Engineer to Excel

SAVEETHA
SCHOOL OF ENGINEERING
Approved by AICTE | IET-UK Accreditation

SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES
(Deemed to be University)
2005
Knowledge Conquers All

TITLE OF PROJECT

SENTENCE AUTO COMPLETE

REGISTER NUMBER

192210403

NAME

LINGAM VENKATA SAI

CONSULTANT PROFESSOR

DR.K.V. KANIMOZHI

A T BLE OF CONTENTS:

ABSTRACT:

Sentence auto-complete is a feature commonly found in text-based applications and platforms that aims to enhance user experience and productivity by predicting and suggesting completions for partially typed sentences or phrases. This technology leverages natural language processing (NLP) and machine learning techniques to analyze the context of the input text and generate relevant suggestions in real-time.

As the library website and its online searching tools become the primary "branch" many users visit for their research, methods for providing automated, context-sensitive research assistance need to be developed to guide unmediated searching toward the most relevant results. This study examines one such method, the use of autocompletion in search interfaces, by conducting usability tests on its use in typical academic research scenarios. The study reports notable findings on user preference for autocomplete features and suggests best practices for their implementation.

INTRODUCTION:

Autocompletion, a searching feature that offers suggestions for search terms as a user types text in a search box (see figure 1), has become ubiquitous on both larger search engines as well as smaller, individual sites. Debuting as the "Google Suggest" feature in 20041, autocomplete has made inroads into the library realm through inclusion in vendor search interfaces, including the most recent ProQuest interface and in EBSCO products. As this feature expands its presence in the library realm, it is important to understand how patrons include it in their workflow and the implications for library site design as well as for reference, instruction, and other library services.

An analysis of search logs from our library federated searching tool reveals both common errors in how search queries are entered, as well as patterns in the use of library search tools. For example, spelling suggestions are offered for more than 29 percent of all searches, and more than half (51 percent) of all searches appear to be for known items.2 Additionally, punctuation such as commas and a variety of correct and incorrect uses of Boolean operators are prevalent. These patterns suggest that providing some form of guidance in keyword selection at the point of search term entry could improve the accuracy of composing searches and subsequently the relevance of search results.

This study investigates student use of an autocompletion implementation on the initial search entry box for a library's primary federated searching feature. Through usability studies, the authors analyzed how and when students use autocompletion as part of typical library research, asked the students to assess the value and role of autocompletion in the research process, and noted any drawbacks of implementing the feature.

