Chương 3: Phân tích câu truy vấn

3.1 Giới thiệu:

Trong hệ GIR, các câu truy vấn thường xuất hiện với những thông tin về chủ đề cần tìm và kèm theo là các thông tin chỉ không gian một cách tường minh hoặc không tường minh nhằm mô tả vùng không gian mà người dùng mong muốn tìm thấy các chủ đề ấy liên quan đến nơi đó (*ví dụ: khách sạn ở Hà Nội, café ở quận 1, TP.HCM, v.v…*). Vì vậy vấn đề trong tiền xử lý câu truy vấn trong hệ GIR sẽ là làm sao xác định được các chủ đề tìm kiếm và vùng không gian mà câu truy vấn đề cập đến và mối quan hệ giữa hai thành phần ấy nhằm giúp cho quá trình tìm kiếm sau đó có "mục tiêu" rõ ràng hơn và các kết quả "chính xác" hơn.

Trong vấn đề này, từ lâu, phương pháp khai khoáng văn bản (text mining) đã từng được áp dụng thành công với hệ GIR để phát hiện các từ/cụm từ chỉ nơi chốn trong nội dung văn bản, hoặc xác định các vùng không gian trong tài liệu. Tuy nhiên, phần chính của các nghiên cứu đó là tập trung vào các tài liệu web hoặc các tài liệu văn bản thông thường. Trong khi đó, đối với một câu truy vấn, thì vấn đề phát hiện và xử lý các thông tin về nơi chốn trở nên khó khăn hơn do nó quá ngắn gọn và thường phụ thuộc vào chủ quan của người dùng bên cạnh các vấn đề về cách phát âm sai, dùng từ nước ngoài, hay từ viết tắt trong truy vấn, v.v… Vì vậy, làm cách nào để có thể tự động biết được người dùng đề cập đến chủ đề gì, giới hạn ở những nơi chốn nào chỉ qua một câu truy vấn đơn giản là công việc cần phải được giải quyết tốt, để từ đó có thể tìm ra được những tài liệu thích hợp nhất với yêu cầu tìm kiếm từ phía người sử dụng.

Trong chương 3, luận văn xin giới thiệu và đề xuất cải tiến một phương pháp đơn giản và hiệu quả dùng để phân tích câu truy vấn, đặc biệt là những câu truy vấn cho hệ GIR Việt Nam.

3.2 Các khái niệm và công việc liên quan:

Phân tích truy vấn trong một hệ GIR là từ một câu truy vấn làm sao để xác định được một bộ ba thành phần <*what, relation, where*>, trong đó *what* là những gì mà người dùng muốn tìm kiếm (các chủ đề tìm kiếm, v.v…), *where* là những nơi mà người dùng chỉ ra để giới hạn phạm vi tìm kiếm trong câu truy vấn (tên các địa danh, v.v…) và *relation* chính là mối quan hệ không gian (quan hệ topology) giữa *what* và *where.* Chúng ta cần lưu ý một chi tiết rằng trong bộ ba đó, không nhất thiết lúc nào cũng đầy đủ cả 3 thành phần, và đặt biệt thành phần *relation* và *where* có thể được hiểu ngầm định.

Hiệu suất của một máy tìm kiếm phụ thuộc vào khả năng nắm bắt được ý nghĩa của câu truy vấn càng gần với mục đích của người dùng càng tốt [13]. Do đó, các cải tiến gần đây của máy tìm kiếm là thêm vào các chức năng xử lý, khả năng phân tích địa lý, thực hiện kết hợp giữa hệ thống thông tin địa lý (GIS) và rút trích thông tin (IR) nhằm xây dựng các máy tìm kiếm thông minh. Khả năng nhận biết và nắm bắt chính xác các đối tượng địa lý cũng như là chủ đề tìm kiếm xuất hiện trong nội dung tài liệu và câu truy vấn của người dùng là vấn đề cốt yếu trong những hệ truy vấn thông tin địa lý (GIR) như thế này.

Trích chọn và phân biệt các đối tượng từ trong văn bản thường liên quan đến mảng nhận biết đối tượng (Named Entity Recognition - NER), và NER đã rất thành công trong các yêu cầu có dạng như thế. Tuy nhiên, trong trường hợp cụ thể về nhận biết và phân biệt các đối tượng không gian của hệ GIR thì NER đã gặp phải nhiều vấn đề trở ngại. Khi làm việc với các đối tượng địa lý với mức độ chi tiết cao, các vấn đề trùng lấp ngữ nghĩa thường xuyên xuất hiện. Với cùng một tên địa danh, nó có thể được dùng để chỉ 2 địa điểm khác nhau hoàn toàn (ví dụ *huyện Châu Thành, tỉnh Tiền Giang* và *huyện Châu Thành, tỉnh Long An*) và ngược lại với chỉ 1 địa điểm nhưng lại có nhiều hơn 1 tên gọi về địa điểm đó (ví dụ *Thành phố Hồ Chí Minh* và *Sài Gòn* đều chỉ 1 nơi).

Do đó vấn đề không thể dừng lại ở việc áp dụng NER cho quá trình phân tích truy vấn. Để có thể xử lý các trường hợp nhập nhằng như thế, một ontology về quan hệ giữa các tên địa danh là rất cần thiết. Dựa vào ontology đó, quá trình phân tích sẽ hiệu quả hơn trong việc xác định một hay nhiều tên địa danh đang được đề cập đến trong ngữ cảnh câu truy vấn, và từ đó xác định chính xác vùng không gian mà người tìm kiếm muốn giới hạn trong truy vấn.

Hiện tại, trên thế giới, các hệ GIR hàng đầu như *Google Maps*[21], *Live Maps*[22] (của *Microsoft*) đều xây dựng cho riêng mình một phương pháp phân tích riêng theo tiêu chuẩn mà họ đặt ra. B. Martins [9] cũng có đề xuất một cách tiếp cận cho vấn đề này. Tuy nhiên, giữa các cách tiếp cận này với nhau vẫn còn tồn tại những mâu thuẫn khó có thể giải thích được hoặc chỉ có thể giải thích theo quan điểm cá nhân. Vấn đề này sẽ được đề cập nhiều hơn trong các phần tiếp theo của chương.

