Build a recommender system for users to live better with Yelp

Jiachen Zhou 901091 Xintong Yao 991477 Yiran Wang 987751 Ruoyu Wu 947132

Abstract

This report will attempt both content-based and Natural Language Processing **(NLP)** content-based recommender systems and select the best model to recommend individualized businesses for Yelp users on Yelp dataset. **Key Methods applied**: Stochastic Gradient Descent **(SGD)** Baseline, Random Forest Regressor **(RFR)**, Basic Recurrent Neural Network **(RNN)**, Long short-term Memory **(LSTM)** and Bidirectional Encoder Representations from Transformers **(BERT)**. **Key Evaluation Metrics**: Root-Mean-Square Error **(RMSE)**, Mean Absolute Error **(MAE)**.

1 Introduction

1.1 Background

Yelp as a business review website provides a valuable platform for users to search for their desired business through the ratings and reviews provided by customers. Businesses with higher ratings in turn are able to attract more consumers and enhance their visibility which creates a win-win situation for both parties. In order to better facilitate the matching process between business and users, Yelp also introduced preference filtering functionality to the website, allowing only businesses which meet the individual preferences to surface as search results (Southern, 2019).

1.2 The Problem

Although Yelp provides detailed and thorough business reviews as references for customers, the reality is that users are often overwhelmed by the lengthy, chunky text reviews. It becomes very time-consuming to read through the texts, which defeats the original purpose of Yelp. Moreover, Yelp only recommends a generic top n most trendy businesses to all users rather than providing an individualized list. The current filtering preference settings also fail to address indecisive customers that might be hesitant about the initial business category.

Therefore, the report aims to build a better recommender system for Yelp users to maximize their satisfaction. It also benefits both the Yelp company and businesses for profit earning.

Hence, a recommender system is proposed to be built in this project to offer users more precise and intelligent business recommendations.

2 Dataset and Attribute Selection

2.1 Yelp Dataset

The dataset is directly obtained from the official Yelp website (Yelp, 2019) or from Kaggle (Kaggle, 2020). Due to the hardware limitation, 12.5% of the data (around one million instances) was extracted from the original dataset as the final dataset which satisfies both data diversity and quantity sufficiency.

2.2 Choice of Attributes

The SQL-like dataset (linked by IDs) is composed of five JSON files. Users, businesses and reviews contain ratings are chosen to represent a typical structure of a recommender system for this project. (Ricci et al., 2010; Yelp, 2020)

2.2.1 Review

- **“review_text” (a.k.a.** “text_ review”) – The NLP-based attribute describes what the user or a similar user thinks of a business. Therefore, to use this attribute, an assumption is that the predicting business should have at least one review.
- **“stars”** – It is the integer rating (from 1 to 5) that a user scored a business based on the text review. This is also the target attribute to be predicted.

2.2.2 Users

- **“user_id”** – This attribute is used for merging user information to the attributes in Section 2.2.1. It can also uniquely link to meaningful a user’s name.
- **“user_review_count”** – An integer that describes the number of reviews a particular user has written. The reason for choosing this attribute is that a user with more reviews may tend to leave a fairer review.
- **“user_elite”** – It is an array of integers that illustrates the years the user was elite. Similarly, an elite user may have more experience and more fair justification for businesses.
- **“user_average_stars”** – The average rating of all reviews related to the user.
- **“user_total_compliments”** – The total number of all types of compliments (e.g. hot, cool, funny etc.) that the user received.

2.2.3 Business

- **“business_id”** – Similar to the user id in Section 2.2.2, business id also helps merge and locate a unique business.
- **“business_stars”** – Each business has a float type star rating (round to half-stars) to reflect the overall score made by all users who commented. This information is important as it serves as users’ first impression of the business and may directly impact their decision.

3 Data Preprocessing

The text review in Section 2.2.1 is defined as **NLP data**, while the attributes in Section 2.2.2 and 2.2.3 except for IDs are defined as the **metadata**. The dataset was separated into 67% for training and 33% for validation.

