AUTOMATIC TEXT COMPLETION

Prof.Uday Nayak, Ahad khan, Piyush Shende, Binoy Bangera

Department of Information Technology

Don Bosco Institute Of Technology

Premier Automobiles Road, Kurla, Mumbai - 400070

Abstract: **Imagine that you were a representative replying to customer online and you are asking more or less the same questions over and over to your customer. Would you like to get automatic suggestions instead of typing the same thing again and again? An autocomplete can be helpful, faster, and convenient and also correct any grammatical / spelling error at the same time.**

Keywords - regex rules, sklearn tfidf tool, similarity matrix

I. Introduction

A. Overview

Imagine that you were a representative replying to customer online and you are asking more or less the same questions over and over to your customer. Would you like to get automatic suggestions instead of typing the same thing again and again? An autocomplete can be helpful, faster, and convenient and also correct any grammatical / spelling error at the same time. In the jupyter notebook in this project, we select a history of sentences written by the representatives and the customer, format and correct them using a few regex rules and count them so we can estimate their frequency and likeliness to be useful again.

B. Scope

Text prediction algorithms for automatic or semi-automatic sentence completion have been vastly discussed in literature. They are widely used to enhance the speed of communication as well as reducing the total time taken to compose text. Autocomplete is designed to help people complete a search they were intending to do, it is an important feature to implement that can improve the user's experience of your product as well leave a good impression on the company's customer care departmen

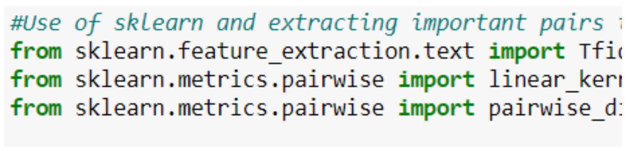
C: Problem Statement: The objective of this project was to be able to apply techniques and methods learned in Natural Language Processing course to a rather famous real-world problem, the task of sentence completion using text prediction. An autocomplete can be helpful, faster, and convenient and also correct any grammatical / spelling error at the same time. In the jupyter notebook in this project, we select a history of sentences written by the representatives and the customer, format and correct them using a few regex rules and count them so we can estimate their frequency and likeliness to be useful again.

D: Need of the proposed System: The word completion system applies prediction criteria to avoid annoying the user by displaying an excessive number of wrong suggestions. This makes it easy for the user to compare the suggestion to the partial data entry. It also saves time for the people working in the representative sector/company service providers to handle the same queries of the customer again and again and provide with accurate /timely solutions as fast as possible .

E: Summary of the results- This work proposes a Autocomplete word completion system detection method using NLP. This method of autocomplete of text using ML can be very useful to the employees as well as students to write proper content as well as save their time. The system automatically predicts your best context based on certain of your neural network. In the jupyter notebook in this project, we select a history of sentences written by the representatives and the customer, format and correct them using a few regex rules and count them so we can estimate their frequency and likeliness to be useful again. After the calculation of a similarity matrix based on the sklear tool (frequency and normalization of words), we use this matrix to calculate the similarity between the new few words written by the representative and the history of messages written in the past. The Autocomplete will recognize the closest sentences and rank 3 final proposals.

II. METHODOLOGY

Document/Text classification is one of the important and typical task in supervised machine learning (ML). Assigning categories to documents, which can be a web page, library book, media articles, gallery etc. has many applications like e.g. spam filtering, email routing, sentiment analysis etc.
Let's divide the classification problem into below steps:
Prerequisite and setting up the environment.
Loading the data set in jupyter.
Extracting features from text files.
Running ML algorithms.
Grid Search for parameter tuning.
Useful tips and a touch of NLTK.
Scikit learn – Text files are actually series of words (ordered). In order to run machine learning algorithms we need to convert the text files into numerical feature vectors. We will be using bag of words model for our example. Briefly, we segment each text file into words (for English splitting by space), and count # of times each word occurs in each document and finally assign each word an integer id. Each unique word in our dictionary will correspond to a feature (descriptive feature).



```
#Use of sklearn and extracting important pairs i
from sklearn.feature_extraction.text import Tfi
from sklearn.metrics.pairwise import linear_ker
from sklearn.metrics.pairwise import pairwise_d
```

TF-IDF
TF-IDF is a method which gives us a numerical weightage of words which reflects how important the particular word is to a document in a corpus. A corpus is a Collection of documents. **Tf** is Term frequency, and **IDF** is Inverse document frequency. This method is often used for information retrieval and text mining.
It is a statistical measure that evaluates how relevant a word is to a document in a collection of documents. This is done by multiplying two metrics: how many times a word appears in a document, and the inverse document frequency of the word across a set of documents.

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

$tf_{i,j}$ = number of occurrences of i in j
df_i = number of documents containing i
N = total number of documents

