UE21CS342BA2 : Algorithms for Information Retrieval and Intelligence Web

Project Report

"Query Auto-completion using Trie and LSTM model"

Submitted by:

Charutha Rajeesh	**PES1UG21CS152**
Pragya Srivastava	**PES1UG21CS416**

Dr. Sujatha R Upadhyaya

January - May 2024

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
FACULTY OF ENGINEERING
PES UNIVERSITY
(Established under Karnataka Act No. 16 of 2013)
100ft Ring Road, Bengaluru – 560 085, Karnataka, India

Table of contents

Objective of the project

In the realm of information retrieval, where users interact with search engines to find relevant information, query formulation can sometimes be a cumbersome task. Users may struggle with articulating their search intent, especially when dealing with long or complex terms. This challenge can lead to suboptimal search results or even frustration among users.

Query autocompletion, also known as query suggestion or search suggestion, addresses this issue by dynamically predicting and suggesting completions for partially entered queries in real-time. As users type into the search bar, the search engine algorithm analyzes the input and offers suggestions based on various factors such as popular search terms, previous user queries, and contextual relevance.

The objective of this project is to develop an efficient and accurate query autocompletion system that leverages both Trie-based word completion and LSTM-based sentence completion techniques. By combining these two approaches, the project aims to enhance user experience in search interfaces by providing relevant and timely suggestions as users type their queries.

Motivation

Query auto-completion stems from a desire to enhance human-computer interaction and improve user experiences by leveraging AI and natural language processing techniques. The following factors drive the motivation behind this project:

1. **User Efficiency**: Query auto-completion aims to save time and effort for users by assisting them in quickly and accurately formulating search queries. By providing real-time suggestions as users type, the system streamlines the query formulation process, allowing users to navigate through vast amounts of information more efficiently.

2. **Enhanced Search Experience**: Real-time suggestions provided by query auto-completion significantly improve the overall search experience. By reducing the need for manual typing and offering relevant suggestions, users experience less frustration and are more likely to find the information they are looking for quickly and effortlessly.

3. **Information Access**: Efficient query formulation facilitated by auto-completion leads to better access to information. Users can express their search intent more effectively, resulting in more relevant search results and a higher likelihood of discovering valuable information.

4.**Technological Advancement**: The development of query auto-completion techniques contributes to the advancement of AI and natural language processing technologies. By exploring innovative approaches such as combining LSTM and Trie models, this project aims to push the boundaries of what is achievable in query autocompletion, paving the way for future advancements in the field.

5.**Practical Applications**: Query auto-completion technology has practical applications across various domains, including search engines, e-commerce platforms, and information retrieval systems. By improving access to information and enhancing user experiences, this technology benefits users in diverse contexts, from casual web browsing to professional research endeavors.

