# AUTO-COMPLETE TOOL FOR YELP REVIEWS

*Submitted by:*
*Raveena Vemula (rv913)*
*Sanjitha Udipi (su472)*

## 1. What did you propose to do? What is the motivation/background?

The primary motivation behind the work was that the current auto-completion systems are typically restricted to search features. A user is faced with an uphill task of unnecessary typing. An autocomplete suggestion mechanism for reviews would certainly help people save their time and efforts. Users who had a very bad or very good experience are the ones sharing their reviews. The majority reviews are lost.

The text of Yelp reviews reveal a more complicated story. Our goal is to provide an autocomplete review mechanism based on the user's preferences and the reviews of restaurants. This will not only save time in the review but also reduce cognitive burden by supporting minimal edits in the review (if any). A user is more likely to be biased in using the words he/her usually uses. We wanted to analyze the text and reveal the underlying subtopics and the terms that contributed the most to the restaurant. We also wanted to analyze a user's review and predict his/her preference for each sub-topic. Based on these two factors we would define a weight and rank our suggestions.

## 2. Explain the data you used and model in detail.

We would be using the open dataset from the Yelp Dataset Challenge. The Yelp dataset has a set of five JSON files:

- Description of the business
- Data relating to business check-ins
- Reviews of the businesses
- Tips
- User data

Below is the frequency of each category in the business dataset.

| | Category | Frequency |
|---|---|---|
| 1 | Restaurants | 26729 |
| 2 | Shopping | 12444 |
| 3 | Education | 877 |
| 4 | Health & Medical | 6106 |
| 5 | Hotels & Travel | 2673 |
| 6 | Religious Organizations | 244 |
| 7 | Public Services & Government | 573 |
| 8 | Beauty & Spas | 7490 |

We considered the restaurant category since it has the highest number of reviews.

Reviews dataset for restaurants category is filtered using Java.
Features included:

| | | |
|---|---|---|
| Review | Restaurant review written by the user. | Text |
| User Id | Identifies the user | Text |
| Business Id | Identifies the business corresponding to the review. | Text |

There are a few alternatives to transform texts before performing an exploratory analysis. We are using the functions within the tm package in the following order:

- Stemming
- Remove URL
- Remove punctuation
- To lowercase
- Remove numbers
- Remove all non-alphanumeric characters
- Strip whitespace

We are not removing stop words as they play an important role for an autocomplete tool. Instead, their frequency is reduced by tf-idf (Term frequency-inverse document frequency).

**Vector Space Model:** VSM is an algebraic model representing textual information as a vector. Term Document Matrix is created which lists all occurrences of words in the corpus, by document. In the TDM, the documents are represented by columns and the terms (or words) by rows. If a word occurs in a particular document, then the matrix entry for corresponding to that row and column is 1, else it is 0 (multiple occurrences within a document are recorded – that is, if a word occurs twice in a document, it is recorded as "2" in the relevant matrix entry).

**Term Frequency – Inverse Document Frequency:** The tf-idf weight is composed by two terms: the first computes the normalized Term Frequency (TF), aka. the number of times a word appears in a document, divided by the total number of words in that document; the second term is the Inverse Document Frequency (IDF), computed as the logarithm of the number of the documents in the corpus divided by the number of documents where the specific term appears.
In our case, certain terms, stop words appear a lot of times but have little importance. Thus we need to weigh down these while scale up the rare ones.

**Latent Dirichlet Allocation (LDA):** We use the Latent Dirichlet Allocation (LDA) factor model to approach the unsupervised learning of factors and topics for the Yelp restaurant review data. This model treats the probability distribution of each document over topics as a

K-parameter hidden random variable rather than a large set of individual parameters (K is the number of hidden topics). We performed LDA for K=10 and generated top 40 terms in each topic.
The LDA algorithm returns an object that contains a lot of information. We are particularly interested in the document to topic assignments, the top terms in each topic and the probabilities associated with each of those terms. Each term is assigned the probability of the topic it belongs to.

