Sentiment Analysis: Predicting Yelp Scores

Bhanu Prakash Reddy Guda   Mashrin Srivastava   Deep Karkhanis
Carnegie Mellon University
`{bguda,mashrins,dkarkhan}@andrew.cmu.edu`

Abstract

In this work, we predict the sentiment of restaurant reviews based on a subset of the Yelp Open Dataset. We utilize the meta features and text available in the dataset and evaluate several machine learning and state-of-the-art deep learning approaches for the prediction task. Through several qualitative experiments, we show the success of the deep models with attention mechanism in learning a balanced model for reviews across different restaurants. Finally, we propose a novel Multi-tasked joint BERT model that improves the overall classification performance.

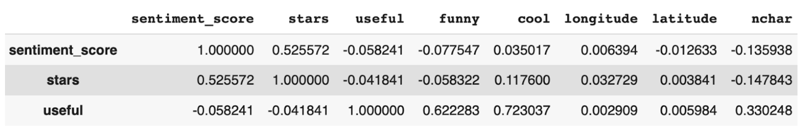
1 Introduction

Analyzing online reviews is crucial for many local and large-scale businesses that aim to excel by prioritizing customer satisfaction. Positive feedback from customers may prosper the store businesses, while negative one could have opposite consequences. Manually going through the reviews for understanding the overall tone could be a laborious process. An *automated sentiment analysis* approach that examines the review and predicts its tone by looking at its structured (meta features) and unstructured information (review text) could benefit the businesses in quickly filtering and emphasizing their focus on improving the negative reviews. While several datasets have product reviews associated with them, in this work, we use the reviews submitted on the Yelp platform, a platform with an open dataset that has proven to be good and accurate for research purposes, to achieve this objective in a restaurant setting.

Among the several attributes that are available in the Yelp data, we focus mostly on the star rating, review text, some useful meta features, and sentiment score values. We organize the rest of the paper as follows. We provide a detailed description of the feature analysis and feature selection in Section 3 and build towards a novel sentiment prediction model in Section 4 by progressively evaluating several machine learning baselines and the key ideas of prominent milestones in the field of deep learning approaches for natural language processing. We provide our intuitions behind using a particular architecture and consequently contribute a novel BERT based multi-tasked deep model with both (joint) meta features and review text as the input features for sentiment classification. In Section 5 we extensively evaluate the baselines along with the proposed model through quantitative and qualitative results.

2 Related Work

Sentiment analysis is a popular and well-explored task in several domains. Among many others, the most common gold standard datasets that academia focuses on are the IMDb movie reviews [2], Amazon product reviews [1], Stanford Sentiment Treebank [4], and Yelp restaurant reviews [7]. Several of the proposed approaches in the past aim to learn a model specific to one dataset or a general model that performs well across all datasets; the latter one being more prevalent. The choice of model innovation to achieve competitive state-of-the-art performances among these works varies widely. [27, 23, 25] attempt to develop sophisticated ML models with careful feature engineering of meta data and review texts. [30, 29, 11, 28] propose deep representations of the review text with



	sentiment_score	stars	useful	funny	cool	longitude	latitude	nchar
sentiment_score	1.000000	0.525572	-0.058241	-0.077547	0.035017	0.006394	-0.012633	-0.135938
stars	0.525572	1.000000	-0.041841	-0.058322	0.117600	0.032729	0.003841	-0.147843
useful	-0.058241	-0.041841	1.000000	0.622283	0.723037	0.002909	0.005984	0.330248

Figure 1: Correlation Matrix over features and labels

an additional component of attending on parts of the input text to improve the overall performance. [19, 17] summarizes and reports the performances of these methods specifically on the Yelp dataset.

Our objective of developing a model specifically for the Yelp review sentiment classification task enables us to differ from the prior approaches that learn a general language model and later fine-tune it for downstream tasks using only text. Our end-to-end novel architecture for the Yelp review sentiment classification task is inspired by the more recent idea of multi-tasked joint learning [12] for a closely related task of empathy prediction using text and meta attributes.

3 Dataset and pre-processing

The dataset used for this paper is a subset of the Yelp Review dataset[7] which is a commonly used publicly available dataset for sentiment analysis. This dataset includes 8,021,122 reviews from 209,393 businesses in 10 metropolitan areas. This data is structured in JSON files, including business, review, user, check-in, tip, and photo data. The subset of the data we have used has 36,692 customer reviews of restaurants. This data has a total of 42 features. However, some of these 36,692 reviews have some features missing. This dataset can be found here. [6]

Before the data could be used for training any machine learning model, it is necessary to do some pre-processing. Sentiment analysis is viewed as a supervised learning task. Of the 42 attributes provided in this dataset, two attributes: star rating and sentiment score are considered labels and the rest are taken as input features to define our supervised learning problem. **Star rating** – Stars of the review. This is a typical one (worst) to five (best) scale which takes values as integers and is only available in the training data. **Sentiment score** – A quantification of the text's sentiment using the AFINN lexicon. The value ranges from -5 (very negative) to 5 (very positive).

