In [25]:
```
import spacy
import nltk
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer
from nltk.corpus import wordnet
from nltk.stem import WordNetLemmatizer
from nltk import pos_tag

# Load spaCy model for Named Entity Recognition
nlp = spacy.load("en_core_web_sm")
```

In [26]:
```
import docx

# Load the resume document
resume = docx.Document("Resume_Summary.docx")
# Extract text from the document
text = "\n".join([para.text for para in resume.paragraphs])

text
```

Out[26]:
```
'Resume 1\n"Alice Johnson is a Data Scientist at Microsoft. She previously worked
at Amazon in Seattle. She joined Microsoft in 2021 and has seven years of experien
ce in the industry."\nResume 2\n"Robert Smith is a Machine Learning Engineer at Fa
cebook. He started working at Facebook in 2019 after graduating from Stanford Univ
ersity. He currently lives in California."\nResume 3\n"Emily Davis is a Software D
eveloper at IBM. Before joining IBM in 2020, she worked at Google. Emily completed
her studies at MIT in 2018 and now resides in New York."\nResume 4\n"Michael Brown
is a Cybersecurity Analyst at Cisco. He was previously employed at Oracle and relo
cated to Texas in 2022. Michael completed his Master\'s degree at Harvard Universi
ty in 2017."\nResume 5\n"Sophia Lee is a Product Manager at Tesla. She joined Tesl
a in 2023 after working at Apple for three years. Sophia is originally from Los An
geles."\n'
```

In [27]:
```
# 1. Tokenization
tokens = word_tokenize(text)
print("Tokenized Words:\n", tokens, "\n")
```

```
Tokenized Words:
 ['Resume', '1', "''", 'Alice', 'Johnson', 'is', 'a', 'Data', 'Scientist', 'at', 'Mi
crosoft', '.', 'She', 'previously', 'worked', 'at', 'Amazon', 'in', 'Seattle', '.',
'She', 'joined', 'Microsoft', 'in', '2021', 'and', 'has', 'seven', 'years', 'of', 'e
xperience', 'in', 'the', 'industry', '.', "''", 'Resume', '2', "''", 'Robert', 'Smit
h', 'is', 'a', 'Machine', 'Learning', 'Engineer', 'at', 'Facebook', '.', 'He', 'star
ted', 'working', 'at', 'Facebook', 'in', '2019', 'after', 'graduating', 'from', 'Sta
nford', 'University', '.', 'He', 'currently', 'lives', 'in', 'California', '.',
"''", 'Resume', '3', "''", 'Emily', 'Davis', 'is', 'a', 'Software', 'Developer', 'a
t', 'IBM', '.', 'Before', 'joining', 'IBM', 'in', '2020', ',', 'she', 'worked', 'a
t', 'Google', '.', 'Emily', 'completed', 'her', 'studies', 'at', 'MIT', 'in', '201
8', 'and', 'now', 'resides', 'in', 'New', 'York', '.', "''", 'Resume', '4', "''", 'M
ichael', 'Brown', 'is', 'a', 'Cybersecurity', 'Analyst', 'at', 'Cisco', '.', 'He',
'was', 'previously', 'employed', 'at', 'Oracle', 'and', 'relocated', 'to', 'Texas',
'in', '2022', '.', 'Michael', 'completed', 'his', 'Master', "'s", 'degree', 'at', 'H
arvard', 'University', 'in', '2017', '.', "''", 'Resume', '5', "''", 'Sophia', 'Le
e', 'is', 'a', 'Product', 'Manager', 'at', 'Tesla', '.', 'She', 'joined', 'Tesla',
'in', '2023', 'after', 'working', 'at', 'Apple', 'for', 'three', 'years', '.', 'Soph
ia', 'is', 'originally', 'from', 'Los', 'Angeles', '.', "''"]
```

