# Extracting Summarizing Sentences from Yelp Reviews

Narek Dshkhunyan — MIT
Saadiyah Husnoo — MIT
John Mikhail — MIT
Dana Mukusheva — MIT

## Abstract

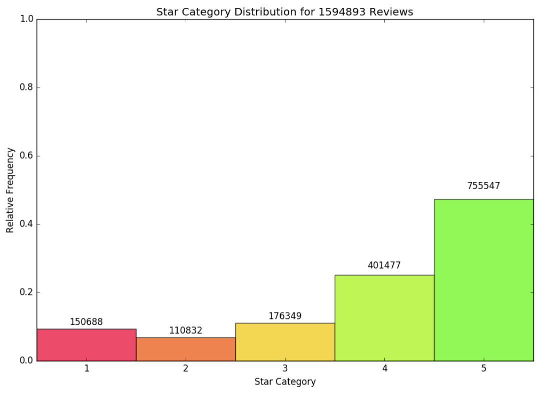
This paper describes an approach to extracting summaries from Yelp reviews. The approach is extractive rather than abstractive. It chooses one or more sentences from the corresponding review that most closely represent the review as a whole, without containing all the sentences. We mainly experiment with two methods. The first uses Naive Bayes to learn important features then picks sentences with those features that are most prevalent. The latter uses Long short-term memory (LSTM) Recurrent Neural Network to predict the number of stars (used inter-changeably with "rating") for every sentence then picks the sentence that has predicted the number of stars closest to the real rating. We provide a comparison of both approaches while highlighting the main differences.

**Keywords:** Extractive, Naive Bayes, LSTM.

## 1 Introduction

Yelp is one of the most used websites for local businesses reviews, in particular restaurants. Yelp users write reviews in varying styles, a lot of which compromise of multiple paragraphs each and use non-standard grammar and slang. It can be time consuming for business owners to extract important information from these long reviews, specially because they tend to contain a lot of unnecessary information. It would be useful for them to have an automatically generated summary of what a review is about. More specifically, they want to know why they were given a one star or two star rating, what customers liked or not... etc. Our approach is to explore ways to condense the review into those sentences that provide the most rich information about the business while excluding those that contain unnecessary information.

We used the dataset provided for participants of the Yelp Dataset Challenge. The dataset contains detailed information about the businesses, users, reviews for businesses and tips. For the purposes of our project, we focused only on the reviews and their ratings (from 1 star to 5 stars). We used review texts as our samples and ratings as our labels. The dataset consists of a total of 2.7 million samples, and the reviews vary in length and style. Due to the nature of the dataset, we have a skewed distribution over the star ratings both before and after the reviews have been filtered. Nearly 50% of the reviews have a 5-star rating, 25% have a 4-star rating and the rest is distributed almost evenly between 1, 2, and 3 star ratings (Figure 1).



= Figure 1: The distribution of star labels in the filtered Yelp dataset is heavily skewed towards 4 and 5 star ratings.

## 2 Related Work

Due to the popularity of the Yelp Dataset Challenge and the availability of its large dataset, there are multiple related projects that attempt to solve sentiment analysis and summarization problem. In this section we will describe the related works and current state-of-the art approaches.

Yu and Chang [3] attempted multiclass sentiment analysis using the Yelp business dataset. They tried two approaches for this task - Recurrent Neural Networks (RNN) with LSTM trained on word2vec, and Convolutional Neural Networks (CNN) trained on GloVe. CNN performed particularly poor in their implementation, the accuracy being near 30% for different paddings, while RNN with LSTM achieved 58% accuracy, which was the highest. This prompted us to not use CNNs for our project and focus on RNNs instead.

Lu and Grafe [6] used multiclass sentiment analysis with the Opentable dataset. The authors have done extensive work on feature selection to accomplish their goal. They removed stopwords, experimented with unigrams, bigrams, and trigrams, implemented an autonomous spelling corrector, and developed a keyword parser. Their highest reported accuracy is under 58% with a combination of uni-, bi-, and trigrams with spell-check and no stopwords, trained on a Support Vector Machine (SVM) model. We used the methods described in this paper as a useful guide during our own feature engineering endeavors.