Feature engineering

We split the data into train, validation and test set (70:15:15) for the purpose of training and evaluating our models. Moreover, we only use a subset of the features for training. Specifically, we **dropped majority of the binary features** indicating an occurrence of particular words like terrible, disgusting, amazing, ... This is because natural language models are prone to overfitting on such word occurrence features. Since natural language has a lot of freedom, most words can be used in most sentiments. Thus, models which do not rely on such bag-of-words approaches usually learn much better since they don't overfit on the training data's words.

We also calculated the **correlation between all features** and labels in the train data. As seen in figure 1, the "stars" feature or the start rating is highly positively correlated with the sentiment score. Hence, predicting the star rating gives us an idea of the sentiment of the review. Thus for the purpose of labeling positive vs negative sentiments, apart from the binary word occurrence indicators, we also **drop the sentiment score** and instead only use the star rating, hence we use the terms *star rating prediction* and *sentiment prediction* interchangeably. For this binary sentiment classification problem, our model labels a one to three star rating as a negative sentiment (0) and a four to five star rating as a positive sentiment (1).

Further, we also drop features that clearly don't contain any useful sentiment information like reviewer name, number of characters and words in the review and the city name since most reviews are from the same city. Note that attributes that help identify the restaurant itself (like latitude and longitude) hurt the problem of sentiment analysis. This is because the model will simply start associating good restaurants to good reviews and not learn to infer sentiment from natural language. We also convert the date to UNIX format, normalize the numerical attributes and convert the categories features to a one-hot encoding of 201 categories. For classical ML models, we also perform standard text pre-processing like removal of stop words, tokenization, converting to lower-case, lemmatization,

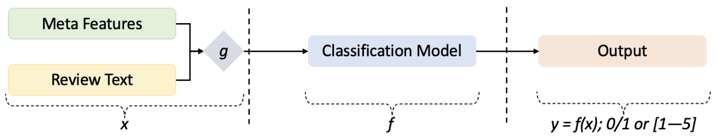


Figure 2: Classification setup used for the baselines and methods for our task. The function g selects one of the feature sets or both of them by simply concatenating or applying a joint transformation.

punctuation removal, and **TF-IDF calculations** [22]. For deep learning models, we directly give the full text since we need to have the original sentence structure preserved.

4 Methodology

In this section, we describe various machine learning baselines and deep learning models used in our task. All our models follow the typical classification setup as shown in Figure 2, input features - model - output class. As mentioned in the dataset section, we have 2 sets of input features, **review text** and **meta features**. The output is either a binary classification i.e good star rating or a bad star rating, or a more granular 5-star classification i.e an integer in the range [1, 5]. For the model part, we provide our intuitions behind using a particular model, and in Section 5, we validate our intuitions by analyzing the model interpretability.

4.1 Machine Learning Baselines

In the first part of our experimentation, we apply 6 machine learning models, i.e Decision Trees (DT), Random Forest Classifier (RF), K-Nearest Neighbour Classifier (KNN), Support Vector Machines (SVM), Gradient Boosting Classifier (GB), and Multi-Layer Perceptron (MLP). To understand *I1: Impact of the meta features and the review text individually and jointly on the classification task*, we experiment with 3 different setups of these models. (1) Using only the meta features as input for both binary as well as 5-star classification tasks. (2) Providing only the **review text** represented as a TF-IDF [22] vector as input. (3) Concatenating the TF-IDF vector with meta features as input to each of these models. In all the setups, we use the standard SCIKIT-LEARN [3] implementations for these models with hyperparameter tuning using the validation set.

4.2 Deep Review Representation (DRR)

While the simple machine learning models achieved good performance values (Table 5) in both binary and 5-star classification setups, one of the bottlenecks is to represent convoluted information i.e review text through a simple TF-IDF vector notation. To this end, we have experimented with a few notable advancements in deep learning techniques for better representation of text. Broadly the advancements have one of the following backbones as the underlying architecture: *convolutional neural networks*, *long short term memory networks*, and *transformer* units. These networks are further boosted in their performance by a technique called *attention*. Through these experiments, we try to understand *I2: Impact of the deep high dimensional representations of the review text in the classification performance*.