In [28]:
```python
# 2. Stemming
stemmer = PorterStemmer()
stemmed_words = [stemmer.stem(word) for word in tokens]
print("Stemmed Words:\n", stemmed_words, "\n")
```

```
Stemmed Words:
 ['resum', '1', "''", 'alic', 'johnson', 'is', 'a', 'data', 'scientist', 'at', 'micr
osoft', '.', 'she', 'previous', 'work', 'at', 'amazon', 'in', 'seattl', '.', 'she',
'join', 'microsoft', 'in', '2021', 'and', 'ha', 'seven', 'year', 'of', 'experi', 'i
n', 'the', 'industri', '.', "''", 'resum', '2', "''", 'robert', 'smith', 'is', 'a',
'machin', 'learn', 'engin', 'at', 'facebook', '.', 'he', 'start', 'work', 'at', 'fac
ebook', 'in', '2019', 'after', 'graduat', 'from', 'stanford', 'univers', '.', 'he',
'current', 'live', 'in', 'california', '.', "''", 'resum', '3', "''", 'emili', 'dav
i', 'is', 'a', 'softwar', 'develop', 'at', 'ibm', '.', 'befor', 'join', 'ibm', 'in',
'2020', ',', 'she', 'work', 'at', 'googl', '.', 'emili', 'complet', 'her', 'studi',
'at', 'mit', 'in', '2018', 'and', 'now', 'resid', 'in', 'new', 'york', '.', "''", 'r
esum', '4', "''", 'michael', 'brown', 'is', 'a', 'cybersecur', 'analyst', 'at', 'cis
co', '.', 'he', 'wa', 'previous', 'employ', 'at', 'oracl', 'and', 'reloc', 'to', 'te
xa', 'in', '2022', '.', 'michael', 'complet', 'hi', 'master', "'s", 'degre', 'at',
'harvard', 'univers', 'in', '2017', '.', "''", 'resum', '5', "''", 'sophia', 'lee',
'is', 'a', 'product', 'manag', 'at', 'tesla', '.', 'she', 'join', 'tesla', 'in', '20
23', 'after', 'work', 'at', 'appl', 'for', 'three', 'year', '.', 'sophia', 'is', 'or
igin', 'from', 'lo', 'angel', '.', "''"]
```

In [29]:
```python
# 3. Lemmatization
lemmatizer = WordNetLemmatizer()
lemmatized_words = [lemmatizer.lemmatize(word) for word in tokens]
print("Lemmatized Words:\n", lemmatized_words, "\n")
```

```
Lemmatized Words:
 ['Resume', '1', "'", 'Alice', 'Johnson', 'is', 'a', 'Data', 'Scientist', 'at', 'Mi
crosoft', '.', 'She', 'previously', 'worked', 'at', 'Amazon', 'in', 'Seattle', '.',
'She', 'joined', 'Microsoft', 'in', '2021', 'and', 'ha', 'seven', 'year', 'of', 'exp
erience', 'in', 'the', 'industry', '.', "'", 'Resume', '2', "'", 'Robert', 'Smit
h', 'is', 'a', 'Machine', 'Learning', 'Engineer', 'at', 'Facebook', '.', 'He', 'star
ted', 'working', 'at', 'Facebook', 'in', '2019', 'after', 'graduating', 'from', 'Sta
nford', 'University', '.', 'He', 'currently', 'life', 'in', 'California', '.', "'",
'Resume', '3', "'", 'Emily', 'Davis', 'is', 'a', 'Software', 'Developer', 'at', 'IB
M', '.', 'Before', 'joining', 'IBM', 'in', '2020', ',', 'she', 'worked', 'at', 'Goog
le', '.', 'Emily', 'completed', 'her', 'study', 'at', 'MIT', 'in', '2018', 'and', 'n
ow', 'resides', 'in', 'New', 'York', '.', "'", 'Resume', '4', "'", 'Michael', 'Bro
wn', 'is', 'a', 'Cybersecurity', 'Analyst', 'at', 'Cisco', '.', 'He', 'wa', 'previou
sly', 'employed', 'at', 'Oracle', 'and', 'relocated', 'to', 'Texas', 'in', '2022',
'.', 'Michael', 'completed', 'his', 'Master', "'s", 'degree', 'at', 'Harvard', 'Univ
ersity', 'in', '2017', '.', "'", 'Resume', '5', "'", 'Sophia', 'Lee', 'is', 'a',
'Product', 'Manager', 'at', 'Tesla', '.', 'She', 'joined', 'Tesla', 'in', '2023', 'a
fter', 'working', 'at', 'Apple', 'for', 'three', 'year', '.', 'Sophia', 'is', 'origi
nally', 'from', 'Los', 'Angeles', '.', "'"]
```

