Resume Keyword Extractor

Abstract

The aim of the project is to develop a process to analyze a folder containing resumes and extract keywords for each resume. The project uses a bag of words approach over the entire data set and extracts keywords for a resume and creates a JSON file with the top 5 keywords.

About the dataset

The example data set provided contained resumes of people having backgrounds in the field of data science, product management and business development. The resumes are in PDF (portable document format) format, and would need to parse them in ASCII text format to process and analyze.

Quickstart

To run the python script please follow the usage below -

```
Usage: python ld_resume.py [ARGS]
        Required Arguments:
        --dir </dir/to/folder/>                 Path to folder containing resumes
        --result-file <output json file>        Path to Output Json File

        Optional Arguments:
        --min-df <min_df value>                  Min DF value for the CountVectorizer - default : 0.2
        --max-df <max_df value>                  Max DF value for the CountVectorizer - default : 0.8
        --stop-words <stop words>                Custom Stop Words to include if any  - default : []
        --num-common-words-ignore <Number>       Number of common words to ignore - default : 0
        --num-keywords <Number>                  Number of keywords to extract to JSON. default = 5

       --help-print                              Prints the usage section of the script

        Example Usage: python ld_resume.py --dir /path/to/folder_containing_resumes/ --result-file /path/output/result.json
```

Descriptions

The process of extracting keywords from a set of documents comprises of the following steps –

1) Text Processing
    a. Normalization – Using lemmatization
    b. Noise Removal – Removing redundant words, punctuations, stopwords, etc.
2) Converting the text corpus to a vector of word counts
3) Transforming the text corpus to a vector of term frequencies
4) Extracting the top N keywords by filtering unigrams/bigrams

Process

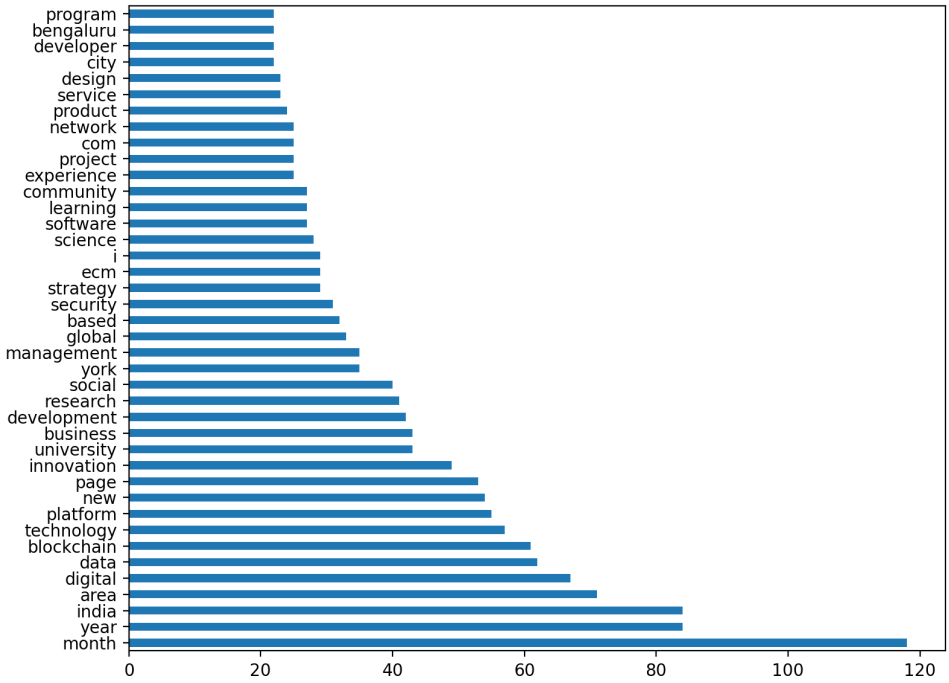
1) **Text processing**

   The data set contains resumes in pdf formats. There are many off the shelf packages available to parse pdf documents, but for the project I am using the textract package, which parses a wide range of documents. The textract package also allows to specify the type of method to process the pdf e.g. PDFMiner and the output encoding to use – ascii.

   ```
   resume_extract_text = textract.process(resume_path, method='pdfminer', encoding='ascii')
   ```

   The sample data set of resumes is bound to contain words which are redundant and occur frequently. These words cannot be used as a discerning factor to extract keywords. In this process we extract the most common words from

the word corpus of the entire text and analyze any outliers, to include in the list of stop words.
From the function ***MostCommonWords(resume_store_df, num_common_words)***
Analyzing the top 40 keywords we see the graph below –



After analyzing the data, and taking into consideration the number of documents provided we create a custom list of stop words. This list of stop words , including the custom stopwords passed to the program are processed with the corpus in the CountVectorizer.

Before feeding the text corpus to the CountVectorizer, the words in the text are processed by removing unwanted fields such as special characters, words with digits, punctuations, etc. The words are lemmatized and a final text corpus is created.