We will now describe the core components in each of these techniques.

4.2.1 Convolutional Neural Networks

In this experiment, we tackle the bottleneck by training convolutional neural networks (CNNs) trained on top of pre-trained word vectors. The word vectors are high-dimensional dense word representations that capture both global and local contexts. Our implementation of the CNNs is inspired by the model proposed by [30]. Given a review of length n (padded with zeros where necessary for facilitating matrix multiplications), a representation of the text is generated by applying convolution operations over various continuous windows using a *filter* **w**.

In Figure 3, $\mathbf{x}_i$ is the embedded representation of the i^{th} word of sentence s_2, **k** convolution filters are applied, with a final max-over-time pooling operation [10] to take max values per filter. The final concatenation of the max pool outputs is considered as the final *DRR*.

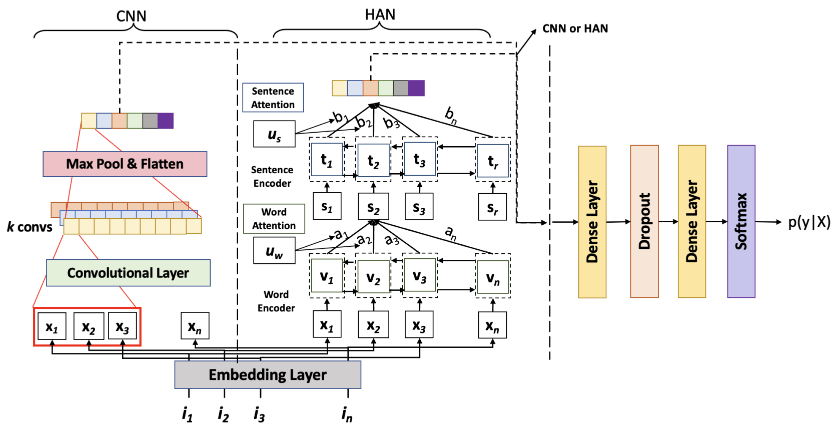


Figure 3: Left: Text encoding with CNN model; Middle: Text encoding with HAN model; Right: Final classification layers after textual representation.

4.2.2 Hierachical Attention Network (HAN)

Our next experiment is more inspired by the structural nature of text “words constitute sentences". [29] proposed a hierarchical attention-based LSTM network for classifying text. The *three components* that improve performance of the model over the plain CNN architecture are the *LSTM* modules, *hierarchical modeling*, and the *attention* modules. As shown in Figure 3-middle, one word is passed as input at each time step, and hidden state representation ($\mathbf{v_t}$) is computed for the next time step i.e. the next word in the LSTM unit.

The *first component* - LSTMs [13] are chosen for their capability of remembering information over long sequences i.e. retaining required information about the 1^{st} word when dealing with the n^{th} word. Since combination of words represent a sentence, the *word encoder* is placed below the sentence encoder (*second component*) as shown in Figure 3-middle. The word encoder encodes the words $i_1 \dots i_n$ to generate an overall representation of a sentence s (s_2 in Figure 3). Once all sentences obtain their representations from the same word encoder model, the sentence encoder generates an overall representation of the review. Sentence encoder is exactly similar to the word encoder, the only difference being sentences are the basic units rather than words. In both word and sentence encoders, we use the BiLSTM [14] variant of LSTMs that capture information in both a left-to-right and right-to-left manner.

The *third component* that improved the performance of HAN is the attention component [8]. A weight matrix ($\mathbf{u_w}$ for word encoder and u_s for sentence encoder) is learnt that computes a score per unit representation, $\mathbf{a_l}$ for l^{th} hidden time step representation $\mathbf{v_l}$ and $\mathbf{b_l}$ for l^{th} hidden time step representation $\mathbf{t_l}$, and computes a weighted sum of hidden representations at each encoder level. The intuition is to weigh certain words and sentences more than others. We will cover this intuition in more detail in the BERT model section. Mathematically, the network can be represented as:

$$\mathbf{v}_l = \overleftrightarrow{\text{LSTM}}(\mathbf{x}_l); \quad \text{a}_l = \mathbf{u_w}.\mathbf{v}_l; \quad \text{s}_k = \sum_{n}^{i=1} \text{a}_i * \mathbf{x}_i \tag{1}$$

$$\mathbf{t}_l = \overleftrightarrow{\text{LSTM}}(\mathbf{s}_l); \quad \text{b}_l = \mathbf{u_s}.\mathbf{t}_l; \quad DRR = \sum_{r}^{i=1} \text{b}_i * \mathbf{t}_i \tag{2}$$

In Figure 3-middle, $\mathbf{x}_1, \mathbf{x}_2, \dots \mathbf{x}_n$ are the embedded words of a sentence $\mathbf{s}_2$, $\mathbf{s}_1, \mathbf{s}_2, \dots \mathbf{s}_r$ are the encoded representations of the sentences of a review, $\overleftrightarrow{\text{LSTM}}$ is the Bi-directional LSTM module, and *DRR* is the final deep representation of the review text.

