TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

———————— * ————————

ĐỒ ÁN
TỐT NGHIỆP ĐẠI HỌC

NGÀNH CÔNG NGHỆ THÔNG TIN

CHUẨN HÓA ĐỊA CHỈ TIẾNG VIỆT

Sinh viên thực hiện : **Đặng Đức Tùng**
Lớp CNTT-TT 2.02 - K59
Giáo viên hướng dẫn: **TS. Trần Việt Trung**

HÀ NỘI 5-2019

PHIẾU GIAO NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP

1. Thông tin về sinh viên
   Họ và tên sinh viên: Đặng Đức Tùng.
   Điện thoại liên lạc: 0989293703 Email: dangductungcfc@gmail.com
   Lớp: CNTT-TT 2.02 - K59 Hệ đào tạo: Đại học chính quy.
   Đồ án tốt nghiệp được thực hiện tại: Trường đại học Bách Khoa Hà Nội.
   Thời gian làm ĐATN: Từ ngày 11/02/2019 đến 20/05/2019.

2. Mục đích nội dung của ĐATN
   Nghiên cứu ứng dụng, cài đặt thử nghiệm và đánh giá hiệu quả các mô hình học máy, phương pháp Sagel vào bài toán *chuẩn hóa địa chỉ tiếng Việt*.

3. Các nhiệm vụ cụ thể của ĐATN

   - Thử nghiệm, đánh giá mô hình CRF/LSTM
   - Thử nghiệm, đánh giá phương pháp Sagel
   - Cải thiện các mô hình, phương pháp với địa chỉ tiếng Việt

4. Lời cam đoan của sinh viên:
   Tôi - *Đặng Đức Tùng* - cam kết ĐATN là công trình nghiên cứu của bản thân tôi dưới sự hướng dẫn của *TS. Trần Việt Trung*. Các kết quả nêu trong ĐATN là trung thực, không phải là sao chép toàn văn của bất kỳ công trình nào khác.

   Hà Nội, ngày tháng năm 2019
   Tác giả ĐATN

   Đặng Đức Tùng

5. Xác nhận của giáo viên hướng dẫn về mức độ hoàn thành của ĐATN và cho phép bảo vệ:
   .....
   .....
   .....
   .....

   Hà Nội, ngày tháng năm 2019
   Giáo viên hướng dẫn

   TS. Trần Việt Trung

LỜI CẢM ƠN

Trước khi trình bày nội dung chính của đồ án, em xin gửi tới thầy, cô của trường Đại học Bách khoa Hà Nội lời chào trân trọng, lời chúc sức khỏe và lời cảm ơn sâu sắc. Với sự quan tâm, dạy dỗ, chỉ bảo tận tình chu đáo của thầy cô, em có thể tiếp thu những kiến thức quan trọng, để em có thể hoàn thành đồ án tốt nghiệp và những công việc của em sau này.

Em xin gửi lời cảm ơn chân thành nhất tới giảng viên – TS. Trần Việt Trung đã hướng dẫn và giúp đỡ em rất nhiều không chỉ trong quá trình thực hiện đồ án mà còn ở một số môn học trong khoảng thời gian ngồi trên giảng đường. Cảm ơn thầy đã giúp đỡ em.

Nhân dịp này em cũng xin được gửi lời cảm ơn chân thành tới gia đình, bạn bè đã luôn bên em, cổ vũ, động viên, giúp đỡ em trong suốt quá trình học tập và thực hiện đồ án tốt nghiệp. Em xin gửi lời cảm ơn đến anh Nguyễn Đức Quang, trong khoảng thời gian anh làm leader của team Platform - thuộc phòng Big data, công ty cổ phần VCCorp, anh đã giúp đỡ em rất nhiều, luôn trả lời những câu hỏi của em, giúp em có thêm nhiều kinh nghiệm thực tế trong thời gian em làm việc tại công ty.

Hà Nội, ngày 08 tháng 04 năm 2019

Sinh viên

Đặng Đức Tùng

Mục lục

LỜI MỞ ĐẦU

Bài toán chuẩn hóa địa chỉ, hay còn được biết đến phổ biến trên thế giới với cái tên Address Parser hoặc Address Normalization, được các các công ty lớn, các nhà phát triển trên thế giới quan tâm và nghiên cứu. Điển hình như Google, trong quá trình phát triển Google Maps, họ cũng đã giải quyết bài toán này, tuy nhiên phương pháp thì không được công bố.

Bài toán chuẩn hóa địa chỉ dựa vào một chuỗi văn bản, có thể chứa địa chỉ, từ đó bóc tách ra được các đơn vị nhỏ hơn cấu thành địa chỉ như tòa nhà, đường, thành phố ... hoặc có thể một số đơn vị khác khi với địa chỉ của nước ngoài.

Bài toán chuẩn hóa địa chỉ có rất nhiều cách tiếp cận, đơn giản nhất có thể sử dụng luật, kết hợp vị trí tiền tố trong chuỗi văn bản từ đó xác định ra các nhãn từ cần tìm. Gần đây, với sự bùng nổ của học máy, học sâu, bài toán chuẩn hóa địa chỉ có thể được coi là một bài toán gán nhãn chuỗi (sequence labeling) mà qua đó, việc xác định các đơn vị có thể dựa trên một thuật toán học máy, học sâu và một tập dữ liệu huấn luyện.

Vài năm gần đây, các nhà phát triển ở Ấn Độ đã công bố một bài báo về bài toán chuẩn hóa địa chỉ, trong đó họ chỉ ra các nhược điểm của phương pháp sử dụng trí tuệ nhân tạo và đề xuất một phương pháp có tên gọi là SAGEL, một phương pháp theo kết quả đánh giá từ bài báo thì khá ấn tượng với tập địa chỉ của Ấn Độ.

Trong phạm vi của đồ án, tôi đã cố gắng để thử nghiệm một vài phương pháp đã được đề cập như sử dụng học máy và SAGEL với tập địa chỉ tiếng Việt vốn khác rất nhiều so với địa chỉ Ấn Độ về đơn vị hành chính. Ngoài ra, tôi có thử nghiệm một phương pháp kết hợp học máy và SAGEL để có thể khắc phục các nhược điểm của các phương pháp trên với tiếng Việt.

Chương 1

Giới thiệu đề tài

1.1. Đặt vấn đề

Với sự phát triển gần đây của thương mại điện tử, bán hàng trực tuyến tại Việt Nam, việc thực hiện các đơn hàng, giao hàng nhanh đang trở thành một yếu tố quan trọng mà khách hàng dùng để đánh giá doanh nghiệp hay cửa hàng bán lẻ, shop trực tuyến. Tuy nhiên, để đảm bảo được điều này, hồ sơ đơn hàng cần được xử lý một cách nhanh chóng để có thể cung cấp cho bên vận chuyển. Một trong những thách thức lớn nhất chính là việc xác định chính xác địa chỉ từ dữ liệu nhập của người dùng. Đặc biệt trên môi trường mạng xã hội, việc cung cấp địa chỉ không đồng nhất (thường là các comment), nơi mà người dùng thường nhập trực tiếp cả địa chỉ trong một văn bản đầu vào, thay vì có một chuẩn chung, do vậy mỗi cá nhân có sự độc đáo riêng trong việc cung cấp địa chỉ của họ, và hơn nữa, có thể có sự sai sót trong quá trình nhập của người dùng khiến việc xử lý địa chỉ một cách khó khăn hơn. Việc trích xuất địa chỉ một cách tự động có thể rút gọn thời gian xử lý đơn hàng đi đáng kể và tăng tốc độ giao chuyển sản phẩm tới khách hàng.

