Career Navigator Documentation

Overview

Career Navigator is a web-based application designed to help users navigate their career paths by providing role recommendations based on their skills and preferences.

Prerequisites

Before running the application, ensure the following tools and dependencies are installed:

1. **Operating System**: Ubuntu 22.04 on WSL
2. **Python**: Version 3.12
3. **Git**: Version control system
4. **AWS CLI**: For cloud-related operations
5. **uv**: Dependency and environment manager
6. **npm**: Dependency and environment manager

Project Configuration

To configure the project, create a configuration file at ./config/.ml-env with the following environment variables:

```
CUDA_VISIBLE_DEVICES=""
LABEL_MODEL_PATH="./data/role_label_encoder.pkl"
JOB_PREDICT_MODEL_PATH="./data/job_model.keras"
SENTENCE_TRANSFORMER_MODEL="all-MiniLM-L6-v2"
GEMINI_KEY=<your-api-key>
```

Additionally, install of the SpaCy model may be needed:

```
uv pip install 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
```

Starting the Application

Backend Service

1. Sync dependencies:
   ```
   uv sync
   ```
2. Pull required data files:
   ```
   dvc pull
   ```
3. Navigate to the src directory:
   ```
   cd src
   ```
4. Start the backend service:
   ```
   uv run uvicorn api.app:app
   ```

Frontend UI

1. Navigate to the frontend directory:
   ```
   cd src/frontend
   ```
2. Install dependencies:
   ```
   npm install
   ```
3. Start the development server:
   ```
   npm run dev
   ```

Project Structure

The project is organized as follows:

```
├── .dvc/            # DVC configuration and cache
├── .github/         # GitHub workflows
├── doc/             # Documentation and reports
├── src/             # Source code
|   ├── api/         # Backend API
|   ├── config/      # Configuration files
|   ├── data/        # Data models and files
|   └── frontend/    # Frontend React application
├── LICENSE          # License file
├── README.md        # Project overview
└── pyproject.toml   # Python project configuration
```

Training Models

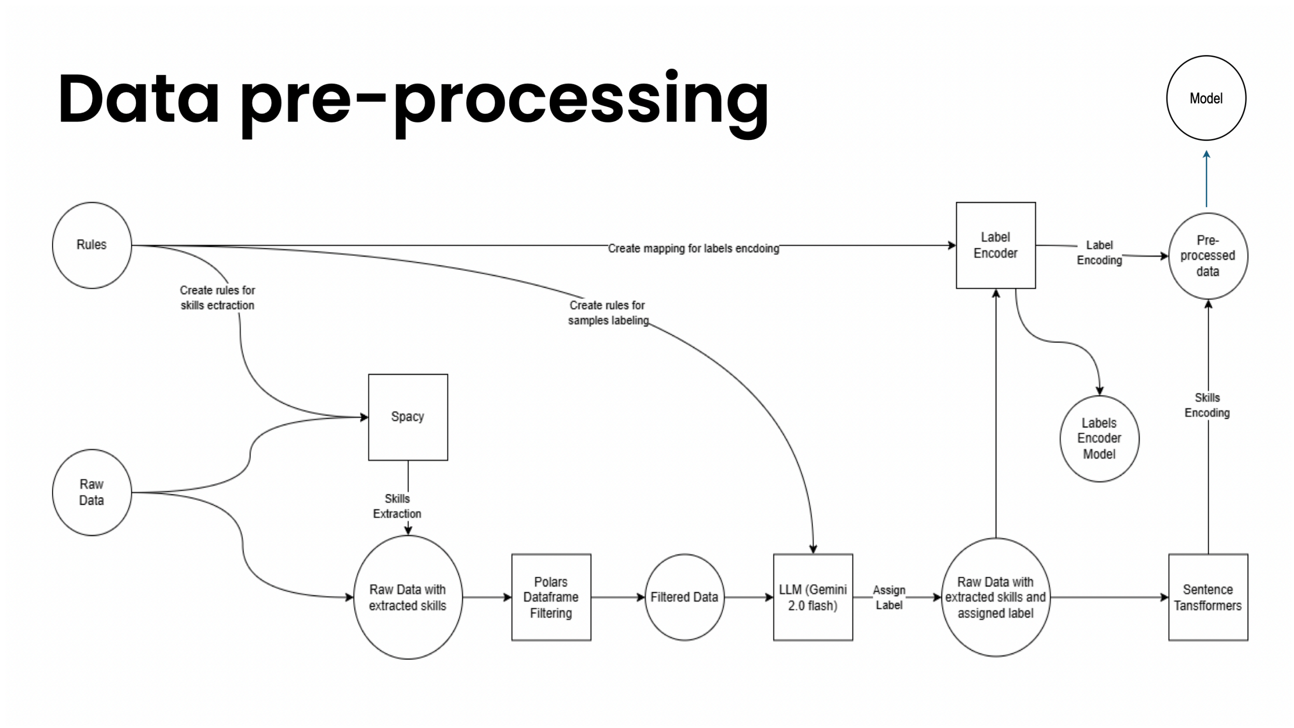
The Career Navigator application uses machine learning models to predict job roles based on user input. Below is a description of the models used:

1. Role Label Encoder:
   - **File**: role_label_encoder.pkl
   - **Purpose**: Encodes job roles into numerical labels for model training and prediction.
   - **Training**: This model was trained using a dataset of predefined job roles and their corresponding labels.
2. Job Prediction Model:
   - **File**: job_model.keras
   - **Purpose**: Predicts the most suitable job role for a user based on their skills and preferences.
   - **Architecture**: A neural network model trained using TensorFlow/Keras.

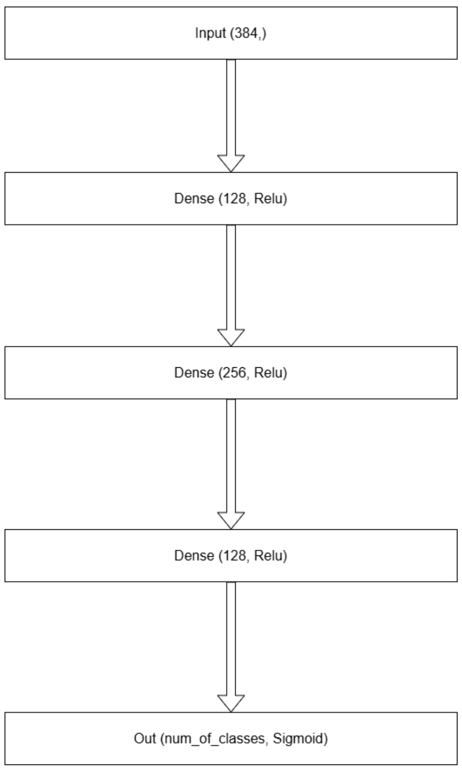
    - **Training**: The model was trained on a dataset of job descriptions, including position titles, job descriptions, company names, experience requirements, keywords, English proficiency levels and publication dates. It uses the encoded labels from the Role Label Encoder.
3. Sentence Transformer:
    - **Model**: all-MiniLM-L6-v2
    - **Purpose**: Converts textual input (e.g., user skills and job descriptions) into vector embeddings for similarity comparison.
    - **Source**: Pretrained model from Hugging Face's Sentence Transformers library.

