tresata

# VIT HACK

## Background

In modern data ecosystems, **auto-tagging** of data columns is a critical step in making raw datasets usable. Organizations often receive files from multiple sources such as customer databases, vendor spreadsheets, and public datasets. These files may contain important values like phone numbers, country names, and company names.

A recurring challenge is that column names are **missing, inconsistent, or misleading,** making it hard for automated systems to understand the true **semantic type** of data in each column.

Once the semantic types of the dataset have been accurately identified, subsequent data processing steps can be applied to enhance data integrity, consistency, and overall quality.

## PROBLEM STATEMENT

## Part A: Semantic Classification

Design a model/pipeline that can **automatically classify the semantic type of a column** based solely on its values, without relying on column headers.
The classification categories are:

1. **Phone Number**
2. **Company Name**
3. **Country**
4. **Date**
5. **Other**

**Example:**

Phone.csv has ph_nb column

| ph_nb |
| --- |
| +1 475-216-2114 |
| (080) 1234 5678 |
| +91 9876543210 |
| VIT VELLORE |

Here the ph_nb number can be semantically classified into phone number

## Part B: Parsing & Normalization

Once a column is classified as **Phone Number** or **Company Name**, the system should further parse the following columns:

1. Phone number: Parse into two fields.
    a. **Country**
    **b. Number**

    **Example:** phone number: +912345678901 can be parsed into the following fields:

    Country: India
    Number: 2345678901

    Phonenum.csv has PhoneNumber column , then output.csv should be as following

| PhoneNumber | Country | Number |
| --- | --- | --- |
| +91 6796233790 | India | 6796233790 |
| +1 2312953582 | US | 2312953582 |
| +44 2028323322 | UK | 2028323322 |
| 4853859590 | | 4853859590 |

2. Company name: Parse into following fields
    a. **Name**
    b. **Legal**

    **Example:** Company name: Tresata pvt ltd. This can be parsed into the following fields:
    Name: tresata
    Legal: pvt ltd

| CompanyName | Name | Legal |
|---|---|---|
| Enno Roggemann GmbH & Co. KG | enno roggemann | gmbh co kg |
| First National Bank | First National Bank | |
| Debrunner Acifer AG | debrunner acifer | ag |

If a file test.csv has both PhoneNumber and CompanyName then running this should result a output.csv with columns: PhoneNumber, Country,Number, CompanyName,Name, Legal

```
None
python3 parser.py --input /path/to/file/test.csv
```

If multiple columns are classified as PhoneNumber or CompanyName, the column with the highest classification probability should be selected for parsing. If you have equal probabilities for a classification, you can parse any one.

## Part C: AGENTIC LAYER

Parts A (semantic classification) and B (parsing) can already handle their tasks independently. The challenge now is to design an agent layer (Part C) that:

- Autonomously orchestrates Parts A and B into a single pipeline
- Runs with Minimal/No human supervision
- Stays robust to noise and inconsistent inputs
- Self-corrects when errors occur
- Ensures smooth, end-to-end processing across diverse datasets

In short: Build an agentic system that turns Parts A + B into a fully autonomous, noise-resilient pipeline.

### Training Dataset

You will be provided with:

1. **4 raw files** containing different types of data (covering the semantic categories).
    a. Company.csv: This file contains data related to various companies.
    b. Countries.txt: This contains a list of all known country names
    c. Dates.csv: This file contains a list of dates
    d. phoneNumber.csv: This file contains a list of phone number
2. A **legal.txt** file containing common company legal suffixes.

These files will serve as the **training dataset** for your solution.

## Deliverables

Please create a zip file containing the following requirements.

1. **Two executable file** that:
    a. Takes a file path and a column as input and returns the semantic classification of that column. Name this files as predict.py

For example:

```
Python
python3 predict.py --input /path/to/file  --column columnName

companyName
```

    b. This program takes a specified file path as input. If  a column is classified as PhoneNumber or CompanyName via your architecture, then you should parse the data as mentioned in PART B of the problem statement. The processed output is subsequently written to an output.csv file. Please make sure that you name the file as **parser.py**

For example:

```
None
python3 parser.py --input /path/to/file
```

Please feel free to add all the supporting files that may be needed to run these executables files in the zip submission.

2. A **2-slide deck**:

    a. **System Architecture** (your approach, workflow, and model design).
    b. **Team Roles** (who did what).