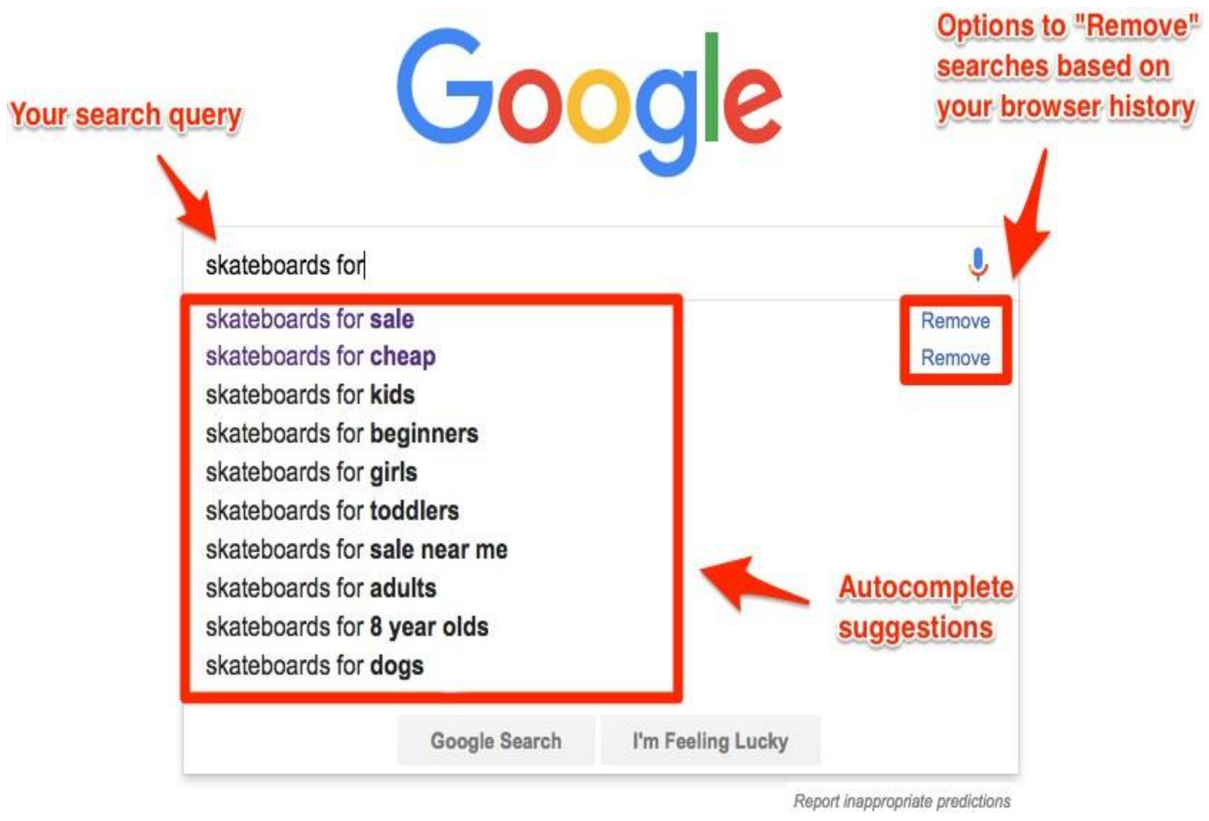


Figure 1. Autocomplete Implementation

LITERATURE REVIEW: Autocomplete as a plug-in has become ubiquitous onsite searches large and small. Research on autocomplete includes a variety of technical terms that refer to systems using this architecture. Examples include Real Time Query Expansion (RTQE), interactive query expansion, Search-as you-Type (SayT), query completion, type-ahead search, auto-suggest, and suggestive searching/search suggestions. The principal research concerns for autocomplete include issues related to both back-end architecture and assessments of user satisfaction and systems for specific implementations.

In defining important design characteristics for AUTOCOMPLETE AS A RESEARCH TOOL | WARD, HAHN, AND FEIST 8 autocomplete implementations, Wu advocates building in a tolerance for misplaced keywords as a critical component. Chaudhuri and Kaushik examine possible algorithms to use in building this type of tolerance into search systems. Misplaced keywords include typing terms in the wrong field (e.g., an author name in a title field), as well as spelling and word order errors.7 Systems that are tolerant in this manner "should enumerate all the possible interpretations and then sort them according to their possibilities," a specification Wu refers to as "interpret-as-you-type."8 Additionally, both Wu and Nandi and Jagadish specify fast response time (or synchronization speed) as a key usability feature in autocomplete interfaces, with Nandi and Jagadish indicating 100ms as a maximum.9,10 Speed also is a concern in mobile applications, which is part of the reason Paek et al. recommend autocomplete as part of mobile search interfaces, in which reducing keystrokes is a key usability feature.11

ABOUT SENTENCE AUTOCOMPLETE: Autocomplete, or word completion, is a feature in which an application predicts the rest of a word a user is typing. In Android and iOS[1] smartphones, this is called predictive text. In graphical user interfaces, users can typically press the tab key to accept a suggestion or the down arrow key to accept one of several.

Autocomplete speeds up human-computer interactions when it correctly predicts the word a user intends to enter after only a few characters have been typed into a text input field. It works best in domains with a limited number of possible words (such as in command line interpreters), when some words are much more common (such as when addressing an e-mail), or writing structured and predictable text (as in source code editors). Many autocomplete algorithms learn new words after the user has written them a few times, and can suggestipnalternatives based on the learned habits of the individual user.