Ngoài ra, việc chuẩn hóa được địa chỉ còn giúp ích được trong nhiều lĩnh vực khác trong nền công nghệ trực tuyến như bất động sản, xe ôm, taxi công nghệ... Trong trường hợp không thể dùng GPS để định vị hoặc, định vị GPS không chính xác. Do đó việc chuẩn hóa được địa chỉ tiếng Việt là một việc làm thiết thực và có nhiều ứng dụng trong thời kỳ công nghệ trực tuyến phát triển mạnh.

1.2. Mục tiêu và phạm vi của đề tài

1.2.1. Giới thiệu bài toán

- Cho một văn bản tiếng Việt, bóc tách địa chỉ và chuẩn hóa để thu được các thông tin từ địa chỉ ấy sao cho các thông tin chi tiết đến mức tối đa và đúng nhất với địa chỉ trong văn bản đầu vào.

- Thông tin thu được phải là một địa chỉ có thể xác định vị trí, tồn tại trên bản đồ.

1.2.2. Phạm vi

- Trong phạm vi của bài toán, chỉ có thể chuẩn hóa được tiếng Việt và tối đa 1 địa chỉ cho mỗi văn bản đầu vào.
- Do giới hạn về dữ liệu địa chỉ tiếng Việt, nên bài toán chỉ có thể đữ ra dữ liệu đúng đến mức đường với quận thuộcthành phố trực thuộc trung ương, ở mức xã với tỉnh trực thuộc trung ương, hoặc xã với huyện thuộc thành phố trực thuộc trung ương.

1.3. Hướng giải quyết

Trong đồ án này, tôi có cài đặt thử nghiệm các phương pháp sử dụng trí tuệ nhân tạo (học máy, học sâu) và SAGEL dựa theo bài báo của các nhà phát triển Ấn Độ nhưng sử dụng trên tập dữ liệu tiếng Việt.

- Với phương pháp trí tuệ nhân tạo, tôi coi bài toán là bài toán gán nhãn chuỗi trong lĩnh vực xử lý ngôn ngữ tự nhiên (NLP). Tôi xây dựng một tập dữ liệu gán nhãn các đơn vị trong địa chỉ tiếng Việt, sau đó tôi sử dụng CRF/ mạng LSTM-CRF để học tham số đồng thời, sau đó so sánh độ chính xác, thời gian huấn luyện, thời gian dự đoán... của từng phương pháp này.
- Với phương pháp SAGEL, tôi cài đặt lại thuật toán từ ý tưởng trong bài báo *SAGEL: Smart Address Geocoding Engine for Supply-Chain Logistics*, sau đó đánh giá độ chính xác dựa trên dữ liệu từ Google với một tập thử nghiệm.
- Ngoài ra trong quá trình cài đặt giải thuật của phương pháp SAGEL, tôi có nhận ra một số nhược điểm của phương pháp này cho tiếng Việt, nên tôi đề xuất một phương pháp kết hợp sử dụng học máy và SAGEL, tôi gọi phương pháp này là SAGEL with Top K Results.

Chi tiết các phương pháp sẽ được trình bày ở các chương sau.

1.4. Các nghiên cứu trên thế giới

Một vài nghiên cứu từ các công ty lớn, hoặc các phần mềm mã nguồn mở:

1.4.1. Google API

Đây là một trong những bộ chuẩn hóa địa chỉ chính xác nhất trên thế giới hiện nay. Với một đội ngũ nhà phát triển hùng hậu, tiềm năng tài chính, cũng như lượng dữ liệu khổng lồ, việc độ chính xác các thuật toán của Google là không phải bàn cãi, nhưng gần đây, Google bắt đầu tính phí cho các dịch vụ API của mình, từ đó khiến việc thử nghiệm của tôi với bài toán cũng trở nên khá khó khăn.

Ngoài ra Google không công bố họ sử dụng công nghệ, thuật toán hoặc phương pháp nào trong bộ chuẩn hóa địa chỉ của mình, từ đó, khiến các công ty muốn sử dụng bộ chuẩn hóa hoặc phải sử dụng của Google API có tính phí, hoặc xây dựng một bộ chuẩn hóa của riêng mình.

Dưới đây là một ví dụ về bộ chuẩn hóa địa chỉ của Google API (ví dụ được lấy trên trang chủ cuả Google Maps Platform):

Ví dụ với chuỗi địa chỉ in đậm dưới đây :

1600 Amphitheatre Parkway, Mountain View, CA 94043, USA

```
"address_components" : [
        {
          "long_name" : "1600",
          "short_name" : "1600",
          "types" : ["street_number"]
        },
        {
          "long_name" : "Amphitheatre Pkwy",
          "short_name" : "Amphitheatre Pkwy",
          "types" : ["route"]
        },
        {
          "long_name" : "Mountain View",
          "short_name" : "Mountain View",
          "types" : ["locality", "political"]
        },
        {
          "long_name" : "Santa Clara County",
          "short_name" : "Santa Clara County",
          "types" : ["administrative_area_level_2", "political"]
        },
        {
          "long_name" : "California",
          "short_name" : "CA",
          "types" : ["administrative_area_level_1", "political"]
        },
        {
          "long_name" : "United States",
          "short_name" : "US",
          "types" : ["country", "political"]
        },
        {
          "long_name" : "94043",
          "short_name" : "94043",
          "types" : ["postal_code"]
        }
    ]
```


Địa chỉ được chuẩn hóa một cách rất chi tiết, có thể cho thấy Google đã rất quan tâm phát triển hệ thống chuẩn hóa địa chỉ của mình.

1.4.2. Thư viện mã nguồn mở libpostal

Đây là một thư viện mã nguồn mở của tác giả có tài khoản github là **openvenues**, tác giả này có mô tả rằng, họ sử dụng machine learning hay chính xác hơn là CRF (trường điều kiện ngẫu nhiên) trên tập dữ liệu huấn luyện hơn 1 tỉ địa chỉ trên thế giới, và sử dụng dữ liệu địa chỉ từ **OpenStreetMap** và **OpenAddresses** là nguồn dữ liệu địa chỉ có cấu trúc.

Trên trang github của thư viện này : *https://github.com/openvenues/libpostal* có đề cập rằng thư viện có thể chuẩn hóa được địa chỉ của Tiếng Việt. Tuy nhiên độ chính xác không được cao:

Ví dụ với chuỗi địa chỉ : **Đường Đại Cồ Việt, quận Hai Bà Trưng, Hà Nội**

Khi sử dụng hàm **parse_address** trong package **postal.parser** sử dụng ngôn ngữ Python trả ra kết quả :

- **road**: 'đường đại cồ việt quận hai bà trưng'
- **city**: 'hà nội'

Nhưng với địa chỉ là tiếng Anh : **The Book Club 100-106 Leonard St Shoreditch London EC2A 4RH, United Kingdom**

- **house**: 'the book club'
- **house_number**: '100-106'
- **road**: 'leonard st'
- **suburb**: 'shoreditch'
- **city**: 'london'
- **postcode**: 'EC2A 4RH'
- **country**: 'united kingdom'

Vì địa chỉ tiếng Anh được sử dụng ở rất nhiều nước trên thế giới, nên có thể nhận thấy dữ liệu địa chỉ cho tiếng Anh cũng rất phổ biến, do đó thư viện libpostal cho kết quả khá chính xác.