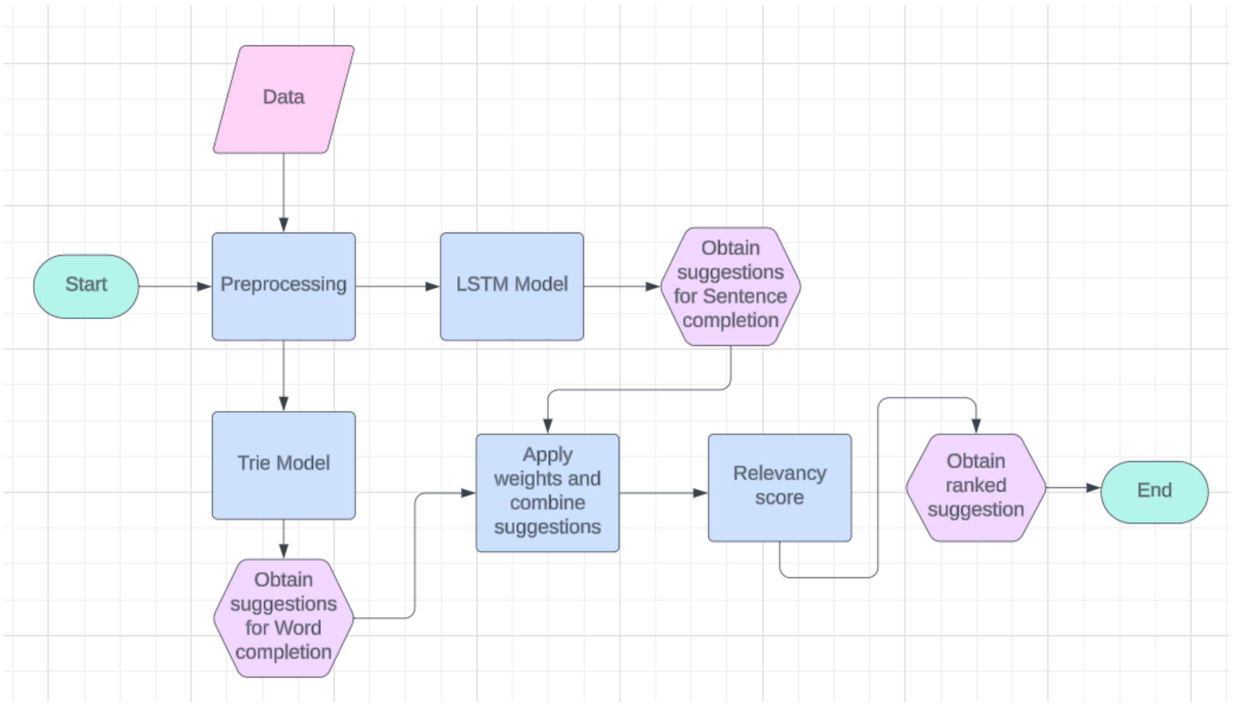
Approach

1. **Data Preprocessing**: Collection and preprocessing of a large corpus of text data to train the LSTM Model and build the Trie data structure.
2. **Model Development**:
    a. LSTM Model: Train an LSTM neural network to predict the next word in a given phrase, utilizing word embeddings and recurrent neural network architecture.

   b. Trie Model: Implement a Trie data structure to efficiently store and retrieve words based on prefixes.

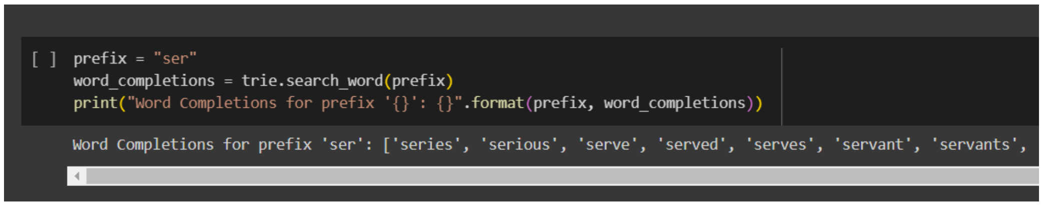
3. **Combination of Model suggestions**: Develop a mechanism to combine the suggestions generated by the LSTM and Trie models, assigning weights based on their respective accuracies and relevancy scores.

Architecture Diagram

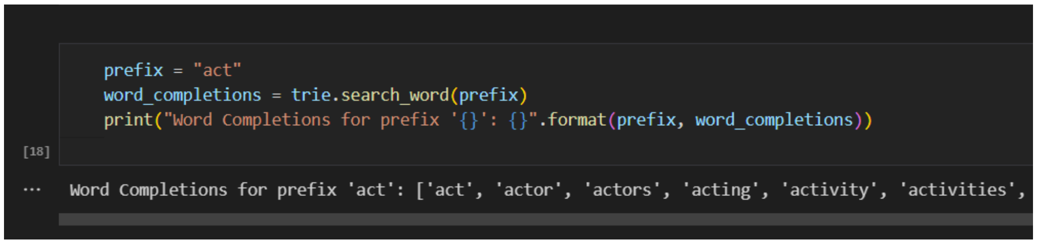

Data
Start
Preprocessing
LSTM Model
Obtain suggestions for Sentence completion
Trie Model
Apply weights and combine suggestions
Relevancy score
Obtain ranked suggestion
End
Obtain suggestions for Word completion

Findings of the project

Results of Trie Model



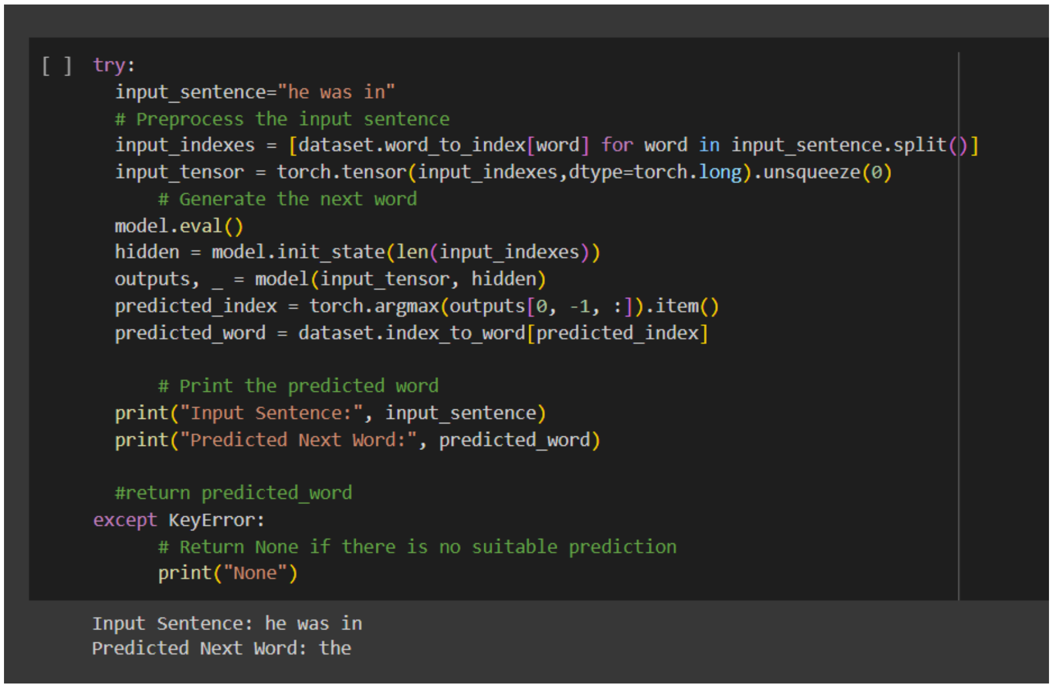
```
[ ] prefix = "ser"
    word_completions = trie.search_word(prefix)
    print("Word Completions for prefix '{}': {}".format(prefix, word_completions))

    Word Completions for prefix 'ser': ['series', 'serious', 'serve', 'served', 'serves', 'servant', 'servants',
```



```
prefix = "act"
word_completions = trie.search_word(prefix)
print("Word Completions for prefix '{}': {}".format(prefix, word_completions))
```

```
[18]
...  Word Completions for prefix 'act': ['act', 'actor', 'actors', 'acting', 'activity', 'activities',
```

