Resume_Analysis_With_Spacy

November 14, 2021

1 Introduction

In this project, we are going to use spacy for entity recognition on 200 Resume and experiment around various NLP tools for text analysis. The main purpose of this project is to help recruiters go throwing hundreds of applications within a few minutes. We have also added skills match feature so that hiring managers can follow a metric that will help them to decide whether they should move to the interview stage or not. We will be using two datasets; the first contains resume texts and the second contains skills that we will use to create an entity ruler.

2 Dataset

livecareer.com resume Dataset

A collection of 2400+ Resume Examples taken from livecareer.com for categorizing a given resume into any of the labels defined in the dataset: Resume Dataset.

2.1 Import

[1]:
```
#spacy
import spacy
from spacy.pipeline import EntityRuler
from spacy.lang.en import English
from spacy.tokens import Doc

#gensim
import gensim
from gensim import corpora

#Visualization
from spacy import displacy
import pyLDAvis.gensim_models
from wordcloud import WordCloud
import plotly.express as px
import matplotlib.pyplot as plt

#Data loading/ Data manipulation
import pandas as pd
import numpy as np
```

```
import jsonlines

#nltk
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
nltk.download(['stopwords','wordnet'])

#warning
import warnings
warnings.filterwarnings('ignore')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     /home/dristanta/nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     /home/dristanta/nltk_data…
[nltk_data]   Package wordnet is already up-to-date!
```

[2]:
```
## For dropdown
import numpy as np
import pandas as pd
import textwrap

import ipywidgets as widgets
from ipywidgets import interact, interact_manual
import IPython.display
from IPython.display import display, clear_output

import plotly.graph_objects as go
```

3 Resume Dataset

Using Pandas read_csv to read dataset containing text data about Resume.

- we are going to randomized Job categories so that 200 samples contain various job categories instead of one.
- we are going to limit our number of samples to 1000 as processing 2400+ takes time.

[3]:
```
df = pd.read_csv("Resume.csv")
```

[4]:
```
df = df.reindex(np.random.permutation(df.index))
data = df.copy().iloc[0:1000,]
data.head()
```

[4]:
```
              ID                                       Resume_str  \
1545   23573064           PROGRAMME FINANCE ASSOCIATE       Pro…
```

```
2310   17033567         VIDEO DIRECTOR, EAST COAST VIDEO FOR …
794    10268614         FITNESS ATTENDANT           Summary  …
308    40018190         IT SUPPORT TECHNICIAN       Education…
770    29134372         REGIONAL RECRUITER       Summary     M…

                                              Resume_html  \
1545   <div class="fontsize fontface vmargins hmargin…
2310   <div class="fontsize fontface vmargins hmargin…
794    <div class="fontsize fontface vmargins hmargin…
308    <div class="fontsize fontface vmargins hmargin…
770    <div class="fontsize fontface vmargins hmargin…

                    Category
1545                 FINANCE
2310                    ARTS
794                  FITNESS
308   INFORMATION-TECHNOLOGY
770               HEALTHCARE
```

[5]:
```
nlp = spacy.load("en_core_web_lg")
```

[6]:
```
skill_pattern_path = "jz_skill_patterns.jsonl"
```

3.0.1 Entity Ruler

To create an entity ruler we need to add a pipeline and then load the .jsonl file containing skills into ruler. As you can see we have successfully added a new pipeline entity_ruler. Entity ruler helps us add additional rules to highlight various categories within the text, such as skills and job description in our case.

[7]:
```
ruler = nlp.add_pipe("entity_ruler")
ruler.from_disk(skill_pattern_path)
nlp.pipe_names
```

[7]:
```
['tok2vec',
 'tagger',
 'parser',
 'attribute_ruler',
 'lemmatizer',
 'ner',
 'entity_ruler']
```

3.0.2 Skills

We will create two python functions to extract all the skills within a resume and create an array containing all the skills. Later we are going to apply this function to our dataset and create a new feature called skill. This will help us visualize trends and patterns within the dataset.

- get_skills is going to extract skills from a single text.

- unique_skills will remove duplicates.