Training Process

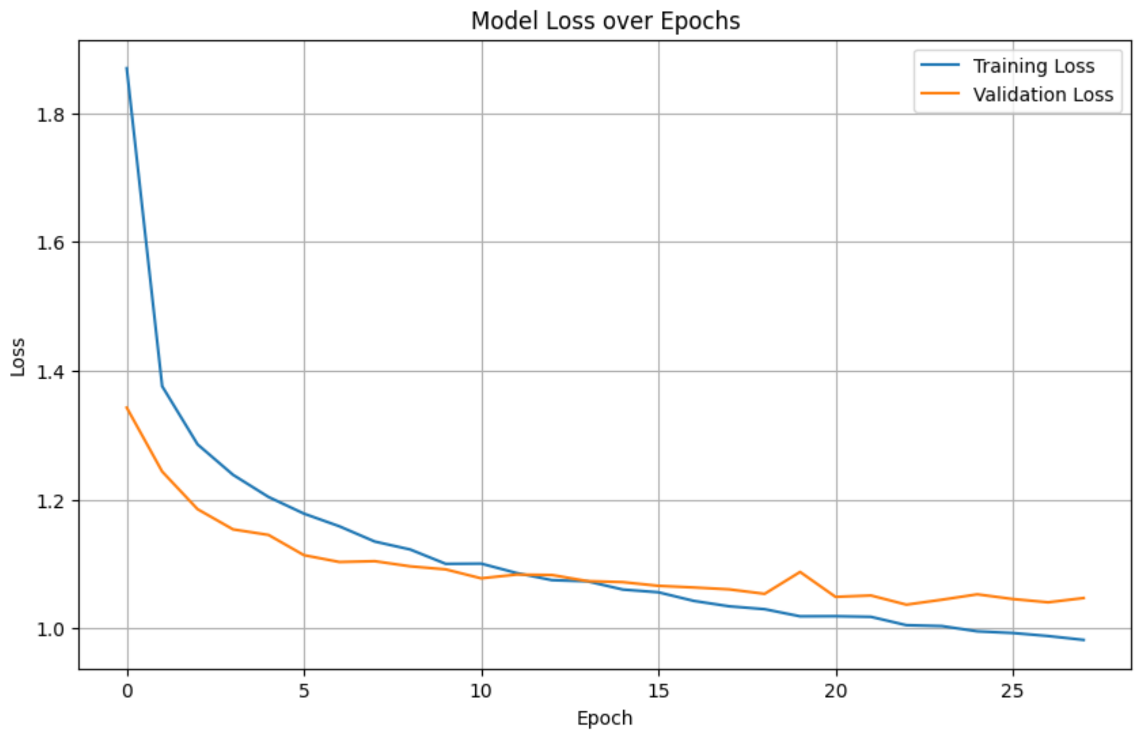
- **Data Preparation**: The training datasets were preprocessed to clean and normalize the input data.
- **Model Training**: The models were trained using Python scripts located in the src/api/training directory.
- **Evaluation**: The models were evaluated using metrics such as accuracy, precision, recall, and F1-score.
- **Version Control**: DVC was used to version control the training datasets and model artifacts.