Sureshra [14] used the entire Yelp dataset for both rating prediction and review summarization. While they also use word2vec for their word representation, they use less than 100K reviews because the authors decided to only use the reviews with more than 10 sentences for the sentence extraction task. Their architecture consists of two layers of Convolution2D units, followed by a dense affine layer and an affine layer, the output of which is fed into an LSTM cell to obtain the sentence vector representation. Finally, the output from the LSTM passes through two affine layers where sentence extraction occurs. The authors achieve less than 33% accuracy on the test set, which might largely be due to the use of a small training set.

M̂eng, Fu, and Wang [8] also tried extractive summarization on the Yelp dataset. For text representation, they explored Bag-Of-Words, Topic, and word2vec models, eventually settling on word2vec. Then they suggest two methods for sentence extraction, which they call the Density-based method and the Graph-based method. To evaluate the extracted sentences, they apply both BLEU and ROUGE metrics, achieving 46% and 54% test accuracy, respectively. We decided to use ROUGE as well in our project.

The next two articles explored state-of-the-art models for abstractive sentence summarization. Rush *et al.* [12] used neural attention models for abstractive sentence summarization. The core of their approach is a feed-forward neural network language model. The authors use an attention-based encoder and a beam-search decoder. They train the model on the annotated Gigaword dataset for both DUC-2003 and DUC-2004 shared tasks, and evaluate the results with ROUGE. This model achieves about 31% accuracy with ROUGE-1 on Gigaword.

Nallapati *et al.* [10] cast the abstractive text summarization problem as a sequence-to-sequence problem and employed Attentional Encoder-Decoder Recurrent Neural Networks for this. In their baseline attention model, the encoder consists of bidirectional Gated Recurrent Unit-RNN (GRU-RNN), while the decoder is a unidirectional GRU-RNN. They have also explored additional variants, such as the Large Vocabulary Trick, Vocabulary Expansion, Feature-rich Encoder, Switching Generator/Pointer, and Hierarchical Attention models. We refer the reader to the original paper for more detailed explanation of these concepts. They used the annotated Gigaword corpus for experiments, trained their models with word2vec vectors, implemented beam search for the decoder, and evaluated the summaries using ROUGE. Overall, the highest accuracy they reached is just under 37% for the test set.

## 3 Technical Approach

### 3.1 Dataset

We chose three text representations to be used for sentiment analysis and summarization. They are Bag of Words (BoW), Term Frequency-Inverse Document Frequency (TF-IDF) and word2vec.

Bag of Words is the simplest representation that creates a vector of size $|V|$, where $V$ is the vocabulary containing all words in the corpus. The entry $i$ of the vector contains the count of the $i$-th word in the vocabulary. The main disadvantage of this representation is that it does not capture the importance of the words and does not take into account the notion of order among the words. The latter is of importance in our case as illustated by the fact that the following two phrases have the same BoW representation: good not bad, bad not good.

The TF-IDF representation is a slightly more sophisticated representation that uses word weights instead of word counts, where the weights are higher for empirically more informative words.

Finally, Word2vec [9] is a representation that embeds the words into a multidimensional vector space, while preserving semantic relations between the words.

In our project, we made use of an open-source pre-trained word2vec model trained on the Google News dataset, which contains 300-dimensional vectors for 3 million words [1]. Since our dataset contains unedited raw texts, they contain many misspelled words, slang words, typos, abbreviations, etc., for which the word2vec model does not have a suitable vector representation. We initially decided to use one randomly generated embedding for all words not contained in our trained model, but due to their sheer number, many reviews tended towards similar ratings, despite being very different. Hence, we instead opted to filter the reviews and retain only those in which all the words were present in our word2vec model, which left us a total of 1.5 million reviews.