4.2.3 Deep Bidirectional Transformers - BERT

Another attempt we made in generating *DRR* is the state-of-the-art language model BERT [11]. Although newer variants of BERT like XLNET [28], RoBERTa [18],…achieve a better performance, we focused our efforts more on explainability of the model rather than quantitative improvements.

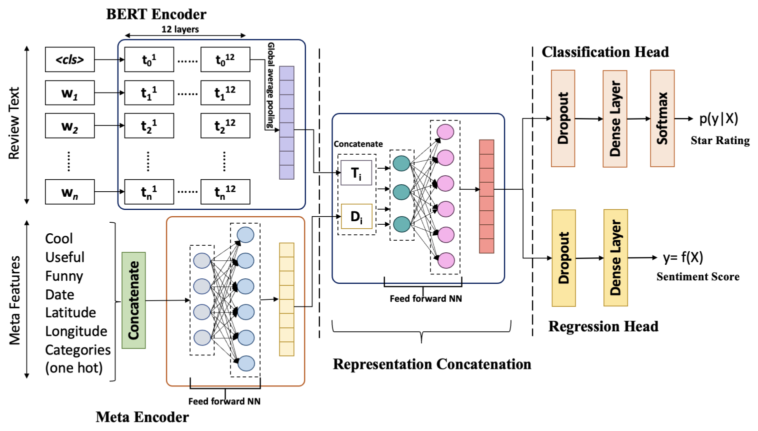


Figure 4: Top-Left: BERT encoder with bert-base-uncased backend; Bottom-Left: Encoding meta features through FNN; Middle: Concatenating text and meta representation and feeding to FNN; Top-right: Star rating classifier head; Bottom-right: Sentiment score regression head.

We briefly explain the intuitions behind the architecture and its applicability in our context. We omit an exhaustive background description of the transformer units and the BERT architecture and refer readers to [26], [11] as well as a detailed guide [5].

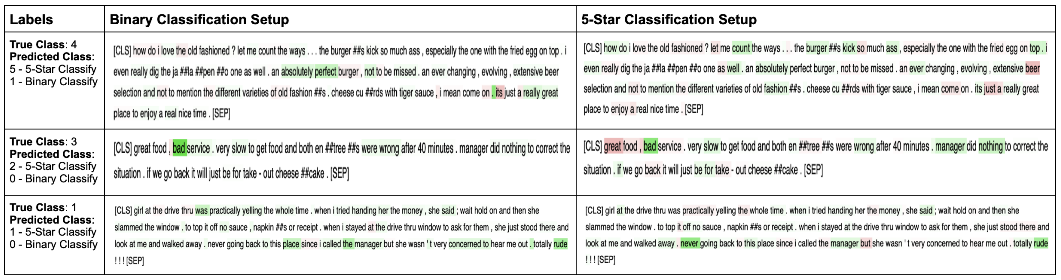
BERT is a language representation model, where deep bidirectional representation from unlabeled text is pre-trained on both left and the right context for masked language model and next sentence prediction tasks. Such a model, when pre-trained on large corpora of natural language texts can be used for wide tasks after being fine-tuned with just one additional output layer since the initial parameters of the model are optimized to learn general English language (language-specific BERT models are also available). The architecture of BERT allows for learning contextual word embeddings based on the *context* by *attending* to the other words in the input sentence. The attention mechanism enforces the model to assign a higher weightage to words that are relatively more important for the downstream task.