The Autocomplete feature creates phrases from any repetitive input you make during your daily routine. The feature is disabled by default and must be enabled at Settings » Autocomplete. Autocomplete recognizes repeatedly entered words, sentences and your manual spelling corrections text during your normal work.

Autocomplete is a feature within Google Search that makes it faster to complete searches that you start to type. Our automated systems generate predictions that help people save time by allowing them to quickly complete the search they already intended to do. Autocomplete, or word completion, is a feature in which an application predicts the rest of a word a user is typing. In Android and iOS smartphones, this is called predictive text. In graphical user interfaces, users can typically press the tab key to accept a suggestion or the down arrow key to accept one of several.

Software integration:

In web browsers: In web browsers, autocomplete is done in the address bar (using items from the browser's history) and in text boxes on frequently used pages, such as a search engine's search box. Autocomplete for web addresses is particularly convenient because the full addresses are often long and difficult to type correctly. HTML5 has an autocomplete form attribute.

In e-mail programs: In e-mail programs autocomplete is typically used to fill in the e-mail addresses of the intended recipients. Generally, there are a small number of frequently used e-mail addresses, hence it is relatively easy to use autocomplete to select among them. Like web addresses, e-mail addresses are often long, hence typing them completely is inconvenient. For instance, Microsoft Outlook Express will find addresses based on the name that is used in the address book. Google's Gmail will find addresses by any string that occurs in the address or stored name.

In search engines: I n search engines, autocomplete user interface features provide users with suggested queries or results as they type their query in the search box. This is also commonly called autosuggest or incremental search. This type of search often relies on matching algorithms that forgive entry errors such as phonetic Soundex algorithms or the language independent Levinstein algorithm. The challenge remains to search large indices or popular query lists in under a few milliseconds so that the user sees results pop up while typing.

MATERIALS AND METHODS:

We trained a set of deep neural networks using different architectures, tokenization techniques, and software libraries to develop this research work. In the following, we introduce the different materials and methods employed for that purpose. 2.1. Deep Neural Networks and Tokenization Models Used Regarding the DNN architectures employed, we chose to use the following ones: Average Stochastic Gradient Descent (ASGD) Weight- Dropped LSTM (AWD-LSTM) [23], Quasi Recurrent Neural Networks (QRNNs) [24], and Transformer [25]. These DNN architectures have reportedly obtained some state-of-the-art results [23,26–38] recently in the NLP field in several groundbreaking digital products (https://openai.com/blog/openai-api/, https: //blog. Google/products/search/search- language-understanding-bert/) and in some of the most known datasets like the Penn Tree Bank [39], WikiText-2 and WikiText-103 [40], the One-Billion Word benchmark [41], or The Hutter Prize Wikipedia dataset (http://prize. hutter1.net/). The AWD-LSTM is a variation of the famous Long Short-Term Memory (LSTM) architecture [42]. The LSTM is a type of

Recurrent Neural Network (RNN) especially capable of processing and predicting sequences. That ability with sequences is the reason why LSTMs have been employed largely in LMs [21].

The AWD-LSTM includes several optimizations compared to the regular LSTM. Two of the most important ones are the use of Average Stochastic Gradient Descent (ASGD) and the weight dropout. The ASGD is used as the NN's optimizer to consider the previous weights (not only the current one) during each training iteration. The weight dropout introduces the dropout technique [43] to avoid overfitting, but with the characteristic of returning zero, notwith a subset of activations between layers, like in traditional dropout, but with a random subset of weights.

The QRNN is another type of RNN that includes alternate convolutional and pooling layers in the architecture. This design makes the QRNN able to capture better long-term sequences and train much faster since the convolutional layers allow the computation of intermediate representations from the input in parallel. They can be up to 16 times faster at training and test time than LSTMs while having better predictive accuracy than stacked LSTMs of the same hidden size. We use a modified QRNN (AWD-QRNN) to include the same ASGD and weight dropout modifications to improve its stability and optimize its capabilities, as for the AWD-LSTM.