[8]:
```python
def get_skills(text):
    doc = nlp(text)
    myset = []
    subset = []
    for ent in doc.ents:
        if ent.label_ == "SKILL":
            subset.append(ent.text)
    myset.append(subset)
    return subset


def unique_skills(x):
    return list(set(x))
```

3.0.3 Cleaning Resume Text

We are going to use ne ltk library to clean our dataset in a few stepse e :

- We are going to use regex to remove hyperlinks, special characters, or punctuations.
- Lowering text
- Splitting text into array based on space
- Lemmatizing text to its base form for normalizations
- Removing English stopwords
- Appending the results into an array.

[9]:
```python
clean = []
for i in range(data.shape[0]):
    review = re.sub(
        '(@[A-Za-z0-9]+)|([^0-9A-Za-z \t])|(\w+:\/\/\S+)|^rt|http.+?"',
        " ",
        data["Resume_str"].iloc[i],
    )
    review = review.lower()
    review = review.split()
    lm = WordNetLemmatizer()
    review = [
        lm.lemmatize(word)
        for word in review
        if not word in set(stopwords.words("english"))
    ]
    review = " ".join(review)
    clean.append(review)
```

3.0.4 Applying functions

In this section, we are going to apply all the functions we have created previously

- creating Clean_Resume columns and adding cleaning Resume data.
- creating skills columns, lowering text, and applying the get_skills function.
- removing duplicates from skills columns.

[10]:
```
data["Clean_Resume"] = clean
data["skills"] = data["Clean_Resume"].str.lower().apply(get_skills)
data["skills"] = data["skills"].apply(unique_skills)
data.head()
```

[10]:
```
            ID                                         Resume_str  \
1545  23573064         PROGRAMME FINANCE ASSOCIATE       Pro…
2310  17033567         VIDEO DIRECTOR, EAST COAST VIDEO FOR …
794   10268614         FITNESS ATTENDANT           Summary  …
308   40018190         IT SUPPORT TECHNICIAN       Education…
770   29134372         REGIONAL RECRUITER       Summary    M…

                                            Resume_html  \
1545  <div class="fontsize fontface vmargins hmargin…
2310  <div class="fontsize fontface vmargins hmargin…
794   <div class="fontsize fontface vmargins hmargin…
308   <div class="fontsize fontface vmargins hmargin…
770   <div class="fontsize fontface vmargins hmargin…

                    Category  \
1545                 FINANCE
2310                    ARTS
794                  FITNESS
308   INFORMATION-TECHNOLOGY
770               HEALTHCARE

                                           Clean_Resume  \
1545  programme finance associate professional summa…
2310  video director east coast video enterprise bra…
794   fitness attendant summary highly motivated nut…
308   support technician education bachelor science …
770   regional recruiter summary motivated program m…

                                                 skills
1545  [nim, security, support, project management, d…
2310                                         [business]
794   [engineering, schedule, finance, marketing, in…
308   [telephony, support, design, project managemen…
770   [security, support, documentation, accounting,…
```

3.1 Visualization

[11]:
```
fig = px.histogram(
    data, x="Category", title="Distribution of Jobs Categories"
).update_xaxes(categoryorder="total descending")
fig.show()
```

[ ]:
```

```

[12]:
```
Job_cat = data["Category"].unique()
Job_cat = np.append(Job_cat, "ALL")
```

[13]:
```
Job_cat
```

[13]:
```
array(['FINANCE', 'ARTS', 'FITNESS', 'INFORMATION-TECHNOLOGY',
       'HEALTHCARE', 'CONSTRUCTION', 'CONSULTANT', 'DESIGNER', 'BANKING',
       'TEACHER', 'CHEF', 'AVIATION', 'ADVOCATE', 'BPO',
       'BUSINESS-DEVELOPMENT', 'PUBLIC-RELATIONS', 'AUTOMOBILE',
       'ENGINEERING', 'APPAREL', 'AGRICULTURE', 'DIGITAL-MEDIA',
       'ACCOUNTANT', 'SALES', 'HR', 'ALL'], dtype=object)
```

