

- ✦ Installing Dependencies & Libraries

```

!pip install pdfplumber spacy phonenumbers sentence-transformers pandas numpy matplotlib seaborn reportlab
!python -m spacy download en_core_web_sm

Requirement already satisfied: python-dateutil<=2.8.2 in /usr/local/lib/python3.12/dist-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-packages (from pandas) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-packages (from pandas) (2025.3)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (1.3.3)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (4.61.1)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (1.4.9)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (3.2.5)
Requirement already satisfied: filelock in /usr/local/lib/python3.12/dist-packages (from huggingface-hub>=0.20.0->sentence-transformers) (3.16.1)
Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/python3.12/dist-packages (from huggingface-hub>=0.20.0->sentence-transformers) (2025.3.0)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.12/dist-packages (from huggingface-hub>=0.20.0->sentence-transformers) (6.0.2)
Requirement already satisfied: hf-xet<2.0.0,>=1.1.3 in /usr/local/lib/python3.12/dist-packages (from huggingface-hub>=0.20.0->sentence-transformers) (1.1.7)
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.12/dist-packages (from pydantic!=1.8,!<1.8.1,<3.0.0) (0.7.0)
Requirement already satisfied: pydantic-core==2.41.4 in /usr/local/lib/python3.12/dist-packages (from pydantic!=1.8,!<1.8.1,<3.0.0) (2.41.4)
Requirement already satisfied: typing-inspection>=0.4.2 in /usr/local/lib/python3.12/dist-packages (from pydantic!=1.8,!<1.8.1,<3.0.0) (0.10.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packages (from python-dateutil<=2.8.2->pandas) (1.17.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12/dist-packages (from requests<3.0.0,>=2.13.0->spacy) (3.10.1)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-packages (from requests<3.0.0,>=2.13.0->spacy) (2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.12/dist-packages (from requests<3.0.0,>=2.13.0->spacy) (2025.1.1)
Requirement already satisfied: blis<1.4.0,>=1.3.0 in /usr/local/lib/python3.12/dist-packages (from thinc<8.4.0,>=8.3.4->spacy) (1.4.0)
Requirement already satisfied: confection<1.0.0,>=0.0.1 in /usr/local/lib/python3.12/dist-packages (from thinc<8.4.0,>=8.3.4->spacy) (0.10.0)
Requirement already satisfied: sympy>=1.13.3 in /usr/local/lib/python3.12/dist-packages (from torch>=1.11.0->sentence-transformers) (1.13.3)
Requirement already satisfied: networkx>=2.5.1 in /usr/local/lib/python3.12/dist-packages (from torch>=1.11.0->sentence-transformers) (3.4.1)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.12/dist-packages (from transformers<6.0.0,>=4.41.0) (2024.11.6)
Requirement already satisfied: tokenizers<=0.23.0,>=0.22.0 in /usr/local/lib/python3.12/dist-packages (from transformers<6.0.0,>=4.41.0) (0.20.1)
Requirement already satisfied: safetensors>=0.4.3 in /usr/local/lib/python3.12/dist-packages (from transformers<6.0.0,>=4.41.0) (0.5.2)
Requirement already satisfied: click>=8.0.0 in /usr/local/lib/python3.12/dist-packages (from typer-slim<1.0.0,>=0.3.0->spacy) (8.1.8)
Requirement already satisfied: cloudpathlib<1.0.0,>=0.7.0 in /usr/local/lib/python3.12/dist-packages (from weasel<0.5.0,>=0.4.1) (0.19.0)
Requirement already satisfied: smart-open<8.0.0,>=5.2.1 in /usr/local/lib/python3.12/dist-packages (from weasel<0.5.0,>=0.4.1) (7.0.5)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.12/dist-packages (from jinja2->spacy) (3.0.3)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.12/dist-packages (from scikit-learn->sentence-transformers) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.12/dist-packages (from scikit-learn->sentence-transformers) (3.5.0)
Requirement already satisfied: cffi>=1.12 in /usr/local/lib/python3.12/dist-packages (from cryptography>=36.0.0->pdfminer.six) (1.17.1)
Requirement already satisfied: wrapt in /usr/local/lib/python3.12/dist-packages (from smart-open<8.0.0,>=5.2.1->weasel<0.5.0,>=0.4.1) (1.17.0)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.12/dist-packages (from sympy>=1.13.3->torch>=1.11.0) (1.3.0)
Requirement already satisfied: pycparser in /usr/local/lib/python3.12/dist-packages (from cffi>=1.12->cryptography>=36.0.0->pdfminer.six) (2.23)
Downloading pdfplumber-0.11.8-py3-none-any.whl (60 kB)
   60.0/60.0 kB 3.4 MB/s eta 0:00:00
Downloading pdfminer_six-20251107-py3-none-any.whl (5.6 MB)
   5.6/5.6 MB 56.4 MB/s eta 0:00:00
Downloading phonenumbers-9.0.21-py2.py3-none-any.whl (2.6 MB)
   2.6/2.6 MB 66.7 MB/s eta 0:00:00
Downloading reportlab-4.4.6-py3-none-any.whl (2.0 MB)
   2.0/2.0 MB 62.2 MB/s eta 0:00:00
Downloading pypdfium2-5.2.0-py3-none-manylinux_2_17_x86_64_manylinux2014_x86_64.whl (3.0 MB)
   3.0/3.0 MB 75.2 MB/s eta 0:00:00
Installing collected packages: reportlab, pypdfium2, phonenumbers, pdfminer.six, pdfplumber
Successfully installed pdfminer.six-20251107 pdfplumber-0.11.8 phonenumbers-9.0.21 pypdfium2-5.2.0 reportlab-4.4.6
Collecting en-core-web-sm==3.8.0
  Downloading https://github.com/explosion/spacy-models/releases/download/en_core_web_sm-3.8.0/en_core_web_sm-3.8.0-py3-none-any.whl (12.8 MB)
    12.8/12.8 MB 59.5 MB/s eta 0:00:00

✓ Download and installation successful
You can now load the package via spacy.load('en_core_web_sm')
⚠ Restart to reload dependencies
If you are in a Jupyter or Colab notebook, you may need to restart Python in order to load all the package's dependencies. You can do this by selecting the 'Restart kernel' or 'Restart runtime' option.