3.3 Phân tích các thành phần trong câu truy vấn:

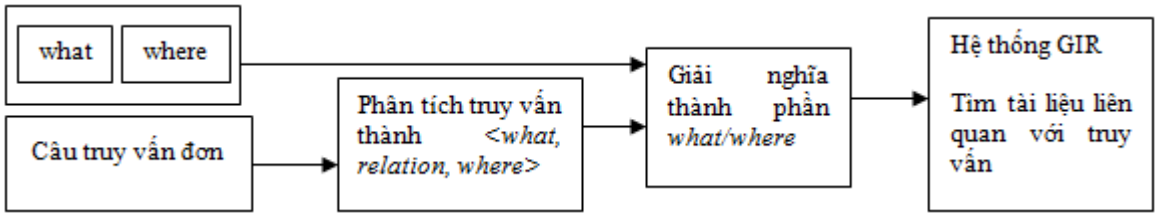
Với đặc trưng của hệ thống là các thông tin liên quan đến các khái niệm, các chủ đề và thông tin về vị trí đi kèm theo chúng, do đó ta có thể thấy rằng đặc trưng trong truy vấn của hệ thống cũng tương tự, nghĩa là các câu truy vấn sẽ thường xuyên có dạng tìm kiếm cái gì và muốn tìm cái đó ở chỗ nào (“*object in place name”* [12]). Vì vậy, hầu hết mọi câu truy vấn trong hệ GIR đều có thể được phân tích thành một bộ ba <*what, relation, where*> trong đó *what* dùng để chỉ đối tượng tìm kiếm, thường là các chủ đề tìm kiếm, *where* dùng để chỉ nơi chốn mà câu truy vấn giới hạn lại, và *relation* chỉ mối quan hệ không gian giữa *what* và *where* và có vai trò quyết định phương thức tìm kiếm của hệ thống (tìm trong vùng, tìm xung quanh điểm hay tìm trên đường, v.v…). Trong khi *what* có thể được cấu thành một cách tự do miễn sao nêu lên được thông tin cần tìm là gì thì *relation* và *where* lại phải theo một danh sách các từ/cụm từ có kiểm soát. Cụ thể *relation* thường phải là những quan hệ không gian phổ biến mà hệ GIR có thể hiểu được (ví dụ: *“ở”, “gần”, “ở tại”, “ở trong”, “trên”, “ở trên”,* v.v…), còn *where* thì

phải được phân tích sao cho kết quả sau cùng nằm trong một tập phân biệt các địa danh được định nghĩa trong ontology địa lý mà hệ thống sử dụng.

Các hệ thống khác nhau có thể có nhiều cách thức khác nhau để tiếp nhận câu truy vấn từ phía người dùng. Thông thường thì có 3 cách:

- Người dùng nhập vào một câu truy vấn duy nhất. Lúc này hệ thống sẽ phải tự phân tích ra các thành phần <*what, relation, where*>.
- Người dùng nhập câu truy vấn vào những ô nhập liệu tách biệt nhau, mỗi ô cho một thành phần. Lúc này hệ thống có thể bỏ qua việc phân tích câu truy vấn thành một bộ ba <*what, relation, where*> mà chỉ cần phân tích cụ thể cho từng thành phần riêng biệt mà người dùng nhập vào.
- Người dùng nhập câu truy vấn với nội dung là thành phần *what*, nghĩa là chỉ cho hệ thống biết chủ đề tìm kiếm. Sau đó, người dùng sẽ cung cấp cho hệ thống vùng giới hạn tìm kiếm bằng cách chọn trực tiếp từ bản đồ hoặc chọn từ một danh sách các tên địa danh phân biệt sẵn có. Lúc này, công việc của hệ thống trở nên đơn giản hơn rất nhiều vì chỉ cần phân tích chủ đề mà người dùng muốn tìm, các thành phần còn lại được hệ thống cung cấp sẵn coi như đã rõ ràng.

Hình 3-1 thể hiện các quá trình từ khi tiếp nhận câu tury vấn đến khi phân tích ra kết quả mà hệ thống có thể hiểu được.



Hình 3-1: Qui trình phân tích câu truy vấn từ người dùng.

3.4 Giải thuật phân tích các thành phần trong câu truy vấn:

3.4.1 Xác định bộ ba *<what, relation, where>*:

Như đã trình bày, một hệ thống GIR có thể cung cấp cho người dùng nhiều cách thức khác nhau để truy vấn thông tin. Nếu hệ thống sử dụng giao diện gồm có các ô nhập liệu riêng biệt cho từng thành phần thì phần việc này coi như được bỏ qua vì khi đó dữ liệu nhập từ phía người dùng đã cho biết rõ đâu là *what*, đâu là *where* và *rel* giữa *what* và *where* là gì. Tuy nhiên, nhằm tạo ra sự đơn giản, tự nhiên hơn ở phía người sử dụng cũng như là giao diện của hệ thống và khả năng tích hợp chung với các hệ tìm kiếm khác thì những hệ thống GIR ngày nay thường luôn cho phép người dùng nhập vào một ô nhập liệu duy nhất mô tả yêu cầu tìm kiếm một cách ngắn gọn nhưng đủ nghĩa để hệ thống có thể phân tích ra các thành phần <*what, rel, where*> phục vụ cho việc tìm kiếm. Đó chính là xu hướng, cách tiếp cận người sử dụng của các hệ *local search* nổi tiếng trên thế giới như *Google Maps* [21] hay *Live Maps* [22] của *Microsoft*. Vậy nếu một hệ thống theo cách tiếp cận như thế sẽ giải quyết vấn đề phân tích câu truy vấn như thế nào để đạt được hiệu quả tìm kiếm cao? B. Martins [9] đã dựa vào đặc điểm truy vấn của hệ GIR để đề xuất một cách tiếp cận khá đơn giản với ý tưởng là sẽ tách các thành phần *what* và *where* trong câu truy vấn bằng những từ/cụm nhận biết được trong một tập hợp hữu hạn các từ khóa chỉ quan hệ không gian *rel* (ví dụ như: *in, at, near, around, v.v…*). Theo đó, thành phần đứng trước *rel* trong câu truy vấn là *what* và thành phần theo sau sẽ là *where*. Tuy nhiên, tác giả của đề xuất đó lại không đề cập đến việc nếu câu truy vấn với quan hệ giữa *what* và *where* không tường minh thì sẽ phân tích như thế nào (ví dụ: *Hotel New York*). Bên cạnh cách tiếp cận đó, qua việc khảo sát các dạng câu truy vấn khác nhau trên hai hệ *Google Maps* [21] và *Live Maps* [22], ta có thể nhận thấy rằng còn có những hướng tiếp cận khác để giải quyết bài toán này nhưng vì đó là những hệ thống mang tính thương mại nên không có một công trình nào liên

quan đến vấn đề của họ được công bố. Ta có thể xem qua một vài kết quả khảo sát trong bảng dưới đây:

