Yelp Data-Set Challenge - Natural Language Processing

Manav Mehra*
Masters of Computer Science
University of Illinois at Urbana-Champaign
manavm3@illinois.edu

Amber Gupta*
Masters of Computer Science
University of Illinois at Urbana-Champaign
amberg4@illinois.edu

Jatin Arora*
Master of Science in Computer Science
University of Illinois at Urbana-Champaign
jatin2@illinois.edu

1. INTRODUCTION

In this project, we work on the review classification task of the Yelp Dataset Challenge[1]. More specifically, the task is, given some review text, we have to classify it with a rating on the scale of 1 to 5. With the recent boom in the e-commerce industry, the amount of textual content present on these e-commerce websites is increasing in volume. A large share of this textual content is present in the form of reviews/opinions given by users for various products/services. This text is a very rich source of information for product manufacturers as well as service providers as it provides them first hand feedback through which they can understand the market response for their products or services. This is also very insightful for users as they get the opinions of their fellow customers and can take better decisions on whether to buy a specific product or not.

Due to the motivations highlighted above, review classification has been a popular research area in the natural language community for long. This problem comes under the general category of document classification, which was initially viewed from a bag-of-words model perspective. Each document is converted into a bag of important words to get a fixed-size vector representation of the document. This feature vector is then fed to standard classifiers like logistic regression or SVM for standard multi-class classification.

With the recent advances in deep learning, there has been a shift from the bag-of-words based approach to get richer representations of documents using CNNs or RNNs etc. In this project, we mostly focus on these recently developed techniques popular for this task. The contributions of this work are threefold:

1. We present a detailed comparison of popular deep learning approaches for solving this task.
2. Instead of treating this problem as a general multi-class classification problem, we utilize the property that ratings (output classes) have an ordinal nature. Hence, we present a different perspective of looking at this problem as an ordinal classification task.
3. To model the ordinal nature of class labels, we propose two novel approaches to modify BERT framework's loss function: `BERT-Ordinal`, `BERT-Focal` and present our analysis.

In the next section, we describe the various approaches we have experimented with and their novel variants followed by a description of the experimental setup. Then, we present our results and analysis, followed by the conclusions and possible extensions.

2. DATASET

The Yelp dataset consists of around 6,880,000 reviews with each review having **10.7 sentences** (average) or **113 words** (average). Due to the computational limitation, we randomly sample and work with **10%** of the entire dataset and divide this data into a 80-10-10 split for train-dev-test sets. Table 1 gives a description of our dataset splits.

Dataset Type	# Samples
Train	550,749
Dev	68,843
Test	68,870

Table 1: Yelp Dataset Details

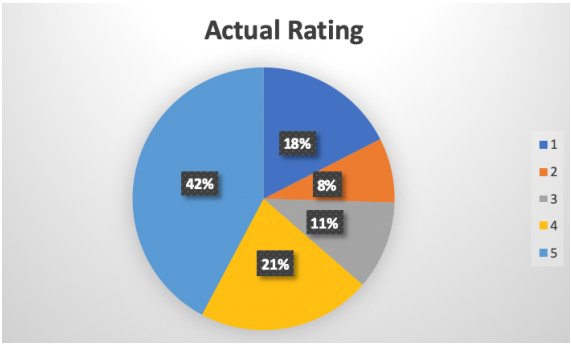

Figure 1: Distribution of the Data Labels - Stars

*Equal Contribution
[1]https://www.yelp.com/dataset/challenge

DOI: 10.1145/1235

3. APPROACHES

In this section, we discuss the various approaches we have applied to solve the task at hand. With each approach, we also present the hyper-parameter settings we work with.

1. **Kim-CNN (Baseline 1)**: As proposed by [3], in this approach, we use a CNN architecture which tries to learn the importance of n-grams for the end task of predicting the class/rating of the review. It is based on the fact that in a sentence, the current word is implicitly dependent on the surrounding words. In the training process, we freeze the word embeddings which are initialized from Word2Vec[5] model pre-trained on general GoogleNews corpus[2]. The embedding dimension is 300.

2. **HAN (Baseline 2)** : Hierarchical Attention Network [9] works on the idea that "Words make sentences and sentences make documents". The architecture is two level. Initially, words are embedded and fed to a bi-directional LSTM, followed by an attention layer, since some words contain more context with respect to characterization of the sentence. This gives a fixed length vector representation for each sentence. In the second level, these sentence vectors are fed to another bi-directional LSTM with attention layer followed by softmax to get a probability distribution among the output labels. Just like Kim-CNN, the main advantage of this architecture is that it is able to capture the essence of the whole document/review, for the end task. Again like Kim-CNN, we use 300-dimensional Word2Vec embeddings prepared from GoogleNews corpus and freeze them during the training process. The hidden layers of both LSTMs are of size 50.