3.1 Feature Engineering

3.1.1 Metadata

Some feature engineering details have been mentioned in Section 2.2.2. What’s more, “user_elite” is converted to an integer that shows the years of the user being elite. Since the particular years a user was active may not be as important as his active period of time (record in how many years).

The reason behind engineering all types of compliments is to reduce the dimension of x-axis attributes.

3.1.2 NLP data

All strings are converted to lower case to avoid two exact words being recognized as different ones merely due to casing. An **assumption** for this process is that only reviews written in English are studied and processed. Words are then converted to tokens by NLTK (Bird et al., 2009) to be able to revert words to their word stems by Porter stemmer (Cambridge-University, 2009) to reduce noise.

3.2 Data Cleaning

3.2.1 Metadata

After examining Metadata, no missing values and extreme values seems to be observed. For example, the elite years are not duplicated and are all within the boundary between the start year and the current year for any user. The users' average stars are also all bounded by 1 to 5 and calculated correctly. Since all data should be saved in the backend Yelp database, it will be collected by fixed rules and calculated by correct algorithms online. Moreover, no outliers were discovered in the metadata.

3.2.2 NLP data

Although all reviews are non-empty, some of them, such as non-English reviews, are still considered as missing values hence dropped due to the assumption raised in section 3.1.2. Stopwords provided by Spacy (Honnibal, 2017) are detected and removed since they mostly may not have much meaning (Tutorialspoint, 2020). Websites, symbols (e.g. " ", " | "), punctuations, abbreviations (e.g. " 'll ", " 're ") and entities (e.g. countries, numbers), names were removed as they may be meaningless and noisy.

3.3 Word Embedding for NLP data

Before training, NLP data needs to be embedded as a matrix. Each sentence is represented by a vector. Tensorflow Tokenizer was used to index words uniquely by taking all sentences from the training dataset and the testing dataset respectively then fit each sentence into vectors respectively (Tensorflow, 2020b).

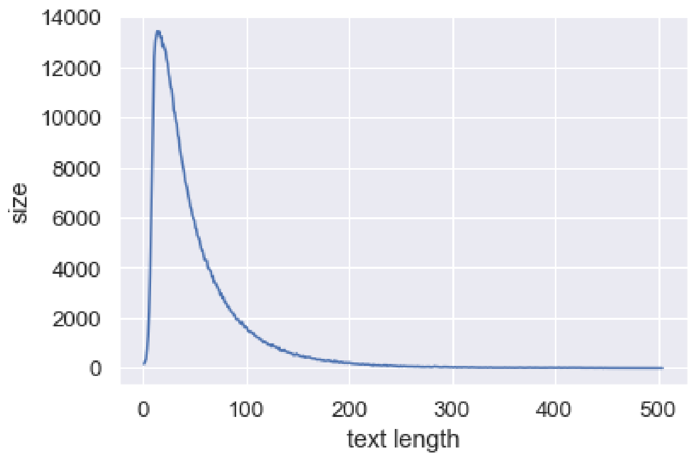


Figure 1: Line plot to show the size of sentences for each possible length

Outliers – Figure 1 shows a right-skewed pattern which means most sentences are within 150 words. Therefore, the first 150 words were taken out for any sentence that exceeds this threshold, to reduce data dimension and improve performance. In addition, the maximum of 50,000 unique words is applied to Tensorflow Tokenizer to reduce the matrix dimension. The remaining words appeared at most 3 times in the whole training dataset and most of them may be meaningless or typo (e.g. "pizzasa"). Hence they are considered as outliers in this project since these words may decrease model performance, while users may hardly use them.