[14]:
```
def dropdown_menu_widget(Job_Category):
    output=widgets.Output()
    drop_Job=widgets.Dropdown(options = Job_Category, value=None,␣
 ↪description='Job Category:')

    for cat in Job_Category:
        def dropdown_Job_eventhandler(change):
            display(input_widgets)
            job_choice=change.new
            IPython.display.clear_output(wait=True)
    drop_Job.observe(dropdown_Job_eventhandler,names='value')
    input_widgets=widgets.HBox([drop_Job])
    display(input_widgets)
    IPython.display.clear_output(wait=True)
```

[15]:
```
dropdown_menu_widget(Job_cat)
```

```
HBox(children=(Dropdown(description='Job Category:', index=3,␣
 ↪options=('FINANCE', 'ARTS', 'FITNESS', 'INFORMAT…
```

[20]:
```
Job_Category="INFORMATION-TECHNOLOGY"
```

[21]:
```
Total_skills = []
if Job_Category != "ALL":
    fltr = data[data["Category"] == Job_Category]["skills"]
    for x in fltr:
        for i in x:
```

```
            Total_skills.append(i)
else:
    fltr = data["skills"]
    for x in fltr:
        for i in x:
            Total_skills.append(i)
```

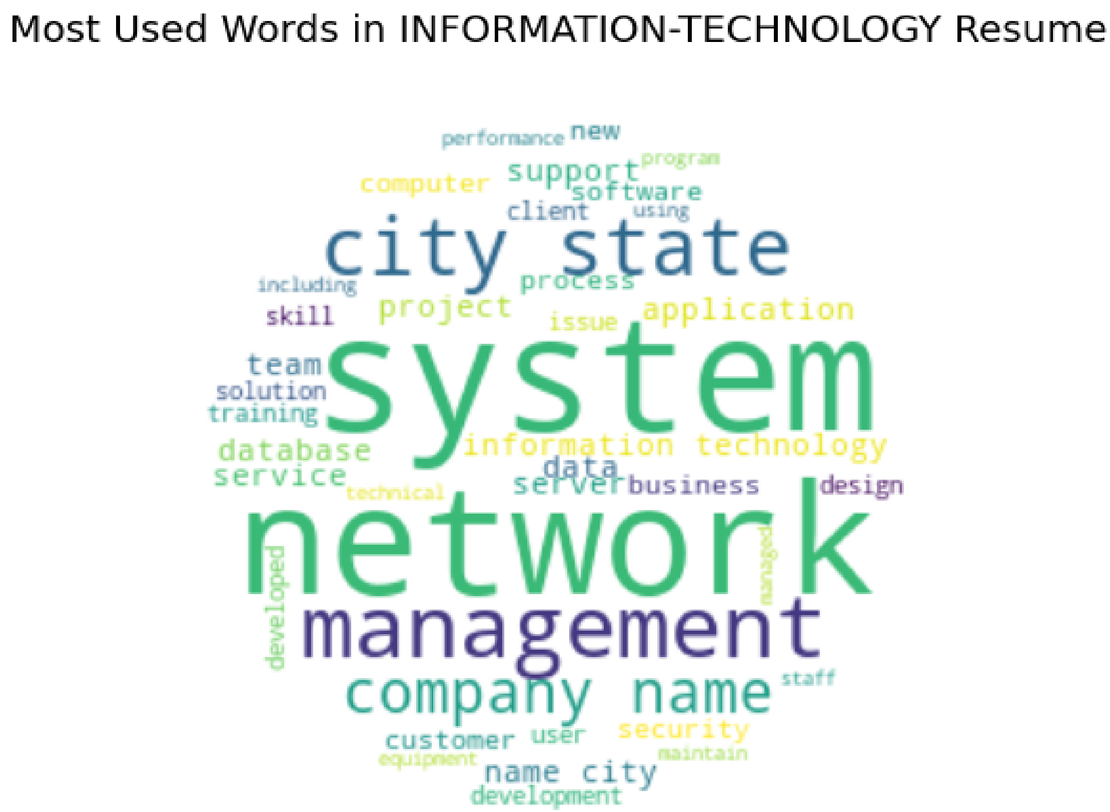
[22]:
```
fig = px.histogram(
    x=Total_skills,
    labels={"x": "Skills"},
    title=f"{Job_Category} Distribution of Skills",
).update_xaxes(categoryorder="total descending")
fig.show()
```

3.1.1 Most used words

In this part, we are going to display the most used words in the Resume filter by job category. In Information technology, the most words used are system, network, and database. We can also discover more patterns by exploring the word cloud below.

