to allow online functionality, possibly using a follow-the-perturbed-leader training and testing protocol.

Finally, we interviewed a machine learning practitioner regarding training convolutional neural nets. He observed that the convolutional nets that he has used requires around 2000 training epochs before it is usable. Naturally, we expect improved performance with more epochs. Unfortunately, we were limited by the number of training epochs.

References

[1] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. (2011). *Learning Word Vectors for Sentiment Analysis*. The 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011).

[2] Goldberg, Y., & Levy, O. (2014). *word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method*. arXiv preprint arXiv:1402.3722.

[3] Cortes, C., Mohri, M., Riley, M., & Rostamizadeh, A. (2008, January). *Sample selection bias correction theory*. In Algorithmic learning theory (pp. 38-53). Springer Berlin Heidelberg.