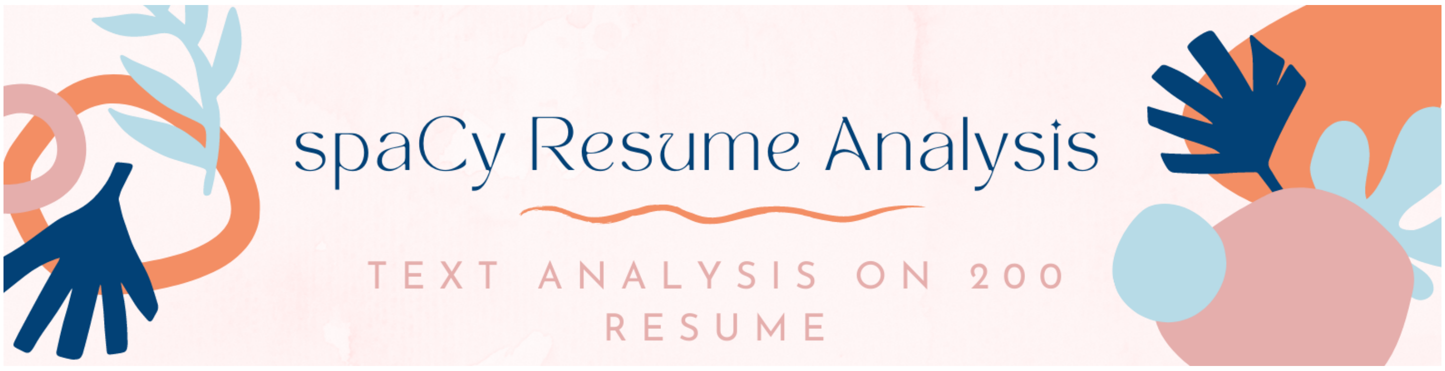


Introduction

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.

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.

Inside the CSV

- ID: Unique identifier and file name for the respective pdf.
- Resume_str : Contains the resume text only in string format.
- Resume_html : Contains the resume data in html format as present while web scrapping.
- Category : Category of the job the resume was used to apply.

Present categories

HR, Designer, Information-Technology, Teacher, Advocate, Business-Development, Healthcare, Fitness, Agriculture, BPO, Sales, Consultant, Digital-Media, Automobile, Chef, Finance, Apparel, Engineering, Accountant, Construction, Public-Relations, Banking, Arts, Aviation

Acknowledgements

Data was obtained by scrapping individual resume examples from www.livecareer.com website. Web Scrapping code present in my Github Repo.

Jobzilla skill patterns

The jobzilla skill dataset is `jsonl` file containing different skills that can be used to create **spaCy** `entity_ruler`. The data set contains `label` and `pattern` → diferent words used to descibe skills in various resume.

```
#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 /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

Loading

In this section, we are going to load the spaCy model, Resume Dataset, and Jobzilla skills dataset directly into the entity ruler.

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 200 as processing 2400+ takes time.

```
df = pd.read_csv("/work/resume-dataset/Resume/Resume.csv")
df = df.reindex(np.random.permutation(df.index))
data = df.copy().iloc[
    0:200,
]
data.head()
```

Loading spaCy model

- You can download spaCy model using `python -m spacy en_core_web_lg`
- Then load spacy model into `nlp`.

```
nlp = spacy.load("en_core_web_lg")
skill_pattern_path = "jz_skill_patterns.jsonl"
```

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.

```
ruler = nlp.add_pipe("entity_ruler")
ruler.from_disk(skill_pattern_path)
nlp.pipe_names
```

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.

```
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))
```

Cleaning Resume Text

We are going to use `nltk` library to clean our dataset in a few steps:

- 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.