[23]:
```
text = ""
for i in data[data["Category"] == Job_Category]["Clean_Resume"].values:
    text += i + " "

plt.figure(figsize=(8, 8))

x, y = np.ogrid[:300, :300]

mask = (x - 150) ** 2 + (y - 150) ** 2 > 130 ** 2
mask = 255 * mask.astype(int)

wc = WordCloud(
    width=800,
    height=800,
    background_color="white",
    min_font_size=6,
    repeat=True,
    mask=mask,
)
wc.generate(text)

plt.axis("off")
plt.imshow(wc, interpolation="bilinear")
plt.title(f"Most Used Words in {Job_Category} Resume", fontsize=20)
```

[23]:
```
Text(0.5, 1.0, 'Most Used Words in INFORMATION-TECHNOLOGY Resume')
```



3.1.2 Entity Recognition

We can also display various entities within our raw text by using spaCy displacy.render. I am in love with this function as it is an amazing way to look at your entire document and discover SKILL or GEP within your Resume.

[24]:
```
sent = nlp(data["Resume_str"].iloc[0])
displacy.render(sent, style="ent", jupyter=True)
```

```
<IPython.core.display.HTML object>
```

3.1.3 Dependency Parsing

We can also visualize dependencies by just changing style to dep as shown below. We have also limited words to 10 which includes space too. Limiting the words will make it visualize the small chunk of data and if you want to see the dependency, you can remove the filter.

[46]:
```
displacy.render(sent[0:10], style="dep", jupyter=True, options={"distance": 90})
```

```
<IPython.core.display.HTML object>
```

[ ]:

```

```

3.2 Custom Entity Recognition

In our case, we have added a new entity called SKILL and is displayed in gray color. I was not impressed by colors and I also wanted to add another entity called Job Description so I started experimenting with various parameters within 'displace.

- Adding Job-Category into entity ruler.
- Adding custom colors to all categories.
- Adding gradient colors to SKILL and Job-Category

[26]:

```
patterns = df.Category.unique()
for a in patterns:
    ruler.add_patterns([{"label": "Job-Category", "pattern": a}])
```

[27]:

```
# options=[{"ents": "Job-Category", "colors": "#ff3232"},{"ents": "SKILL",␣
↪"colors": "#56c426"}]
colors = {
    "Job-Category": "linear-gradient(90deg, #aa9cfc, #fc9ce7)",
    "SKILL": "linear-gradient(90deg, #9BE15D, #00E3AE)",
    "ORG": "#ffd966",
    "PERSON": "#e06666",
    "GPE": "#9fc5e8",
    "DATE": "#c27ba0",
    "ORDINAL": "#674ea7",
    "PRODUCT": "#f9cb9c",
}
options = {
    "ents": [
        "Job-Category",
        "SKILL",
        "ORG",
        "PERSON",
        "GPE",
        "DATE",
        "ORDINAL",
        "PRODUCT",
    ],
    "colors": colors,
}
sent = nlp(data["Resume_str"].iloc[5])
displacy.render(sent, style="ent", jupyter=True, options=options)
```

```
<IPython.core.display.HTML object>
```