2) CountVectorizer

A CountVectorizer is used to convert the collection of text corpus to a matrix of token counts. In this project we are using the sklearns.CountVectorizer as it provides the capability to preprocess the text corpus prior to generating the vector matrix representation.
The sklearn's CountVecctorizer also lets us create a ***corpora specific stopwords***, using the min_df and max_df values.

We create the CountVectorizer as below –

```
cv = CountVectorizer(min_df=0.2, max_df=0.8, stop_words=stop_words, ngram_range=(1, 2),
analyzer='word',token_pattern=r"(?u)\b\w[\w-]*\w\b")
```

We use the min_df (minimum data frequency) to ignore words having lesser frequency in a few documents. The min_df value used in the code is 0.2, as we would like to ignore words that have occurred in 20 percent of the documents. For e.g. the word Boston appears in only 1 resume out of 10, would be ignored by the count vectorizer. The max_df value used is 0.8 i.e. ignore words which occur in more than 80 percent of the documents. An ngram_range of (1,2) is provided to consider words which are single words and bi-grams.

E.g.
Unigram – Programming
Bigram – Machine Learning

By default the CountVectorizer removes all special characters, punctuations and single characters. In our current data set we are expecting hyphenated words such as data-scientist or deep-learning to exist. To resepect the hyphenated words – we provide a custom tokenizer to respect hyphenated words

```
e.g – token_pattern=r"(?u)\b\w[\w–]*\w\b"
```

3) **TfidfTransformer**
Sklearn's TfidfTransformer is used to transform the sparse coo matrix to a normalized tf or tf-idf representation. The TfidfTransformer is used to scale down the impact of the words/tokens that occur very frequently in a given text corpu, by penalizing them and therby increasing the value point of the words that occur in less fraction of the training corpus. Thus tf-idf scores words that are more relevant to the context of the document, rather than the word frequency that frequently appears across documents. The TfIdfTransformer is used by setting the use_idf=True to enable inverse-document-frequency reweighting.

Notes about modules and packages used –

Vanilla Python installations may not have the necessary python packages. Installing them may be trivial using pip, e.g "pip install name_of_package"
But some of the packages depending on the system being used might throw errors and exceptions.

Textract :

1) Installing textract requires a dependency on the libxml2 library. If your machine does not have the libxml2 library, please update your pip and install the lxml library and then install textract.

2) On some of the platforms depending on how you are running Python – textract may throw an error - *UnboundLocalError: local variable 'pipe' referenced before assignment*

   This issue is due to your platform or virtual environment unable to find the pdf2txt.py script in the Scripts section folder by the Python Interpreter.
   This issue is resolved by adding "shell=True" to the subprocess.Popen() call at line 82 of utils.py utilities python file.
   Link to discussion - https://github.com/deanmalmgren/textract/issues/154pip

Nltk :

1) The project uses nltk module to download a list of stop words.
   nltk.download('stopwords')

   The nltk.download command goes over the internet, and may throw an error during download if the SSL certificate cannot be verified. The issue is resolved by disabling the SSL check.

```
try:
  _create_unverified_https_context = ssl._create_unverified_context
except AttributeError:
  pass
else:
  ssl._create_default_https_context = _create_unverified_https_context
nltk.download('stopwords')
```

Results

Running the Python Script creates the result JSON file as below –

result.json


```
{
  "Resumes": [
    {
      "FileName": "Anna Prokofieva.pdf",
      "Keywords": [
        "Google",
        "Feature",
        "Experiment",
        "Columbia University",
        "Classification"
      ]
    },
    {
      "FileName": "Pranay Anchuri.pdf",
      "Keywords": [
        "Institute",
        "Ibm",
        "Python",
        "Programs",
        "Pattern"
      ]
    },
    {
      "FileName": "Gokul Alex.pdf",
      "Keywords": [
        "Digital",
        "Blockchain",
        "Platform",
        "Innovation",
        "Technology"
      ]
    },
    {
      "FileName": "Chris Gonsalves.pdf",
      "Keywords": [
        "Financial",
        "Community",
        "Analyst",
        "Business Development",
        "Group"
      ]
    },
    {
      "FileName": "Raul Popa.pdf",
      "Keywords": [
        "Product Management",
        "Best",
        "Startup",
        "Multiple",
        "Machine Learning"
      ]
    },
    {
      "FileName": "Danning Sui.pdf",
      "Keywords": [
        "Mathematical",
        "Consensys",
        "Prize",
        "Degree",
        "Working"
      ]
    },
    {
      "FileName": "Tyler Swartz.pdf",
      "Keywords": [
        "Draft",
```

```
        "Team",
        "Dollar",
        "Advisory Board",
        "Million"
      ]
    },
    {
      "FileName": "Nidhiya V Raj.pdf",
      "Keywords": [
        "Community",
        "Technology",
        "Project",
        "Young",
        "Partnership"
      ]
    },
    {
      "FileName": "Edward Luo.pdf",
      "Keywords": [
        "Python",
        "Machine Learning",
        "Finance",
        "Sql",
        "Ibm"
      ]
    },
    {
      "FileName": "Timothy Yu.pdf",
      "Keywords": [
        "Analysis",
        "Python",
        "Fundamental",
        "Learning",
        "Technical"
      ]
    }
  ]
}
```