Matthew Goldgirsh

IS 4200 Information Retrieval Final Report

Professor Smith

December 12, 2024

Evaluation of Language Models for Query Auto-Completion

Introduction

Whenever a query is typed into a search engine such as Google there are always autocompletions for the query generated. Although autocompletions are sometimes unnoticed by the user this feature is helpful in helping the user construct the perfect query. Autocompletions specifically for search engines can be constructed in many ways: through the popularity of the search term, the location the user is located, or the common frequency of words used in the query. Furthermore, finding different representations of the language model that the autocompletion runs on will produce different relevant autocompleted queries that the user may find useful. This paper will focus on the evaluating the different language models that may be used for the query autocompletion problem (QAC).

This task is important to evaluate because the results will show which language model is the most efficient and will provide the most relevant results to the user. For an autocomplete system to be effective it must be able compute autocompleted queries in a

time efficient manner and give relevant information to the user. To add this feature to the search engine the evaluation of language models for the QAC must be understood.

Methodology

This paper will evaluate 3 separate language models that perform the QAC task and determine which one is best suited to perform query autocompletion. To evaluate these models, the SIGIR 2017 dataset was used. This dataset involves 75,198 prefixes for training a varying different query and provides a dataset for testing involving 32,559 prefixes.

The first language model this paper investigates is the one hot vector embedded language model. This language model uses the idea of one hot vector which are vectors which show the presence of a word or token in the sentence. By using one hot vector to model a query you can keep track of what words are in the partial query and therefore allow comparisons between partial queries and the full query set. The partial queries are turned into one hot vector based on the vocabulary of the dataset and then the dot product is performed between the partial query and the set of full queries to generate the number of similar elements and to determine the highest similarity to produce a full query. Therefore, the one hot vector model is very useful at finding exact similar words that should be present within the autocompleted query.

In contrast to the one hot vector language model a bag of words model with tf-idf vectorizing was also created. The tf-idf metric is used to determine the importance of word in a document within the collection of documents. This metric compared to that of the one hot vector metric is more effective at capturing the context in relation to set of full

autocompleted queries. Because the tf-idf captures more context in the way that the tokens are embedded in the partial query the generated full queries are different from the one hot vectors.

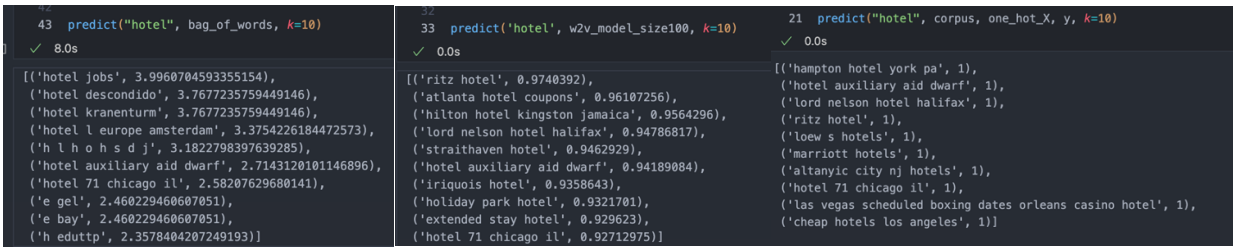
The last language model evaluated was the word2vec model. The word2vec model is a well-known way to represent words or phrases of words in the set size vector space. The word2vec model is the best of both worlds between the one hot vector model and tf-idf model where it can capture the context between words as well as exactly capture the words used like the one hot vector model. Furthermore, the word2vec model uses a LSTM architecture to train the network which again captures the context of the words to generate the full query that would work best within the context.

Experimentation

First the various models were trained with the train dataset so that they could be evaluated with the test dataset. To evaluate the models, it was important to return the full completed query in order of highest score metric. Furthermore, the top 10 scores were analyzed in terms of binary relevance manually based on the context provided and the partial query given.

Results and Analysis

To obtain the results of the model a sample partial query of “hotel” was used to determine what full queries should be autocompleted.