1.5. Bố cục đồ án

Đồ án được trình bày theo bố cục sau:

- **Chương 1 - Mở đầu**: giới thiệu đề tài, đặt vấn đề, mục tiêu phạm vi đề tài, định hướng giải pháp và có đưa ra một số sản phẩm hoặc nghiên cứu trên thế giới.

- **Chương 2 - Dữ liệu địa chỉ Việt Nam**: đặc điểm phân cấp hành chính, đặc điểm địa chỉ Việt Nam, các dữ liệu địa chỉ sử dụng trong đồ án.
- **Chương 3 - Cơ sở lý thuyết**: cơ sở lý thuyết về các phương pháp, công nghệ sử dụng trong đồ án.
- **Chương 4 - Phương pháp giải quyết**: trình bày về việc áp dụng các phương pháp với bài toán chuẩn hóa hóa địa chỉ.
- **Chương 5 - Đánh giá và cài đặt thử nghiệm**: trình bày về dữ liệu thử nghiệm, các kịch bản thử nghiệm, đánh giá độ chính xác của các phương pháp theo các kịch bản này.
- **Chương 6 - Kết luận và hướng phát triển**: Nhận xét tổng quan về đề tài, những khó khăn, thiếu sót trong đề tài cũng như phương hướng phát triển trong tương lai.

Chương 2

Dữ liệu địa chỉ Việt Nam

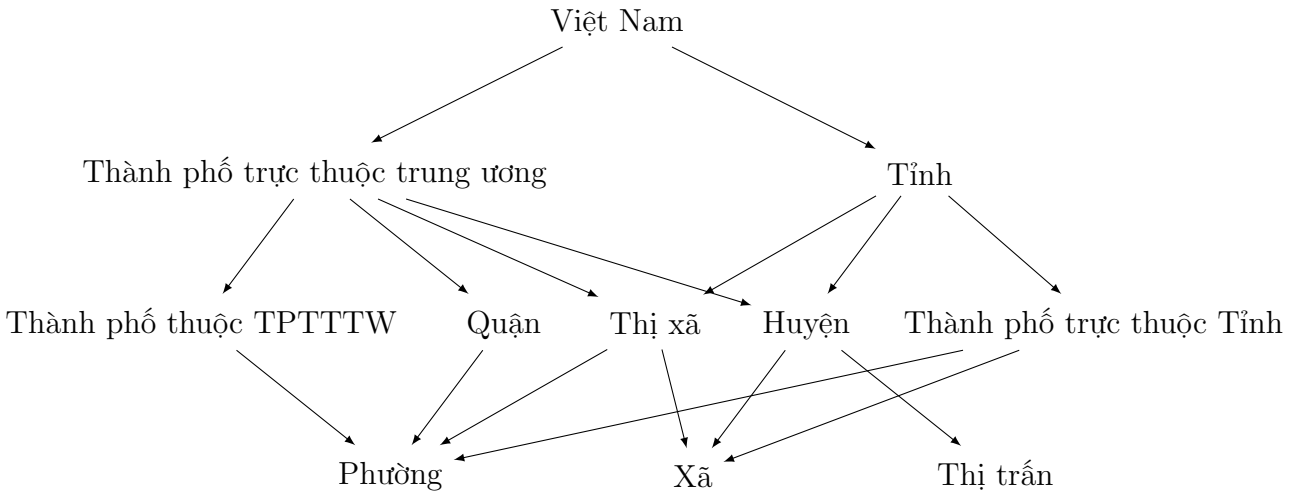
Địa chỉ là tập hợp các thông tin, thường có hình thức biểu diễn cố định, nhằm miêu tả vị trí của một tòa nhà, một căn hộ, hay một cấu trúc hoặc một diện tích đất nào đó. Địa chỉ thường sử dụng đường biên giới chính trị và tên phố để miêu tả, cùng với các thông tin nhận dạng khác như số nhà hoặc số căn hộ. Một số địa chỉ còn chứa một số loại mã để giúp các cơ quan phụ trách vận chuyển và thư từ dễ tìm kiếm như mã ZIP và mã bưu chính. Địa chỉ thông thường dựa trên các phân cấp hành chính của quốc gia.

2.1. Phân cấp hành chính Việt Nam

Phân cấp hành chính Việt Nam là sự phân chia các đơn vị hành chính của Việt Nam thành từng tầng, cấp theo chiều dọc. Theo đó cấp hành chính ở trên (cấp trên) sẽ có quyền quyết định cao hơn, bắt buộc đối với cấp hành chính ở dưới (hay cấp dưới).

Phân cấp hành chính Việt Nam theo Điều 110 Hiến pháp 2013 và Điều 2 Luật Tổ chức chính quyền địa phương gồm 3 cấp hành chính là:

- Cấp Tỉnh, gồm: Tỉnh/ Thành phố trực thuộc trung ương
- Cấp Huyện, gồm: Quận/ Huyện/ Thị xã/ Thành phố thuộc tỉnh/ Thành phố thuộc thành phố trực thuộc trung ương
- Cấp Xã, gồm: Xã/ Phường/ Thị trấn



Hình 1. *Phân cấp hành chính Việt Nam*

Ngoài ra còn có Đơn vị hành chính - kinh tế đặc biệt do Quốc hội thành lập (còn được gọi là đặc khu) là đơn vị hành chính thuộc tỉnh, do Quốc hội quyết định thành lập, có cơ chế, chính sách đặc biệt về phát triển kinh tế - xã hội, có tổ chức chính quyền địa phương và cơ quan khác của Nhà nước tinh gọn, bảo đảm hoạt động hiệu lực, hiệu quả.

Ở dưới các đơn vị hành chính trong **hình 1** là cấp cơ sở không pháp nhân, phục vụ cho quản lý dân cư nhưng không được xem là cấp hành chính, và những người tham gia quản lý hoạt động ở cấp này chỉ hưởng phụ cấp công tác mà không được coi là công chức.

- Dưới xã có làng/ thôn/ bản/ buôn/ sóc/ấp.., khi lượng dân cư đông thì thôn làng dưới xã có thể chia ra các xóm.
- Dưới phường/thị trấn có khu dân cư/khu phố/khu vực/khóm/ấp, khi lượng dân cư đông, khu dân cư ở phường/thị trấn thì chia ra tổ dân phố, dưới tổ dân phố còn chia ra cụm dân cư.

2.2. Địa chỉ Việt Nam

Địa chỉ ở Việt Nam thường theo địa hạt hành chính. Việc sử dụng mã bưu chính hay mã ZIP không thật sự phổ biến. Định dạng của địa chỉ ở Việt Nam cách mới gồm các phần:

Phần số nhà và tên đường

Phần số nhà và tên đường được ghi theo quy định ở Quyết định số 05/2006/QĐ-BXD ngày 08 tháng 03 năm 2006. Định dạng phổ biến nhất như sau:

- Chỉ gồm số nhà và tên đường, ví dụ: 123 đường Lê Lợi. Đây là định dạng cơ bản và phổ biến nhất.

- Số nhà có thể thêm các ký tự ở cuối: 123A đường Lê Lợi, hoặc 123B đường Lê Lợi. Trường hợp này là do khu đất trước kia là số 123, nhưng sau đó mới được tách làm 2 căn nhà có 2 địa chỉ khác nhau.

- Nếu nhà trong hẻm thì có thêm dấu gạch chéo: 123/3 đường Lê Lợi. Trong đó, 123 là địa chỉ của con hẻm, còn 3 là số nhà trong con hẻm đó.