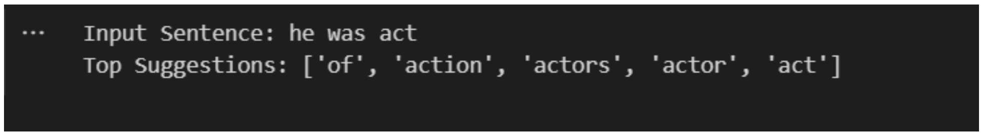
Results of LSTM Model



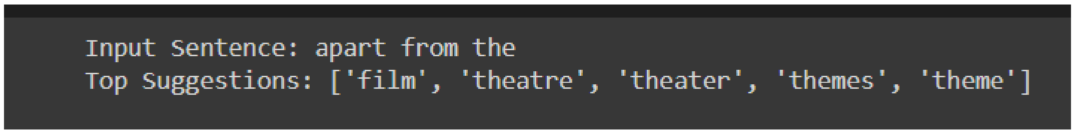
```
[ ] try:
      input_sentence="he was in"
      # Preprocess the input sentence
      input_indexes = [dataset.word_to_index[word] for word in input_sentence.split()]
      input_tensor = torch.tensor(input_indexes,dtype=torch.long).unsqueeze(0)
          # Generate the next word
      model.eval()
      hidden = model.init_state(len(input_indexes))
      outputs, _ = model(input_tensor, hidden)
      predicted_index = torch.argmax(outputs[0, -1, :]).item()
      predicted_word = dataset.index_to_word[predicted_index]

          # Print the predicted word
      print("Input Sentence:", input_sentence)
      print("Predicted Next Word:", predicted_word)

      #return predicted_word
    except KeyError:
          # Return None if there is no suitable prediction
          print("None")
```

```
    Input Sentence: he was in
    Predicted Next Word: the
```

Result of combined suggestions



```
...  Input Sentence: he was act
     Top Suggestions: ['of', 'action', 'actors', 'actor', 'act']
```



```
     Input Sentence: apart from the
     Top Suggestions: ['film', 'theatre', 'theater', 'themes', 'theme']
```

The combination of LSTM-based sentence completion and Trie-based word completion methods resulted in a contextually relevant autocompletion system compared to traditional methods. The Trie based model has demonstrated high scalability and efficiency, making it suitable for handling large vocabularies.

Future enhancements

1. **Improve Relevancy Score Calculation**: Explore more sophisticated methods for calculating relevancy scores, such as incorporating semantic similarity measures or context-awareness. Experiment with machine learning techniques, such as learning-to-rank algorithms, to better weigh the importance of different features in determining relevancy. Consider user feedback mechanisms to continuously refine and improve the relevancy scoring mechanism based on user interactions and preferences.

2. **Use Larger Dataset for Training the Models**: Acquire or generate a larger and more diverse dataset for training both the LSTM and Trie models. This can help improve the robustness and generalization capabilities of the models. Explore techniques for data augmentation to artificially increase the size of the dataset, such as paraphrasing or text generation methods.Consider utilizing pre-trained word embeddings or language models trained on large-scale corpora to enhance the representation learning process.

3. **Model Expansion**: Extend the capabilities of the autocompletion system to handle additional types of queries or input formats, such as voice-based queries or queries in multiple languages. Explore the integration of other NLP techniques, such as part-of-speech tagging or named entity recognition, to enhance the contextual understanding and accuracy of the autocompletion suggestions. Investigate the potential of incorporating user context, such as browsing history or user profiles, to personalize and tailor the autocompletion suggestions to individual user preferences and behavior.