CODE:

```
class TrieNode:
   def___init__(self):
      self.children = {}
      self.is_end_of_word =
      False


class
   AutoComplete:
   def___init__(self):
      self.root = TrieNode()


class TrieNode:
   def___init__(self):
      self.children = {}
      self.is_end_of_word =
      False
```

```
class
  AutoComplete:
  def___init__(self):
    self.root = TrieNode()

  def insert(self, sentence):
    node = self.root
    for char in sentence:
      if char not in node.children:
        node.children[char] =
        TrieNode()
      node = node.children[char]
    node.is_end_of_word = True

  def search(self, prefix):
    node = self.root
    for char in prefix:
      if char not in node.children:
        return []
      node = node.children[char]
    return self._find_sentences_from_node(node, prefix)

  def _find_sentences_from_node(self, node, prefix):
    results = []
    if node.is_end_of_word:
      results.append(prefix)
    for char, child_node in node.children.items():
      results.extend(self._find_sentences_from_node(child_node,     prefix   +
char))
```

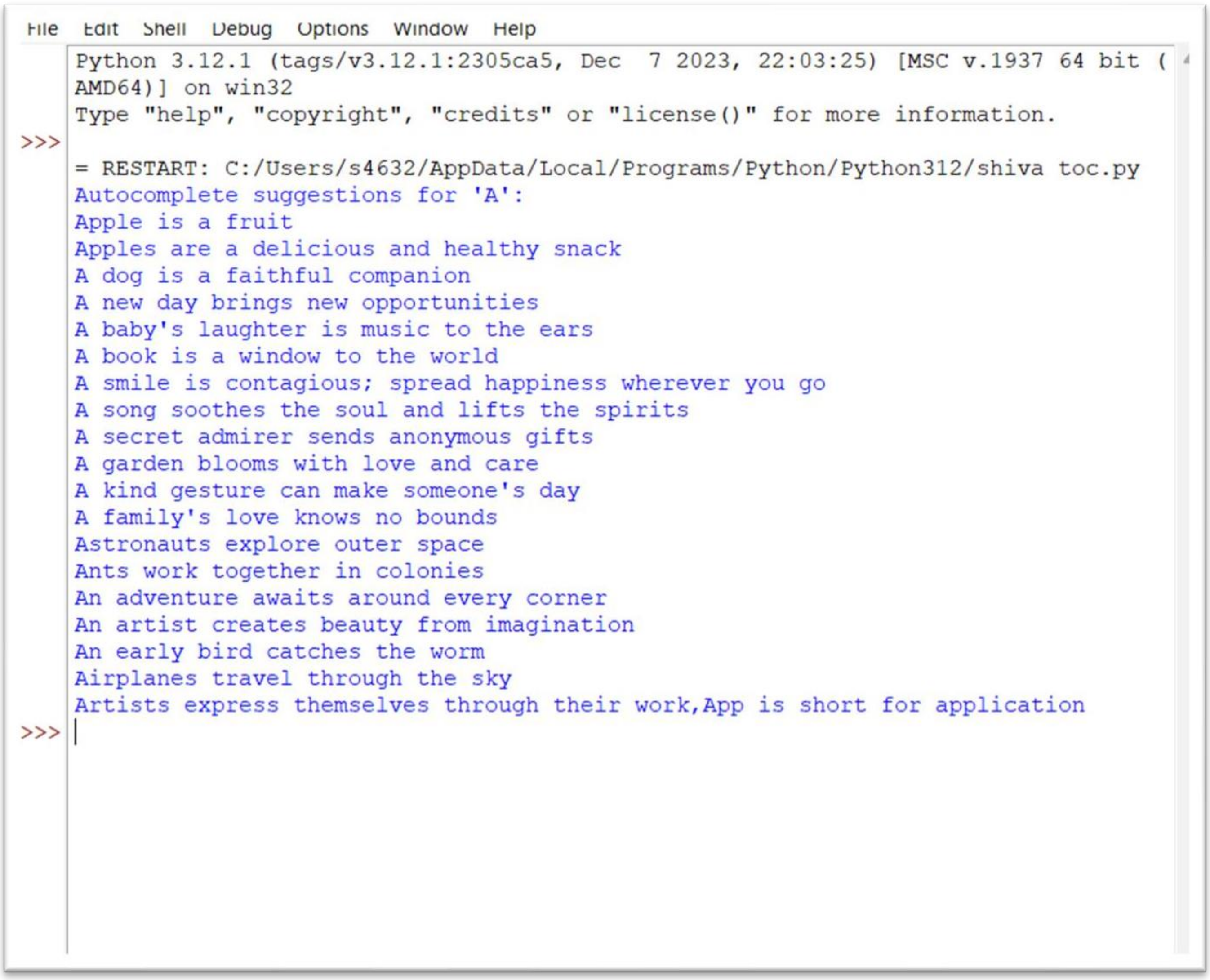
```
        return results

# Example usage:
auto_complete = AutoComplete()
sentences = [
    "Apple is a fruit",
    "A dog is a faithful companion",
    "Apples are a delicious and healthy snack",
    "Astronauts explore outer space",
    "Ants work together in colonies",
    "Airplanes travel through the sky",
    "Artists express themselves through their work,"
    "App is short for application",
    "banana is a tropical fruit",
    "bat is a mammal",
    "cat is a pet",
    "A new day brings new opportunities",
    "A baby's laughter is music to the ears",
    "A book is a window to the world",
    "An adventure awaits around every corner",
    "A smile is contagious; spread happiness wherever you go",
    "A garden blooms with love and care",
    "An artist creates beauty from imagination",
    "A song soothes the soul and lifts the spirits",
    "A kind gesture can make someone's day",
    "An early bird catches the worm",
    "A family's love knows no bounds",
    "A secret admirer sends anonymous gifts",
]
for sentence in sentences:
    auto_complete.insert(sentence)

prefix = "A"
suggestions = auto_complete.search(prefix)

print ("Autocomplete suggestions for
'{}':".format(prefix))for suggestion in suggestions:
    print(suggestion)
```