[ ]:

```

```

3.3 Your Resume Anlaysis

[28]:
```
input_resume="""
Dristanta Das,

Data Science,
Data Analytics,



Languages,
Python,
R,
Systems,
Docker,
CVAT, MySQL, Neo4J
,Python Modules,
Numpy ,
Pandas,
Matplotlib,
TorchAudio ,
Pytorch ,
Scikit-learn,
Machine Learning,
Regression ,
Classification,
Cross-validation, Naive Bayes,
Dimensionality Reduction, K-Means,
Decision Tree, Boosting,
Deep Learning,
Multilayer Perceptron, CNN, RNN,
LSTM, Transfer Learning,
Experience
Aug,21'- • Summer Project Intern IITKGP
Under the guidance of professor C.S Kumar ,Mechanical Engineering
Project
• Machine Learning
Feb-June,21' • 2.5D Visual Sound
Implemented a deep learning model to convert common monaural
audio into binaural audio by leveraging video.
• Computer Vision
Feb,21' • Hybrid Image Formation
Hybrid images are images with two interpretation. The interpretation changes␣
↪with viewing distance.
Mar,21' • Harris Corner Detection and SIFT
The Harris Corner detector is a standard technique for locating interest points␣
↪on an image.
Apr,21' • Hough Line and Circle Detector
```

```
May,21’ • Fundamental Matrix using RANSAC
The objective of this project was to improve upon image matching by
leveraging epipolar geometry of a stereo image pair while applying
RANSAC to reject poorly matched feature points.
• Exploratory Data Analysis
Sept-Dec,20’ • Zomato Restaurants Data
Analyzed the best restaurants of the cities all around India.
Nov,20’-
Dec,20’
• Player Data of FIFA21 Game
Analyzed the best players and their performance stats.
• Statistical Data Analysis
Mar-Apr,21’ • Bankruptcy Data
Analyzed the Bankruptcy data and drew inference out of it. Used
LDA to discriminate the bankrupt and non-bankrupt firms.
Education
2020-Present M.Sc. in Big Data Analytics RKMVERI
GPA: 7.67
Courses
• Machine Learning • Optimisation Algorithm
• Multivariate Statistics • Computer Vision
• Optimisation for Machine Learning • Linear Algebra
• Probability and Stochastic Process • Data Structure and Algorithm
• Artificial Intelligence • Time Series and Survival Analysis
• Online Learning • Data Mining
2017-2020 B.Sc. in Mathematics Presidency University
CGPA: 7.05
2015-2017 Higher Secondary Jodhpur Park Boys’ School
Marks: 84.2%
2005-2015 Secondary Jodhpur Park Boys’ School"""
```

[29]:

```
sent2 = nlp(input_resume)
displacy.render(sent2, style="ent", jupyter=True, options=options)
```

```
<IPython.core.display.HTML object>
```

[ ]:

```

```

3.4 Match Score

In this section, I am allowing recruiters to add skills and get a percentage of match skills. This can help them filter out hundreds of Resumes with just one button.

[30]:

```
input_skills="Data Science,Data Analysis,Database,MySQL,Machine Learning,Deep␣
↪Learning,Analytics,Artificial Intelligence,python,pytorch"
```

[31]:

```
req_skills = input_skills.lower().split(",")
resume_skills = unique_skills(get_skills(input_resume.lower()))
```

```
score = 0
for x in req_skills:
    if x in resume_skills:
        score += 1
req_skills_len = len(req_skills)
match = round(score / req_skills_len * 100, 1)

print(f"The current Resume is {match}% matched to your requirements")
```

```
The current Resume is 80.0% matched to your requirements
```

[32]:
```
print(resume_skills)
```

```
['python', 'decision tree', 'machine learning', 'dimensionality reduction',
'artificial intelligence', 'multivariate statistics', 'exploratory data
analysis', 'pandas', 'docker', 'inference', 'big data', 'computer vision', 'data
science', 'mechanical engineering', 'languages', 'time series', 'algorithm',
'analytics', 'data analysis', 'pytorch', 'deep learning', 'data structure',
'data mining', 'corner detection']
```

3.4.1 Topic Modeling - LDA

LDA, or Latent Dirchlet Allocation is arguably the most famous topic modeling algorithm out there. Out here we create a simple topic model with 4 topics. The code was inspired by Allan's project: Topic Modeling of NLP GitHub repositories

[33]:
```
docs = data["Clean_Resume"].values
dictionary = corpora.Dictionary(d.split() for d in docs)
bow = [dictionary.doc2bow(d.split()) for d in docs]
lda = gensim.models.ldamodel.LdaModel
num_topics = 4
ldamodel = lda(
    bow,
    num_topics=num_topics,
    id2word=dictionary,
    passes=50,
    minimum_probability=0
)
ldamodel.print_topics(num_topics=num_topics)
```