Số thứ tự	**Câu truy vấn**	**Google Maps**	**Live Maps**	**B. Martins**
1	New York Hotel	*What* := "**Hotel**" *Rel* := "**Near**" *Where* := "**New York, NY**" (**)	*What* := "**Hotel**" *Rel* := "**Near**" *Where* := "**New York, NY**"	*What* := "**New York Hotel**" *Rel* := "" *Where* := ""
2	Paris Hotel	*What* := "**Hotel**" *Rel* := "**Near**" *Where* := "**Paris, France**"	*What* := "**Paris Hotel**" *Rel* := "" *Where* := ""	*What* := "**Paris Hotel**" *Rel* := "" *Where* := ""
3	Hotel in Paris, Lyon	*What* := "**Hotel**" *Rel* := "**Near**" *Where* := "**Place de Paris, 69009 Lyon, France**"	*What* := "**Hotel Lyon**" *Rel* := "**Near**" *Where* := "**Paris, France**"	*What* := "**Hotel**" *Rel* := "**Near**" *Where* := "**Rue de Lyon, 75012 Paris, France**" (*)
4	Hotel in Lyon, Paris	*What* := "**Hotel**" *Rel* := "**Near**"	*What* := "**Hotel**" *Rel* := "**Near**"	*What* := "**Hotel**" *Rel* := "**Near**"

		Where := "**Place de Paris, 69009 Lyon, France**"	*Where* := "**Rue de Lyon, 75012 Paris, France**"	*Where* := "**Place de Paris, 69009 Lyon, France**" (*)
5	Paris, Lyon	*What* := "**Paris**" *Rel* := "**Near**" *Where* := "**Lyon, France**"	*What* := "**Lyon**" *Rel* := "**Near**" *Where* := "**Paris, France**"	*What* := "**Paris**" *Rel* := "**Near**" *Where* := "**Lyon, France**"
6	Hotel with name Novotel	*What* := "**Hotel with name Novotel**" *Rel* := "" *Where* := "" (Kết quả tìm kiếm bao gồm cả những **Hotel** không có tên là **Novotel**)	*What* := "**Hotel with name Novotel**" *Rel* := "" *Where* := "" (Không tìm thấy kết quả)	*What* := "**Hotel with name Novotel**" *Rel* := "" *Where* := ""
7	Hotel with name Novotel in New York	*What* := "**Hotel with name Novotel**" *Rel* := "**Near**" *Where* := "**New**	*What* := "**Hotel with name Novotel**" *Rel* := "**Near**" *Where* := "**New**	*What* := "**Hotel with name Novotel**" *Rel* := "**Near**" *Where* := "**New**

		York, NY” (Kết quả tìm kiếm bao gồm cả những **Hotel** không có tên là **Novotel**)	**York, NY**” (Không tìm thấy kết quả)	**York, NY**”
8	Hotel around White House	*What* := “**Hotel**” *Rel* := “**Near**” *Where* := “**White House, Washington, DC**”	*What* := “**Hotel House**” *Rel* := “**Near**” *Where* := “**White”**	*What* := “**Hotel**” *Rel* := “**Near**” *Where* := “**White House, Washington, DC**”

(*) Giả sử cách tiếp cận của B. Martins sử dụng cùng một ontology địa lý với Google Maps.

(**) Trong bảng khảo sát trên, thành phần where đã được các hệ thống thực hiện phân tích giải nghĩa cụ thể chứ không còn là nguyên gốc từ câu truy vấn.

Bảng 3-1: Bảng khảo sát một số câu truy vấn trên các local search nổi tiếng.

Dựa vào quá trình khảo sát các hệ thống lớn và phương pháp đề nghị của B. Martins, nhận thấy rằng các cách tiếp cận ấy vẫn còn tồn tại một số vấn đề cần giải quyết cũng như là việc những cách tiếp cận đó chưa đặt trong môi trường câu truy vấn tiếng Việt. Vì vậy, luận văn này xin đề xuất một số cách thức cải tiến nhằm mục tiêu xây dựng các phương pháp phân tích câu truy vấn, các thành phần của câu truy vấn sao cho đáp ứng được những vấn đề mà các cách tiếp cận trước không đề cập đến hoặc không giải quyết được và quan trọng là các phân tích phải dựa trên những tính chất đặc trưng trong cách nói, cách dùng từ, quan điểm của người dùng Việt Nam về cách mô tả chủ đề tìm kiếm

và các địa danh ở Việt Nam. Dưới đây là ý tưởng của giải thuật xác định 3 thành phần chính của câu truy vấn dựa theo cách tiếp cận của Martins có cải tiến:

Giải thuật ***GIRQueryAnalyzer*:**

Công cụ hỗ trợ: Ontology địa lý *O*, Danh sách các quan hệ không gian *R*.

Input: Câu truy vấn *Q*.

Output: <*What, Rel, Where*>.

arrTokens := mảng các token của *Q*.

SizeOfTokens := SizeOf(*arrTokens*).

For i := 0 to *SizeOfTokens-1* **Do**

    What := Concatenate(arrTokens, 0, SizeOfTokens - 1)

    For j := i to *SizeOfTokens-1* **Do**

        Rel2 := Concatenate(arrTokens, i, j)

        If *IsValidRelation(Rel2)* **Then**

            What2 := Q.Substring(0, IndexOf(Rel2))

            Where := Concatenate(arrTokens, j + 1, SizeOfTokens - 1)

            If *IsValidWherePhrase(Where)* **Then**

                Rel := *Rel2*

                i := *SizeOfTokens-1*

                Break.

            End If

            Rel := *Rel2*

        End If

    End For

End For

If (*Rel* == null AND *Where* == null) **Then**

    i := Vị trí xuất hiện của địa danh đầu tiên trong *Q*.

    If (*i* > *0*) **Then**

        What2 := Chuỗi *Q* từ 0 đến *i*.

        Where := Chuỗi *Q* từ *i* đến cuối chuỗi.

        Rel := "Ở".

    End If

End If

If Not *IsValidWherePhrase(Where)* **Then**

    What2 := *What*

    Rel := null

    Where := null

End If

If (*What2* <> null) **Then**

    What := *What2*.