Additionally, we experimented with including and excluding stopwords, using n-gram models (n=1 and 2 only due to the limits on our hardware) and stemming the words. We have also run experiments with both uniformized data (uniform distribution over the labels) and data having the original skew.

### 3.2 Rating prediction

In the first part of our project, we experimented with sentiment analysis techniques in order to predict the review rating and later apply them for the summarization tasks. In this section we will discuss our choices for the baselines to be used as benchmarks and our technical approach.

#### 3.2.1 Baselines

The first, most natural baseline for the review rating prediction is to predict a particular category (a number) regardless of the content. This way, if we leave the original star distribution and always predict 5-star rating, the accuracy of the prediction is already ∼50%. If we uniformize the distribution of samples, then we will have roughly ∼20% accuracy.

Another baseline that we used in our project is a Naive Bayes classifier. We made use of the open-source python library scikit-learn [11] as their Naive Bayes implementation was reliable. We used BoW and TF-IDF text representations (unigrams, bigrams, and trigrams), and again also experimented with applying stemming techniques and removing stopwords.

#### 3.2.2 Our method

Our aim was to capture the overall complex sentiment present in the reviews, since some of the reviews contained mixed emotionally-charged words with more weights on some rather than the others. We chose the LSTM RNN model due to its ability to take into account the order in which the words appear. [5]. The architecture of our models included an embeddings layer, that converted the words to their word2vec 300 dimensional representation in the embedding space, an LSTM layer and an output layer with a simple linear activation function. The output of our function was a rating prediction (discrete value) for the input review.

We have also experimented with Bidirectional RNNs [13], which we expected to perform better than a forward-pass RNN, as the latter mostly remembers the most recent words, that happen to be the last words of the review. Bidirectional RNN solves this problem by remembering future information from the future time steps as compared to LSTM that only remembers information from the previous time steps. We have experimented with various dimension (number of units) for the hidden layer, as well as the choice for the optimizer and the loss function used. We used open source python libraries such as Keras [4] and TensorFlow [2].

#### 3.2.3 Evaluation

In order to evaluate the performance of our LSTM RNN classifier, we used metrics such as accuracy, precision, recall and F1 score. We used 10-fold cross validation to ensure stable and consistent results and averaged the scores across each fold to produce our final score.

### 3.3 Summarization

The next task in our pipeline was to perform extractive summarization. We decided that extracting 1-3 sentences that capture the main message of the review will fulfil this task. We have hand-annotated about 400 sentences from 33 randomly selected reviews with at least 10 sentences each by marking the sentences that can potentially serve as a 1-sentence summary. We made sure to include an adequate number of reviews from each of the 5 rating value groups. The hand-annotated samples were used as our gold values to train and evaluate our system on.

#### 3.3.1 Our methods

We have devised two different methods for extracting the summarizing sentences from the reviews. The first method involved selecting the most emotionally charged features and choosing the sentences that contain one or several of them. We obtained the most important features (unigrams, bigrams, and trigrams) from the Naive Bayes classifier by choosing the features with the highest probability given one of the labels and picked the best sentences accordingly.

The second method involved training our LSTM model to predict the rating for each sentence in the review rather than for the entire review. In the training phase, for each review, we labeled all the individual sentences with the star rating of that review. In the testing phase, we fed in each sentence of the test review and chose that sentence, whose predicted review is numerically closest (smallest square distance) to the actual review rating. In order to differentiate between the sentiments of each sentence and to be able to pick the best sentences, we could no longer use the discrete 1-5 ratings as these did not capture subtle rating prediction differences. Many sentences would score the same ratings though varying in usefulness. Hence we needed a continuous value prediction in order to accurately compute an accuracy value and reconfigured our LSTM RNN model to predict a continuous value in the range [1,5] inclusive.