```

Information Extraction From pdf CV's, ATS\_Scoring , & Matching with suitable Internships , then Visualization of Results

```
import os
import re
import json
import warnings
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```

import pdfplumber
import spacy
import phonenumbers
from sentence_transformers import SentenceTransformer, util

# Configuration
warnings.filterwarnings('ignore')
MODEL_DIR = "cv_ai_models"
OUTPUT_DIR = "output"
os.makedirs(MODEL_DIR, exist_ok=True)
os.makedirs(OUTPUT_DIR, exist_ok=True)

# =====
# 0. DATA PREPARATION (CSV DATASET)
# =====
def create_csv_dataset():
    """Creates the internships CSV file from the provided data."""
    csv_content = """internship_id,company,position,field,description,required_skills,preferred_skills,location,duration,monthly_pay
INT0001,PalTel Group,Cloud Networking Intern,Network Engineering,"Join PalTel Group as a Cloud Networking Intern in Gaza. Work with Python, Docker, and AWS.",["Python", "Docker", "AWS"],"Gaza", "3 months", 1500
INT0002,PalTel Group,Azure Intern,Cloud Computing,"Join PalTel Group as a Azure Intern in Bethlehem. Work with Python, Docker, S3, and Azure.",["Python", "Docker", "S3", "Azure"],"Bethlehem", "3 months", 1500
INT0021,Bank of Palestine,Financial Technology Intern,FinTech,"Join Bank of Palestine in Ramallah. Node.js, Blockchain, React.",["Node.js", "Blockchain", "React"],"Ramallah", "3 months", 1500
INT0041,ASAL Technologies,Statistical Analysis Intern,Data Science,"Join ASAL Technologies in Bethlehem. NLTK, Spark, TensorFlow, and Scikit-Learn.",["NLTK", "Spark", "TensorFlow", "Scikit-Learn"],"Bethlehem", "3 months", 1500
INT0061,Exalt Technologies,Digital Forensics Intern,Cybersecurity,"Join Exalt Technologies in Ramallah. Wireshark, Metasploit, Kali Linux, and Python.",["Wireshark", "Metasploit", "Kali Linux", "Python"],"Ramallah", "3 months", 1500
INT0082,Arab American University,Business Intelligence Intern,Data Science,"Join AAU in Jenin. Latex, Statistical Analysis, Python, and Power BI.",["Latex", "Statistical Analysis", "Python", "Power BI"],"Jenin", "3 months", 1500
INT0122,Palestine Islamic Bank,React Native Intern,Mobile Development,"Join PIB in Ramallah. React, Flutter, MongoDB.",["React", "Flutter", "MongoDB"],"Ramallah", "3 months", 1500
INT0161,Palestine Polytechnic University,Full Stack Developer Intern,Software Engineering,"Join PPU in Hebron. IoT Protocols, Raspberry Pi, and Python.",["IoT Protocols", "Raspberry Pi", "Python"],"Hebron", "3 months", 1500
INT0174,Palestine Polytechnic University,NLP Engineering Intern,Artificial Intelligence,"Join PPU in Hebron. Embedded Linux, IoT, TensorFlow, and Python.",["Embedded Linux", "IoT", "TensorFlow", "Python"],"Hebron", "3 months", 1500
INT0051,ASAL Technologies,Python Developer Intern,Software Engineering,"Join ASAL in Ramallah. Python, PyTorch, OpenCV, Kubernetes, and Docker.",["Python", "PyTorch", "OpenCV", "Kubernetes", "Docker"],"Ramallah", "3 months", 1500
"""
    if not os.path.exists("palestinian_internships_200.csv"):
        with open("palestinian_internships_200.csv", "w") as f:
            f.write(csv_content)
        print("✅ CSV Dataset created.")

# =====
# 1. AI ENGINE INITIALIZATION
# =====
def initialize_models():
    print("\n" + "="*60)
    print("🚀 INITIALIZING AI ENGINES")
    print("="*60)

    # 1. NLP
    try:
        nlp = spacy.load("en_core_web_sm")
        print("✅ Spacy NLP Model Loaded.")
    except:
        print("⚠️ Downloading Spacy Model...")