End If

Return <*What, Rel, Where*>

Ý tưởng của giải thuật trên là duyệt qua từng token trong câu truy vấn. Khi hệ thống nhận biết được từ/cụm từ chỉ quan hệ không gian dài nhất trong câu thông qua hàm kiểm tra *IsValidRelation*, hệ thống sẽ chọn hai thành phần trước và sau từ/cụm từ đó làm thành phần *what* và *where*. Tại một thời điểm khi đã tạm thời tách được *what* và *where*, hệ thống sẽ kiểm tra thành phần *where* ấy có phải là tập hợp các tên nơi chốn hay không qua hàm *IsValidWherePhrase*. Nếu thỏa, hệ thống sẽ dừng phân tích và

chọn các thành phần <*what, rel, where*> đó làm kết quả của thuật giải. Ngược lại, hệ thống sẽ tiếp tục duyệt qua các token của câu truy vấn để chọn các thành phần khác nếu có. Sau cùng, nếu thành phần *where* cuối cùng không được hệ thống nhận biết là một hay nhiều địa danh, nơi chốn nào đó qua hàm kiểm tra *IsValidWherePhrase* thì hệ thống sẽ xem như toàn bộ câu truy vấn là thành phần *what*.

3.4.2 Xác định ý nghĩa thành phần *where*:

Đây là giải thuật được áp dụng sau khi hệ thống đã biết được rõ ràng từng thành phần trong câu truy vấn. Giải thuật này nhằm giúp cho hệ thống nắm bắt được chính xác vùng không gian được đề cập đến trong thành phần *where* (vị trí địa lý hay vùng giới hạn của nơi đó, v.v…). Rõ ràng là công việc không hề đơn giản do các vấn đề về sự nhập nhằng ngữ nghĩa trong thành phần *where* khá phức tạp [14]. Từ đây, công việc đặt ra là cần có một phương pháp nhận biết tên địa danh và "hiểu" chính xác ý nghĩa của tên ấy cùng với sự hỗ trợ của một ontology địa lý như trong [6]. Để giải quyết vấn đề này thì B. Martins [9] cũng đã đề xuất một giải thuật với ý tưởng là trước tiên sẽ nhận biết các tên địa danh từ câu truy vấn, sau đó các tên địa danh ấy sẽ lần lượt được kết hợp với ontology địa lý để xem xét các mối quan hệ không gian giữa các địa danh tìm thấy. Kết quả sau cùng sẽ là địa danh mà hệ thống "hiểu" được từ câu truy vấn của người dùng. Tuy nhiên, thuật toán này chỉ xét đến các trường hợp mà tồn tại giữa các địa danh có ít nhất một mối quan hệ phân cấp trong không gian trong khi việc các địa danh không có mối quan hệ phân cấp không gian cũng là trường hợp không hiếm khi diễn ra. Và chính điều đó đã gây nên những trường hợp không thể nhận biết được chính xác thành phần *where* nếu như người dùng đề cập đến nhiều hơn một địa danh trong câu truy vấn (ví dụ tìm *"khách sạn ở Hà Nội, Thành Phố Hồ Chí Minh"*).

Để phân tích thành phần *where* đầy đủ hơn, luận văn xin trình bày một phương pháp cải tiến từ cách làm của B. Martins [9] nhằm khắc phục những thiếu xót đã đề cập. Sau

đây là đoạn mã giả trình bày ý tưởng thuật toán cải tiến, tạm gọi tên là *DisambiguateWhere*:

DisambiguateWhere(*OriginalWhere*)

Công cụ hỗ trợ: Ontology địa lý *O*.

Input: *OriginalWhere.*

Output: *AnalyzedWhere*.

lPlace := Danh sách các địa danh nhận biết được từ *OriginalWhere* qua *O*.

Chuẩn hóa *lPlace.*

For *i := 0* **To** *LengthOf(lPlace) - 1* **Do**

    If *lPlace[i]*.***IsAncestorOf****(lPlace[i+1])* **Then**

        AnalyzedWhere := *AnalyzedWhere*.Add(*lPlace[i+1]*).

        lPlace.Remove(*lPlace[i]*).

    Else If *lPlace[i]*.***IsDescendantOf****(lPlace[i+1])* **Then**

        AnalyzedWhere := *AnalyzedWhere*.Add(*lPlace[i]*).

        lPlace.Remove(*lPlace[i+1]*).

    Else // Không có quan hệ phân cấp

        AnalyzedWhere := *AnalyzedWhere*.Add(*lPlace[i]*).

        AnalyzedWhere := *AnalyzedWhere*.Add(*lPlace[i+1]*).

    End If

End For

Return *AnalyzedWhere*.

Giải thuật này như giải thuật của B. Martins cũng dùng để giải quyết các tình huống có nhiều hơn một tên địa danh được tìm thấy trong thành phần *where* của câu truy vấn, và

các tên địa danh có các mối quan hệ phân cấp với nhau (ví dụ: "Hoàn Kiếm, Hà Nội"). Ý tưởng ở đây là giải thuật sẽ phát hiện ra các thành phần trong *where* thuộc từng cấp khác nhau (ví dụ: "Hà Nội" thuộc cấp "Thủ đô/Thành phố/Tỉnh", "Hoàn Kiếm" thuộc cấp "Quận/Huyện"), sau đó giải thuật sẽ dựa vào ontology địa lý để kiểm tra mối quan hệ giữa các thành phần con vừa được phát hiện đó. Thông thường, giữa hai thành phần liền kề nhau, theo cách viết, cách gọi tên chung của người dùng thì sẽ có một thành phần là cha của thành phần còn lại, hay nói cách khác là sẽ có một thành phần mang nghĩa rộng hơn và một thành phần mang nghĩa cụ thể hơn, trong ví dụ trên thì ta có "Hà Nội" là cha của "Hoàn Kiếm" theo những khái niệm định nghĩa trong ontology địa lý. Và do đó, trong trường hợp như thế thì vùng địa danh được hệ thống nhận biết sẽ là vùng địa danh mang nghĩa cụ thể hơn và hệ thống sẽ sử dụng các khái niệm qui định trong ontology về địa danh đó để phục vụ cho các giai đoạn tìm kiếm tiếp theo. Trong trường hợp ví dụ "Hoàn Kiếm, Hà Nội" thì thuật toán sẽ nhận biết ý nghĩa thật sự là đề cập đến *"Quận Hoàn Kiếm"* thuộc *"Thủ Đô Hà Nội"* của nước *"Việt Nam"*.

Hơn thế, giải thuật này còn có thể giải quyết các trường hợp mà giải thuật trước không nhắc đến. Một trong các trường hợp đó dễ thấy là khi các tên địa danh nhận biết được không có mối quan hệ phân cấp với nhau. Ví dụ cho trường hợp này là câu truy vấn tìm "khách sạn ở Hà Nội, Thành Phố Hồ Chí Minh", với câu truy vấn này, thành phần *where* sẽ là *"Hà Nội, Thành Phố Hồ Chí Minh"* và sẽ được phân tích ra thành 2 địa danh là *"Thủ Đô Hà Nội"* và *"Thành Phố Hồ Chí Minh"*.