In [30]:

```
# 4. POS Tagging
pos_tags = pos_tag(tokens)
print("POS Tags:\n", pos_tags, "\n")
```

```
POS Tags:
 [('Resume', '$'), ('1', 'CD'), ("'''", "'''"), ('Alice', 'NNP'), ('Johnson', 'NNP'),
('is', 'VBZ'), ('a', 'DT'), ('Data', 'NNP'), ('Scientist', 'NN'), ('at', 'IN'), ('Mi
crosoft', 'NNP'), ('.', '.'), ('She', 'PRP'), ('previously', 'RB'), ('worked', 'VB
D'), ('at', 'IN'), ('Amazon', 'NNP'), ('in', 'IN'), ('Seattle', 'NNP'), ('.', '.'),
('She', 'PRP'), ('joined', 'VBD'), ('Microsoft', 'NNP'), ('in', 'IN'), ('2021', 'C
D'), ('and', 'CC'), ('has', 'VBZ'), ('seven', 'CD'), ('years', 'NNS'), ('of', 'IN'),
('experience', 'NN'), ('in', 'IN'), ('the', 'DT'), ('industry', 'NN'), ('.', '.'),
("'''", "'''"), ('Resume', 'VBD'), ('2', 'CD'), ("'''", "'''"), ('Robert', 'NNP'), ('Smi
th', 'NNP'), ('is', 'VBZ'), ('a', 'DT'), ('Machine', 'NNP'), ('Learning', 'NNP'),
('Engineer', 'NNP'), ('at', 'IN'), ('Facebook', 'NNP'), ('.', '.'), ('He', 'PRP'),
('started', 'VBD'), ('working', 'VBG'), ('at', 'IN'), ('Facebook', 'NNP'), ('in', 'I
N'), ('2019', 'CD'), ('after', 'IN'), ('graduating', 'VBG'), ('from', 'IN'), ('Stanf
ord', 'NNP'), ('University', 'NNP'), ('.', '.'), ('He', 'PRP'), ('currently', 'RB'),
('lives', 'VBZ'), ('in', 'IN'), ('California', 'NNP'), ('.', '.'), ("'''", "'''"), ('R
esume', 'VBD'), ('3', 'CD'), ("'''", "'''"), ('Emily', 'RB'), ('Davis', 'NNP'), ('is',
'VBZ'), ('a', 'DT'), ('Software', 'NNP'), ('Developer', 'NNP'), ('at', 'IN'), ('IB
M', 'NNP'), ('.', '.'), ('Before', 'IN'), ('joining', 'VBG'), ('IBM', 'NNP'), ('in',
'IN'), ('2020', 'CD'), (',', ','), ('she', 'PRP'), ('worked', 'VBD'), ('at', 'IN'),
('Google', 'NNP'), ('.', '.'), ('Emily', 'RB'), ('completed', 'VBD'), ('her', 'PR
P'), ('studies', 'NNS'), ('at', 'IN'), ('MIT', 'NNP'), ('in', 'IN'), ('2018', 'CD'),
('and', 'CC'), ('now', 'RB'), ('resides', 'VBZ'), ('in', 'IN'), ('New', 'NNP'), ('Yo
rk', 'NNP'), ('.', '.'), ("'''", "'''"), ('Resume', 'VBD'), ('4', 'CD'), ("'''", "'''"),
('Michael', 'NNP'), ('Brown', 'NNP'), ('is', 'VBZ'), ('a', 'DT'), ('Cybersecurity',
'NNP'), ('Analyst', 'NNP'), ('at', 'IN'), ('Cisco', 'NNP'), ('.', '.'), ('He', 'PR
P'), ('was', 'VBD'), ('previously', 'RB'), ('employed', 'VBN'), ('at', 'IN'), ('Orac
le', 'NNP'), ('and', 'CC'), ('relocated', 'VBD'), ('to', 'TO'), ('Texas', 'NNP'),
('in', 'IN'), ('2022', 'CD'), ('.', '.'), ('Michael', 'NNP'), ('completed', 'VBD'),
('his', 'PRP$'), ('Master', 'NN'), ("'s", 'POS'), ('degree', 'NN'), ('at', 'IN'),
('Harvard', 'NNP'), ('University', 'NNP'), ('in', 'IN'), ('2017', 'CD'), ('.', '.'),
("'''", "'''"), ('Resume', 'VBD'), ('5', 'CD'), ("'''", "'''"), ('Sophia', 'NNP'), ('Le
e', 'NNP'), ('is', 'VBZ'), ('a', 'DT'), ('Product', 'NN'), ('Manager', 'NNP'), ('a
t', 'IN'), ('Tesla', 'NNP'), ('.', '.'), ('She', 'PRP'), ('joined', 'VBD'), ('Tesl
a', 'NNP'), ('in', 'IN'), ('2023', 'CD'), ('after', 'IN'), ('working', 'VBG'), ('a
t', 'IN'), ('Apple', 'NNP'), ('for', 'IN'), ('three', 'CD'), ('years', 'NNS'), ('.',
'.'), ('Sophia', 'NNP'), ('is', 'VBZ'), ('originally', 'RB'), ('from', 'IN'), ('Lo
s', 'NNP'), ('Angeles', 'NNP'), ('.', '.'), ("'''", "'''")]
```