        os.system("python -m spacy download en_core_web_sm")
        nlp = spacy.load("en_core_web_sm")

    # 2. SBERT
    sbert_path = os.path.join(MODEL_DIR, "sbert_model")
    if not os.path.exists(sbert_path):
        print("⚠️ Downloading SBERT Model...")
        model = SentenceTransformer('all-MiniLM-L6-v2')
        model.save(sbert_path)
    else:
        model = SentenceTransformer(sbert_path)
    print("✅ SBERT Transformer Loaded.")

    return nlp, model

# =====
# 2. ELITE EXTRACTOR CLASS (Fixed)
# =====
class EliteCVExtractor:
    def __init__(self, nlp):
        self.nlp = nlp
        # Expanded Headers to capture Academic CV sections
        self.headers = {
            'education': ['EDUCATION', 'ACADEMIC BACKGROUND', 'QUALIFICATIONS'],
            'experience': ['EXPERIENCE', 'ACADEMIC EXPERIENCE', 'TEACHING EXPERIENCE', 'PROFESSIONAL EXPERIENCE', 'WORK HISTORY'],
            'skills': ['SKILLS', 'TECHNICAL SKILLS', 'TECHNOLOGIES', 'COMPETENCIES'],
            'projects': ['PROJECTS', 'RESEARCH & PROJECTS', 'PUBLICATIONS', 'RESEARCH INTERESTS']
        }

```

```

def process(self, pdf_path):
    print(f"\n📖 Reading PDF: {pdf_path}")
    lines = []
    try:
        with pdfplumber.open(pdf_path) as pdf:
            for page in pdf.pages:
                text = page.extract_text()
                if text: lines.extend(text.split('\n'))
    except Exception as e:
        print(f"❌ Error reading PDF: {e}")
        return None

    full_text = "\n".join(lines)

    # 1. Extract Name (Smart NLP + Heuristic)
    name = self._extract_name(lines)

    # 2. Extract Phone (Fixed Regex for (+970)599...)
    phone = self._extract_phone(full_text)

    # 3. Extract Email
    email = self._extract_email(full_text)

    # 4. Smart Section Segmentation
    sections = self._segment_sections(lines)

    # 5. Education (Handle "Bachelor's" with apostrophe)
    education_data = self._refine_education(sections['education'])

    # 6. Skills (NLP Extraction)
    refined_skills = self._refine_skills(sections['skills'], full_text)

    # Combine different experience types for analysis
    combined_experience = sections['experience'] + sections['projects']

    return {
        "personal": {"name": name, "phone": phone, "email": email},
        "education": education_data,
        "experience": combined_experience,
        "skills": refined_skills,
        "raw_text": full_text
    }

def _extract_name(self, lines):
    """
    Uses NLP to find a Person entity in the first 10 lines.
    Fallback: First line if it looks like a name (uppercase, < 5 words).
    """
    for line in lines[:10]:
        clean = line.strip()
        if not clean: continue

        # 1. Spacy Check
        doc = self.nlp(clean)
        for ent in doc.ents:
            if ent.label_ == "PERSON" and len(clean.split()) < 5:
                return ent.text.strip()

        # 2. Formatting Check (All Caps, short line)
        if clean.isupper() and len(clean.split()) < 4:
            return clean

    return lines[0].strip() if lines else "Unknown"

def _extract_phone(self, text):
    # Specific regex for the format in the PDF: (+970)599320207
    # Also handles +970 599... and 059...
    patterns = [
        r'\(\+97[02]\)\s?\d{9}',          # (+970)599320207
        r'\+97[02]\s?\d{3}\s?\d{6}',     # +970 599 320207
        r'\b05\d{8}\b'                  # 0599320207
    ]

    for p in patterns:
        match = re.search(p, text)
        if match:
            return match.group(0).strip()