4 Descriptive Analysis

Figure 2 does not exhibit a strong correlation between parameter features and the target feature. It can be found that the business stars may have less impact on final ratings than expected. It may be

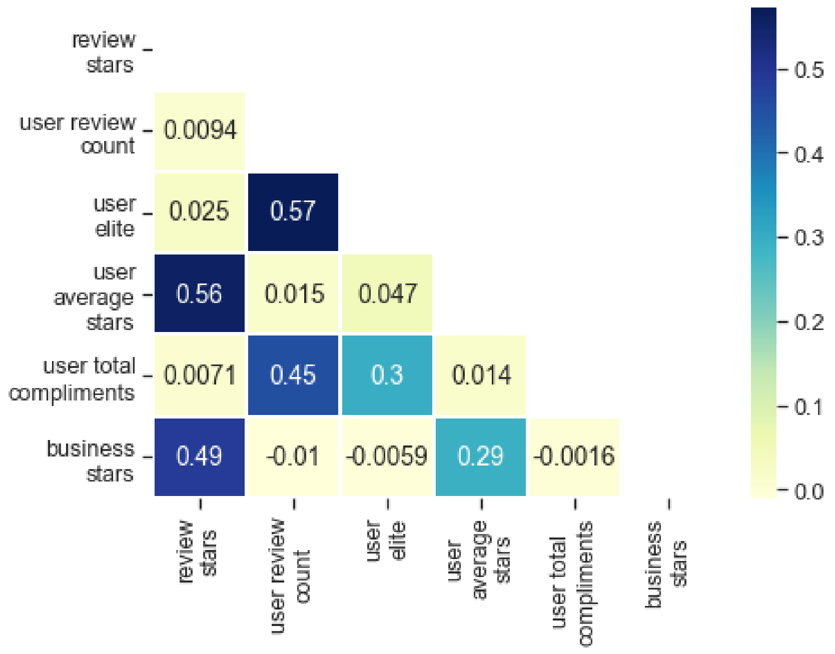


Figure 2: Correlation heatmap between metadata and the target feature (review stars)

more suitable to predict future rating based on the previous user rating. Because users are more likely to visit their favourite type of business with the highest possibility. The user average stars may reveal how harsh a user is on ratings, which can be more crucial than the business stars.

5 Evaluation Metric

5.1 Root-Mean-Square Error

A Root-Mean-Square Error (**RMSE**) has the following equation:

$$RMSE = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2} \tag{1}$$

where n is the number of instances involved, y_j is the actual value and $\hat{y}_j$ is the predicted value. The difference is squared before averaging by n, giving large errors a higher weight. It is useful when trying to eliminate larger errors. The square root was taken outside the sum of squares to the original rating scale and to make the comparison with other metrics easier.

5.2 Mean Absolute Error

Mean Absolute Error (**MAE**) has equal weight on individuals with the following equation.

$$MAE = \frac{1}{n}\sum_{j=1}^{n}|y_j - \hat{y}_j| \tag{2}$$

The variables involved are the same as the ones in Section 5.1. It represents the expected average error magnitude, which has no bias to extreme values. Therefore, a holistic view of the Recommender system can be obtained.

Overall, RMSE and MAE are the main popular measures for evaluating recommender system performance (Rackaitis, 2019). The macro accuracy was not considered in evaluation since we assume the exact star a user will give is not important. We also assume that users may desire to know about the ranking of the businesses of their interests. Overall, the problem is regarded as a supervised regression task rather than classification.

6 Modelling

The same training and testing data in Section 3 are used in all models. Also, cosine similarity will be applied for all models to detect the similarity for user profiles.

6.1 Content-Based Recommender System

A Content-based recommendation system tries to recommend items to users based on their profile , where the profile is the metadata in this project(Aghabozorgi & Santarcangelo, 2020).

6.1.1 Stochastic Gradient Descent (Baseline)

The Stochastic Gradient Descent (SGD) baseline has the following equations:

$$Q(w) = \frac{1}{n}\sum_{i=1}^{n} Q_i(w) \tag{3}$$

$$w := w - \lambda \nabla Q(w) = w - \frac{\lambda}{n}\sum_{i=1}^{n} \nabla Q_i(w) \tag{4}$$

where the parameter w that minimizes Q(w) is to be estimated. The i-th observation in the training dataset is related to each summand function Q_i. (Wikipedia, 2020) The second equation aims to minimize the result in first equation, where λ is the learning rate to update the loss.