Một trong những bước quan trọng trong giải thuật cải tiến này cần lưu ý chính là quá trình chuẩn hóa danh sách các địa danh nhận biết được trước khi kết hợp với các khái niệm trong ontology để phân tích các tên địa danh đó. Danh sách tên các địa danh này thường được tạo thành từ những danh sách con tương ứng với từng cấp, ta có thể giả thiết là lP cho danh sách các tên địa danh ở cấp tỉnh thành được nhận biết, và tương tự là lD cho quận huyện, lW cho phường xã và lR cho đường xá. Lúc này, có 2 trường

hợp có thể xảy ra. Thứ nhất là khi các lP, lD, lW và lR giao nhau bằng rỗng, nghĩa là mọi tên địa danh đều trở nên rõ ràng giữa những phân cấp địa lý với nhau. Khi đó, điều duy nhất ta còn phân vân là liệu 1 tên địa danh nào đó có thể nói về bao nhiêu nơi khác nhau trong cùng 1 phân cấp địa lý (ví dụ *Châu Thành* là *Châu Thành* thuộc tỉnh nào?). Trong trường hợp này ta sẽ dựa vào thông tin của những địa danh đi bên cạnh nó (trước hoặc sau). Trường hợp thứ hai, rất hay xảy ra là các tập lP, lD, lW và lR có chung một vài giá trị, khi đó hệ thống cần phải có những tiêu chí đặc biệt để quyết định xem nên chọn giá trị trùng lắp đó từ tập nào. Ví dụ với "Hà Nội", ta sẽ có lP = {"Hà Nội"} (cho Thủ Đô Hà Nội), lD = {}, lW={}, lR = {"Hà Nội"} (cho đường xa lộ Hà Nội), lúc này ta sẽ có L = {"Hà Nội"} nhưng vấn đề là "Hà Nội" này thuộc lP hay lR? B. Martins [9] cũng có đề nghị một vài Heuristic để giải quyết các trường hợp này nhưng nếu áp dụng vào một hệ GIR Việt Nam thì vẫn chưa đủ hợp lý vì một vài đặc trưng trong cách dùng từ chỉ địa danh của người Việt Nam. Vì vậy, trong phần này giải thuật cải tiến sẽ kế thừa các Heuristic của B. Martins [9] và bổ sung thêm các Heuristic mới để phù hợp hơn khi áp dụng cho hệ GIR Việt Nam, đó là các Heuristic sau:

- Nếu một Thành phố, một Quận/Huyện, một Phường/Xã, một con đường có cùng một tên gọi thì khi người dùng đề cập đến tên gọi đó mà không mô tả rõ ràng nó thuộc cấp nào sẽ được hiểu ngầm định là cấp cao nhất.

- Tuy nhiên, trường hợp đặc biệt nếu một Phường/Xã và một con đường có cùng tên thì nếu không nói rõ sẽ được hiểu ngầm định là con đường. Đây là 1 Heuristic đặc trưng cho cách gọi tên địa danh của người Việt. Ví dụ ta có *"Phường Nguyễn Văn Cừ"* và *"đường Nguyễn Văn Cừ"* nhưng khi người ta nói ngắn gọn là *"ở Nguyễn Văn Cừ"* thì khả năng người ta muốn nói đến *"ở đường Nguyễn Văn Cừ"* cao hơn là *"ở phường Nguyễn Văn Cừ"*.

3.4.3 Xác định ý nghĩa thành phần *what*:

Một vấn đề khác trong phân tích truy vấn cũng đóng vai trò khá quan trọng ảnh hưởng đến tính hợp lý và mức độ thỏa mãn yêu cầu tìm kiếm của người dùng chính là hiểu nghĩa chủ đề cần tìm kiếm.

Nguyên lý cơ bản của bất kỳ một máy tìm kiếm nào trước tiên là sẽ dựa vào độ tương quan giữa các từ tìm kiếm và các tài liệu có chứa các từ đó, sau đó sẽ có một hàm đánh giá để xếp hạng các kết quả. Theo đó, khi hệ thống nhận được yêu cầu tìm kiếm "khách sạn New World", hệ thống sẽ tách truy vấn thành từng từ đơn và tìm kiếm trên những từ đơn đó, những kết quả có được sẽ bao gồm những tài liệu có đầy đủ 4 từ "khách", "sạn", "New", "World" hoặc những tài liệu chỉ cần có ít nhất một trong 4 từ đó (điều này còn tùy thuộc vào các toán tử tìm kiếm mà hệ thống lựa chọn – AND hoặc OR, v.v…). Rõ ràng với tập kết quả đó, người tìm kiếm sẽ cảm thấy thừa mà thiếu, thiếu mà thừa. Có một cách đơn giản để giải quyết vấn đề đó là dùng cặp dấu "" để giúp cho hệ thống tìm chính xác hơn, tuy nhiên nó lại gây nên cảm giác thiếu tự nhiên khi tìm kiếm, với lại không phải lúc nào đối tượng người sử dụng muốn tìm cũng là một cụm từ đi liền nhau để có thể sử dụng dấu nháy kép ("). Vậy để làm sao khi hệ thống nhận câu truy vấn "*khách sạn New World*" mà "hiểu" được nó và chỉ tìm đúng *khách sạn* có tên là *New World* chứ không tìm ra *khách sạn New Shanghai*, hoặc khi người dùng muốn tìm *"khách sạn New World, nhà hàng Sinh Đôi"* thì hệ thống sẽ tìm ra đúng những *khách sạn* có tên là *New World* và những *nhà hàng* có tên là *Sinh Đôi* chứ không có những kết quả ngoài ý muốn kèm theo. Đó là nhiệm vụ mà một hệ tìm kiếm cần làm tốt. Ở luận văn này, đối với hệ GIR Việt Nam thì thành phần phân tích nghĩa của *what* sẽ nhằm mục đích làm rõ nghĩa chủ đề mà người sử dụng cần tìm kiếm để kết quả sau cùng chính xác và hợp lý hơn.