#### 3.3.2 Evaluation

We evaluated the quality of our summaries using the ROUGE-N metric [7]. The ROUGE-N metric computes an n-gram recall between the gold summaries and the extracted summary. However, in our case gold summaries were bits of different information rather than the paraphrasing of the same idea, as we chose different sentences that express related sentiments. ROUGE-N gives higher scores if more n-grams in the generated summary match n-grams from the gold summaries, which is not quite applicable for our case, as the extracted summary and its bigrams are more likely to match only one of the gold summaries, out of the many possible sumamrizing sentences.

Nonetheless, this metric was the most suitable evaluation metric we could find, other than simply determining if the summary sentence we produce is one of the possibly many summary sentences for our review. ROUGE-N is more adaptable and easy to use rather than having to develop a different piece of code when we try out 2 and 3 sentence summaries.

| Distr. over labels | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|
| Non-uniform | 61.5% | 62.6% | 61.5% | 61.4% |
| Uniform | 56.5% | 58.1% | 56.5% | 56.5% |

= Table 1: The results of the experiments with the best performing configuration for Naive Bayes

| Distr. over labels | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|
| Non-uniform | 59.5% | 58.1% | 57.5% | 57.8% |
| Uniform | 40.2% | 37.1% | 41.5% | 39.18% |

= Table 2: The results of the experiments with the best performing configuration for LSTM RNN.

## 4 Experiments

### 4.1 Rating prediction

We have conducted experiments with 40 000 reviews, with a randomly chosen 1% of them serving as a test set. The best results we achieved with Naive Bayes on both uniformized and non-uniform dataset was with using unigrams, including stopwords and after stemming the data. In Table 1 we included the scores for our best-performing Naive Bayes configurations on the test set.

Our LSTM RNN model, implemented with the Keras library, has achieved similar results on the non-uniform data and lower numbers for the uniform data. We have noticed, that running Bidirectional LSTM RNN does not significantly improve the accuracy of the predictions, due to the fact that our input samples are not long enough. Unfortunately, we have not outperformed our baselines. We believe that our model suffers from underfitting since we ran it on a relatively small number of samples due to the limitations of our hardware, and achieved up to 67% accuracy during the training phase. We have included the details of our LSTM model performance in Table 2.

### 4.2 Extracting summarizing sentences.

We have trained our Naive Bayes classifier on 1M samples, using unigrams, bigrams and trigrams features. We have selected the most meaningful features, which have the highest probability within one of the categories. We set the threshold to be 0.02 and the resulting features look as follows :

```
UNIGRAMS:
1: ["bad", "place", "service", "food", "
    worst", "horrible", "closed"],
2: ["great", "place", "good", "service",
    "food", "better", "ok"],
3: ["ok", "great", "service", "food", "
    good"],
4: ["place", "service", "food", "great",
    "good"],
5: ["service", "place", "best", "food", "
    great", "love"]
BIGRAMS:
1: ["don't waste", "bad service", "stay
    away", "ow ow", "customer service", "
    terrible service", "place closed", "
    horrible service"],
2: ["good food", "just ok", "good service
    ", "customer service", "food good", "
    food ok"],
3: ["service good", "good service", "
    pretty good", "good food", "food good
    ", "food ok"],
4: ["food great", "great service", "food
    good", "great food", "good food", "
    good service"],
5: ["highly recommend", "food great", "
    love place", "great food", "great
    service", "service great"]
TRIGRAMS:
1: ["don't waste money", "worst customer
    service", "horrible customer service
    ", "don't' waste time", "ow ow ow", "
    poor customer service"],
2: ["food great service", "food mediocre
    best", "food good service", "service
    good food", "food just ok", "meh ve
    experienced", "ve experienced better
    "],
3: ["food pretty good", "food just ok", "
    service good food", "good food good",
     "food good service"],
4: ["good food great", "good food good",
    "food good service", "great food
    great", "food great service"],
5: ["love love love", "great service
    great", "great customer service", "
    food great service", "great food
    great"]
```

We can see that the most influential features are those that represent emotionally charged words. Therefore, the method picks the most polar sentences, and might not be able to perform well in the cases when the review is overall neutral (3-4 stars) but contains at least highly emotionally loaded sentence. The ROUGE-N scores for such summaries are not consistent and vary from 0.15 to 0.3, and should not be taken as a main evaluation technique due to the reasons discussed in the previous section.