- Trong hẻm cũng có quy tắc đặt tên giống số nhà và có thể có hẻm con, ví dụ: 123/3E đường Lê Lợi, hoặc 123/3/5B đường Lê Lợi.

  Một cách khác là ở các khu gọi là cư xá. Trong cư xá sẽ có đường và số nhà kèm theo, quy tắc đặt tên sẽ là:

- Bao gồm cả số nhà, đường và tên cư xá, ví dụ: 123 đường số 4 cư xá Bình Thới.

Phần tên thôn/ấp

Trường hợp không có số nhà và đường, phần đầu bao gồm tên thôn/ấp, một số nơi còn kèm theo tên đội. Ví dụ: Đội 2 thôn Phú Lợi, hoặc ấp Thái Hòa.

Như vậy, một thôn/ấp sẽ có nhiều nhà có cùng địa chỉ. Nhưng điều này vẫn không phải trở ngại lớn khi gửi thư, vì người dân ở các thôn/ấp này đều đa phần là biết nhau, nên chỉ cần hỏi tên người là xác định được.

Hiện vẫn chưa có cách đặt địa chỉ nào chính xác hơn cho trường hợp này. **Phần phân cấp hành chính Việt Nam**

Phần này được xác định theo phân cấp hành chính Việt Nam gồm 3 cấp : xã, huyện, tỉnh (như đề cập ở mục **2.1**)

Tên nước thường không được bao gồm khi gửi trong nội bộ Việt Nam.

Ngoài ra ở các quận tại Việt Nam, một đường có thể thuộc nhiều phường khác nhau và vì diện tích của một đường (tại các quận) không lớn, kết hợp với số được đánh theo dạng chẵn/lẻ tại hai bên đường, có thể xác định được vị trí của địa điểm dễ dàng hơn phường, vốn khá rộng và không được đánh số. Do tính phân loại không cao, nên phường thông thường có thể không cần đưa vào chuỗi địa chỉ tiếng Việt.

Ví dụ về một địa chỉ tiếng Việt: "Đại học Bách Khoa Hà Nội, số 1 Đại Cồ Việt, Hai Bà Trưng, Hà Nội"

- Đại học Bách Khoa Hà Nội : Địa điểm, tòa nhà, khu đất

- Số 1 : Số của đường

- Đại Cồ Việt : Đường, phố

- Hai Bà Trưng : Quận

- Hà Nội : Thành phố

2.3. Dữ liệu dùng trong bài toán

Trong bài toán này, tôi sử dụng nhiều loại dữ liệu khác nhau, mỗi loại được sử dụng trong một phương pháp. Tôi xin chân thành cảm ơn TS. Trần Việt Trung đã giúp đỡ tôi trong việc tìm kiếm và thu thập bộ dữ liệu phục vụ trong bài toán. Dữ liệu này được chia làm 2 loại.

2.3.1. Dữ liệu huấn luyện cho phương pháp sử dụng trí tuệ nhân tạo

Trong phạm vi của đồ án lần này, tôi sử dụng hai mô hình cho bài toán gán nhãn từ loại trên cùng một tập dữ liệu, đó là :

- CRF : một mô hình phổ biến cho bài toán sequence labeling nói chung và bài toán nhận dạng thực thể có tên (Named Entity Recognition - Nhận dạng thực thể có tên), ưu điểm thời gian huấn luyện ngắn, không cần số lượng lớn dữ liệu huấn luyện.

- Mạng Bi-LSTM-CRF : là một mô hình học sâu (deep learning) nổi bật hiện nay, tôi lựa chọn phương pháp này từ ý tưởng bài viết trên **medium** của *Zalo TechBlog : Nhận diện tên riêng (NER) với Bidirectional Long Short-Term Memory và Conditional Random Field.*

Về dữ liệu huấn luyện, gồm 2 dạng chính:

- Dữ liệu chỉ chứa chuỗi địa chỉ cần được chuẩn hóa, theo cấu trúc địa chỉ Việt Nam, có tiền tố chỉ đơn vị như đường, quận, huyện,... có thể được viết theo một thứ tự chuẩn theo quy định - **dữ liệu địa chỉ có cấu trúc**

  Ví dụ : Đường Đại Cồ Việt, quận Hai Bà Trưng, Hà Nội

- Dữ liệu từ mạng xã hội facebook : được thu thập từ các trang bán, hàng ngoài chứa chuỗi địa chỉ, văn bản còn chứa những từ ngữ cảm thán, danh từ, động từ,... hoặc các trường trong chuỗi địa chỉ không sắp xếp theo thứ tự thông thường, không có ngăn cách giữa các đơn vị nhưng vẫn đảm bảo rằng người đọc có thể hiểu được - **dữ liệu địa chỉ không có cấu trúc**

  Ví dụ : Ship cho mjnh đến 2 Đại Cồ Việt nha !

Trong phạm vi của đồ án lần này, tôi sẽ chỉ ra ưu nhược điểm của các phương pháp với từng loại dữ liệu (có cấu trúc và không có cấu trúc)

2.3.2. Dữ liệu cho phương pháp Sagel

Trong chương này, tôi chỉ tập trung vào giới thiệu dữ liệu tôi sử dụng cho bài toán mà không tập trung vào việc tôi sử dụng dữ liệu đó như thế nào. Việc sử dụng dữ liệu đó sẽ được tôi đề cập trong các chương sau.

Để có thể giải quyết bài toán Address Parser, Sagel yêu cầu có 1 bộ dữ liệu địa chỉ chuẩn và thuật toán của họ tập trung vào việc chuẩn hóa theo bộ dữ liệu chuẩn này.

Nhân đây tôi xin cảm ơn T.S Trần Việt Trung đã hỗ trợ tôi trong việc thu thập bộ dữ liệu địa chỉ chuẩn tiếng Việt, nguồn trên website batdongsan.com.vn, giúp tôi có thể cài đặt thử nghiệm giải thuật theo phương pháp Sagel.

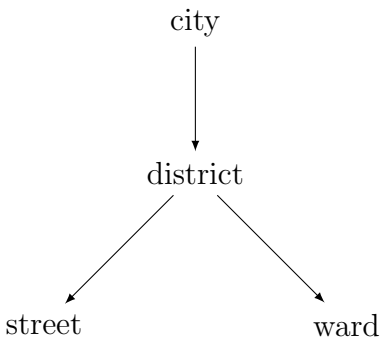
Bộ dữ liệu địa chỉ chuẩn này, gồm các phân cấp như **Hình 1.** *Phân cấp hành chính Việt Nam*, tuy nhiên có bổ sung thêm mức đường. Hơn nữa, tại mỗi mức phân cấp, bộ dữ liệu không chia ra thành từng phân cấp nhỏ hơn mà coi chung là một.

Gồm các mức như sau:

- **ward** : gồm xã, phường, thị trấn
- **district** : gồm quận, huyện, thị xã, thành phố trực thuộc tỉnh
- **city** : tỉnh, thành phố trực thuộc trung ương

Ngoài ra có mức **street**, không nằm trong phân cấp hành chính, nhưng thường xuất hiện trong địa chỉ nói chung và địa chỉ tiếng Việt nói riêng.