Bag of Words Prediction Word2Vec Prediction One Hot Vector Prediction

(taken from https://github.com/mgoldgirsh/autocomplete-evaluation)

These results show that both the one hot vector prediction and the word2vec prediction all generate 10 relevant results when used with the partial query of “hotel”. However, the bag of words prediction with tf-idf tries to obtain the context of the partial query to much and has one 6 relevant results out of the 10 results asked for. Therefore, from this sample evaluation with the sample partial query of hotel it is evident that each model has different results that certainly affect autocomplete. Furthermore, there is a time efficiency evaluation that needs to be evaluated for query autocompletion. It should be noted that the bag of words models with tf-idf weights has a slower wait time to generate autocomplete predictions whilst the other two models do not have the slow wait time. Therefore, the analysis shows the drawbacks of using the bag of words model and the hallmarks of the word2vec model and the one hot vector model in comparison to it. However, the one hot vector model is rudimentary in the fact it only looks for exact matches while the word2vec model looks at the context as well making it the superior choice for the QAC task.

Conclusion and Future Considerations

Overall, from this paper there was a relevance analysis made off 3 language models and relevance judgements were performed on a sample prefix. The goals set in my grade contract were certainly met. I was able to train 3 separate models that were different from each other and exploited different features required for the QAC task. Furthermore, I was able to achieve the evaluation part of my grade contract as I was able to analyze the relevance judgements of the three separate models on a predetermined dataset of partial queries.

In the future I plan to work on implementing the LambdaMART model which is a predetermined model on how autocompletion systems work. This model is based on a LSTM architecture that can classify partial queries into groups and generate full queries that are like those groups with the specified context. Overall this paper was important in providing me with an understanding into how autocompletion system work and which systems are more effective than other systems.

Works Cited

GitHub Repo: https://github.com/mgoldgirsh/autocomplete-evaluation

Dataset: “A neural language model for query auto-completion”
https://sites.google.com/site/daehpark/Resources/data-set-for-query-auto-completion-sigir-2017

Word2Vec:
https://radimrehurek.com/gensim/models/word2vec.html#gensim.models.word2vec.Word2Vec.predict_output_word

Sample Queries and Relevance Judgements

Sample 1

Narrative: You are trying to find what Hilton hotels are near the Washington DC area.

Partial Query: Hotel

Responses: Hilton Hotel in Arlington VA (1), Hilton Hotel in Balitmore MD (1), Fairfax VA hotels (1), Hyatt Hotel in Hyde Park (0)

Annotations: The responses for the above partial query and narrative pair are examples of both relevant and non-relevant results. This example was taken from the sample query used in the Github project and it shows how the autocomplete system still generates results that are out of context ("Hyatt Hotel in Hyde Park") of the narrative.

(taken from https://github.com/mgoldgirsh/autocomplete-evaluation)

Sample 2

Narrative: You are trying to find specifically high schools in/near New York

Partial Query: School

Responses: Ithaca High School (1), School Dress Code (0), High School Musical (0), Eastern State School (1), American State School (1)

Annotations: The responses for the above partial query and narrative pair are examples of both relevant and non-relevant results. This example was taken from the sample query used in the datasets training data and it shows how the autocomplete system still generates results that are out of context of the narrative even though they contain the partial.

(taken from https://sites.google.com/site/daehpark/Resources/data-set-for-query-auto-completion-sigir-2017)

Sample 3

Narrative: You are trying to learn more about natural language processing techniques and specifically natural language common architectures

Partial Query: Natural Language

Responses: Natural Language Prcoessing (1), Natural Language Prcoessing Examples (1), Natural Language (0), NLTK, Natural Language ToolKit (1)

Annotations: This example was generated using Google's query autocomplete system. Google has its own algorithm for autocompleting queries and Google search is a prime example of their QAC system at play. This partial query was searched with the narrative in mind the responses from the autocomplete were recorded.

(taken from Google)

Sample 4

Narrative: You are trying to access your grades on the Northeastern Canvas website

Partial Query: Canvas

Responses: Northeastern Canvas (1), Canvas Login (1), Canvas Prints (0), Canvas Delete All (0), Canva (0)

Annotations: This example was generated using Google's query autocomplete system. Google has its own algorithm for autocompleting queries and Google search is a prime example of their QAC system at play. Furthermore, the partial query was using the user cookies to generate specific responses at frequently click links in order to generate more relevant autocomplete.

(taken from Google)

Sample 5

Narrative: You are trying to find cats to adopt.

Partial Query: Cats

Responses: Cats for sale (1), cats photos (0), cute cats (0), cats musical (0), cats for adoption (1)

Annotations: This example was generated using Google's query autocomplete system. Google has its own algorithm for autocompleting queries and Google search is a prime example of their QAC system at play. The partial query is a broad topic that can have many different sectors that the query could be autocompleted to. The Google algorithm tries to

cover all these sectors by referencing the most popular ones first and then covering the other autocomplete sectors.

(taken from Google)