The stochastic gradient is uniformly bounded and converges successfully so it satisfies the assumption (Nguyen et al., 2018). This model is not the study focus as it acts as baseline. The learning rate of 0.0001 was applied which gives the result of 1.2774 (RMSE) and 1.4844 (MAE).

6.1.2 Random Forest Regressor

The random forest regressor (RFR) generates multiple decision trees based on the input categories and uses the MSE equation (take square on RMSE equation in section 5.1) to predict.

RFR uses the bagging ensemble algorithm Bootstrap Aggregation with no model underneath, so that the only assumption may apply is that sampling should be representative (Laptev, 2013). This should comply with the assumption since the target only has five values.

Tuning Process – To normalize the attributes does not give an improvement on model performance since RFR may be tolerant of inputs. Grid Search Cross-Validation (GRCV) (Sklearn, 2020) is then used to tune (hyper)parameters automatically. The outcome is to increase the number of trees to 2000 (limit by computer memory) and to decrease the maximum number of features by taking log2.

6.2 NLP Content-Based Recommender System

Similar to Section 6.1, NLP content-based recommender systems recommend items to users based on their reviews. The batch size was set to 128 to control parameters. A larger batch size can improve computational performance but requires more (GPU) memory and may decrease the model performance. The number of epochs was set to three initially.

6.2.1 Basic Recurrent Neural Network (Baseline)

A basic recurrent neural network (**RNN**) takes a word matrix to predict with the following equations (Nabi, 2019):

$$Input : h^{(t)} = \sigma_h(Ux^{(t)} + Wh^{(t-1)} + b) \tag{5}$$

$$Output: y^{(t)} = \sigma_y(Uh^{(t)} + b) \tag{6}$$

where $x^{(t)}$ represents the input word vector, $h^{(t)}$ means the hidden layer inside the model, while $y^{(t)}$ is the output vector which is the predictions in this project. W and U represent parameter matrices while b stands for vector. σ_h (default to tanh in Tensorflow) and σ_y are the activation functions for input and output respectively. The input activation function (tanh) remains as default because it is one of the requirements to use the cuDNN implementation (Tensorflow, 2020a), where cuDNN may be the only way to train models by NVIDIA GPU for Windows users on Tensorflow.

RNNs can model sequences of data so that each sample is assumed to be dependent on previous ones (Mittal, 2019). Since words in one sentence are dependent syntactically and each review of the same business should be related, the above assumption is satisfied. For example, a user may decide to visit a business or leave comments for either satisfaction or dissatisfaction based on other users' reviews.

The baseline model is made up of an embedded word matrix as input, one hidden Dense layer with 16 units, an average pooling layer for smoothing result, and one output Dense layer without the activation function.

6.2.2 Long Short-term Memory

A long short-term memory (LSTM) is a type of RNN and has the same assumption as RNN described in Section 6.2.1 (Tao & Liu, 2018). Compared to the RNN model, LSTM units include a "memory cell" that stores past words and sentences in memory for a long period. The input equation for LSTM is:

$$f^{(t)} = \sigma_g(U^{(f)}x^{(t)} + W^{(f)}h^{(t-1)} + b_f) \tag{7}$$

$$i^{(t)} = \sigma_g(U^{(i)}x^{(t)} + W^{(i)}h^{(t-1)} + b_i) \tag{8}$$

$$o^{(t)} = \sigma_g(U^{(o)}x^{(t)} + W^{(o)}h^{(t-1)} + b_o) \tag{9}$$

$$\tilde{c}^{(t)} = \sigma_c(U^{(c)}x^{(t)} + W^{(c)}h^{(t-1)} + b_c) \tag{10}$$

$$c^{(t)} = f^{(t)} \circ c^{(t-1)} + i^{(t)} \circ \tilde{c}^{(t)} \tag{11}$$

$$h^{(t)} = o^{(t)} \circ \sigma_h c^{(t)} \tag{12}$$

where $f^{(t)}$, $i^{(t)}$ and $o^{(t)}$ represent the forget gate's activation vector, input gate's activation vector and output gate's activation vector respectively with a default sigmoid activation function. A sigmoid activation function has the equation $y = 1/(1 + exp(-x))$ which helps to converge all possibilities to 1 when having a large x value. The initial values for $c^{(0)}$ and $h^{(0)}$ are both zero and the operator $\circ$ denotes the Hadamard product (element-wise product) (Wikipedia-contributors, 2018).