```

```

        return "Not Found"

def _extract_email(self, text):
    match = re.search(r'\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}\b', text)
    return match.group(0) if match else "Not Found"

def _segment_sections(self, lines):
    sections = {k: [] for k in self.headers.keys()}
    current_sec = None

    for line in lines:
        clean = line.strip().upper()
        if not clean: continue

        # Check if header
        is_header = False
        for sec, keys in self.headers.items():
            # Allow exact match OR starts with (e.g., "ACADEMIC EXPERIENCE")
            if any(clean == k or clean.startswith(k) for k in keys):
                # Safety: Headers usually aren't very long sentences
                if len(clean) < 40:
                    current_sec = sec
                    is_header = True
                    break

        if is_header: continue
        if current_sec:
            sections[current_sec].append(line.strip())

    return sections

def _refine_education(self, edu_lines):
    # If section extracted, return it.
    # Logic to ensure "Bachelor's" is captured is implicitly handled by segment_sections
    # getting the lines under "EDUCATION".
    # We can also filter for relevance.
    refined = [line for line in edu_lines if len(line) > 5]
    return refined

def _refine_skills(self, skill_lines, full_text):
    """NLP extraction of technical terms."""
    # Combine skill section lines
    text = " ".join(skill_lines)

    # If skills section is empty/sparse, scan full text for known keywords
    if len(text) < 20:
        text = full_text

    doc = self.nlp(text)
    skills = []

    # 1. Noun Chunks/Entities
    for token in doc:
        # Keep Proper Nouns (Python, Java) and Nouns
        if token.pos_ in ['PROPN', 'NOUN'] and len(token.text) > 2:
            skills.append(token.text.capitalize())

    # 2. Specific Whitelist (things spacy might miss or tag as verbs)
    whitelist = ['C++', 'C#', 'Go', 'React', 'Vue', 'Node.js', 'Latex', 'Docker', 'Kubernetes', 'SQL', 'NoSQL', 'Git', 'Linux']
    for w in whitelist:
        if w.lower() in text.lower():
            skills.append(w)

    return list(set(skills))

# =====
# 3. ACCURATE ATS SCORING
# =====
class InternshipMatcher:
    def __init__(self, sbert_model, csv_path):
        self.model = sbert_model
        self.df = pd.read_csv(csv_path).fillna('')

        print("🔄 Computing Job Embeddings...")
        self.df['combined_text'] = (
            self.df['position'] + " " +
            self.df['required_skills'] + " " +