OUTPUT:



```
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec  7 2023, 22:03:25) [MSC v.1937 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/s4632/AppData/Local/Programs/Python/Python312/shiva toc.py
Autocomplete suggestions for 'A':
Apple is a fruit
Apples are a delicious and healthy snack
A dog is a faithful companion
A new day brings new opportunities
A baby's laughter is music to the ears
A book is a window to the world
A smile is contagious; spread happiness wherever you go
A song soothes the soul and lifts the spirits
A secret admirer sends anonymous gifts
A garden blooms with love and care
A kind gesture can make someone's day
A family's love knows no bounds
Astronauts explore outer space
Ants work together in colonies
An adventure awaits around every corner
An artist creates beauty from imagination
An early bird catches the worm
Airplanes travel through the sky
Artists express themselves through their work,App is short for application
>>>
```

RESULT: Finaly we concluded that sentence auto completion is very helpful everyone. In this project we found some examples how sentence was completed for given letter. by using python code, we got some output examples example : “A” we got Apple is fruit, apple is yellow in color, Airplane travel through the sky. we got this kind of full sentence for giving letter A. It is very helpful for in software integration like web browser, e-mail program and search engine etc. Here we funded some examples for sentence autocomplete by using python code for giving single value.

CONCLUSION: Implementing autocomplete functionality that accounts for the observed research tendencies and preferences of users makes for a compelling search experience. Participant selection of autocomplete suggestions varied between the types of searches studied. Spelling correction was the one universally acknowledged use. For subject-based searching, confidence in the topic searched and the stage of research emerged as indicators of the likelihood of autocomplete suggestions being taken. The use and effectiveness of providing subject suggestions requires further study, however. Students expect suggestions to produce usable results within a library’s collections, so the source of the suggestions should incorporate known, viable subject taxonomies to maximize benefits and not lead students down false search paths. There is an ongoing need to investigate possible search-term dictionaries outside of Google, such as lists of library holdings, journal titles, article titles, and controlled vocabulary from key library databases. The “brainstorming” aspect of autocomplete for subject searching is an intriguing benefit that should be more fully explored and supported. In combination with these findings, participant’s positive responses to some of the assessment questions (including first impressions of autocomplete and willingness to recommend it to friends) indicate that autocomplete is a viabletool to incorporate site-wide into library search interfaces.