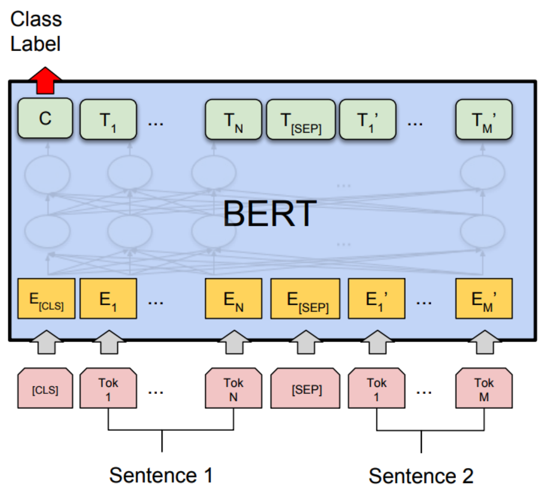
Then a LSTM layer with dimension of 100 replaced the hidden Dense layer in the RNN model.

Tuning Process: A tuning parameter process was accessed to refine the model. By training the same model with ten epochs, we found the validation loss kept decreasing in the first five epochs but increased after that. To achieve the best model performance without overfitting, the number of epochs were reset to five. Also, a dropout rate of 0.2 was added on LSTM to avoid overfitting. Since some predictions may still be negative, a relu activation function was added to the output Dense layer. It has a simple equation which is max(x, 0) that can make the prediction closer to the actual result to improve the model performance. Moreover, a bidirectional LSTM (Bi-LSTM), which is an extension of the traditional LSTM model, was built not only to "see" the past instances but also the coming ones by reversing half of the original LSTM model.

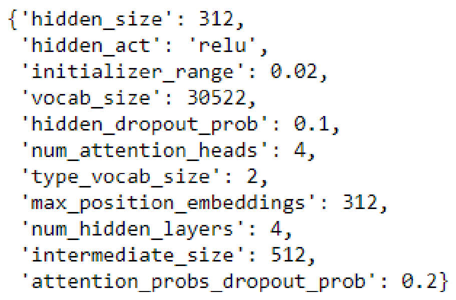
6.2.3 Bidirectional Encoder Representations from Transformers

Bidirectional Encoder Representations from Transformers (BERT) fine-tuning can further analyze the review sentiment. It is designed to do NLP tasks so that the basic assumption should be to take the embedded word matrix as input. Another assumption is made on Next Sentence Prediction (NSP) (Horev, 2018) which is not the intention in this project.

Figure 3 indicates the operation process of BERT which is to separate sentences by [CLS] and [SEP]. Each sentence will then be tokenized (similar to Tensorflow Tokenizer) to build a pre-trained NLP model.



(a) Figure 3: BERT pre-trained model structure (Huang, 2020)



```
{'hidden_size': 312,
 'hidden_act': 'relu',
 'initializer_range': 0.02,
 'vocab_size': 30522,
 'hidden_dropout_prob': 0.1,
 'num_attention_heads': 4,
 'type_vocab_size': 2,
 'max_position_embeddings': 312,
 'num_hidden_layers': 4,
 'intermediate_size': 512,
 'attention_probs_dropout_prob': 0.2}
```

(b) Figure 4: BERT-Custom Model Configuration

The introduced models of BERT-Tiny and BERT-Mini are then built for a start.