```

```

        self.df['description']
    )
    self.job_embeddings = self.model.encode(self.df['combined_text'].tolist(), convert_to_tensor=True)

def calculate_ats_score(self, cv_data):
    """
    Calibrated scoring for Academic/High-Level CVs.
    """
    score_breakdown = {}

    # 1. Contact (15 pts)
    c_score = 0
    if cv_data['personal']['name'] != "Unknown": c_score += 5
    if cv_data['personal']['phone'] != "Not Found": c_score += 5
    if cv_data['personal']['email'] != "Not Found": c_score += 5
    score_breakdown['Contact'] = c_score

    # 2. Education (35 pts) - Recognizes PhD/Masters/Bachelors
    edu_text = " ".join(cv_data['education']).upper()
    e_score = 10 # Base for having section
    if 'PHD' in edu_text or 'DOCTOR' in edu_text: e_score += 25
    elif 'MASTER' in edu_text or 'M.SC' in edu_text: e_score += 15
    elif 'BACHELOR' in edu_text or 'B.SC' in edu_text: e_score += 10
    score_breakdown['Education'] = min(35, e_score)

    # 3. Experience (30 pts) - Recognizes "Professor", "Chair", "Years"
    exp_text = " ".join(cv_data['experience']).upper()
    ex_score = 10 # Base
    # Keywords for high-level experience
    if any(x in exp_text for x in ['PROFESSOR', 'LECTURER', 'MANAGER', 'LEAD', 'CHAIR']):
        ex_score += 20
    elif len(cv_data['experience']) > 15: # High volume of bullets
        ex_score += 15
    elif len(cv_data['experience']) > 5:
        ex_score += 5
    score_breakdown['Experience'] = min(30, ex_score)

    # 4. Skills/Research (20 pts)
    s_score = len(cv_data['skills'])
    # Cap at 20 (roughly 1 pt per skill detected)
    score_breakdown['Skills'] = min(20, s_score + 5) # +5 base

    total = sum(score_breakdown.values())
    return total, score_breakdown

def match(self, cv_data, ats_score):
    # Candidate Embedding
    cand_text = (
        " ".join(cv_data['skills']) + " " +
        " ".join(cv_data['experience']) + " " +
        " ".join(cv_data['education'])
    )
    cand_emb = self.model.encode(cand_text, convert_to_tensor=True)

    # Similarity
    cos_scores = util.cos_sim(cand_emb, self.job_embeddings)[0]

    matches = []
    for idx, score in enumerate(cos_scores):
        semantic_pct = float(score) * 100

        # Weighted Final Score: 60% Semantic Match + 40% CV Quality (ATS Score)
        # This ensures a high-quality CV (PhD) gets matched highly even if semantic overlap varies
        final_score = (semantic_pct * 0.6) + (ats_score * 0.4)

        matches.append({
            "company": self.df.iloc[idx]['company'],
            "position": self.df.iloc[idx]['position'],
            "location": self.df.iloc[idx]['location'],
            "stipend": self.df.iloc[idx]['monthly_stipend'],
            "final_score": final_score
        })

    return sorted(matches, key=lambda x: x['final_score'], reverse=True)

# =====
# 4. ADVANCED VISUALIZATION

```

```

# =====
def visualize_dashboard(cv_data, matches, ats_score, ats_breakdown):
    print("\n" + "="*60)
    print("📊 GENERATING DASHBOARD")
    print("="*60)

    sns.set_theme(style="whitegrid")
    fig = plt.figure(figsize=(20, 14))

    name = cv_data['personal']['name']
    fig.suptitle(f"Executive CV Analysis: {name}", fontsize=24, fontweight='bold', y=0.96)

    # 1. ATS Breakdown (Donut)
    ax1 = fig.add_subplot(231)
    ax1.pie(ats_breakdown.values(), labels=ats_breakdown.keys(), autopct='%1.1f%%', startangle=140, colors=sns.color_palette("pa
    ax1.add_artist(plt.Circle((0,0),0.70,fc='white'))
    ax1.text(0, 0, f"{ats_score}", ha='center', va='center', fontsize=24, fontweight='bold', color='#333')
    ax1.set_title("ATS Score", fontweight='bold')

    # 2. Profile Readiness (Radar)
    ax2 = fig.add_subplot(232, polar=True)
    cats = list(ats_breakdown.keys())
    vals = list(ats_breakdown.values())
    max_vals = [15, 35, 30, 20] # Max pts per section defined in rubric
    norm_vals = [(v/m)*100 for v, m in zip(vals, max_vals)]

    angles = np.linspace(0, 2*np.pi, len(cats), endpoint=False).tolist()
    norm_vals += [norm_vals[0]]
    angles += [angles[0]]

    ax2.plot(angles, norm_vals, color='#4c72b0', linewidth=2)
    ax2.fill(angles, norm_vals, color='#4c72b0', alpha=0.25)
    ax2.set_xticks(angles[:-1])
    ax2.set_xticklabels(cats, size=12, fontweight='bold')
    ax2.set_title("Profile Strength", pad=20, fontweight='bold')

    # 3. Skills Bar
    ax3 = fig.add_subplot(233)
    skills = cv_data['skills'][:10]
    if skills:
        sns.barplot(x=list(range(len(skills))), y=skills, palette="mako", ax=ax3, orient='h')
    ax3.set_title("Key Skills Detected", fontweight='bold')
    ax3.set_xlabel("Relevance")

    # 4. Top Matches Confidence
    ax4 = fig.add_subplot(234)
    top_5 = matches[:5]
    lbls = [f"{m['company']}\n{m['position']}" for m in top_5]
    scs = [m['final_score'] for m in top_5]
    sns.barplot(x=scs, y=lbls, palette="rocket", ax=ax4)
    ax4.set_xlim(0, 100)
    ax4.set_title("Top 5 Match Confidence", fontweight='bold')

    # 5. Stipend vs Fit
    ax5 = fig.add_subplot(235)
    top_10 = matches[:10]
    stips = [float(m['stipend']) for m in top_10]
    fits = [m['final_score'] for m in top_10]
    comps = [m['company'] for m in top_10]
    sns.scatterplot(x=stips, y=fits, hue=comps, s=300, palette="deep", ax=ax5, legend=False)
    ax5.set_title("Stipend vs Fit", fontweight='bold')
    ax5.set_xlabel("Stipend ($)")
    ax5.set_ylabel("Fit Score")

    # 6. Location
    ax6 = fig.add_subplot(236)
    locs = [m['location'] for m in matches[:10]]
    from collections import Counter
    lc = Counter(locs)
    ax6.pie(lc.values(), labels=lc.keys(), autopct='%1.1f%%', colors=sns.color_palette("Set3"))
    ax6.set_title("Location Distribution", fontweight='bold')

    plt.tight_layout()
    plt.savefig(os.path.join(OUTPUT_DIR, "executive_report.png"), dpi=300)
    plt.show()

# =====
# MAIN EXECUTION