**Kneser-Ney Smoothing (KN):** N-gram probabilities are estimated from relative counts of a training corpus. Due to finite training data, count c of word wa might be 0. Thus, probability for the unigram case is zero and undefined for the bigram, trigram and the quadgram case.
We use modified Kneser-Ney (KN), which applies interpolation of target n-gram and lower order distribution, for probability estimation. At the end we interpolate the unigrams with the uniform distribution. The strength of this model is that it sets uni-gram probability proportional to the number of different contexts it appears in, rather than to the number of occurrences of the word. The highest probability P_KN from the n-gram with matching first words yields the next-word prediction wi.

**Katz BackOff Model:** A generative n-gram language model that estimates the conditional probability of a word given its history in the n-gram. It accomplishes this estimation by "backing-off" to models with smaller histories under certain conditions. By doing so, the model with the most reliable information about a given history is used to provide the better results. The most detailed model that can provide sufficiently reliable information about the current context is used. Works well in practice in combination with smoothing.
Whenever data sparsity is an issue, smoothing and backoff model can help performance, and data sparsity is almost always an issue in statistical modeling. Smoothing combined with back-off model significantly improved the performance of our auto-completion tool.

## 3. What did you end up doing?

We have implemented all the models as mentioned in the proposal.
Based on the feedback received for the presentation, in addition to the existing models we implemented Kneser-Ney Smoothing.
We used Jaccard index as a distance metric for comparing similarities between the written review and the test data.

## 4. What if anything did you change about your approach and why?

Initial intuition was to remove stop words but later we realised that stop words are necessary for an auto-complete tool in order to prompt well formed grammatically correct

sentences. So we retained stop words but **Tf-idf** was added to reduce the weightage and hence the importance of stop words.

**KN Smoothing** was performed so as to interpolate the probabilities of n-grams. Smoothing plus a back-off model helped improve the performance of the auto-complete text review.

We had implemented a UI tool using HTML and Javascript but realised that the UI input should be sent back to the server code to predict a word based on the typed input. So we used **Shiny Apps**. Shiny combines the computational power of R with the interactivity of the modern web.

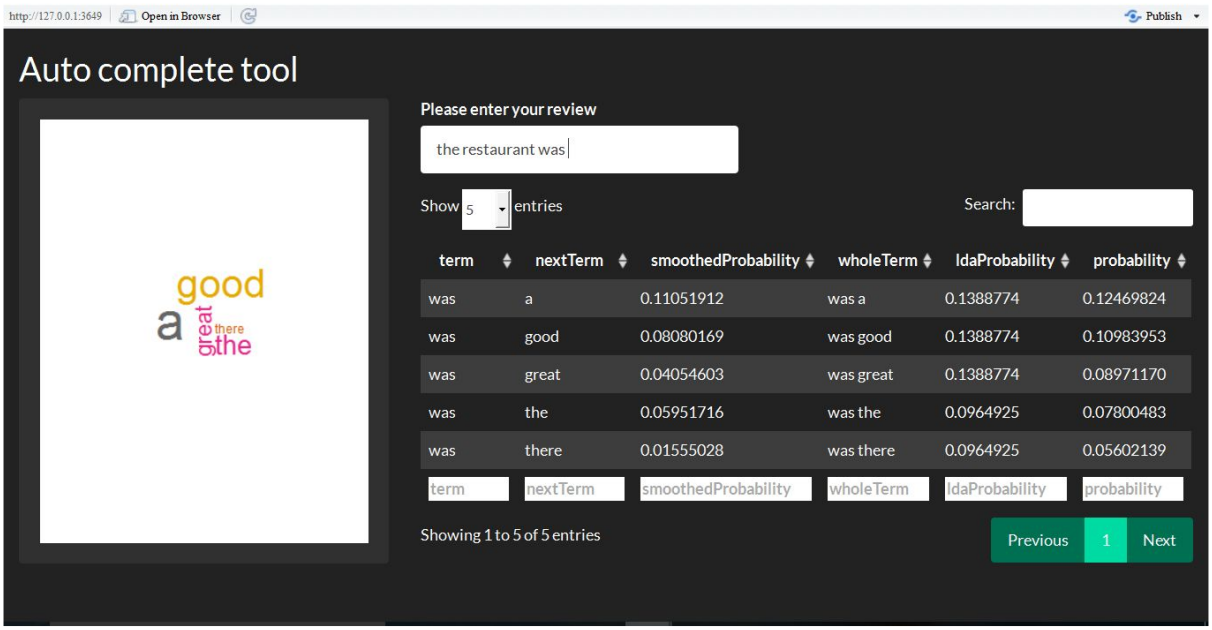
Cosine similarity was the initial approach used to measure the similarity between the test reviews and the generated review by the auto-complete tool. But we used Jaccard index. **Jaccard index** is used for comparing the similarity and diversity of sample sets. Cosine index could be used to identify plagiarism, but will not be a good index to identify mirror sites on the internet. Whereas Jaccard index, will be a good index to identify mirror sites, but not so great at catching plagiarism.