3. **BERT**: As proposed by Devlin et al. [1], we pass in the review to the sequence classification head of pre-trained BERT model for multi-class classification. We use the `bert-base-uncased` model and fix the maximum sequence length to 256. So, reviews with length less than that, get padded, while longer length reviews get truncated. Compared to the approaches discussed above, the limitation of this approach is that, for longer length reviews, it truncates the content giving no significance to content that goes beyond the sequence length threshold. As discussed further in the results section, this model gives us the best results.

4. **BERT-Ordinal**: This is a variation of the BERT model. In the above discussed approaches, we naively use the cross-entropy loss for multi-class classification. Consequently, our model loses out on the information about the ordering of the ratings. The model does not understand the intuitive difference between ratings 1 and 2 vs 1 and 5. For example, if the actual review rating is 1, the model penalizes a wrong prediction of 2 as well as 5 in the same way.

   To handle this issue, instead of training a single multi-class classifier, we train *k-1* binary classifiers where *k* is the number of class labels / ratings. This approach is inspired from work done by [2]. More concretely, in our case of 5 rating classes, we train 4 binary classifiers, with the first one predicting if review is rated greater than 1. Similarly the 2nd classifier predicts if the review is greater than 2 and the 4th predicts if the review is rated greater than 4 (i.e 5).

   To achieve this, we convert our data-set labels from a one-hot to an ordinal encoding. Hence, rating 1 `[10000]` gets converted to `[0000]`, rating 3 `[00100]` is converted to `[1100]` and so on. Table 2 shows the encoded values for each rating in the dataset. Ultimately, for classification, we use sigmoid function to convert the logits output from the model into individual binary classifier probability values. The first dimension gives $P(rating > 1)$. The second dimension gives $P(rating > 2)$. Taking the difference gives, $P(rating = 1)$. Similarly, all rating probabilities are calculated and we take the `argmax` to obtain the classification. Apart from this, rest of the experimental setup is the same as naive BERT model described above.

Rating	Encoding (One-Hot)	Encoding (Ordinal)
1	10000	0000
2	01000	1000
3	00100	1100
4	00010	1110
5	00001	1111

Table 2: Dataset Labeling Description

5. **BERT-Ordinal-Focal**: Although BERT-Ordinal approach captures the ordinal nature of output ratings, the binary classifiers mentioned above faces the challenge of biased data-set. Consider the binary classifier which predicts $Prob(rating > 1)$ as described above. For this classifier, the number of positive samples are all the samples in the data-set with rating 2 to 5. However, the only negative samples for this classifier are the reviews rated 1. To tackle this bias in the training data, we take inspiration from focal loss, as proposed by [4]. Since our setting is of ordinal classification, with continuous streak of 1's, naive application of focal loss on the logits output from the model does not work. Hence, we calculate the focal loss for each dimension / classifier separately and then sum it up to get the overall loss per sample. We fix parameters, gamma = 2 and alpha = 0.25 which are heuristically found to work well, as proposed by [4]. Other model parameters are same as the BERT-Ordinal model.

4. EXPERIMENTAL SETUP

For implementing various approaches discussed above, we make use of the hedwig library[3]. The deep learning models are written in PyTorch and for BERT, we make use of the HuggingFace `transformers` [4] python package. For implementing focal loss, we derive our understanding from its PyTorch implementation on GitHub[5]. The experiments are run on Nvidia GPUs.

[2]https://code.google.com/archive/p/word2vec/

[3]https://github.com/castorini/hedwig

[4]https://github.com/huggingface/transformers

[5]https://github.com/clcarwin/focal_loss_pytorch

5. RESULTS

Table 3 presents the results of various approaches. These results are on 10% of the entire Yelp dataset created as described in the dataset section above.

Approach	Accuracy (%)
Kim-CNN	62.5
HAN	64.0
BERT	**75.3**
BERT-Ordinal	44.9
BERT-Focal	42.6

Table 3: Comparison of Results

Note that, in order to get a better understanding of the generalization of our models, we also run some light approaches described above, like, **HAN** and **Kim-CNN** approaches on a larger dataset as well. The larger dataset is still 20% of the entire Yelp dataset (created in the same way as the 10% data described above). The comparative study is presented in Table 4.

