My take-home assignment involved taking a list of patient data as input and outputting a list of clinical trials that each patient is eligible for.

1. Input Data

Patient Data

The patient dataset is sourced from the **100 Sample Synthetic Patient Records (CSV format)**, available in the `"synthea_sample_data_csv_latest"` folder. The following files were utilized:

- **`patients.csv`** – Contains demographic details such as:
  - **ID**: Unique identifier for each patient.
  - **BirthDate**: Used to calculate patient age.
  - **Gender**: Used for gender-based trial eligibility filtering.
- **`conditions.csv`** – Contains:
  - **Patient ID**: Links conditions to individual patients.
  - **Condition Description**: Lists diagnosed medical conditions.

Clinical Trial Data

The eligibility criteria for trials were extracted from XML files obtained from `ClinicalTrials.2021-04-27.part1.zip`. The focus was on a subset of trials (`NCT00000102.xml` to `NCT00000200.xml`), each containing:

- **Minimum and maximum age requirements**
- **Gender eligibility**
- **Inclusion criteria** (required conditions or characteristics)
- **Exclusion criteria** (conditions that disqualify a patient)

---

2. Data Preprocessing

Patient Data Processing

1. Extracted relevant columns (**ID, BirthDate, Gender**).
2. Converted **BirthDate** to **Age** using Python's `datetime` module.

3. Ensured **ID consistency** by removing unnecessary fields like Passport and SSN.
4. Standardized **Gender formatting** to align with clinical trial data.

Condition Data Processing

1. Extracted **Patient ID** and **Condition Description**.
2. Ensured correct **Patient ID formatting** for merging.
3. **Aggregated conditions per patient** to create a consolidated medical history.
4. Replaced missing values in **Condition Description** with empty lists.

Merging Patient and Condition Data

- Merged `patients.csv` and `conditions.csv` on **Patient ID**.
- Stored the merged dataset in `merged_patient_data.csv` for further processing.

Clinical Trial Data Processing

1. Extracted **eligibility text** from XML files.
2. Parsed **minimum and maximum age** fields.
3. Standardized **gender requirements** (e.g., converting `"Both"` to `"All"`).
4. Identified **inclusion and exclusion criteria** from unstructured text.

---

3. Thought Process

Step 1: Data Collection & Standardization

- **Challenges:** Patient and trial data originate from different formats (CSV & XML) and have inconsistencies such as missing values, varying date formats, and unstructured eligibility criteria.
- **Solution:** Standardize key attributes (**Age, Gender, Conditions**) to enable direct comparison.

Step 2: Text Preprocessing & Cleaning

- Raw text often contains **noise** (punctuation, inconsistent capitalization, and redundant words).
- Cleaning includes:
    - Lowercasing text
    - Removing punctuation
    - Preserving **key medical terms** (ensuring clinical relevance)

Step 3: Feature Engineering for Matching

- **Challenge:** Representing patient conditions and trial eligibility criteria in a comparable format.
- **Approaches:**
    - **String-Based Matching** – Basic keyword presence check.
    - **Word2Vec Embeddings** – Converts words into vectors to measure semantic similarity.
    - **BERT Embeddings** – Uses deep learning to capture contextual meaning in text.

Step 4: Matching Strategy

- **Cosine Similarity** was chosen as the metric for comparing patient embeddings with trial embeddings.
- **Threshold tuning:**
    - **Word2Vec-based similarity threshold** set to **0.4**.
    - **BERT-based similarity threshold** reduced from **0.6 to 0.4** to improve recall.
- The system **iterates over patients and trials**, computing similarity scores and recording best matches.

Step 5: Storing & Interpreting Results

- Results were stored in:
    - **CSV format** (`word2vec_matched_patients.xlsx`, `bert_matched_patients.xlsx`)
    - **JSON format** (`word2vec_matched_patients.json`, `bert_matched_patients.json`)
- **Enhanced readability**:
    - Each match includes the **trial ID, name, and conditions met**.

---

4. Debugging Insights & Improvements

After initial runs produced **zero matches**, multiple refinements were made:

Issue 1: Eligibility Filtering Too Strict

- **Problem:** Patients had to match every inclusion criterion **exactly**.
- **Fix:** Required **partial matching** of at least **3 inclusion criteria**.

Issue 2: Over-Cleaning of Medical Terms

- **Problem:** Removing stopwords unintentionally removed important medical terms.
- **Fix:** Used **raw patient conditions** for embeddings instead of cleaned text.

Issue 3: Weak Word2Vec Model

- **Problem:** Training on a small dataset resulted in poor vector representations.
- **Fix:** Used a **pretrained biomedical Word2Vec model** for better embeddings.

Issue 4: BERT Model Not Trained on Medical Data

- **Problem:** `all-MiniLM-L6-v2` was a general-purpose model.
- **Fix:** Switched to **`nlpaueb/biobert-base`**, a biomedical NLP model.

Issue 5: Similarity Threshold Too High

- **Problem:** The **0.6 similarity threshold** was rejecting valid matches.
- **Fix:** Reduced threshold to **0.4 for Word2Vec & BERT**.

---

5. Final Results

After refining the system:

- **Word2Vec-based matching found: 6,135 patient-trial pairs**.
- **BERT-based matching found: 4 patient-trial pairs**.

These results confirm that **advanced embeddings significantly improve recall** but require careful tuning.

---

6. Conclusion

This system demonstrates a **scalable approach** to patient-trial matching:

1. **Rule-based methods** (string matching) provide a **baseline**.
2. **NLP-based embeddings** (Word2Vec, BERT) improve **semantic matching**.
3. **Threshold tuning** optimizes precision-recall trade-offs.

Future improvements could include:

- Using **ClinicalBERT** for **contextualized medical embeddings**.
- Integrating **Graph Neural Networks (GNNs)** for relationship-based matching.
- Automating **active learning** to improve model adaptation.