Cây địa chỉ tôi dựng được từ bộ dữ liệu chuẩn có dạng như sau:



Hình 2. *Cây địa chỉ từ dữ liệu chuẩn tại batdongsan.com.vn*

So với dữ liệu của phương pháp Sagel với địa chỉ Ấn Độ, họ sử dụng thêm một số trường như : building, zip code, lat long, short name, ... Phần lớn các trường trong đó đều không quá phổ biến trong địa chỉ tiếng Việt, ngoài ra cũng do hạn chế về mặt dữ liệu nên tôi chỉ có thể chuẩn hóa dựa trên cây địa chỉ ở trên.

Một đặc điểm về cây địa chỉ là Sagel không đưa số nhà vào trong cây địa chỉ. Trong bài báo họ không đưa lý do, nhưng cá nhân tôi cho rằng có thể do mức độ phân tách không cao, dễ gây nhiễu.

Ví dụ : Số 1 Đường X, thì đường X mang tính phân loại cao hơn với giải thuật của họ, còn số 1 thì đều xuất hiện trên tất cả các con đường.

Sau khi chuẩn hóa thì đầu ra bài toán của tôi dựa trên tổ hợp các đơn vị của cây địa chỉ có thể có là :

- street - district - city
- ward - district - city

Một số thông tin đi kèm trong chuỗi địa chỉ thì tôi có thể dùng luật để bóc tách ra ngoài các trường nêu trên.

Chi tiết về giải thuật tôi cài đặt thử nghiệm sẽ được đề cập ở chương 4.

Chương 3

Cơ sở lý thuyết

Ở chương này, tôi tập trung vào việc giới thiệu các thuật toán, mô hình, các độ đo hoặc công nghệ mà tôi sử dụng trong đồ án và đi sâu về mặt lý thuyết.

3.1. Mô hình CRF

CRF (Conditional Random Fields) - Trường ngẫu nhiên có điều kiện là một mô hình dựa trên xác suất có điều kiện, là một dạng của mô hình phân biệt (Discriminative model), thường được sử dụng trong các bài toán dự đoán cấu trúc, gán nhãn dữ liệu dạng chuỗi tuần tự (Sequences Tagging).

3.1.1. Discriminative classifier

Các mô hình học máy nhìn chung thường được phận loại thành 2 dạng chính, đó là *Generative model (Mô hình sinh)* và *Discriminative model (Mô hình phân biệt)*. CRF là một loại của Discriminative model, và do đó, chúng mô hình hóa ranh giới quyết định giữa các lớp khác nhau. Generative model, theo một cách khác, chúng mô hình hóa cách dữ liệu được sinh ra và sau khi được huấn luyện, mô hình có thể được sử dụng như một bộ phân loại.

Một ví dụ đơn giản và phổ biến của bộ phân loại theo xác suất là Naive Bayes - một bộ phân loại dựa trên Maximum Likelihood Estimation (MLE), là một Discriminative model.

Bộ phân loại Naive Bayes dựa trên giải thuật Naive Bayes, có công thức tổng quát:

$$P(y \mid X) = \frac{P(X \mid y)\, P(y)}{P(X)}$$

3.1. *Công thức Bayes*

Dự đoán mà bộ phân loại đưa ra, có thể được biểu diễn dưới dạng xác suất có điều kiện, và có thể được phân tách bằng giải thuật Naive Bayes:

Giả sử ta có các điểm dữ liệu $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_3$,.., $\mathbf{x}_n$. Giả sử thêm rằng ta đã biết các điểm dữ liệu này tuân theo một phân phối nào đó được mô tả bởi bộ tham sốy. Dựa trên giả

định các điểm dữ liệu là độc lập, ta có :

$$P(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, .., \mathbf{x}_n \mid y) \approx \prod_{n=1}^{N} P(\mathbf{x}_n \mid y)$$

3.2. *Công thức xấp xỉ theo giả định các điểm dữ liệu là độc lập*

Thay vào **3.1** ta được :

$$P(y \mid X) \approx \frac{\prod_{n=1}^{N} P(\mathbf{x}_n \mid y)\, P(y)}{P(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, .., \mathbf{x}_n)}$$

3.3. *Công thức Bayes theo giả định độc lập*

Bởi vì mẫu số là hằng số, nên ta có thể rút gọn:

$$P(y \mid X) \approx \prod_{n=1}^{N} P(\mathbf{x}_n \mid y)\, P(y)$$

3.4. *Công thức Bayes rút gọn theo giả định độc lập*

Bài toán trở thành bài toán tối ưu y sao cho xác suất $P(y \mid X)$ đạt cực đại:

$$\hat{y} = argmax_y \prod_{n=1}^{N} P(\mathbf{x}_n \mid y)\, P(y)$$

3.5. *Công thức tối ưu tham số theo MLE*

Trên đây là một ví dụ về cách bộ phân loại Naive Bayes, là một bộ phân loại thuộc Discriminative model, tối ưu các tham số dựa trên MLE.

3.1.2. Conditional Random Fields (CRF)

Để thuận tiện cho việc trình bày, trong phần này, tôi sẽ sử dụng kí hiệu $\underline{w}$ thay cho vector có các thành phần $w_1, w_2, ...w_d$, sử dụng $\exp(x)$ thay cho e^x.

3.1.2.1. Log-linear models

Giả sử có hai tập $\mathbf{X}$ và $\mathbf{Y}$, tập $\mathbf{Y}$ là tập hữu hạn. Mục tiêu của chúng ta là xây dựng một mô hình để ước lượng xác suất có điều kiện $p(y \mid x)$ là xác suất của một nhãn $y \in \mathbf{Y}$ với đầu vào là $x \in \mathbf{X}$. Ví dụ với bài toán gán nhãn từ loại, x có thể là một từ, y có thể là các nhãn N (danh từ), V (động từ),... Một feature-vector $\underline{\phi} : \mathbf{X} \times \mathbf{Y} \longrightarrow \mathbb{R}^d$, một vector trọng số $\underline{w} \in \mathbb{R}^d$. Log-liner models được định nghĩa với công thức như sau:

$$p(y \mid x; \underline{w}) = \frac{exp(\underline{w} \cdot \underline{\phi}(x,y))}{\sum_{y' \in \mathbf{Y}} exp(\underline{w} \cdot \underline{\phi}(x,y'))}$$

3.6. *Công thức log-linear models*

Đây là xác suất có điều kiện của y khi biết đầu vào x, với tham số $\underline{w}$. Tích số

$$\underline{w} \cdot \underline{\phi}(x,y)$$

3.7. *Tích trọng số và feature-vector*

có thể nhận giá trị dương hoặc âm, có thể được coi như một định lượng về mức độ hợp lý của nhãn y với đầu vào x. Với một đầu vào x, ta có thể tính toán được tích số **3.7** với mỗi nhãn $y \in \mathbf{Y}$. Để có thể chuyển về dạng phân phối xác suất một cách chính xác, tích số **3.7** phải luôn dương. Do đó ta sẽ sử dụng hàm exp():

$$exp(\underline{w} \cdot \underline{\phi}(x,y))$$

Giá trị này luôn lớn hơn 0. Cuối cùng, chuẩn hóa bằng cách chia cho tổng của tất cả tích số **3.7** trên tất cả các nhãn

$$\sum_{y' \in \mathbf{Y}} exp(\underline{w} \cdot \underline{\phi}(x,y'))$$

ta có thể đảm bảo rằng xác suất $p(y \mid x; \underline{w})$ luôn nằm trong khoảng $[0; 1]$ dù tích số **3.7** có thể nhận giá trị âm hoặc dương.