Approach	Dataset Size (%)	Accuracy
Kim-CNN	10	62.5
Kim-CNN	20	62.8
Kim-CNN	50	62.0
HAN	10	64.0
HAN	20	62.0

Table 4: Accuracy with Dataset Variation

6. OBSERVATIONS AND INFERENCES

From our experiments, we observe that out-of-the-box BERT model outperforms all other approaches discussed. It gives better results than the baselines, HAN and Kim-CNN, which can be attributed to the advantages of the transformer architecture over vanilla CNNs and LSTMs.

The Kim-CNN model uses 1-dimensional convolutions with filter sizes of 3, 4 and 5. On top of it, ultimately it uses a linear layer. Hence, it is able to capture n-grams of size not more than 5 and does not consider the sequence structure for n-grams beyond that (due to the linear layer). HAN model captures the sequence structure and in fact the hierarchy that words build up sentences which in turn build up the review. To achieve this, HAN uses a bi-directional LSTM architecture at its core. BERT model on the other hand, uses a transformer architecture which is found to capture the representation of information better than LSTMs [7]. In addition to this, BERT model captures the bi-directionality better than naive LSTMs, using the masked language modelling loss function.

Although in this work we don't present ablations of the BERT model, but we propose certain variants to capture the ordinal nature of output class labels, i.e. BERT-Ordinal and BERT-Focal (as described in the sections above). However, contrary to our belief, the BERT-Ordinal model does not perform better than BERT. In the next paragraph we discuss a possible reasoning for the BERT-Ordinal and BERT-Focal models to not perform well.

Both the models, BERT-Ordinal and BERT-Focal require a continuous streak of classifiers to predict correctly. So, for classifying a review of rating (say) 7, we require 6 classifiers to predict correctly. Although, this seems plausible theoretically, it might actually be a high expectation. Even if one of these 6 classifiers does not give proper probability values, since, we take a running difference of probabilities (as described in the Bert-Ordinal section above), the classification may go wrong.

Also, the BERT-Focal model performs comparably with the BERT-Ordinal model. This suggests that the individual binary classifiers (trained in the BERT-Ordinal approach) are not biased in the first place itself and adding the additional focal loss term is slightly reducing the quality of learning.

Also, from Table 4, we observe that accuracy does not vary much as the dataset size increases. This observation can be extrapolated to conclude that the dataset we work with and the model, generalizes over the bigger Yelp corpus.

6.1 Error Diff Analysis

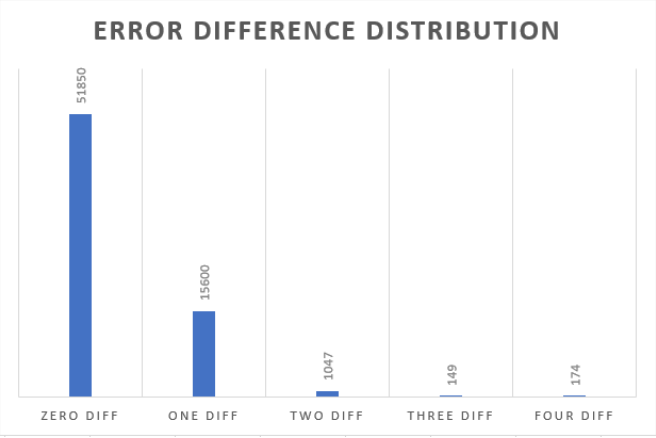

Figure 2: Label Error Difference Distribution for BERT

Fig 2. shows the distribution of difference between the predicted and actual labels on the test set. We observe that apart from the Diff = 0 cases (which are captured by the reported accuracy of 75%). If we relax the accuracy measure to consider mis-classifications with Diff = 1, then the relaxed accuracy is 98 % which is a good observation. Extreme mis-classifications are very limited and can be considered as outliers.

7. CONCLUSIONS AND FUTURE WORK

In this work, we experimented with several popular approaches for handling review classification over the Yelp dataset. We observe that just like many SQuAD[6] and GLUE[8] tasks, BERT performs well for fine-grained review rating classification task as well.

Nevertheless, we observe from the diff. analysis, that the next major limitation of the BERT model is, in capturing the Diff = 1 cases well. As future work, we aim to qualita-

tively analyse the review texts for such mis-classifications to identify if certain entity words or sentiment words may be given higher weightage through some architecture to better handle such cases.

8. REFERENCES

[1] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[2] E. Frank and M. Hall. A simple approach to ordinal classification. In *European Conference on Machine Learning*, pages 145–156. Springer, 2001.

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

[4] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[5] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[6] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

[7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[8] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

[9] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy. Hierarchical attention networks for document classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 1480–1489, 2016.