```
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)
```

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.

> As you can see below that we have cleaned the resume and skills columns.

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

Visualization

Now that we have everything we want, we are going to visualize Job distributions and skill distributions.

Jobs Distribution

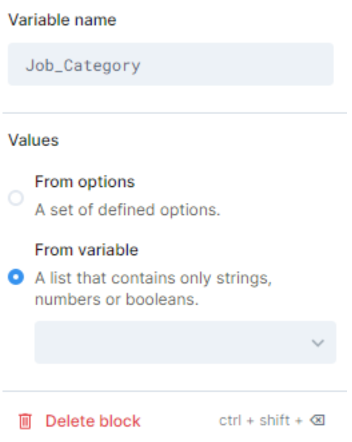
As we can see our random 200 samples contain a variety of job categories. Accountants, Business development, and Advocates are the top categories.

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

Skills

In this part, we are going to use the Deepnote input cell to create category variables and then visualize the distribution of skills based on selected Job Descriptions.

> First, we need to create variables from the dataset using `unique()` and then add the `ALL` category so that we can also visualize the overall skills trend in the dataset. lastly, we are going to use input cells and import categories from variables, as shown below.



Variable name

Job_Category

Values

From options
A set of defined options.

[✓ From variable
A list that contains only strings, numbers or booleans.]

Delete block ctrl + shift + ⌫

As we can observe `INFORMATION-TECHNOLOGY` job category's skills distributions.

Top Skills

- Software
- Support
- Business

> If you are looking to improve your chance of getting hired by a software company try focusing on software engineering, Support, and Business skills.

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

Job_Category

INFORMATION-TECHNOLOGY

```
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)

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

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.

```
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)
```

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.

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

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.

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

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

> You can see the result below as the new highlighted texts look beautiful.

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

```
# 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)
```

Your Resume Anlaysis

In this part, I am allowing users to **copy&paste** their resumes and see the results.

> As we can see my I have added my Resume and the results are amazing. The model has successfully highlighted all the skills.

`input_resume`

Abid Ali Awan Data Scientist I am a certified dat

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

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.

> **Please add the skills that are required by the job description without space in between commas and it will print out the percentage of match skills within the resume.**

`input_skills`

Data Science,Data Analysis,Database,SQL,Mach

```
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 66.7% matched to your requirements
```

We can also see the skills mentioned in your resume.

```
print(resume_skills)
```

```
['testing', 'time series', 'speech recognition', 'simulation', 'text processing', 'ai', 'pytorch', 'communications', 'ml', 'engineering', 'machin
```

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

```
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)
```

pyLDAvis

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

- topic #1 appears to relate to the customer, state, and city.
- topic #2 relates to management and marketing.
- topic #3 relates to systems and projects.
- topic #4 relates to financial and company.

```
pyLDAvis.enable_notebook()
pyLDAvis.gensim_models.prepare(ldamodel, bow, dictionary)
```

Conclusion

In this project, we have used an entity ruler to create additional entities and then displayed them using custom colors. We have also visualized categories and skills distributions and allowed the user to add resumes directly which includes skills match percentage. Finally, we have used LDA for topic modeling and used pyLDAvis to visualize various topics.

Overall, it was a learning experience for me as I have never used spaCy in depth. I have also discovered various ways on how my project can be used to improve the hiring process in filtering out the perfect candidate for the job. I hope you like my project and don't forget to give ♥.

Learning Resources

Blogs

- spaCy 101: Everything you need to know · spaCy Usage Documentation
- spaCy Tutorial | spaCy For NLP | spaCy NLP Tutorial (analyticsvidhya.com)
- Named Entity Recognition with NLTK and SpaCy | by Susan Li | Towards Data Science
- How I used NLP (Spacy) to screen Data Science Resume | by Venkat Raman | Towards Data Science

Projects

- Resume-and-CV-Summarization-and-Parsing-with-Spacy-in-Python
- AI models for automatic job application pipeline (user CV, job description analysis (customized NER/SpaCy) and artificial cover letter generation (trained GPT-2 model)
- Github API & SpaCy | Deepnote
- Spacy Food Entities (deepnote.com)

Next Step

- We can train a classification model to predict Job Categories from the text.
- T-SNE visualization of topics using bokeh.
- Improving spaCy model by adding a variety of skills and categories.
- Experimenting with various spaCy functions.