# CV Scan improved

Our algorithm to extract keywords from a resume document has improved dramatically down to six lines of code after importing libraries and opening the object.

Per our latest series of postings, we recognized the value of keywords in the job market and specially to the candidate, who needs to manage it to increase the chance of landing an interesting job. By now we have also highlighted the role those keywords play in the automatic selection of millions of candidates. The best way to learn a tool is putting it to work!

I encourage you to use the Pisani method below to raise your digital presence by a smart selection of the tools at your disposal. If you are lucky to have a job, the invitation goes to share it openly with your network, you maybe helping someone who will use the code when most needed.

Code from Notebook

**Resume X-ray re 01**

Abstract

Key words in your resume are used to filter against an opening, to understand how they can be efficiently extracted using a simple Python script is useful for improving your resume. The algorithm below is commented to highlight the purpose of each line or cell. Suggestions are welcomed!

Revision 01 04.18.21 - Importing nltk library for tokens and stopwords for natural language processing capacities.

**import** string

**from** collections **import** Counter

**import** nltk

**from** nltk.corpus **import** stopwords

nltk**.**download('stopwords')

nltk**.**download('punkt')

**from** nltk.tokenize **import** word\_tokenize

resumexray **=** open('Gustavo\_Resume.txt')**.**read()**.**lower()**.**translate(str**.**maketrans('','',string**.**punctuation)) *# file opened, normalized with no capitals and no punctuation.*

tokens **=** nltk**.**word\_tokenize(resumexray) *# tokenizing*

tokens\_key **=** [word **for** word **in** tokens **if** **not** word **in** stopwords**.**words()] *# creating list of tokens*

count\_key\_words **=** Counter(tokens\_key) *# creating a dictionary of tokens and ocurrence.*

lst **=** list() *# extracts the data out of the dictionary and create a tuple list.*

**for** key, val **in** list(count\_key\_words**.**items()):

lst**.**append((val,key)) *# appending to the list. A list of tuple in value, key order*

lst**.**sort(reverse**=True**) *# sort backwards the value key tuple lst and assign it back to the list lst variable. Descending order*

**for** val,key **in** lst[0:10]: *# Look for the top 10 relevant words.*

print(key) *# Printing it in the order of the members of the tuple; val*

data

supply

chain

management

venezuela

digital

business

category

sap

responsible