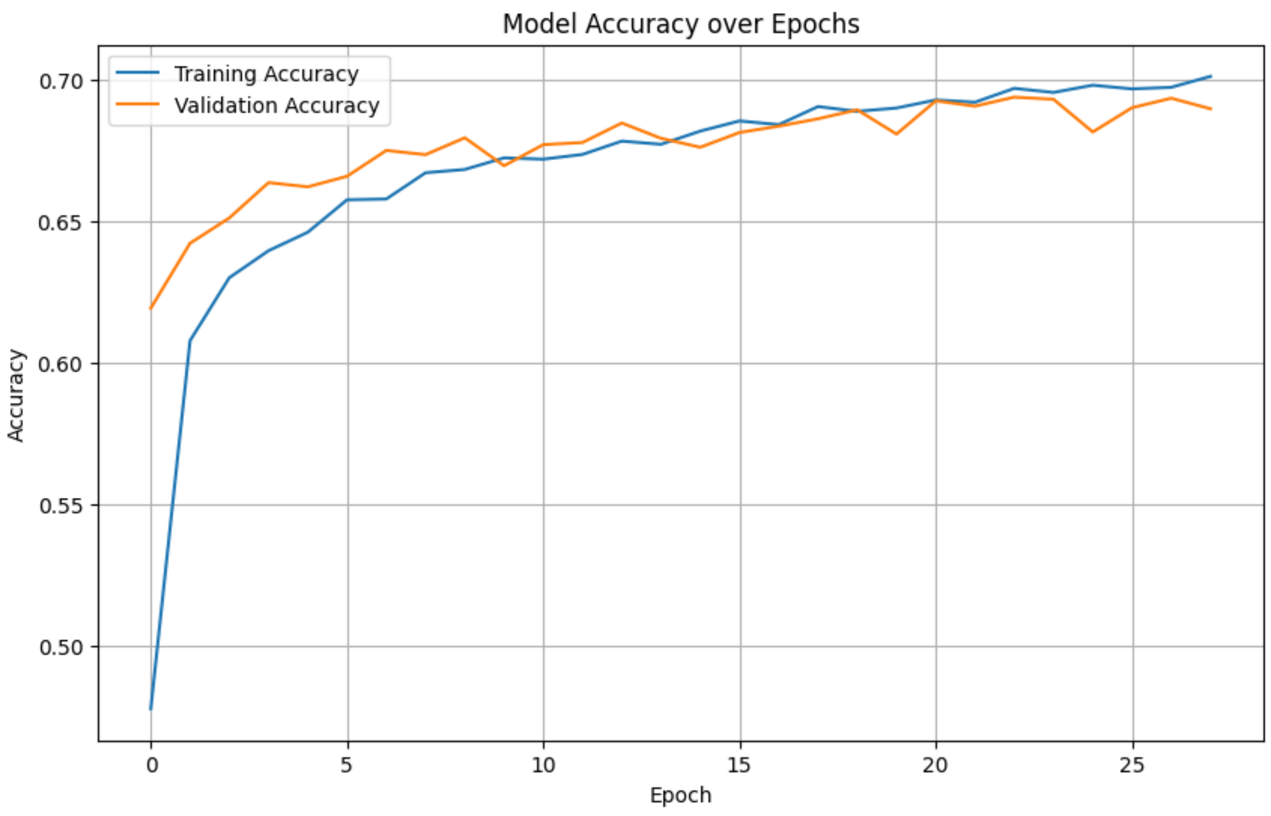


Model architecture



Model metrics





```
              precision    recall  f1-score   support

           0      0.500     0.447     0.472        38
           1      0.673     0.584     0.625       197
           2      0.000     0.000     0.000         9
           3      0.791     0.849     0.819       378
           4      0.436     0.567     0.493        60
           5      0.644     0.565     0.602       154
           6      0.571     0.429     0.490        28
           7      0.782     0.868     0.823       965
           8      0.808     0.700     0.750        30
           9      0.412     0.583     0.483        12
          10      0.478     0.256     0.333       258
          11      0.453     0.400     0.425        85
          12      0.000     0.000     0.000        13
          13      0.683     0.819     0.745       698
          14      0.553     0.512     0.532        41
          15      0.800     0.222     0.348        18
          16      0.654     0.770     0.707       466
          17      0.801     0.895     0.845       455
          18      0.000     0.000     0.000        35
          19      0.734     0.798     0.765       336
          20      0.562     0.461     0.506       178
          21      0.507     0.516     0.512       275
          22      0.727     0.170     0.276        47
          23      0.000     0.000     0.000         5
          24      0.000     0.000     0.000        52
          25      0.571     0.407     0.475        59
          26      0.000     0.000     0.000        17
          27      0.744     0.652     0.695       374
          28      0.470     0.477     0.473        65
          29      0.533     0.444     0.485        18

    accuracy                          0.694      5366
   macro avg      0.496     0.446     0.456      5366
weighted avg      0.669     0.694     0.675      5366
```

API

Overview

The Job Role Recommendation API is a FastAPI-based service that provides:

- A list of available skills known by the system.
- Job role recommendations based on the user's provided skills.

