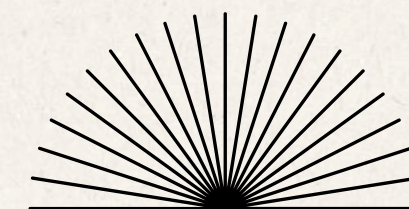




ATS INSIGHT - INTELLIGENT RESUME ANALYZER

Dominika Piechota

**Final project for the course
Data Management**



Agenda

03	Why do we need it?
04	Research questions
05	Data
06	Extracting text
07	NLP
09	ML
10	Output
11	Visualization
12	If time allows...

Why do we need CV analysis?

- **Many companies use ATS** (Applicant Tracking Systems) that automatically analyse and filter CVs.
 - People looking for a job often submit well-prepared CVs but **do not receive replies**.
 - It's hard to know whether your **document is ATS-compatible** or contains the keywords that companies look for.
-

Research Questions

- How can information such as skills, education and experience be effectively extracted from a CV?
 - Is it possible to calculate an objective CV quality indicator?
 - Which elements – skills, education or description – have the greatest impact on the result?
-

Datasets

- Kaggle Resume Dataset
 - Overleaf resume templates
 - Canva resume templates
 - Synthetic CVs (generated)
 - My own + friends' CVs (as a validation set)
-

Extracting text from PDF

I use a two-layer approach to process PDF documents:

- **pdfminer.six** - for precise extraction of raw text
 - **PyMuPDF** (fitz) - analyse the document structure - identify page layout, text blocks, headings, fonts, and separate sections such as Skills, Education, or About Me.
-

NLP Extraction Layer

1. **Basic spaCy model** for tokenisation, POS tagging and detection of basic entities (NAME, COMPANIES, DATES, LOCATIONS)
 2. **Rule-Based Matching (spaCy Matcher)**, for example for recognition of tool, library and framework names from a predefined list (e.g. TensorFlow, Git, Linux)
 3. **Regex and industry heuristics** for:
 - detection of programming languages (Python, C++, Java, SQL)
 - detection of technical links (GitHub, LinkedIn, Portfolio)
 - parsing of seniority levels (junior/mid/senior)
-

NLP Extraction Layer

4. Analysis and standardisation of CV sections

- recognition of sections by headings (SKILLS, EDUCATION, EXPERIENCE, ABOUT ME)
- mapping all versions of headings to a single standard
- organising content into a JSON structure

5. JSON output as a unified data format

```
{
  "contact_info": {
    "full_name": "...",
    "email": "...",
    "phone": "..."
  },
  "about_me": "...",
  "skills": ["...", "..."],
  "programming_languages": ["...", "..."],
  "experience": [
    {
      "job_title": "...",
      ...
    }
  ],
  "education": [
    {
      ...
    }
  ],
  "scores": {
    ...
  }
}
```


ML Modelling

Input variables:

- number of matched skills
- “About Me” section vectorized with TF-IDF
- education and experience score
- keyword relevance (cosine similarity with job offer)

Models to test:

1. Logistic Regression
2. Random Forest Classifier

Evaluation: F1 score, ROC AUC, Confusion matrix, z-score for scailing

Output

Each CV will receive a score on a scale of 1 to 5 indicating its suitability for a specific job offer.

Visualization

Planned plots:

- Frequency of skills across all CVs
- Education level distribution
- Most important words for a sample job offer
- z-score plots → CVs above/below average
- Wordcloud: most common keywords in “About me”

Tools:

- Matplotlib
 - Seaborn
-

Web interface (if time allows)

Streamlit dashboard

1. Upload CV
 2. Display parsed fields
 3. Show scores & visualizations
 4. Explain suggestions (skills missing, weak descriptions, etc.)
-

Timeline