The second method should overcome the issue and not just pick the most emotionally charged sentences but rather get the sentence that communicates the key point of the reviewer that defines the rating. Here is an example of a randomly picked review and its extracted summary.

```
Review:
```

```
Clearly this location does not emphasize
customer service. There were only two
people working during peak hour. One
couldn't get the smoothies straight.
The other greeted what seemed to be an
out of uniform employee who asked him to
get something from the back. In the middle
of a packed store with 8 people each
waiting for multiple smoothies, the guy
stopped making smoothies, went to the back,
came back with some papers and proceeded to
chat it up with his friend. Slow as snails.
No sense of urgency what so ever. Customers
come first. The guy's friend should have
waited for things to die down before he
received what he needed. This place needs
to get it together! You clearly have the
traffic so get some more competent
employees. Smoothie making is not that
difficult!On a side note, it felt like
there was no A/c running which is dangerous
for customers and employees.

Summary:
clearly this location does not emphasize
customer service
```

However, we ran into the issue that the model tends to predict 5 stars for each sentence, regardless of the actual rating of the review. We believe that the root of this issue is again in the underfitting of our data. After training on a larger sample set, the model should learn the patterns and predict the sentence rating more precisely.

## 5 Conclusion

Overall, our project consists of two main goals: multi-class classification and extractive summarization. For both goals, we started first with the more direct, if not brute force approaches, and developed those into more subtle and ”smarter” models.

For classification, we used Mutinomial Naive Bayes as our baseline. While we have experimented with many other classifiers - SVMs with radial basis function kernel, $k$-Nearest Neighbors, Decision Trees, Logistic Regression - none of them offered significantly higher accuracy scores for test sets. Therefore, we based all further analysis on the remarkably faster Naive Bayes (a carefully tuned Random Forest resulted in better accuracy and precision scores for 40K samples, but was too slow to run on all 1M reviews). We experimented with removing stopwords, including bigrams and trigrams, and alternating between BoW and TF-IDF representations. Next, we developed a Recurrent Neural Network classifier with the hope of achieving a higher accuracy score. In particular, we implemented an LSTM RNN model which would not only take into account the word ordering but also solve the vanishing gradients problem. Furthermore, we also implemented a bidirectional LSTM to avoid the bias for most recently seen words.

For extraction purposes, we again started with Naive Bayes as our baseline. We chose the five most important features (that is, uni/bi/trigrams) from both BoW and TF-IDF representations and extracted only those sentences from the reviews which contain at least one such feature. While very simple, this proved to be a very powerful extracting system, on par with an LSTM RNN regressor approach we developed further. For this, we trained our LSTM RNN model to predict each sentence as opposed to the entire review, and then in the testing phase chose the sentence with a predicted start label closest to the actual review rating. Finally, we evaluated summary sentences obtained through both methods with ROUGE-N and analyzed the produced results.

## 6 Future Directions

Over the course of the project, we ran into multiple interesting approaches for different tasks that unfortunately we could not implement due to both time constraints and the lack of computation power at our disposition. One big improvement for multi-class classification using both Naive Bayes and LSTM would be to more carefully engineer our features. The Yelp dataset comes not only with the reviews and their star labels, but also the number of helpful up-votes on each review, the average rating of each user, and so on. These features are likely to be very informative and could increase our accuracy and precision scores. For example, some users tend to rate businesses with a generally higher rating than others and some users have more variance in their ratings in general. Further, we could use more nuanced features, such as the time period during which the review was posted (during certain big holidays such as Thanksgiving and Christmas both the number and types of reviews change), the average ratings of the friends of the user, the length of the review and its correlation with the star label, and so on.