In the standard implementation of BERT, the input is concatenated with a special token $<cls>$ that captures the entire sentence representation after L self-attended transformer layers. We consider the Global Average Pooled output of the $< cls >$ representation as the dense representation of our review text. In the Figure 4, the **BERT ENCODER** would generate a representation $\mathbf{T_i}$ for i^{th} review text with words $\mathbf{w}_1, \mathbf{w}_2 \ldots, \mathbf{w}_n$, which when fed to the **CLASSIFICATION HEAD** would predict the probability of the review text belonging to a star rating class. Our intuition behind choosing attention-based models HAN and BERT among many competing deep learning architectures is because of our final intuition *I3: Are there some signal words that indicate the sentiment of review and hence concentrating more on these terms would improve the classification performance?*

4.3 Multi-Task Learning

To reiterate, due to the strong positive correlation between the sentiment scores and the star rating, we refrained from using the sentiment score as an input to the machine learning models as it can cause data leak (final label) into the model. However, we could still utilize the sentiment scores to boost the performance of the aforementioned deep models through the concept of Multi-task learning(MTL) [9]. MTL states that joint learning of correlated tasks would improve the model by learning the shared representations in a joint space that would lead to a better generalization, in addition to boosting up the performance of one/both the tasks.

Towards this end, we propose our *first novel change* on top of the deep models by adding a classification head and regression head tasks for predicting the star rating class and estimating the sentiment score respectively. To facilitate this, in the Figure 3, we add the **CLASSIFICATION** and **REGRESSION HEADS** from Figure 4 and pass the representation generating from CNN or HAN model to both the heads. For the joint loss optimization, we experiment with 2 variants. A simple variant is a weighted loss $\lambda * \mathcal{L}_1 + (1 - \lambda) * \mathcal{L}_2$ where λ is a hyperparameter. The second variant is the same as the weighted loss, with the exception that λ is learned along with the model. We employ



Labels	Binary Classification Setup	5-Star Classification Setup
True Class: 4 **Predicted Class**: 5 - 5-Star Classify 1 - Binary Classify	[CLS] how do i love the old fashioned ? let me count the ways . . . the burger ##s kick so much ass , especially the one with the fried egg on top . i even really dig the ja ##la ##pen ##o one as well . an absolutely perfect burger , not to be missed . an ever changing , evolving , extensive beer selection and not to mention the different varieties of old fashion ##s . cheese cu ##rds with tiger sauce , i mean come on . its just a really great place to enjoy a real nice time . [SEP]	[CLS] how do i love the old fashioned ? let me count the ways . . . the burger ##s kick so much ass , especially the one with the fried egg on top . i even really dig the ja ##la ##pen ##o one as well . an absolutely perfect burger , not to be missed . an ever changing , evolving , extensive beer selection and not to mention the different varieties of old fashion ##s . cheese cu ##rds with tiger sauce , i mean come on . its just a really great place to enjoy a real nice time . [SEP]
True Class: 3 **Predicted Class**: 2 - 5-Star Classify 0 - Binary Classify	[CLS] great food , bad service . very slow to get food and both en ##tree ##s were wrong after 40 minutes . manager did nothing to correct the situation . if we go back it will just be for take - out cheese ##cake . [SEP]	[CLS] great food , bad service . very slow to get food and both en ##tree ##s were wrong after 40 minutes . manager did nothing to correct the situation . if we go back it will just be for take - out cheese ##cake . [SEP]
True Class: 1 **Predicted Class**: 1 - 5-Star Classify 0 - Binary Classify	[CLS] girl at the drive thru was practically yelling the whole time . when i tried handing her the money , she said ; wait hold on and then she slammed the window . to top it off no sauce , napkin ##s or receipt . when i stayed at the drive thru window to ask for them , she just stood there and look at me and walked away . never going back to this place since i called the manager but she wasn ' t very concerned to hear me out . totally rude ! ! ! [SEP]	[CLS] girl at the drive thru was practically yelling the whole time . when i tried handing her the money , she said ; wait hold on and then she slammed the window . to top it off no sauce , napkin ##s or receipt . when i stayed at the drive thru window to ask for them , she just stood there and look at me and walked away . never going back to this place since i called the manager but she wasn ' t very concerned to hear me out . totally rude ! ! ! [SEP]

Figure 5: Qualitative examples with ground truth label and the BERT-MT-Joint model prediction in both binary and 5-star setup. Green color indicates that the word is weighted more in the correct class prediction (*not positive class prediction*), and red indicate more weight in the opposite class prediction. Please zoom in for more clear visibility of the picture.

an uncertainty-based weighing of losses technique proposed in [15]. It is defined as

$$\mathcal{L}_{joint} = \frac{1}{2\sigma_1^2}\mathcal{L}_1 + \frac{1}{2\sigma_2^2}\mathcal{L}_2 + \log(\sigma_1\sigma_2) \tag{3}$$

where σ_1, σ_2 are learnable parameters and $\mathcal{L}_1, \mathcal{L}_2$ are classification loss, cross-entropy, and regression losse, mean squared error, respectively.