Log-Likelihood Function Trong phần trước, tôi đã đề cập đến việc tối ưu tham số cho bộ phân loại Naive Bayes bằng MLE. Trong phần này, tôi sẽ trình bày chi tiết hơn về cách MLE tối ưu tham số.

Giả sử ta có một tập gồm n điểm dữ liệu, $\{(x_i, y_i)\}_{i=1}^{n}$. Log-likehood function được định nghĩa là:

$$L(\underline{w}) = \sum_{i=1}^{n} \log p(y_i \mid x_i; \underline{w})$$

3.8. *Log-likelihood function*

Chúng ta có thể hiểu $L(\underline{w})$ là một hàm mà với một giá trị $\underline{w}$ cho trước, ta có thể đo được $\underline{w}$ mô tả dữ liệu tốt như thế nào. Ví dụ, một giá trị $\underline{w}$ tốt sẽ cho giá trị $p(y_i \mid x_i; \underline{w})$ cao với mọi $i = 1...n$, và do vậy chúng ta cũng có một giá trị $L(\underline{w})$ cao.

Maximum-likelihood estimates là đi tìm một giá trị $\underline{w}$ để cực đại hóa xác suất $p(y_i \mid x_i; \underline{w})$ hay "khớp" nhất với bộ dữ liệu huấn luyện dựa trên tiêu chí $L(\underline{w})$:

$$\underline{w}^* = \arg\max_{\underline{w} \in \mathbb{R}^d} \sum_{i=1}^{n} \log p(y_i \mid x_i; \underline{w})$$

3.9. *Công thức tối ưu tham số MLE*

Để có thể giải quyết bài toán MLE, chúng ta sử dụng *Gradient Decent* và sau khi tính toán sẽ thu được tham số $\underline{w}^T$ là một tham số tối ưu của mô hình.

Regularized log-likelihood Trong nhiều ứng dụng đã được kiểm chứng là đạt hiệu quả cao, log-likelihood thường được thêm vào một số *regularization term*, khi đó log-likelihood trở thành :

$$L(\underline{w}) = \sum_{i=1}^{n} \log p(y_i \mid x_i; \underline{w}) - \frac{\lambda}{2}\|\underline{w}\|^2$$

Trong đó, $\|\underline{w}\|^2 = \sum_j w_j^2$ và $\lambda > 0$, việc giải quyết bài toán vẫn là đi tìm $\underline{w}$ sao cho $L(\underline{w})$ đạt cực đại.

$$\underline{w}^* = \arg\max_{\underline{w} \in \mathbb{R}^d} L(\underline{w})$$

Tương tự như trong phần log-likelihood, ta vẫn sử dụng *Gradient Descent* để tìm ra giá trị $\underline{w}^*$ tối ưu cho bài toán.

3.1.2.2. CRFs model

Trong phần này, tôi cũng sẽ sử dụng $\underline{x}$ thay cho một chuỗi đầu vào $x_1...x_m$, và $\underline{s}$ thay cho một chuỗi các trạng thái $s_1...s_m$. Tập hợp các trạng thái có thể xảy ra là S. Tập hợp tất cả các chuỗi trạng thái có thể có là S^m.

Trong CRFs, chúng ta sẽ xây dựng lại một model của

$$p(s_1...s_m \mid x_1...x_m) = p(\underline{s} \mid \underline{x})$$

Đầu tiên, CRFs sẽ định nghĩa một feature vector

$$\underline{\Phi}(\underline{x}, \underline{s}) \in \mathbb{R}^d$$

vector này sẽ ánh xạ toàn bộ chuỗi đầu vào $\underline{x}$ với toàn bộ chuỗi trạng thái $\underline{s}$ tương ứng theo cặp thành một số feature vector d chiều .Feature vector này được định nghĩa như sau:

$$\underline{\Phi}(\underline{x}, \underline{s}) = \sum_{j=1}^{m} \underline{\phi}(\underline{x}, j, s_{j-1}, s_j)$$

3.10. *Công thức feature vector*

Trong đó :

- $\underline{x}$ là toàn bộ câu đang xét
- i là vị trí được gắn thẻ (giá trị từ 1 đến m)
- s_{j-1} là giá trị trạng thái liền trước của trạng thái hiện tại(có thể là bất kì giá trị nào trong S)
- s_j là giá trị trạng thái mới(có thể là bất kì giá trị nào trong S)

Giả sử có 1 feature thứ k nào đó trong khoảng 1 ... d, feature vectore tại k sẽ là:

$$\Phi_k(\underline{x}, \underline{s}) = \sum_{j=1}^{m} \phi_k(\underline{x}, j, s_{j-1}, s_j)$$

3.11. *Công thức feature vector tại một giá trị k bất kì*

Từ đó, ta có thể tính được giá trị Φ_k bằng cách tính tổng của tất cả các ϕ_k trên toàn bộ m trạng thái khác nhau từ $s_1 ... s_m$

Sau đó chúng ta sẽ xây dựng log-linear model,

$$p(\underline{s} \mid \underline{x}; \underline{w}) = \frac{\exp(\underline{w} \cdot \underline{\Phi}(\underline{x}, \underline{s})}{\sum_{\underline{s'} \in S^m} \exp(\underline{w} \cdot \underline{\Phi}(\underline{x}, \underline{s'})}$$

3.12. *Công thức log-linear model cho CRFs*

Đây là một công thức cần lượng tính toán rất lớn, do không gian các giá trị có thể xảy ra với $\underline{s}$ là S^m, vô cùng lớn, và việc tính toán tổng ở dưới mẫu trong **3.12** cũng bị lặp lại nhiều lần. Tuy nhiên nếu đưa ra các giả định thích hợp, ta có thể huấn luyện và tính toán mô hình này một cách khá hiệu quả.

Decoding với CRFs Vấn đề này được phát biểu như sau: Cho một chuỗi đầu vào $\underline{x} = x_1, x_2, ... x_m$, chúng ta phải tìm ra được một chuỗi trạng thái thích hợp nhất với chuỗi đầu vào theo mô hình

$$\arg max_{\underline{s} \in S^m} p(\underline{s} \mid \underline{x}; \underline{w})$$

Biến đổi dựa vào **3.10** thu được:

$$\arg max_{\underline{s} \in S^m} p(\underline{s} \mid \underline{x}; \underline{w}) = \arg \max_{\underline{s} \in S^m} \sum_{j=1}^{m} \underline{w} \cdot \underline{\phi}(\underline{x}, j, s_{j-1}, s_j)$$

Giải quyết vấn đề decoding chính là đi tìm một chuỗi trạng thái sao cho cực đại hóa:

$$\arg \max_{\underline{s} \in S^m} \sum_{j=1}^{m} \underline{w} \cdot \underline{\phi}(\underline{x}, j, s_{j-1}, s_j)$$

Mỗi khi chuyển trạng thái từ s_{j-1} sang s_j đều có một số điểm liên kết:

$$\underline{w} \cdot \underline{\phi}(\underline{x}, j, s_{j-1}, s_j)$$

Để giải bài toán này, ta sử dụng biến thể của giải thuật Viterbi

- Khởi tạo : for $s \in S$

  $$\pi[1, s] = \underline{w} \cdot \underline{\phi}(\underline{x}, 1, s_0, s)$$

  s_0 là một giá trị khởi tạo đặc biệt "initial"

- For $j = 2...m, s = 1...k$:

  $$\pi[j, s] = max_{s' \in S}[\pi[j-1, s'] + \underline{w} \cdot \underline{\phi}(\underline{x}, j, s', s)]$$

Cuối cùng ta được :

$$\max_{s1...sm} \sum_{j=1}^{m} \underline{w} \cdot \underline{\phi}(\underline{x}, j, s_{j-1}, s_j) = \max_{s} \pi[m, s]$$

3.13. *Công thức decoding trong CRFs*

Thuật toán này chỉ chạy với độ phức tạp là $O(mk^2)$ nên việc decodeing trong CRFs là hoàn toàn khả thi.