Trong [5], có đề cập đến vấn đề từ có nghĩa có nhiều hơn một âm tiết trong Tiếng Việt trong quá trình lập chỉ mục nhằm hạn chế việc tách các từ có nghĩa có nhiều hơn một

âm tiết đó ra thành những từ đơn khi tìm kiếm. Với cách tiếp cận đó, ta có thể thấy rằng với *"khách sạn New World"* hệ thống chắc chắn sẽ tìm thấy những *"...khách sạn New World..." + những "..khách sạn..." + những "...sạn New..." + những "...New World..."* và có thể có thêm những *"...khách sạn New..." hay những "... sạn New World..."* tùy vào quá trình lập chỉ mục dựa trên nội dung sử dụng phương thức tách từ là bigram hay unigram. Ở một mức độ nào đó, ta thấy nó tốt hơn là phương pháp lập chỉ mục trên từng từ đơn và khi tìm kiếm các từ trong truy vấn được tách riêng biệt ra theo cách cơ bản của mọi máy tìm kiếm. Tuy nhiên, với cách làm này thì các kết quả tìm thấy vẫn còn có khả năng xuất hiện những kết quả "không liên quan" (ví dụ *"nhà hàng New World"* vẫn sẽ được tìm thấy trong khi người dùng muốn tìm *"khách sạn new World"*). Vậy nên đây cũng không phải là một cách tiếp cận tốt nếu áp dụng vào phân tích *what* cho hệ GIR.

Hơn nữa, trong mọi ngôn ngữ, hầu như luôn xuất hiện khái niệm về từ đồng nghĩa. Do đó đối với cùng một chủ đề tìm kiếm, người truy vấn có thể dùng những cách thể hiện khác nhau về mặt từ ngữ để yêu cầu hệ thống tìm kiếm. Điều này đặt ra một vấn đề là nếu có một tài liệu không chứa từ khóa mà người dùng tìm kiếm nhập vào nhưng về mặt chủ đề thì đó chính là cái mà người ta cần tìm thì tài liệu đó vẫn sẽ được tìm thấy hay không? Câu trả lời là nếu hệ thống không tìm thấy những tài liệu đó mà chỉ trả ra những tài liệu thật sự liên quan với từ khóa nhập vào thì hệ thống đó vẫn là một hệ thống hoạt động đúng logic cơ bản. Tuy nhiên, nếu hệ thống có thể trả ra những tài liệu không liên quan gì với từ khóa nhưng có chung chủ đề nội dung mà người dùng muốn tìm thì đó sẽ là một hệ thống thông minh hơn, tốt hơn. Với những yêu cầu ngày càng cao trong việc tìm kiếm thông tin của người dùng thì một hệ thống hoạt động thông minh hơn sẽ là mục tiêu hướng đến của các nhà xây dựng hệ thống.

Đồng thời, ý tưởng của tìm kiếm là sử dụng ngầm định những toán tử AND và OR giữa những từ có nghĩa trong tiếng Việt nhận biết được đã đặt ra những câu hỏi về việc

lúc nào dùng toán tử AND và lúc nào dùng toán tử OR. Trở lại với ví dụ *"khách sạn New World, nhà hàng Sinh Đôi"*, giả sử toán tử OR được áp dụng kết quả sẽ có thể có *khách sạn Sinh Đôi* hay *nhà hàng New World*, nhưng nếu toán tử AND được sử dụng thì có thể sẽ không có một kết quả nào được tìm thấy (trong khi sự thật sẽ có ít nhất là 2 kết quả). Vậy nên trong trường hợp trên, một toán tử OR sẽ được áp dụng giữa 2 đối tượng nhận biết được là *khách sạn New World* và *nhà hàng Sinh Đôi* trong khi toán tử AND sẽ được sử dụng bên trong từng đối tượng ấy ("khách sạn" AND "New World", "nhà hàng" AND "Sinh Đôi") sẽ hy vọng mang đến những kết quả tốt hơn.

Từ những vấn đề nhận thấy ở trên, luận văn đã đề nghị một cách tiếp cận đơn giản trong việc phân tích thành phần *what* của câu vấn nhằm mục đích tìm kiếm được những kết quả có tính hợp lý cao hơn. Giải thuật cũng sử dụng một cơ sở tri thức được tổ chức có dạng như WordNet [20] để tìm ra các mối quan hệ giữa các chủ đề tìm kiếm với nhau, ở đây chủ yếu là quan hệ từ đồng nghĩa [2] để mở rộng câu truy vấn nhằm tăng độ bao phủ (*Recall*) cho tập kết quả. Bên cạnh đó, thuật toán còn có sử dụng một số Heuristic trong khi nhận biết đối tượng/chủ đề được tìm kiếm, đó là:

- Đi sau các chủ đề tìm kiếm đã được nhận biết nếu không phải là một chủ đề tìm kiếm khác thì đó sẽ là thông tin nhằm làm rõ nghĩa cho chủ đề tìm kiếm trước đó (ví dụ: *khách sạn New World nhà hàng Sinh Đôi* thì theo sau *khách sạn* là *New World* có tác dụng làm rõ nghĩa của chủ đề *khách sạn*, và theo sau *nhà hàng* là *Sinh Đôi* có tác dụng làm rõ nghĩa cho *nhà hàng*). Trong những trường hợp như thế thì đối tượng tìm kiếm sẽ được nhận biết là *"khách sạn New World"* hay *"nhà hàng Sinh Đôi"* và chỉ 2 đối tượng đó thôi. Đây cũng là cách tiếp cận trong một số trường hợp mà *Google Maps* hay *Live Maps* lựa chọn.

- Nếu thuật giải không nhận biết được một chủ đề tìm kiếm nào thì sẽ xem như cả thành phần *what* chính là điều người sử dụng muốn tìm (ví dụ: *Novotel* nghĩa là người ta muốn tìm bất cứ cái gì liên quan đến *Novotel*).
- Nếu đứng trước một chủ đề tìm kiếm là một cụm từ không có chức năng bổ nghĩa cho một chủ đề nào khác thì cụm từ đó được xem là bổ nghĩa cho chủ đề đó (ví dụ: *5 sao khách sạn* có thể được hiểu là *khách sạn 5 sao*).

Sau đây là ý tưởng giải thuật:

DisambiguateWhat(*OriginalWhat*)

Công cụ hỗ trợ: Từ điển *D*, Danh sách từ đồng nghĩa có dạng như WordNet *S*.

Input: *OriginalWhat.*

Output: *AnalyzedWhat*.

lObj := Danh sách các chủ đề tìm kiếm nhận biết được từ *OriginalWhat* qua *D*.

Chuẩn hóa *lObj*.

Mở rộng *lObj* bằng danh sách *S*.

Idx := IndexOf(lObj[0])

If Idx > 0 **Then**

    lObj[0].Attribute.Add(Cụm từ đứng trước Idx)

End If

For *i := 0* **To** *LengthOf(lObj) - 1* **Do**

    delta := Khoảng cách giữa *lObj[i]* và *lObj[i+1]*.

    If delta <> 0 **Then**

        lObj[i].Attribute.Add(Chuỗi nằm giữa *lObj[i]* và *lObj[i+1]*).

    End If

End For

If i=0 **Then**

lObj.Add(*OriginalWhere*).