[33]:
```
[(0,
  '0.012*"state" + 0.011*"company" + 0.010*"city" + 0.009*"management" +
0.008*"name" + 0.007*"business" + 0.005*"client" + 0.005*"financial" +
0.005*"skill" + 0.005*"marketing"'),
 (1,
  '0.008*"engineering" + 0.008*"project" + 0.008*"state" + 0.007*"system" +
0.007*"equipment" + 0.007*"city" + 0.006*"company" + 0.005*"training" +
0.005*"management" + 0.005*"name"'),
 (2,
```

```
  '0.021*"customer" + 0.013*"state" + 0.012*"city" + 0.012*"sale" +
0.011*"company" + 0.011*"service" + 0.009*"name" + 0.007*"skill" +
0.007*"product" + 0.007*"food"'),
 (3,
  '0.013*"project" + 0.012*"system" + 0.009*"management" + 0.007*"team" +
0.007*"design" + 0.007*"support" + 0.007*"software" + 0.007*"company" +
0.007*"network" + 0.006*"application"')]
```

3.5 pyLDAvis

The best way to visualize Topics is to use pyLDAvis from GENSIM.

- topic #1 appears to relate to the project,management,system.
- topic #2 relates to management,company,business.
- topic #3 relates to customer,services , state.
- topic #4 relates to state,city,student.

[34]:
```
pyLDAvis.enable_notebook()
pyLDAvis.gensim_models.prepare(ldamodel, bow, dictionary)
```

```
/usr/lib/python3/dist-packages/past/builtins/misc.py:45: DeprecationWarning: the
imp module is deprecated in favour of importlib; see the module's documentation
for alternative uses
  from imp import reload
/usr/lib/python3/dist-packages/past/builtins/misc.py:45: DeprecationWarning: the
imp module is deprecated in favour of importlib; see the module's documentation
for alternative uses
  from imp import reload
/usr/lib/python3/dist-packages/past/builtins/misc.py:45: DeprecationWarning: the
imp module is deprecated in favour of importlib; see the module's documentation
for alternative uses
  from imp import reload
/usr/lib/python3/dist-packages/past/builtins/misc.py:45: DeprecationWarning: the
imp module is deprecated in favour of importlib; see the module's documentation
for alternative uses
  from imp import reload
/usr/lib/python3/dist-packages/past/builtins/misc.py:45: DeprecationWarning: the
imp module is deprecated in favour of importlib; see the module's documentation
for alternative uses
  from imp import reload
/usr/lib/python3/dist-packages/past/builtins/misc.py:45: DeprecationWarning: the
imp module is deprecated in favour of importlib; see the module's documentation
for alternative uses
  from imp import reload
```

[34]:
```
PreparedData(topic_coordinates=              x         y  topics  cluster
Freq
topic
```

```
0     -0.055600 -0.044501       1        1  49.636856
2     -0.120338  0.031467       2        1  23.091396
3      0.087075 -0.086253       3        1  15.843025
1      0.088862  0.099287       4        1  11.428724, topic_info=
Term         Freq        Total Category  logprob  loglift
724     customer  4285.000000  4285.000000  Default  30.0000  30.0000
229      project  2972.000000  2972.000000  Default  29.0000  29.0000
279       system  2765.000000  2765.000000  Default  28.0000  28.0000
637         sale  3225.000000  3225.000000  Default  27.0000  27.0000
529         food   971.000000   971.000000  Default  26.0000  26.0000
..           …           …           …       …       …       …
267        skill   321.287370  3248.224727   Topic4  -5.3966  -0.1445
797    technical   185.610605   862.682767   Topic4  -5.9452   0.6326
92   development   205.488429  2310.775258   Topic4  -5.8435  -0.2509
189          new   202.050632  2376.260081   Topic4  -5.8604  -0.2957
226      program   196.145236  1968.028735   Topic4  -5.8900  -0.1369

[363 rows x 6 columns], token_table=      Topic      Freq        Term
term
8110      4  0.955708        9001
1047      1  0.862114     account
1047      2  0.082276     account
1047      3  0.055447     account
3054      1  0.997378  accountant
…     …        …          …
308       1  0.386266        work
308       2  0.338968        work
308       3  0.104942        work
308       4  0.170469        work
2392      3  0.991147         xml

[726 rows x 3 columns], R=30, lambda_step=0.01, plot_opts={'xlab': 'PC1',
'ylab': 'PC2'}, topic_order=[1, 3, 4, 2])
```

[ ]:

```

```