Ước lượng tham số trong CRFs Để ước lượng tham số, giả sử ta có một tập dữ liệu gồm n ví dụ đã được gán nhãn $\{(\underline{x}^i, \underline{s}^i)\}_{i=1}^n$. Mỗi $\underline{x}^i$ là một chuỗi đầu vào $x_1^i...x_m^i$, mỗi $\underline{s}^i$ là một chuỗi các trạng thái $s_1^i...s_m^i$. Việc tối ưu tham số được thực hiện giống như ở phần *regular log-linear models*:

$$L(\underline{w}) = \sum_{i=1}^{n} \log p(\underline{s}^i \mid \underline{x}^i; w) - \frac{\lambda}{2}\|\underline{w}\|^2$$

Tham số $\underline{w}$ được ước lượng:

$$\underline{w}^* = \arg\max_{\underline{w} \in \mathbb{R}^d} \sum_{i=1}^{n} \log p(\underline{s}^i \mid \underline{x}^i; \underline{w}) - \frac{\lambda}{2}\|\underline{w}\|^2$$

3.14. *Công thức tối ưu tham số trong CRFs*

Để tìm ra được giá trị $\underline{w}^*$, *Gradient Descent* vẫn là sự lựa chọn tối ưu, về chi tiết của giải thuật, tôi không trình bày chi tiết vì trong chương này tôi chỉ muốn giới thiệu cơ bản về các mô hình và công nghệ mà tôi sử dụng.

3.2. Máy tìm kiếm

Ở phần này, tôi sẽ trình bày về các công nghệ cốt lõi mà phương pháp Sagel sử dụng. Về cơ bản, giải thuật của Sagel là một giải thuật *Fuzzy Matching*, giải thuật này cần phải *match* chuỗi địa chỉ đầu vào với một chuỗi địa chỉ trong hàng chục nghìn, thậm chí là hàng trăm nghìn chuỗi địa chỉ chuẩn. Vì vậy, việc đảm bảo tốc độ là vô cùng cần thiết. Máy tìm kiếm là một lựa chọn tối ưu.

Máy tìm kiếm, tên gọi tiếng Anh là *Search Engine*, là một hệ thống truy vấn thông tin, các kết quả trả về thường được biểu diễn trong một danh sách, và được gọi là *hits*. Máy tìm kiếm được xây dựng và thiết kế để giảm thiểu thời gian cần thiết trong quá trình truy vấn. Trong phạm vi của đồ án, tôi sử dụng máy tìm kiếm *Elasticseark*, là một máy tìm kiếm phổ biến hiện nay. Ở trong *section* này, tôi sẽ tập trung giới thiệu về *Elasticseark*.

3.2.1. Chỉ mục ngược

Elasticsearch sử dụng một cấu trúc gọi là *chỉ mục ngược* hay *invered index*. Chỉ mục này được thiết kế để cho phép thực hiện các truy vấn *full-text* với tốc độ rất cao. Dưới đây là một ví dụ về chỉ mục ngược:

1. Tôi học tại trường Bách Khoa Hà Nội
2. Bách Khoa thật đẹp

Có 2 văn bản có chứa các từ như trên, gọi là doc_1 và doc_2. Chỉ mục ngược sẽ không lưu trữ theo thông thường là doc - words mà lưu trữ ngược theo theo dạng word - docs. Đây là lí do vì sao nó được đặt tên là chỉ mục ngược hay chỉ mục đảo.

Với ví dụ trên, kết quả sẽ như sau:

term	doc_1	doc_2
Tôi	X	
học	X	
tại	X	
trường	X	
Bách	X	X
Khoa	X	X
Hà	X	
Nội	X	
thật		X
đẹp		X

Bảng 1. *Ví dụ về chỉ mục ngược*

Với 1 truy vấn *Tôi đến Bách Khoa*, máy tìm kiếm chỉ cần tìm văn bản có xuất hiện các từ *Tôi*, *đến*, *Bách*, *Khoa*. Trong trường hợp này, doc_1 xuất hiện 3 từ *Tôi, Bách, Khoa*, doc_2 xuất hiện 2 từ *Bách, Khoa*. Hai văn bản này đều liên quan đến câu truy vấn, tuy nhiên doc_1 sẽ được cho điểm số cao hơn là doc_2 do chứa nhiều từ hơn.

term	doc_1	doc_2
Tôi	X	
đến		
Bách	X	X
Khoa	X	X

Bảng 2. *Truy vấn sử dụng chỉ mục ngược*

Trên đây là một ví dụ đơn giản về thiết kế và truy vấn sử dụng chỉ mục ngược. Hầu hết các máy tìm kiếm phổ biến hiện nay đều sử dụng cấu trúc này cho hệ thống của mình.

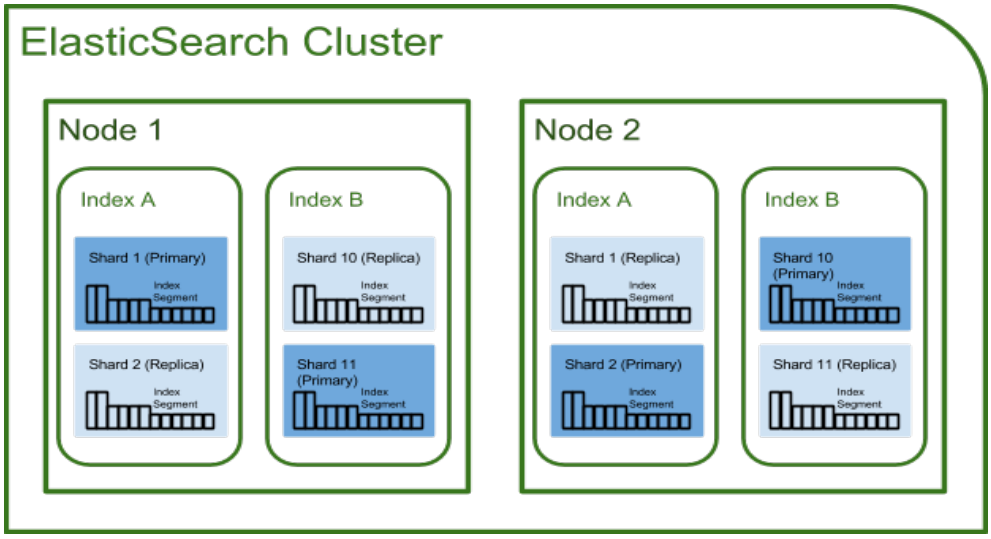
3.2.2. Elasticsearch

Elasticsearch là một máy tìm kiếm, mã nguồn mở, chạy trên nền tảng*Apache Lucene*. *Elasticsearch* có thể chạy phân tán, người sử dụng có thết tương tác với nó thông qua API với giao thức HTTP hoặc TCP. Điểm mạnh của *Elasticsearch* là dễ triển khai và khả năng mở rộng rất tốt.