Our autocomplete use findings draw attention to user needs and library support across search processes; specifically, autocomplete functionality offers support while forming search queries and can improve the results of user searching. For this reason, we recommend that autocomplete functionality be investigated for implementation across all library interfaces and websites to provide unified support for user searches. The benefits that can be realized from autocomplete can be maximized by consulting with reference and instruction personnel on the benefits noted above and collaboratively devising best practices for integrating autocomplete results into search strategy formulation and classroom-teaching workflows.

REFERENCES:

"How to use Auto-Correction and predictive text on your iPhone, iPad, or iPod touch". Apple Support. Apple.

1.^ Jump up to: a b c d e Tam, Cynthia; Wells, David (2009). "Evaluating the Benefits of Displaying Word Prediction Lists on a Personal Digital Assistant at the Keyboard Level". Assistive Technology. 21 (3): 105–114. doi:10.1080/10400430903175473. PMID 19908678. S2CID 23183632.

2.^Anson, D.; Moist, P.; Przywara, M.; Wells, H.; Saylor, H.; Maxime, H. (2006). "The Effects of Word Completion and Word Prediction on Typing Rates Using On-Screen Keyboards". Assistive Technology. 18 (2): 146–154. doi:10.1080/10400435.2006.10131913. PMID 17236473. S2CID 11193172

3.^Jump up to: a b Trnka, K.; Yarrington, J.M.; McCoy, K.F. (2007). "The Effects of Word Prediction on Communication Rate for AAC". NAACL-Short '07: Human Language Technologies 2007: The Conference of the North American Chapter of *the Association for Computational Linguistics. Vol. Companion Volume, Short Papers. Association for Computational Linguistics. pp. 173–6. Cutesier 10.1.1.363.2416.*

4.^ Jump up to: a b Beukelman, D.R.; Mirenda, P. (2005). Augmentative and Alternative Communication: Supporting Children and Adults with Complex Communication Needs (3rd ed.). Baltimore, MD: Brookes. p. 77. ISBN 9781557666840. OCLC 254228982.

5.^ Jump up to: a b c Witten, I.H.; Darragh, John J. (1992). The reactive keyboard. Cambridge University Press. pp. 43–44. ISBN 978-0-521-40375-7.

6.^ Jelinek, F. (1990). "Self-Organized Language Modelling for Speech Recognition". In Waibel, A.; Lee, Kai-Fu (eds.). Readings in Speech Recognition. Morgan Kaufmann. p. 450. ISBN 9781558601246.

7.^ Oster, Jan (2015). "Communication, defamation and liability of intermediaries". Legal Studies. 35 (2): 348–368. doi:10.1111/lest.12064. S2CID 143005665.

8.^ McCulloch, Gretchen (11 February 2019). "Autocomplete Presents the Best Version of You". Wired. Retrieved 11 February 2019.

9. ^ McClure, Max (12 November 2012). "Chinese typewriter anticipated predictive text, finds historian".

10.^ Jump up to: a b Sorrel, Charlie (February 23, 2009). "How it Works: The Chinese Typewriter". Wired.

11.^ Greenwood, Veronique (14 December 2016). "Why predictive text is making you forget how to write". New Scientist.

12.^ Clasohm, Carsten (2011). "Let MeType". Archived from the original on 2012-05-27. Retrieved 2012-05-09.

13.^ "Medical Transcription Software — In tell iComplete". Flash Peak. 2014.

14.^ Davids, Neil (2015-06-03). "Changing Autocomplete Search Suggestions". Reputation Station. Retrieved 19 June 2015.