link thư mục tại [AnNT/research](https://drive.google.com/drive/folders/1Tm1KuWKCiTgh9kWwUNXJbiF09Wkrsqb0?usp=sharing)

# # Tóm tắt các bước chạy

## 1. Xử lý dữ liệu cầu giấy

  - cd 1.\ data\ csv

  - python process\_caugiay\_data.py

  - python check\_caugiay\_data.py

## 2. Token dữ liệu từng loại

  - cd 2.\ token\ each\ data

  - python each\_db\_of\_tab.py

## 3. Tạo dữ liệu huấn luyện

  - cd 3.\ train\ data

  - python create\_data\_train\_1.py

  - python create\_data\_train\_2.py

  - python create\_data\_train\_3.py

  - python concate\_sentence.py

  - python create\_train\_test\_data.py

  - Train lại bộ tách từ với các câu tìm kiếm trong dữ liệu huấn luyện

      - javac -encoding UTF-8 DataPreprocessor.java

      - java DataPreprocessor train/Train\_gold.txt

      - cd train

      - python RDRsegmenter.py train Train\_gold.txt.BI Train\_gold.txt.RAW.Init

  - python create\_token\_test\_data.py để tách câu tìm kiếm của dữ liệu test

## 4. Lấy char\_encode, word\_embedd, char\_embedd của dữ liệu train, dữ liệu val, dữ liệu test

  - python get\_char\_to\_encode.py

  - python get\_tag\_embedd.py

  - chạy get\_word\_embedd.ipynb trên colab

## 5. Huấn luyện mô hình

  - chạy model.ipynb trên colab

## 6. Đánh giá mô hình

  - chạy evaluate\_model\_with\_validate\_data.ipynb trên colab

  - chạy evaluate\_model\_with\_test\_data.ipynb trên colab

## 7. Sử dụng Model thông qua API Flask, Ngrok

  - chạy main\_have\_char\_embedd.ipynb trên colab

<div *style*='page-break-after:always;'></div>

# # Giải thích chi tiết các bước chạy

**---**

# **Cài đặt môi trường**

# **I. Xử lý dữ liệu Cầu Giấy**

# **II. Token dữ liệu**

# **III. Tạo dữ liệu huấn luyện**

# **IV. Lấy char\_encode, word\_embedd, char\_embedd của dữ liệu**

# **V. Huấn luyện mô hình**

**# VI. Đánh giá mô hình**

**# VII. Sử dụng Model thông qua API**

**---**

<div *style*='page-break-after:always;'></div>

## ## Cài đặt môi trường

### **1. Môi trường sử dụng**

    tensorflow 1.15

    keras 2.2.5

    keras-contrib-2.0.8

    python 3.7

    scikit-learn 0.22.2.post1

    matplotlib 3.2.2

    numpy 1.19.5

    flask ngrok 0.0.25

    fasttext 0.9.2

    vncorenlp 1.0.3

### **2. Các file cần cài đặt**

  [cc.vi.300.bin](https://dl.fbaipublicfiles.com/fasttext/vectors-crawl/cc.vi.300.bin.gz) là mô hình CBOW Word vectors được xây dựng sẵn của facebook

  [Công cụ tách từ tiếng việt VNcoreNLP](https://github.com/vncorenlp/VnCoreNLP), lấy thư mục models

  [char\_vocab\_VISCII](https://drive.google.com/file/d/17RPRvk6A0i9TPiqdzsaFY0Z-k260e\_Kw/view?usp=sharing)

  <div *style*='page-break-after:always;'></div>

## ## I. Xử lý dữ liệu Cầu Giấy

### **Thực hiện:**

  cd 1. data csv

  python process\_caugiay\_data.py

### **Kết quả:**

    cho ra 2 file gồm:

      ner.txt lưu tên của các đối tượng được đặt tên

      caugiay\_processed.json là dữ liệu cầu giấy đã tiền xử lý

  ![alt text](./1.%20data%20csv/kết%20quả%20cmd.png)

### **Giải thích:**

    chương trình này sẽ xử lý dữ liệu dữ liệu cầu giấy csv

```mermaid

graph LR

  a(lowercase và bỏ dấu ,)-->b

  b(xử lý name.default)-->c

  c(xử lý tên đường)-->d

  d(xử lý số nhà)-->e(loại bỏ những bản ghi trùng lặp)

```

  > *Bước 1: xử lý name.default: chỉ lấy những tên không phải là ghép của số nhà và tên đường*

  > *Bước 2: xử lý tên đường*

     - nếu tên đường là số => gán là trống

     - nếu có chứa số nhà => bỏ đoạn text là số nhà

     - nếu tên đường có chứa ["ward", "xã", "phường", "thị trấn", "quận", "huyện","district","thị xã","thành phố","province","hà nội","việt nam"] thì bỏ từ chỗ bắt đầu đoạn text đến hết