Else If *OriginalWhat* không kết thúc bằng *lObj[i]* **Then**

lObj[i].Attribute.Add(Chuỗi còn lại của *OriginalWhat* sau *lObj[i]*).

End If

AnalyzedWhat := *lObj*.

Return *AnalyzedWhat*.

3.5 Đánh giá các giải thuật:

Các giải thuật cải tiến, các heuristic bổ sung ở trên đều xuất phát từ những cách tiếp cận rất đơn giản nhưng hiệu quả để giải quyết bài toán phân tích câu truy vấn, phân tích các thành phần trong câu truy vấn của hệ GIR mà những nghiên cứu trước đó trên thế giới đã chứng minh. Độ chính xác của kết quả phân tích sẽ phụ thuộc vào mức độ đầy đủ các khái niệm trong ontology địa lý mà giải thuật sử dụng, hơn nữa nó còn phụ thuộc vào các quan điểm chủ quan của người xây dựng hệ thống và quan điểm trong tìm kiếm thông tin của từng người sử dụng. Vì đây là một dạng hệ thống mang tính phổ thông nên việc đánh giá nó như thế nào là rất khó khăn. Vì giới hạn thời gian cũng như là dữ liệu nên luận văn không có được một bộ truy vấn vài trăm, vài nghìn câu, nhưng chỉ với 8 câu truy vấn từ bảng khảo sát 3-1 cũng đủ cho ta thấy một điều rằng giữa những hệ thống khác nhau chưa có một tiêu chuẩn chung nào để thống nhất việc phân tích các thành phần của câu truy vấn. Với câu truy vấn đầu tiên, tại sao Google Maps phân tích thành <*Hotel, Near, New York*> mà không phải là <*New York Hotel,,*> theo đúng ý nghĩa tự nhiên về mặt ngôn ngữ, cú pháp của câu truy vấn, v.v… là một dấu hỏi lớn. Và nếu *Live Maps* phân tích câu truy vấn (1) giống với *Google Maps* thì ở câu truy

vấn (2) với cấu trúc giống như (1) thì *Live Maps* lại phân tích hoàn toàn khác và lúc này có vẻ như là phù hợp hơn nếu xét ở khía cạnh ngôn ngữ tự nhiên (*Google Maps* phân tích (2) cũng giống như (1)). Với câu truy vấn (3), *Google Maps* có vẻ như là cùng cách tiếp cận với B. Martins nhưng *Live Maps* thì ngược lại và không thể nào lý giải được tại sao kết quả phân tích câu *Hotel in Paris, Lyon* của *Live Maps* là <*Hotel Lyon, Near, Paris*> trong khi với truy vấn (4) có cấu trúc hoàn toàn giống với (3) thì *Live Maps* lại phân tích giống với *Google Maps* (chỉ khác ở phần phân tích cụ thể thành phần *where* lúc sau). Tương tự, trong câu truy vấn (5)(6)(7)(8), ta có thể một lần nữa thấy rằng cách tiếp cận của *Google Maps* và *Live Maps* là hoàn toàn khác nhau. Vậy câu hỏi đặt ra là với những sự khác nhau được nêu ra trên đây thì cách tiếp cận của hệ thống nào tốt hơn hệ thống nào? Hệ thống nào giải quyết vấn đề tốt hơn? Và liệu có một cơ sở nào để đánh giá việc phân tích truy vấn hay không? Câu trả lời là cho đến thời điểm này vẫn chưa có một độ đo nào được đề nghị để đánh giá tính đúng đắn của một cách tiếp cận trong phân tích truy vấn. Việc đánh giá một phương pháp phân tích truy vấn trong bài toán này vẫn phụ thuộc vào cảm tính và quan điểm tìm kiếm của từng người dùng trong hệ thống là chính. Nếu quan điểm tìm kiếm ấy trùng khớp với quan điểm mà hệ thống lựa chọn thì hệ thống sẽ được người dùng đó đánh giá là hệ thống phân tích *đúng* (vì đúng cách nghĩ của họ), ngược lại nếu quan điểm tìm kiếm không trùng khớp với quan điểm của hệ thống thì người dùng đó sẽ không thể hài lòng với kết quả mà hệ thống phân tích được, khi đó hệ thống được đánh giá là phân tích *sai*. Và như thế một hệ thống luôn ở trong hai trạng thái *đúng* và *sai* tùy theo chủ quan người đánh giá. Đối với các thuật giải phân tích truy vấn đề nghị trong luận văn này thì việc đánh giá lại càng trở nên khó khăn hơn vì đây là lần đầu tiên một cách tiếp cận trong vấn đề này đối với tiếng Việt được tiến hành. Chúng tôi không tìm thấy những hệ thống tương tự cũng như những công trình tương tự làm cho hệ GIR tiếng Việt để đánh giá sự tương quan trong khi không thể nào so sánh với những hệ thống lớn trên thế giới làm cho tiếng nước ngoài (mà ngay cả giữa những hệ thống lớn đó cũng đang tồn tại

rất nhiều mâu thuẫn), cũng như là không thể sử dụng những bộ dữ liệu test của những tổ chức uy tín chuyên đánh giá về những hệ thống tìm kiếm trên thế giới như TREC hay GeoCLEF.