Endpoints

1. GET /

- **Description**: Root endpoint that returns a welcome message.
- Response:

```
{
  "message": "Welcome to the Job Role Recommendation API"
}
```

- Response Code: 200 OK

2. GET /available_skills

- **Description**: Returns a list of available skills that can be used for job role recommendations.
- Response:
    - Response Model: SkillsResponse

```
{
  "skills": [
    "Python",
    "Machine Learning",
    "Data Analysis",
    "Project Management",
    "Communication"
  ]
}
```

- Response Code: 200 OK

Details:

- The data is loaded from the data/metadata.json file.
- Skills are aggregated from various job segments and returned as a sorted list.

3. POST /recommend

- **Description**: Generates job role recommendations based on the user's provided skills.

- Headers:
    - Content-Type: application/json
- Input Data:
    - **Input Model**: RecommendationRequest

```
{
 "skills": ["Python", "Machine Learning", "Data Analysis"]
}
```

- Response:
    - **Response Model**: RecommendationResponse

```
{
 "recommendations": [
  {
   "role": "Data Scientist",
   "matchPercent": 85,
   "skills": [
    {
     "skill": "Python",
     "status": true,
     "skillGapPercent": null
    },
    {

     "skill": "Machine Learning",
     "status": true,
     "skillGapPercent": null

    },
    {

     "skill": "Big Data",
     "status": false,
     "skillGapPercent": 40
    }
   ]
  }
```

- Response Code: 200 OK

Details:

- The predict_job_role function generates a list of job roles with matching probabilities.
- The build_recomendation_skill_for_role function creates a detailed list of skills required for each role, indicating which ones the user already possesses.

Data Models

1. RecommendationRequest
   - **Description**: Input data model for the /recommend endpoint.
   - Fields:
     - skills (list of strings): A list of the user's skills.
     - **Validation**: Must contain at least one skill.
2. RecommendationResponse
   - **Description**: Output data model for the /recommend endpoint.
   - Fields:
     - recommendations (list of RecommendationRole objects): A list of recommended job roles.
3. RecommendationRole
   - **Description**: Represents a single recommended job role.
   - Fields:
     - role (string): The name of the job role.
     - matchPercent (float): The percentage match for the role.
     - skills (list of RecommendationSkill objects): A list of skills related to the role.
4. RecommendationSkill
   - **Description**: Represents a single skill required for a job role.
   - Fields:
     - skill (string): The name of the skill.
     - status (bool): Whether the user possesses this skill.
     - skillGapPercent (optional integer): The percentage gap for the skill (currently always null).
5. SkillsResponse
   - **Description**: Output data model for the /available_skills endpoint.
   - Fields:
     - skills (list of strings): A list of available skills.

Error Handling

The API returns appropriate error codes in case of issues:

- 422 Unprocessable Entity: Invalid input data.
- 500 Internal Server Error: Internal server error.

Example error response:

```
{
  "detail": "Invalid input data."
}
```

Technologies

- Framework: FastAPI
- **Middleware**: CORS (Cross-Origin Resource Sharing)
- **Data Source**: data/metadata.json file
- Prediction Model: predict_job_role function from the ml.predictor module.

Frontend

Overview

The Career Navigator Frontend is a SPA display selectable list of skills, give guidance to the user on how many skills he should select and show listed recommendations along with match percent and list of skills required for recommended position:

Linting

Run the linter to ensure code quality:

```
npm run lint
```

Building the Frontend

To build the production-ready frontend:

```
npm run build
```

Deployment

The application can be deployed by running the backend and frontend services as described above. Ensure all dependencies are installed and environment variables are configured correctly.

Troubleshooting

- **Missing Dependencies**: Ensure all dependencies are installed using uv sync and npm install.
- **Data Files Not Found**: Run dvc pull to fetch the required data files.
- **Frontend Issues**: Check the browser console for errors and ensure the development server is running.