Tuning Process – Although jacobdevlin (Google-research, 2019), the author of Bert, recommends 16 or 32 for the batch size to achieve the best model performance, it remains to 128 because a batch size of 32 takes half a day to train the model which is time-consuming for tuning parameters. The tuning process tries to maximize the model size on the limitation of GPU memory.

Figure 4 shows the final custom BERT model. Both attention and hidden layers are limited to 4. Max position embeddings are minimized to avoid meaningless zero in the end of sequences. Dropout rate for both hidden layer and attention heads are set to 0.1 and 0.2 respectively to avoid overfitting. Hidden size was maximized while hidden act changed to relu as mentioned in Section 6.2.2.

7 Analysis of Results

Table 1: Metrics of the selected models

Group	Model	RMSE	MAE
1	SGD	1.2774	1.4844
	RF	1.1968	0.8590
	RF(Tuned)	1.1845	0.8502
2	Basic RNN	0.9721	0.7670
	LSTM	0.7722	0.5718
	LSTM(Tuned)	0.7346	0.5117
	Bi-LSTM(Tuned)	0.7232	0.4899
3	Bert-Tiny	1.2616	1.0497
	Bert-Mini	0.8022	0.5818
	Bert-Custom	0.7365	0.4791

According to Table 1, NLP content-based recommender system (from group 2 and 3) performs better, which concludes that the reviews give more meaning and weight than the metadata for the final rating. In this project, Bi-LSTM and custom BERT work the best with similar model performance. Due to the huge impact of model size configuration, BERT may be expected to have a much better result when working on BERT-Base (the current basic size of BERT which is recommended to run on a GPU with 16GB memory). However, in this case, Bi-LSTM would be chosen as the final model since it uses less computing sources than BERT.

8 Solution to the Existing Problem Based on the Final Model

In fact, the recommendation can be automatically done by the final model. If the model can be embedded into the Backend Database of Yelp, when a source user asks for recommendations, the system can then rank all existing destination users by similarity (including the user him/herself) based on the source user. The destination user with the highest similarity will appear at the top of the rank. The system will then assign the reviews of a particular business for the source user by checking the result of the most similar destination user. For example, the top destination user made comments on business 1, 2, 3. These comments will be permanently added to the source on the same business, meaning it can not be overwritten by different opinions held by the rest destination users with lower similarity. The process will not finish until all businesses are assigned with a review once or all existing users are checked.

Again, the assumption for this model is that all businesses have at least one review. As a result, this source user would possess an array of reviews of all existing businesses. based Although it is impossible for the source user to go through all businesses, this can be achieved by referring to the reviews offered by similar users.

9 Discussion

Both experiments on the modern (BERT) and widely-used (LSTM) models are considered as successful. One regret may be that BERT did not show a dominant performance over LSTM. However, as mentioned in Section 7, if given more time and more computational sources, we may adjust the batch size to 16 and use the newest RTX3090 with 24GB memory to train BERT-Base or even BERT-Large to maximize BERT performance. We can also stack BERT as a pure encoder on LSTM to make it even better if we have more team members and computational sources.

10 Conclusion

In conclusion, adding NLP data on the content-based recommender system by RNN highly improves the model performance. Users do not need to leave comments to get recommendations as long as the business has at least one review.

The main recommendation for Yelp is to embed the final model into the database or improve it based on the advice given in Section 9. The final algorithm in Section 8 offers instruction on how to use the final model. Since the model has low errors, the predictions are regarded as reliable. This is a highly automated predicting process which requires no actions from the user side. Users are able to obtain the Top-K recommendations, where the number of K can be chosen by users.

On the other hand, users are encouraged to utilize the final model and algorithm as it can ultimately guide them to tailored business recommendations while maximising customer satisfaction. Local businesses on Yelp will also benefit from increased sales as the website has successfully engaged customers. As a result, more and more users and businesses are likely to return to Yelp when selecting the desired services, driving Yelp as a platform to grow and expand with a larger customer base. Overall, the final model hopefully can create a sustainable ecosystem among users, businesses and Yelp.