3.2.2.1. Kiến trúc của Elasticsearch

Cluster Cluster là tập hợp một hay nhiều node (hay server) cùng nắm giữ toàn bộ data, cung cấp khả năng lập chỉ mục và tìm kiếm liên kết giữa các node với nhau. mỗi cluster được định danh bằng một tên duy nhất, ví dụ tên của cụm là "elasticsearch", mỗi node là một phần của cụm khi được thiết lập được nối với cụm bằng tên "elasticsearch".

Node Node là một server đơn lẻ, là một phần của cluster, lưu trữ data, tham gia vào chức năng lập chỉ mục và tìm kiếm. Cũng như cluster, node cũng được định danh, mỗi nút được biết đến với một tên thường được đặt mặc định ngẫu nhiên bởi Universally Unique IDentifier (UUID), việc đặt tên được tiến hành khi thiết lập, người dùng cũng có thể tự đặt tên node. Tên của nút rất quan trọng trong việc xác định nút này thuộc cụm nào trong một cụm elasticsearch.



Hình 3. *Kiến trúc Elasticsearch Cluster*

Document Document là đơn vị cơ bản của dữ liệu trong Elasticsearch hay nói cách khác, dữ liệu trong Elasticsearch được tổ chức thành các document. Document trong Elasticsearch có dạng giống như JSON, bao gồm các thuộc tính và giá trị (giá trị có thể là kiểu String, Long, Boolean, Float,..). Người dùng có thể cấu hình các thuộc tính để bắt buộc có hoặc không bắt buộc, có thể định nghĩa trước hoặc không cần định nghĩa trước. Ví dụ một document trong Elasticsearch:

```
{
    "name":         "John Smith",
    "age":          42,
    "confirmed":    true,
    "join_date":    "2014-06-01",
    "home": {
        "lat":      51.5,
        "lon":      0.1
    },
    "accounts": [
        {
            "type": "facebook",
            "id":   "johnsmith"
        },
        {
            "type": "twitter",
            "id":   "johnsmith"
        }
    ]
}
```

Index Index (chỉ mục) là một tập hợp của các document mà chứa những đặc điểm tương tự. Ví dụ, chúng ta có chỉ mục về dữ liệu khách hàng, một chỉ mục khác về danh mục sản phẩm, và những chỉ mục khác cho những data khác. Chỉ mục cũng được định danh bằng tên, tên này được sử dụng khi thực hiện các hoạt động như lập chỉ mục, tìm kiếm, cập nhật hoặc xóa các document trong đó.

Type Type được sử dụng làm danh mục của một chỉ mục, cho phép lưu trữ các loại dữ liệu khác nhau trong cùng một chỉ mục.

Shards & Replicas Một chỉ mục trong Elasticsearch có thể lưu trữ một lượng lớn dữ liệu, thậm chí lớn hơn cả kích cỡ ổ đĩa cứng tại một node. Điều này khiến cho node đó phản hồi chậm với các truy vấn khi lượng dữ liệu tăng. Để giải quyết vấn đề này, Elasticsearch cung cấp khả năng cho phép chia dữ liệu trong một chỉ mục ra thành các phân mảnh nhỏ hơn, gọi là *shard*. Khi tạo mới một chỉ mục, người dùng có thể cấu hình số *shard* mà dữ liệu chỉ mục có thể được chia, nếu không Elasticsearch sẽ để một giá trị mặc định.

Các *shard* có thể được chia ra nhiều node khác nhau, qua đó giảm tải lượng dữ liệu tập trung trên một node khiến node bị quá tải. Ngoài ra, việc chia *shard* như vậy cho phép dữ liệu được lưu trữ phân tán.

Trong một số trường hợp, node bị down hoặc offline, để đảm bảo *high available* cho hệ thống, việc nhân bản dữ liệu là cần thiết. Việc nhân bản dữ liệu trong Elasticsearch được thực hiện bằng việc nhân bản các *shard* và người dùng có thể cấu hình được số lượng nhân bản này. Việc này còn giúp mở rộng kích thước hoặc thông lượng tìm kiếm,

bởi các tìm kiếm có thể được thực hiện song song trên các node.

Tóm tắt lại, để có thể lưu trữ dữ liệu phân tán, Elasticsearch chia dữ liệu trong một chỉ mục ra các *shard*, các *shard* này được phân chia ra các node và có thể nhân bản các shard này. Từ đó giúp tăng khả năng chịu tải cũng như tính *available* cho hệ thống.

3.2.2.2. Truy vấn trong Elasticsearch

Elasticsearch hỗ trợ người dùng rất nhiều kiểu truy vấn, hướng tới những mục đích và công việc khác nhau. Trong phạm vi của phần *cơ sở lý thuyết về Elasticsearch*, tôi chỉ giới thiệu những truy vấn tôi sử dụng trong đồ án.

Truy vấn DSL

Domain Specific Language (DSL) - Ngôn ngữ miền chuyên biệt là một ngôn ngữ máy tính, chuyên dùng cho một lĩnh vực, miền ứng dụng cụ thể. Ở đây, Elasticsearch cung cấp truy vấn DSL và dựa trên JSON để định nghĩa các truy vấn. Truy vấn DSL trong Elasticsearch thông thường được chia thành loại chính :

- Leaf query clauses : Là truy vấn mà tìm kiếm một giá trị cụ thể trong một trường cụ thể, thường là các query *match, term, range*.
- Compound query clauses : Là các truy vấn có thể bao gồm *leaf query clauses* và cả *compound query clauses* khác, thực hiện các truy vấn phức tạp như truy vấn *bool, dis_max*.

Các truy vấn thông dụng

- Truy vấn *match* : đây là một truy vấn tiêu chuẩn để thực hiện các truy vấn *full text*, bao gồm *fuzzy matching* và truy vấn theo cụm từ hoặc truy vấn gần đúng.
- Truy vấn *match_phrase* : giống như truy vấn *match* nhưng được sử dụng để tìm kiếm chính xác một cụm từ hoặc một cụm từ với các từ gần đúng.
- Truy vấn *bool* : Là một truy vấn cấp cao, thường dùng để kết hợp nhiều truy vấn *leaf query clauses*, cho phép truy vấn phức tạp sử dụng các điều kiện *or, and, not*. *Bool query* hỗ trợ các tham số :

  - *must* : phải phù hợp với tất cả các điều kiện và đóng góp vào điểm số, tương đương với *and*.

  - *must_not* : ngược lại với *must*, phải không phù hợp với tất cả các điều kiện, tương đương với câu lệnh *not*.

  - *should* : phù hợp với một hoặc một vài trong số tất cả các điều kiện, tương đương với câu lệnh *or*.

  - *minimum_should_match* : có kiểu dữ liệu là số nguyên, quy định số kết quả mà *should* bắt buộc phải khớp, mặc định là bằng 1.

3.3. Các độ đo sử dụng trong bài toán

3.3.1. Độ tương tự Jascard

Độ tương tự *Jascard* là một chỉ số thống kê, thường được dùng để đánh giá mức độ tương đồng hoặc đa dạng của các tập mẫu với nhau. Độ tương tự *Jascard* được tính bằng cách lấy giao của hai tập hợp chia cho hợp của hai tập hợp:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

3.15. *Độ đo Jascard*

Trong đó A, B là hai tập hữu hạn. Trong trường hợp cả hai tập đều rỗng, thì $J(A,B) = 0$. Vậy nên :

$$0 \leq J(A,B) \leq 1$$

3.3.2. Khoảng cách Levenshtein

Chương 4

Phương pháp giải quyết

Chương 5

Đánh giá và cài đặt thử nghiệm

Chương 6

Kết luận và hướng phát triển