4.4 Joint Multi-Task Learning

After observing the contribution of both *meta features* and the *review text*, and the improvement when concatenated together in ML models, we experimented with concatenating the meta and text features in the dense models as well. Figure 4 represents our *end-to-end novel architecture* with all the above mentioned modules. The overall pipeline is as follows: The text is encoded through the BERT encoder, the meta features are concatenated and passed through a feed-forward neural network (FNN) to generate the meta representation. The two representations, $\mathbf{T_i}, \mathbf{D_i}$, are then concatenated and passed through a FNN to learn the joint representation. Finally, the join representation is fed to the classification head (single task) or both heads (multi-task). Exact changes of Meta encoder and joint representation module are made to the LSTM and HAN models as well in Figure 3.

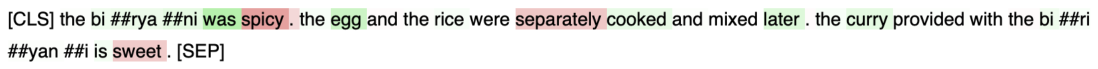
4.5 Model Configurations

We use the following hyperparameter configurations for the various ML baselines tested in this work: **Decision Trees** - *gini entropy, no maximum depth*, **RF** - *100 estimators, gini entropy, no max depth*, **SVM** - *rbf kernel with C value 1.0*, **Gradient boosting** - *learning rate 0.1, max depth of 5, max features of 0.5*, **KNN** - *K=10*, **MLP** - *2 hidden layers with 256, 128 units, learning rate 0.001, adam optimizer*. For deep models CNN and HAN, we use 100-dimensional GloVe embeddings [21] for input words. In the CNN encoder, we use 256 1D filters of size 3 and stride 1. For the HAN model, we set the hidden size of word and sentence encoder LSTM units to 50, which implies that the input to the sentence encoder is of 50 dimensions. For the BERT encoder, we use the base uncased variant which has 12 hidden layers, 12 attention heads and hidden size 768 (embedding dim).

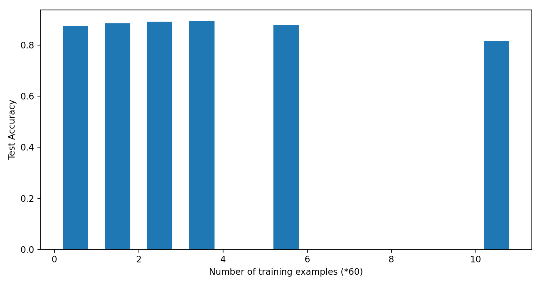
For Meta encoder in Figure 4, we use a single hidden layer FNN with 512 units followed by a dropout layer. For the concatenation module, we use a single layer FNN with 256 units. The dense layers in both classification and regression heads have the number of hidden units equal to the outputs i.e 2 (binary) or 5 (5-star) for star rating prediction and 1 for sentiment score estimation. Loss $\mathcal{L}_1$ is cross-entropy and loss $\mathcal{L}_2$ is mean squared error loss. In all the cases we use RELU as the activation function and use a batch size of 32. The maximum sequence length is set to 50 for CNN and HAN, and 128 for BERT. We use Adam [16] optimizer for CNN and HAN with learning rate 1e-4 and train for 5 epochs, and use AdamW [20] optimizer with learning rate 5e-5 and epsilon 1e-8 and fine-tune for 3 epochs in BERT variants.

5 Results and Model Interpretability

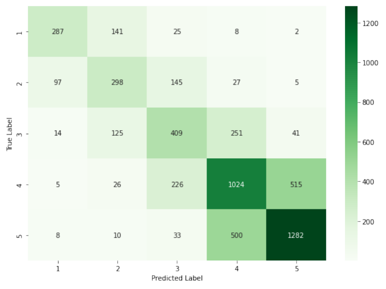
In this section, we first provide the results of various models in different setups. In each of the model setups, we vary one or more of the input features, classification task, objective functions. While we


[CLS] the bi ##rya ##ni was spicy . the egg and the rice were separately cooked and mixed later . the curry provided with the bi ##ri ##yan ##i is sweet . [SEP]

(a)



(b)



(c)

Figure 7: (a)An example where the BERT-MT is not able to classify correctly. (b) Test accuracy values of BERT-MT-Joint model on restaurants with different number of datapoints in the training set, (c) Confusion matrix in 5-star classification setup. F1 values are implied from the confusion matrix and hence we don't repeat in Table 1.