In [31]:
```python
# 5. Named Entity Recognition (NER)
doc = nlp(text)
entity_dict = {"PERSON": [], "ORG": set(), "GPE": set(), "DATE": set()}

for ent in doc.ents:
    if ent.label_ in entity_dict:
        # Auto-correction: Ensure proper classification
        if ent.label_ == "ORG" and any(job in ent.text.lower() for job in ["scienti
            continue  # Skip job roles misclassified as ORG
        if ent.label_ == "GPE" and ent.text.lower() == "cisco":
            entity_dict["ORG"].add(ent.text)  # Correct Cisco to ORG
        elif ent.label_ == "PERSON":
            # Keep only the longest version of a person's name
            existing_names = entity_dict["PERSON"]
            should_add = True
            for i, name in enumerate(existing_names):
                if ent.text in name:
                    should_add = False  # A longer name already exists
                    break
                elif name in ent.text:
                    existing_names[i] = ent.text  # Replace shorter name with longe
                    should_add = False
                    break
            if should_add:
                existing_names.append(ent.text)
        else:
            entity_dict[ent.label_].add(ent.text)

print("Extracted Named Entities:")
for key, value in entity_dict.items():
    print(f"{key}: {value}")
```

```
Extracted Named Entities:
PERSON: ['Alice Johnson', 'Robert Smith', 'Emily Davis', 'Michael Brown', 'Sophia Le
e']
ORG: {'Stanford University', 'Google', 'Oracle', 'MIT', 'Apple', 'Microsoft', 'Harva
rd University', 'IBM', 'Tesla', 'Cisco', 'Amazon'}
GPE: {'Texas', 'Los Angeles', 'California', 'Seattle', 'New York'}
DATE: {'seven years', '2021', '2017', '2022', '2023', 'three years', '2019', '2020',
'2018'}
```