References

Aghabozorgi, S., & Santarcangelo, J. (2020). *Content-based recommender systems - recommender systems*. Retrieved October 31, 2020, from https://www.coursera.org/lecture/machine-learning-with-python/content-based-recommender-systems-jPrfc

Bird, Steven, Loper, E., & Klein, E. (2009). *Natural language toolkit — nltk 3.4.4 documentation*. https://www.nltk.org/

Cambridge-University. (2009). *Stemming and lemmatization*. https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html

Google-research. (2019). *Google-research/bert*. https://github.com/google-research/bert

Honnibal, M. (2017). *Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing*. https://spacy.io/

Horev, R. (2018). *Bert explained: State of the art language model for nlp*. https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270

Huang, J. (2020). *Google colaboratory*. Retrieved October 31, 2020, from https://colab.research.google.com/github/pytorch/tutorials/blob/gh-pages/_downloads/dynamic_quantization_bert_tutorial.ipynb

Kaggle. (2020). *Yelp dataset*. https://www.kaggle.com/yelp-dataset/yelp-dataset

Laptev, D. (2013). *Regression - random forest assumptions*. https://stats.stackexchange.com/questions/59124/random-forest-assumptions

Mittal, A. (2019). *Understanding rnn and lstm*. https://towardsdatascience.com/understanding-rnn-and-lstm-f7cdf6dfc14e

Nabi, J. (2019). *Recurrent neural networks (rnns)*. Retrieved October 31, 2020, from https://towardsdatascience.com/recurrent-neural-networks-rnns-3f06d7653a85

Nguyen, L. M., Nguyen, P. H., van Dijk, M., Richtárik, P., Scheinberg, K., & Takáč, M. (2018). Sgd and hogwild! convergence without the bounded gradients assumption. *arXiv:1802.03801 [cs, math, stat]*. Retrieved October 31, 2020, from https://arxiv.org/abs/1802.03801#:~:text=The%20classical%20convergence%20analysis%20of

Rackaitis, T. (2019). *Evaluating recommender systems: Root means squared error or mean absolute error?* Retrieved October 31, 2020, from https://towardsdatascience.com/evaluating-recommender-systems-root-means-squared-error-or-mean-absolute-error-1744abc2beac

Ricci, F., Rokach, L., & Shapira, B. (2010). Introduction to recommender systems handbook. *Recommender Systems Handbook*, 1–35. https://doi.org/10.1007/978-0-387-85820-3_1

Sklearn. (2020). *Sklearn.model$_s$election.gridsearchcv*. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html?highlight=grid%20search%20cv#sklearn.model_selection.GridSearchCV

Southern. (2019). Yelp introduces personalized search results. search engine journal. *Search Engine Journal*. https://www.searchenginejournal.com/yelp-introduces-personalized-search-results/322727/#close

Tao, F., & Liu, G. (2018). *Advanced lstm: A study about better time dependency modeling in emotion recognition - ieee conference publication*. Retrieved October 31, 2020, from https://ieeexplore.ieee.org/document/8461750

Tensorflow. (2020a). *Tf.keras.layers.lstm | tensorflow core v2.3.0*. Retrieved October 31, 2020, from https://www.tensorflow.org/api_docs/python/tf/keras/layers/LSTM

Tensorflow. (2020b). *Tf.keras.preprocessing.text.tokenizer | tensorflow core v2.3.0*. https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/text/Tokenizer

Tutorialspoint. (2020). *Python - remove stopwords - tutorialspoint*. Retrieved October 31, 2020, from https://www.tutorialspoint.com/python_text_processing/python_remove_stopwords.htm

Wikipedia. (2020). *Stochastic gradient descent*. https://en.wikipedia.org/wiki/Stochastic_gradient_descent

Wikipedia-contributors. (2018). *Long short-term memory*. https://en.wikipedia.org/wiki/Long_short-term_memory

Yelp. (2019). *Yelp dataset*. https://www.yelp.com/dataset

Yelp. (2020). *Yelp dataset*. https://www.yelp.com/dataset/documentation/main