report the results for both 5-star and binary star classification setups, the reader can assume the binary setup unless explicitly stated. Later, we provide an in-depth qualitative analysis of the models to evaluate the **intuitions** which we have specified earlier. We report the complete results of all setups in Table 1. In Table 1, Joint indicates meta features + review text as the input, 0/1 indicates binary mode and 5 indicates 5-star mode classification, and -MT indicates multi-tasked with sentiment score. The BERT-MT model outperforms all other models in all settings. However, multi-tasking when used together with Joint representation didn't improve the performance as expected, in fact in a couple of cases (CNN, BERT) it marginally decreased as well. We hypothesize that this is due to the increased zero or negative correlation that the sentiment score brings in when multi-tasked with star rating (Figure 1). For all the follow-on experiments, we use the BERT-MT-Joint model or BERT-MT model if meta features are not available.

	Model	Input Features					
		Meta Features		Review Text		Joint	
		0/1	5	0/1	5	0/1	5
ML models	Decision Trees	65.42	35.96	70.3	39.32	72.66	43.5
	RF Classifier	68.80	40.21	74.1	48.51	75.17	53.47
	KNN	63.74	34.2	69.5	38.5	70.92	46.42
	SVM	**71.77**	**43.25**	75.72	53.56	77.46	55.10
	Gradient Boosting	70.05	42.34	73.19	51.56	75.50	53.33
	MLP	69.99	41.99	73.22	51.25	75.85	52.01
Dense models	CNN	–	–	81.17	56.01	82.25	56.31
	CNN-MT	–	–	82.32	56.99	82.40	56.80
	HAN	–	–	84.59	58.33	85.98	59.25
	HAN-MT	–	–	85.91	59.14	85.95	59.21
	BERT	–	–	86.44	59.96	87.46	60.87
	BERT-MT	–	–	**87.81**	**61.22**	**87.82**	**61.20**

Table 1: Test accuracy values in various setups

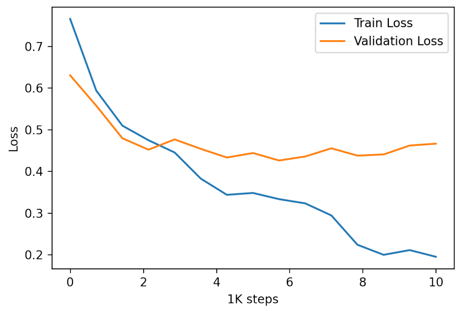


Figure 6: Loss curves

***I1**: Impact of the meta features and the review text individually and jointly on the classification task* – From the table, we can observe that both meta and text features are important in both 5-star and binary classification setups, with text features resulting in a better performance. When fed jointly, all the models report a better performance when compared to using either of the features alone. For deep models, since removing text reduces the model to a simple FNN, we report results for with text and with text + meta features.

***I2**: Impact of the deep high dimensional representations of the review text in the classification performance* – When compared against the simple tf-idf representation, the dense models achieve significantly better performance across all setups. Among the competing deep learning models, the contextual deep representation by the BERT model wins.

***I3**: Are there some signal words that indicate the sentiment of the review and hence concentrating more on these terms would improve the classification performance?* – Since there is no labeled dataset to generate quantitative results, we qualitatively analyzed the attention weights using the inverted gradients technique. Briefly, the IG method computes the input feature value multiplied by its gradient w.r.t loss i.e contribution of this word towards to either class prediction probability, for more details please refer to [24]. In Figure 5, we report results for randomly chosen examples with different star ratings. As observed in the 2^{nd} column, for classifying the 4-star example, the model concentrated more on words like “absolute, perfect, really, great" and words like “bad, rude" in 3-star and 1-star examples. Note that in binary classification, we attribute the 3-star as class 0 i.e negative sentiment. This experiment proves that there are certain words that act as signals for the classification task and attention-based models are indeed focusing more on the signal terms to achieve better performance.

***Q1**: 5-star vs binary classification* – In this experiment, we wanted to observe the differences that are likely causing the poor performance of the 5-star classification. One can observe in Figure 5 that for binary classification, the model simply concentrated on positive terms like “absolutely perfect" to predict the class as positive. However, in a more nuanced setup, the model is concentrating on additional terms like “kick ass, extensive, . . . " and therefore ended up classifying as 5-star. This difference is more pronounced in the 3-star example (2^{nd} row of Figure 5). For simple negative classification (binary mode), the model was sufficient with the word “bad service". However, in a 5-star setup, the attention is distributed between “great food" and “bad service, nothing", which indeed it is supposed to do, to find the neutrality. Due to assigning more weights to “bad" and “nothing", the model ended up in classifying the example as 2-star. For 1-star example, the sentiment is clear from the terms “rude" and “never" and hence in both binary and 5-star, the model was correct. This experiment proves that although the model is attending properly on the terms required, it is confusing between nearby classes and hence the reduced performance of a 5-star setup. We explore this confusion in more detail shortly.