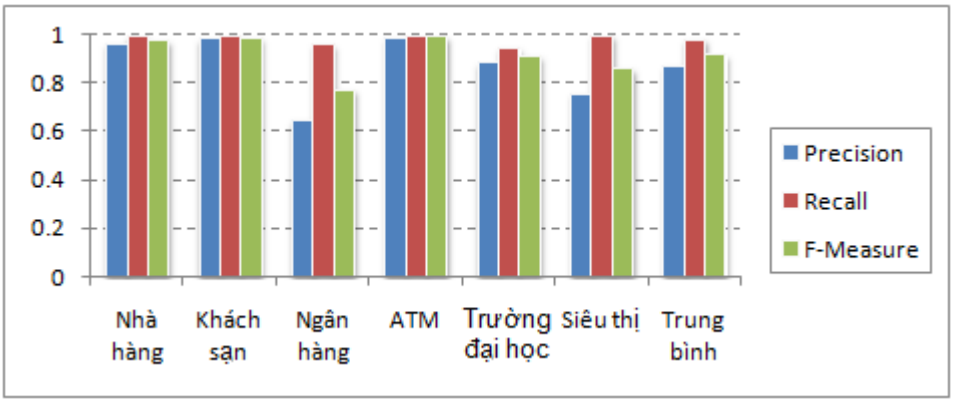
Tuy nhiên, để có một đánh giá tổng quát về khả năng vận hành của giải thuật đề nghị này, hệ thống đã chủ động sử dụng *query log* của trang web tìm kiếm Vietbando [23] cung cấp để kiểm tra. Query log này sau khi loại bỏ các truy vấn giống nhau cũng như là số các truy vấn tìm kiếm địa chỉ thì số lượng truy vấn còn lại là 13858 câu truy vấn do người sử dụng trang web vietbando nhập vào với mục đích tìm kiếm các thông tin về các địa danh hoặc vị trí các điểm dịch vụ ở trên toàn lãnh thổ Việt Nam. Với bộ dữ liệu đó, giải thuật đề nghị đã phân tích được 17% số câu truy vấn có đầy đủ 3 thành phần là <*what, rel, where*>, 38% số câu truy vấn chỉ có thành phần *what*, 44% số câu truy vấn chỉ có thành phần *where* và 1% là số câu truy vấn mà giải thuật phân tích sai (ví dụ như truy vấn với từ chỉ quan hệ không gian không có trong danh sách mà giải thuật sử dụng). Về vấn đề này chúng ta cũng cần phải chú ý rằng, giải thuật chỉ làm công việc phân tích ra các thành phần, làm rõ các thành phần, nhưng các thành phần đó có đúng với ý đồ của người tìm kiếm hay không lại là chuyện khác (như các ví dụ trong bảng 3-1 bên trên). Do đó ở đây vẫn chưa thể đánh giá cách tiếp cận đơn giản này có mức độ hiệu quả (về khía cạnh kết quả phân tích) như thế nào vì việc đánh giá nó cần một quá trình trong trải nghiệm thực tế. Và một khi vẫn chưa có cơ sở rõ ràng nào để khẳng định một cách tiếp cận được cho là hiệu quả hay không hiệu quả thì những cách tiếp cận đơn giản, chi phí xử lý thấp, xử lý được đa số những câu truy vấn mà hệ thống thường xuyên nhận được sẽ là ưu tiên lựa chọn trong những hệ thống đòi hỏi tốc độ như các hệ tìm kiếm thông tin.

Ngoài ra, luận văn cũng đã sử dụng bộ dữ liệu về các POIs (Point of Interested) của Vietbando [23] để đánh giá các độ đo *Precision, Recall* và *F-Measure* của các kết quả tìm kiếm được sau khi thực hiện phân tích, mở rộng câu truy vấn theo các giải thuật đề

nghị ở trên. Bộ dữ liệu thử nghiệm gồm 200.000 điểm đối tượng với nhiều chủ đề nội dung khác nhau, tuy nhiên vì số lượng chủ đề khá lớn nên luận văn chỉ xin trình bày kết quả thử nghiệm với 6 chủ đề trong POIs mà người sử dụng thường xuyên truy vấn trong hệ thống của Vietbando nhất là: *nhà hàng, khách sạn, ngân hàng, ATM, trường đại học, siêu thị*. Kết quả được thể hiện trong bảng dưới đây:

	Đối tượng liên quan	Kết quả tìm thấy	Đối tượng liên quan trong kết quả	Precision	Recall	F-Measure
Nhà hàng	2250	2312	2235	0.9666955	0.993333333	0.979833406
Khách sạn	2873	2906	2861	0.9845148	0.995823181	0.990136702
Ngân hàng	2739	4067	2637	0.64838948	0.962760131	0.774904496
ATM	1444	1459	1444	0.98971899	1	0.994832931
Trường đại học	330	354	313	0.88418079	0.948484848	0.915204678
Siêu thị	234	309	234	0.75728155	1	0.861878453
Trung bình				0.87179685	0.983400249	0.919465111

Bảng 3-2: Bảng thống kê độ đo Precision - Recall của kết quả tìm kiếm.

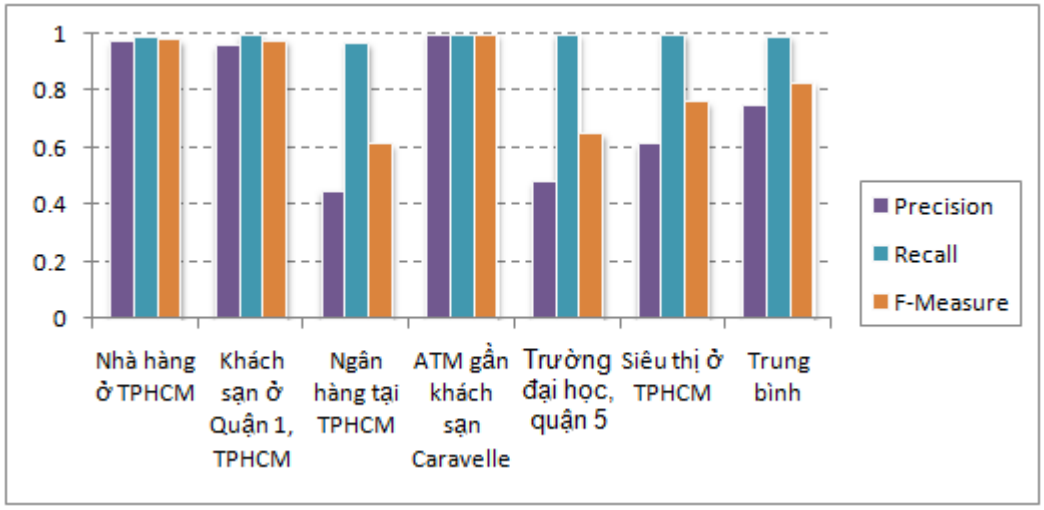


Hình 3-2: Biểu đồ thể hiện các độ đo Precision-Recall của kết quả tìm kiếm.

Còn dưới đây là kết quả thực nghiệm với 6 chủ đề tìm kiếm trên trong những vùng không gian xác định:

	Đối tượng liên quan	Kết quả tìm thấy	Đối tượng liên quan trong kết quả	Precision	Recall	F-Measure
Nhà hàng ở TPHCM	1044	1060	1031	0.972642	0.987548	0.980038
Khách sạn ở Quận 1, TPHCM	380	393	378	0.961832	0.994737	0.978008
Ngân hàng tại TPHCM	890	1916	863	0.450418	0.969663	0.61511
ATM gần khách sạn Caravelle	263	263	263	1	1	1
Trường đại học, quận 5	14	29	14	0.482759	1	0.651163
Siêu thị ở TPHCM	118	192	118	0.614583	1	0.76129
Trung bình				0.747039	0.991991	0.830935

Bảng 3-3: Bảng thống kê độ đo Precision - Recall của kết quả tìm kiếm theo điều kiện.



Hình 3-3: Biểu đồ thể hiện các độ đo Precision-Recall của kết quả tìm kiếm theo điều kiện.