  > *Bước 3: xử lý số nhà*

    - nếu số nhà không chứa số => gán là trống

    - nếu có dạng "số9", thì sẽ chuyển thành "số 9"

    - nếu có dạng "103-A12", thì sẽ chuyển thành "103 - A12"

    - nếu có chứa ["phố ","đường","duong","ngách","hẻm","ngõ","lô","tổ"], thì sẽ bỏ từ chỗ bắt đầu đoạn text đến hết

  > *Bước 4: loại bỏ những bản ghi trùng lặp*

    Các trường số nhà, đường, xã, huyện, tỉnh, quốc gia của một bản ghi được tách bởi VNcoreNLP, ta được một mảng các từ

    - những từ là số=> <number>

    - từ có dạng 12/13=> <hẻm>

    - từ có dạng 12/13/14=> <ngách>

    - từ có dạng 10000=> <postcode>

    sau đó nối các từ trong mảng thành một câu, dùng các câu này để lọc bản ghi trùng lặp

<div *style*='page-break-after:always;'></div>

## ## II. Token dữ liệu

### **Thực hiện:**

      cd 2.\ token\ each\ data

      python each\_db\_of\_tab.py

### **Kết quả:**

      4 file

        obj.txt là obj+feature đã token và đánh tag

        pre.txt là pre đã token và đánh tag

        location.txt là các địa chỉ dữ liệu trong file csv địa chỉ đã đánh tag

        location\_ner.txt là location\_ner và location\_special đã token và đánh tag

<div *style*='page-break-after:always;'></div>

## ## III. Tạo dữ liệu huấn luyện

### **Thực hiện:**

      cd 4.\ train\ data

      python create\_data\_train\_1.py

      python create\_data\_train\_2.py

      python create\_data\_train\_3.py

      python concate\_sentence.py

      python create\_train\_test\_data.py

      Train lại bộ tách từ với các câu tìm kiếm trong dữ liệu huấn luyện

        - javac -encoding UTF-8 DataPreprocessor.java

        - java DataPreprocessor train/Train\_gold.txt

        - cd train

        - python RDRsegmenter.py train Train\_gold.txt.BI Train\_gold.txt.RAW.Init

      python token\_test\_data.py để tách câu tìm kiếm của dữ liệu test

### **Kết quả:**

    thư mục chứa dữ liệu train, validate, test (tỉ lệ 7:1:2), mỗi thư mục có các file:

      data.txt (dữ liệu huấn luyện gán nhãn đầy đủ)

      data\_no\_tag.txt (dữ liệu huấn luyện không có nhãn)

      cautruyvan.txt (câu truy vấn của dữ liệu huấn luyện)

      cautruyvan\_token.txt (câu truy vấn của dữ liệu huấn luyện đã qua bộ tách từ)

<div *style*='page-break-after:always;'></div>

## ## IV.Lấy char\_encode, word\_embedd, char\_embedd của dữ liệu train, dữ liệu val, dữ liệu test

### **Thực hiện:**

      python get\_tag\_embedd.py

      python char\_to\_encode.py

      chạy get\_word\_embedd.ipynb trên colab

### **Kết quả**

    word embedd, char encode, tag embedd của tập dữ liệu train, dữ liệu validate

    word embedd, tag embedd của dữ liệu test

### **Chú ý:**

\* Train bộ tách từ RDR với các câu train để tách các câu tìm kiếm trong bộ test

\* train/cautruyvan\_token.txt là dữ liệu train cho mô hình tách từ RDR, được copy vào file Train\_gold.txt

\* Chú ý: trong file Train\_gold.txt không chứa "\_ " hoặc " \_" hoặc " \_ " vì \_ là ký tự nối của 1 từ

\* [model tách từ RDR đã train lại theo các câu huấn luyện](https://drive.google.com/file/d/1hiYg2Elg-PbXl81KM4IqZuLJF-x8nR-H/view?usp=sharing), đặt vào trong thư mục models/wordsegmenter của VncoreNLP, và sử dụng với giao diện của VncoreNLP

<div *style*='page-break-after:always;'></div>

## ## V. Huấn luyện mô hình

### **Thực hiện**

    Chạy model.ipynb trên colab

### **Kết quả**

    model đã train xong

<div *style*='page-break-after:always;'></div>

## ## VI. Đánh giá mô hình

### **Thực hiện**

  - chạy evaluate\_model\_with\_validate\_data.ipynb trên colab

  - chạy evaluate\_model\_with\_test\_data.ipynb trên colab

### **Kết quả**

    Quan sát được độ acc, confused matrix, độ đo P R F1 của các nhãn với dữ liệu validate và dữ liệu test

<div *style*='page-break-after:always;'></div>

## ## VII. Sử dụng Model thông qua API

### **Thực hiện**

    Chạy main\_have\_char\_embedd.ipynb trên colab

### **Kết quả**

    API sử dụng models

  ![alt text](./7.%20API/review%20sử%20dụng%20API.png)