```

```

# =====
if __name__ == "__main__":
    create_csv_dataset()
    nlp, sbert_model = initialize_models()

    CV_FILE = "Anas_CV.pdf"

    if os.path.exists(CV_FILE):
        # 1. Extract
        extractor = EliteCVExtractor(nlp)
        cv_data = extractor.process(CV_FILE)

        # Save Extracted Data
        with open(os.path.join(OUTPUT_DIR, "cv_extracted.json"), "w") as f:
            json.dump(cv_data, f, indent=4)

        print(f"\n👤 ANALYSIS FOR: {cv_data['personal']['name']}")
        print(f"📞 Phone: {cv_data['personal']['phone']}")
        print(f"🎓 Education Count: {len(cv_data['education'])}")

        # 2. Match
        matcher = InternshipMatcher(sbert_model, "palestinian_internships_200.csv")
        ats_score, ats_breakdown = matcher.calculate_ats_score(cv_data)
        matches = matcher.match(cv_data, ats_score)

        print(f"\n📊 ATS SCORE: {ats_score}/100")
        print(f"    Breakdown: {ats_breakdown}")

        print("\n🏆 TOP RECOMMENDATIONS:")
        print(f"{'#':<3} {'Company':<20} {'Position':<25} {'Loc':<10} {'Score':<5}")
        print("-" * 75)
        for i, m in enumerate(matches[:5], 1):
            print(f"{i:<3} {m['company']:<20} {m['position']:<25} {m['location']:<10} {m['final_score']:.1f}%")

        # 3. Visualize
        visualize_dashboard(cv_data, matches, ats_score, ats_breakdown)
        print("\n✅ Process Complete.")
    else:
        print(f"❌ Error: Please upload '{CV_FILE}' first.")

```

=====

⚙️ INITIALIZING AI ENGINES

=====

✅ Spacy NLP Model Loaded.  
⌚ Downloading SBERT Model...  
✅ SBERT Transformer Loaded.

📄 Reading PDF: Anas\_CV.pdf

👤 ANALYSIS FOR: ANAS MELHEM

📞 Phone: (+970)599320207

🎓 Education Count: 11

⌚ Computing Job Embeddings...

📄 ATS SCORE: 100/100

Breakdown: {'Contact': 15, 'Education': 35, 'Experience': 30, 'Skills': 20}

🏆 TOP RECOMMENDATIONS:

#	Company	Position	Loc	Score
1	Bank of Palestine	Digital Forensics Intern	Ramallah	78.0%
2	Exalt Technologies	Ethical Hacking Intern	Ramallah	75.2%
3	Exalt Technologies	Security Operations Intern	Jerusalem	75.2%
4	Exalt Technologies	Full Stack Developer Intern	Ramallah	73.4%
5	Exalt Technologies	Digital Forensics Intern	Ramallah	71.5%

=====

📊 GENERATING DASHBOARD

=====

### Executive CV Analysis: ANAS MELHEM