***Q2**: Model Failure cases* – While we report the superior performance of the BERT-MT-Joint model both quantitatively (Table 1) and qualitatively (Figure 5), to find failure cases, we manually explored by looking at the predictions of the model on a number of Google maps restaurant reviews. Out of the many good performance cases (binary setup), we found one type of example where our model is dominantly failing. In Figure 7a we report one such example. Here the user specified a negative intention through the *method of cooking* rather than using the standard signal words. In these extreme scenarios where an external knowledge is required, probably a good recipe for restaurants dishes, our model is significantly under-performing.

Q3**: Training convergence, bias, and label confusion* – The last quantitative analysis that we perform pertains to observing the training convergence of the model, identifying any training biases, and labeling confusion in the model predictions. Figure 4 clearly shows that our training loss is decreasing with iterations and validation loss is saturated. Although we observe a slight increase in validation loss at the end, we employ an early stopping mechanism to avoid this. In Figure 7b we plot the mean accuracy of performance on restaurants (identified by the *name* attribute) with different number of examples in training. We observe a balanced performance across restaurants that have zero or low number of examples in training vs restaurants that have a significant presence in the training set. Finally in Figure 7c we plot the confusion matrix in the nuanced 5-star classification setup. We observe that the model is mostly confusing between nearby classes, 1 & 2, 2 & 3, 3 & 4, and 4 & 5. We already discussed an intuition behind this confusion earlier in ***Q1.

6 Conclusion and Future Work

From our extensive evaluations, we conclude that (1) jointly using meta and review text is always beneficial, (2) deep representation of text significantly boosts performance, (3) multi-tasking with sentiment score gives additional improvements, (4) attention mechanisms can enhance performance and interpretability (5) current training setup induces no bias in spite of reviews being from a small number of restaurants. The model is well-generalized with balanced performance.

We also point some future directions of exploration: (1) understanding the nuances between the nearby classes and consequently to devise a better performing model for 5-star classification, (2) incorporating external knowledge using knowledge graphs or commonsense reasoning, and (3) generalizing the model beyond restaurant reviews.

References

[1] *Amazon product reviews datset*. https://jmcauley.ucsd.edu/data/amazon/.

[2] *IMDb movie reviews datset*. http://ai.stanford.edu/%7Eamaas/data/sentiment/.

[3] *Scikit-Learn: Machine Learning in Python*. https://scikit-learn.org/.

[4] *SST: Stanford Sentiment Treebank*. https://nlp.stanford.edu/sentiment/treebank.html.

[5] *A Visual Guide to Using BERT for the First Time*. https://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/.

[6] *Yelp Dataset Subset*. https://drive.google.com/file/d/1wM4xcFuU4_Wa8hIP9h8OQqypn8w_tjUI/view.

[7] *Yelp Open Dataset*. https://www.yelp.com/dataset.

[8] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[9] Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

[10] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(ARTICLE):2493–2537, 2011.

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[12] Bhanu Prakash Reddy Guda, Aparna Garimella, and Niyati Chhaya. Empathbert: A bert-based framework for demographic-aware empathy prediction. *arXiv preprint arXiv:2102.00272*, 2021.

[13] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[14] Zhiheng Huang, Wei Xu, and Kai Yu. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*, 2015.

[15] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7482–7491, 2018.

[16] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[17] Siqi Liu. Sentiment analysis of yelp reviews: A comparison of techniques and models. *arXiv preprint arXiv:2004.13851*, 2020.

[18] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[19] Zefang Liu. Yelp review rating prediction: Machine learning and deep learning models. *arXiv preprint arXiv:2012.06690*, 2020.

[20] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[21] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[22] Anand Rajaraman and Jeffrey David Ullman. *Mining of massive datasets*. Cambridge University Press, 2011.

[23] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.

[24] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International Conference on Machine Learning*, pages 3319–3328. PMLR, 2017.

[25] Tan Thongtan and Tanasanee Phienthrakul. Sentiment classification using document embeddings trained with cosine similarity. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 407–414, 2019.

[26] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[27] Sida I Wang and Christopher D Manning. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 90–94, 2012.

[28] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32, 2019.

[29] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard 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.

[30] Ye Zhang and Byron Wallace. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820*, 2015.