In some papers that we consulted, the authors have used Convolutional Neural Networks instead of/in addition to RNNs. While this did not necessarily result in better scores for them, experimenting with additional architectures could prove potentially useful. In addition, while we exclusively use word2vec for our word embedding representations. GloVe is another popular vector representation that might have performed better in our model.

Finally, while we focused only on extractive summarization for our project, it seems that these days, the state-of-the-art approach is abstractive summarization. We started some initial experiments with open-source seq2seq models, but the scope of that approach was too overwhelming for us to be able to implement a fully functional attention model in the limited amount of time we had. However, while we might not have necessarily gotten better ROUGE scores than what our current extractive methods achieve, it is still plausible the newly generated reviews could be more useful to the business owners than simply the sentences extracted from the existing reviews.

## 7 Team work

All team members were involved in the research phase for data representation, classification and summarization. We collectively wrote and proofread the report and made the presentation. We opted for a pair programming approach, so we would all be involved in all parts of the project but also because our code would only run on two of our four computers.

### 7.1 Narek Dshkhunyan

Narek has written most of the linear classifiers throughout the project (SVM, *k*-Nearest Neighbors, Decision Trees, Logistic Regression, and Random Forest), and tuned our main Naive Bayes for different ngram ranges(uni/ni/trigram) and representations (BOW and TF-IDF). He has implemented some of the initial RNN code in Tensorflow, although it turned out to be too difficult to apply to our current project and was later replaced with similar architecture serving different functions. He has also developed our baseline sentence extraction model, by deriving the most important uni/bi/trigram features from both BOW and TF-IDF representations in Naive Bayes and selecting sentences that contain these features.

### 7.2 Saadiyah Husnoo

Saadiyah initially wrote a script that would set up a database of all the JSON data as these were not fitting in memory, but after filtering the reviews, this script was no longer needed. She then helped in writing and debugging Naive Bayes and LSTMs. She uniformized the filtered reviews used as the data was heavily skewed ratings-wise. She was responsible for annotating all the reviews used for sentence summarization and wrote a script to randomly select reviews based on various criteria (length of sentence, number of sentences and rating), split them into sentences and prompt for a rating. She was also responsible for taking all meeting minutes.

### 7.3 John Mikhail

John was responsible of cleaning and filtering the data. He wrote scripts to provide useful statistics on the content of the data so we can figure out sensible criteria to use for filtering. Additionally, he wrote scripts that filter the reviews based on different factors, including Google News word2vec vocabulary. He also helped writing and debugging both the Naive Bayes classifier and LSTM model. Last but not least, John was responsible for running many of the experiments.

### 7.4 Dana Mukusheva

Dana has developed and implemented the core code functions required for the RNN LSTM model and all related experiments using both Keras and TensorFlow python libraries for the rating prediction and summary extraction tasks. Dana has also helped implement scripts for data representation and the initial code for Naive Bayes.

## References

[1] Google Code: word2vec. `https://code.google.com/archive/p/word2vec/`. Accessed: 2016-12-10.

[2] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[3] Y. April and C. Daryl. Multiclass sentiment prediction using yelp business reviews. 2015.

[4] F. Chollet. keras. `https://github.com/fchollet/keras`, 2015.

[5] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, Nov. 1997.

[6] M. Lee and P. Grafe. Multiclass sentiment analysis with restaurant reviews. 2010.

[7] C.-Y. Lin. Rouge: A package for automatic evaluation of summaries. In S. S. Marie-Francine Moens, editor, *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[8] R. Meng, J. Fu, and M. Wang. Tl;dr? tips generation for yelp reviews.

[9] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[10] R. Nallapati, B. Zhou, Ç. glar Gulçehre, and B. Xiang. Abstractive text summarization using sequence-to-sequence rnns and beyond.

[11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[12] A. M. Rush, S. Chopra, and J. Weston. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*, 2015.

[13] M. Schuster and K. K. Paliwal. Bidirectional recurrent neural networks - signal processing, ieee transactions on. 1997.

[14] S. Sureshra. Summarizing reviews and predicting rating for yelp dataset. 2015.