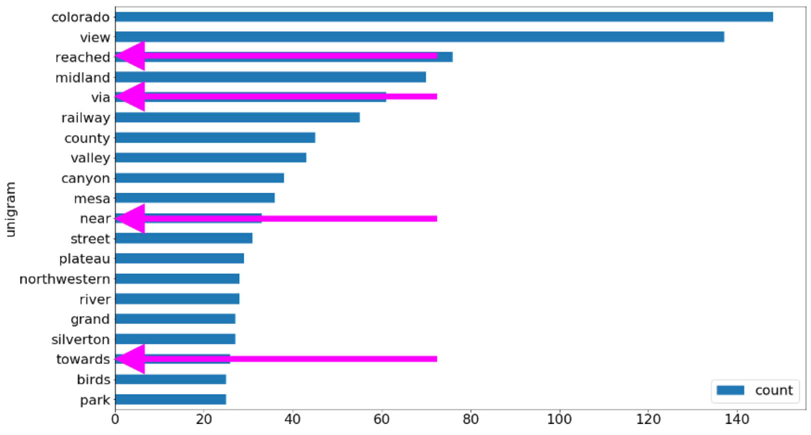
TF*IDF is an information retrieval technique that weighs a term's frequency (**TF**) and its inverse document frequency (**IDF**). Each word or term that occurs in the text has its respective **TF** and **IDF score**. The product of the **TF** and **IDF scores** of a term is called the **TF*IDF** weight of that term.
If you have short documents, like we do in this case, the TF part of TF-IDF may be pretty low-signal, in which case you can try using IDF alone or set TF to 1.
Here's what the top terms look like if we use IDF and limit to 10 terms per document:

Scikit-Learn

- $\text{IDF}(t) = \log\frac{1+n}{1+\text{df}(t)} + 1$

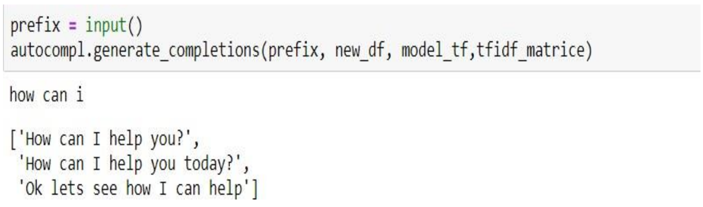
Standard notation

- $\text{IDF}(t) = \log\frac{n}{\text{df}(t)}$



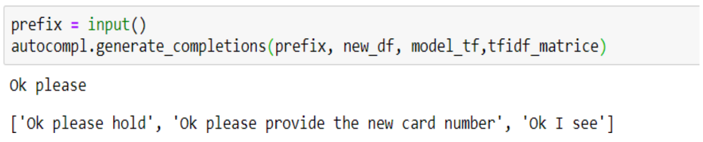
III. RESULTS

In the jupyter notebook in this project, we select an history of sentences written by the representatives and the customer, format and correct them using a few regex rules and count them so we can estimate their frequency and likeliness to be useful again. After the calculation of a similarity matrix based on the sklearn tfidf tool (frequency and normalization of words), we use this matrix to calculate the similarity between the new few words written by the representative and the history of messages written in the past. The Autocomplete will recognize the closest sentences and rank 3 final proposals:



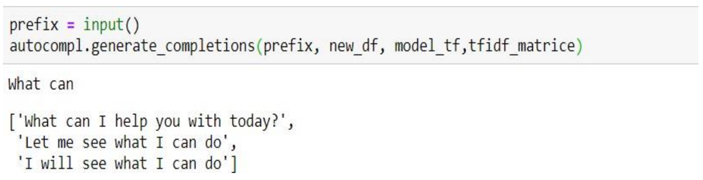
```
prefix = input()
autocompl.generate_completions(prefix, new_df, model_tf,tfidf_matrice)
```

```
how can i

['How can I help you?',
 'How can I help you today?',
 'Ok lets see how I can help']
```



```
prefix = input()
autocompl.generate_completions(prefix, new_df, model_tf,tfidf_matrice)
```

```
Ok please

['Ok please hold', 'Ok please provide the new card number', 'Ok I see']
```



```
prefix = input()
autocompl.generate_completions(prefix, new_df, model_tf,tfidf_matrice)
```

```
What can

['What can I help you with today?',
 'Let me see what I can do',
 'I will see what I can do']
```

V. CONCLUSIONS

Autocomplete, or word completion, is a feature in which an application predicts the rest of a word a user is typing. .... 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. A word completion user interface allows the user to customize each suggestion list with user- defined name- completion pairs on an on-going basis. This work proposes a Autocomplete word completion system detection method using NLP. This method of autocomplete of text using ML can be very useful to the employees/representatives to have a interaction with different types of users as well as save their time. The system automatically predicts your best context based on certain training of your neural network.

VI: References

[1] MahdiehPootchi, George Thomas, ‘Nlp analysis and machine learning for detecting patterns [Online] Available: https://www.sciencedirect.com/science /article/pii/S193152441 730333X [Accessed 17 May 2020].

[2] Dipanjan (Dj) Sarkar, ‘NLP with Deep Learning’, 2019. [Online] Available: https://towardsdatascience.com/detecting-patterns-with- deep-learning-9e45c1e34

[3] Poostchi, M., Silamut, K., Maude, R., Jaeger, S. and Thoma, G., 2020. ‘Text Analysis and Machine Learning.’ [Online] Available at: https://www.Ncbi.nlm.nih.gov/pmc/articles/PMC5840030 / [Accessed 17 May 2020].

[4]P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. Squad: 100,000+ questions for machine comprehension of text. EMNLP, 2016.
[5] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. Squad: 100,000+ questions for machine comprehension of text. EMNLP, 2016.

[6] P. Ramachandran, P. J. Liu, and Q. V. Le. Unsupervised pretraining for sequence to sequence learning. ArXiv preprint arXiv: 1611.02683, 2016.

[7] M. Ranzato, C. Poultney, S. Chopra, and Y. LeCun. Efficient learning of sparse representations with an energy-based model. In Advances in neural information processing systems, pages 1137–1144, 2007.
[8] X. Zhu. Semi-supervised learning literature survey. 2005.