## 5. What visualizations have you included? Explain what is conveyed in the visualization and why.

- Plot to show count of each n-gram in the data: It gives an idea of the total number of n-grams we are dealing with.
- Wordcloud for uni-gram, bi-gram and tri-gram: Shows the top words in each n-gram.
- Plot for top 10 n-grams and their frequencies: Top 10 n-grams.
- Snapshot of smoothed data (top 10 records): Probability of each term and its nextTerm after smoothing.
- Snapshot of topic probabilities for top 10 documents: Output of LDA model.
- Snapshot of top 10 terms under each topic: Gives an idea of how topics are created. Terms in each topic have high probability of occurring together.
- Plot for topic probability for each topic across all the documents: Gives an idea about which topic has high probability of occurrence.
- Plot to depict similarity (evaluation).
- Video of the tool.

## 6. What evaluation method did you propose?

We have divided the dataset in the ratio of 9:1 where 90% is the training set and the 10% belongs to the test set. The input to the tool would be the user's review entered using the UI platform. The screenshot of the UI screen can be seen below:




We perform a similarity analysis between the text produced and the test data to evaluate the performance. Jaccard Similarity Index has been used as an evaluation distance metric.

## 7. How did your model perform according to this evaluation?

The model performed well. The similarity was between 0.6-0.9

## 8. Based on your results what conclusions do you draw?

Autocomplete operation needs to be very fast as it is an interaction with the user and latency should be as small as possible maybe < 5 or 10. Our tool is fast with a query time of <4 ms. We have been able to detect what users care about most in their reviews of restaurants and have been able to pinpoint the areas of interest for a restaurant using LDA. Based on the topics we have discovered using LDA and language modelling, we could predict the next word a user is about to type. On analyzing the reviews we found that the tool prompts the user with the words he is likely to use based on his/her history and preferences.

## 9. Based on your results what further studies would you do or are warranted?

Through future works, we expect to explore more accurate insights by taking into account the votes and time factor. We also intend to provide an autocorrect technology to identify typos and misspelled words.

**REFERENCES:**


http://cseweb.ucsd.edu/~jmcauley/cse255/reports/fa15/035.pdf

https://www.linkedin.com/pulse/termfrequencyinversedocumentsanjaysingh1

http://www.lkozma.net/nlp_report.pdf

http://rstudiopubsstatic.s3.amazonaws.com/96423_1733e504dbe045ab9eebc148e9f762fd.html

http://rstudiopubsstatic.s3.amazonaws.com/139113_83fcd61d53c84672b7d806c985f76909.html

https://rstudiopubsstatic.s3.amazonaws.com/69717_91e500e66f784451a5126c405ceaa738.html

http://dbgroup.eecs.umich.edu/files/fcpaper07.pdf

https://github.com/ThachNgocTran/KatzBackOffModelImplementationInR

https://github.com/arttuK/word-prediction/blob/master/shiny/prediction.R

https://rpubs.com/jbonsak/capstonefinal

https://rpubs.com/BerniePilgram/47782

https://github.com/j-wang/DataScienceCapstone/tree/master/final/shiny-app

https://eight2late.wordpress.com/2015/09/29/a-gentle-introduction-to-topic-modeling-using-r/

https://github.com/groberts0429/Data-Science-Capstone/blob/master/knitr.md

https://github.com/arttuK/word-prediction/blob/master/shiny/prediction.R