

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

*



BÁO CÁO LẦN MỘT
XỬ LÝ NGÔN NGỮ TỰ
NHIÊN

SPEECH AND LANGUAGE PROCESSING
CHƯƠNG 21, 22, 27, 28 VÀ 29

GVHD: TS. NGUYỄN KIÊM HIẾU

—o0o—

HVTH: CAO MẠNH HẢI - CB190206

HÀ NỘI, 7/2020

Mục lục

1 Information Extraction (IE) (Chapter 21)[1]	1
1.1 Named Entity Recognition (NER) (Nhận diện thực thể có tên)	1
1.1.1 NER as Sequence Labeling (Nhận diện thực thể có tên như việc gán nhãn tuần tự)	2
1.1.2 Evaluation of Named Entity Recognition (Dánh giá hệ thống nhận diện thực thể có tên)	2
1.1.3 Practical NER Architectures (Các kiến trúc NER thực tế)	2
1.2 Relation Extraction (Trích xuất quan hệ)	3
1.2.1 Using Patterns to extract relations (Việc sử dụng các mẫu để trích xuất các quan hệ)	3
1.2.2 Relation Extraction via Supervised Learning (Trích xuất quan hệ thông qua việc học có giám sát)	4
1.2.3 Semisupervised Relation Extraction via Bootstrapping (Trích xuất quan hệ bán giám sát thông qua Bootstrapping)	5
1.2.4 Distant Supervision for Relation Extraction	6
1.2.5 Unsupervised Relation Extraction (Trích xuất quan hệ không giám sát)	7
1.2.6 Evaluation of Relation Extraction (Dánh giá hệ thống trích xuất quan hệ)	7
1.3 Extracting Times	8
1.3.1 Temporal Expression Extraction	8
1.3.2 Temporal Normalization	9
1.4 Extracting Events and their Times	9
1.4.1 Temporal Ordering of Events	10
1.5 Template Filling	10
1.5.1 Statistical Approaches to Template Filling	10
1.5.2 Earlier Fine-State Template-Filling Systems	11
2 Semantic Role Labeling (Chapter 22)[1]	12
2.1 Semantic Roles	12
2.2 Diathesis Alternations	13
2.3 Semantic Roles: Problems with Thematic Roles	15
2.4 The Proposition Bank	15
2.5 FrameNet	15
2.6 Semantic Role Labeling (SRL)	16
3 Question Answering (QA) (Chapter 27)[1]	18
3.1 IR-based Factoid QA	18

3.1.1	Question Processing	18
3.1.2	Answer type detection (Question Classification)	19
3.1.3	Query Formulation	19
3.1.4	Passage Retrieval	19
3.1.5	Answer Processing	20
3.2	Knowledge-based Question Answering	21
3.2.1	Rule-based Methods	21
3.2.2	Supervised Methods	21
3.2.3	Dealing with variation: Semi-Supervised Methods	22
3.3	Evaluation of Factoid Answer	22
4	Dialog Systems and Chatbots (Chater 28)[1]	23
4.1	Chatbots	23
4.1.1	Rule-based chatbots: ELIZA và PARRY	24
4.1.2	Corpus-based chatbots	24
4.2	Frame Based Dialog Agents	26
4.2.1	Control structure for frame-based dialog	26
4.2.2	Natural language understanding for filling slots	27
4.2.3	Evaluating Slot Filling	29
4.2.4	Other components of frame-based dialog	30
4.3	Evaluating Dialogue Systems	30
5	Advanced Dialog Systems (Chapter 29)[1]	31
5.1	Dialog Acts	32
5.2	Dialog State: Interpreting Dialog Acts	35
5.2.1	Sketching an algorithm for dialog act interpretation	36
5.2.2	A special case: detecting correction acts	37
5.3	Dialogue Policy	37
5.3.1	Generating Dialogue Acts: Confirmation and Rejection	37
5.4	Natural language generation in the dialog-state model	38
5.5	Advanced: Markow Decision Processes	39

Danh sách hình vẽ

1.1	Các đặc trưng được sử dụng phổ biến trong việc huấn luyện các hệ thống nhận dạng thực thể có tên	2
1.2	Ví dụ về lexico-syntactic patterns	3
1.3	Các mẫu lexico-syntactic được xây dựng bằng tay cho việc tìm kiếm hypernyms	4
1.4	Mẫu của các đặc trưng đã trích rút trong quá trình phân loại của <American Airlines, <Tim Wager> tuple	5
1.5	Bootstrapping from seed entity pairs to learn relations	5
1.6	Công thức tính độ tin cậy	6
1.7	The distant supervision algorithm for relation extraction	6
1.8	Công thức tính Precision	8
1.9	Examples of absolute, relational and durational temporal expressions	8
1.10	Ví dụ về cách đánh nhãn IBO	8
1.11	Typical features used to train IBO-style temporal expression taggers	9
1.12	Ví dụ về event extraction	9
2.1	Some commonly used thematic roles with their definitions	12
2.2	Ví dụ về một đại diện sự kiện	13
2.3	Ví dụ 1	14
2.4	Ví dụ 2	14
2.5	Ví dụ 3	14
2.6	Ví dụ 4	15
2.7	The frame elements in the change-position-on-a-scale frame from the FrameNet Labelers Guide	16
2.8	A generic semantic-role-labeling algorithm	17
3.1	Ví dụ, hệ thống có thể có các chuỗi cho từng thực thể trong hệ thống (Texas, Ada Lovelace)	21
3.2	Công thức tính MRR	22
4.1	A simplified sketch of the ELIZA algorithm. The power of the algorithm come from the particular transforms associated with each keyword	24
4.2	A simple finite-state automaton architecture for frame-based dialog	27
4.3	Phía bên trái của mỗi luật tương ứng với các thực thể ngữ nghĩa	28
4.4	A semantic grammar parse for a user sentence, using slot names as the internal parse tree nodes	28
4.5	An LSTM architecture for slot filling, mapping the words in the input (represented as 1-hot vectors or as embeddings) to a series of IBO tags plus a final state consisting of a domain concatenated with an intent	29
4.6	Công thức tính Slot Error Rate for a Sentence	30

5.1	Architecture of a dialog-state system for task-oriented dialog	32
5.2	Dialog Acts	32
5.3	Sự liên tục của các phương thức mà người nghe B có thể sử dụng để nói lên cách nói của người nói, được sắp xếp từ yếu đến mạnh	33
5.4	The 18 high-level dialog acts for a meeting scheduling task	34
5.5	Dialogue acts used by the HIR restaurant recommendation system	34
5.6	A sample dialog from the HIS system using the dialog acts in Fig. 5.4	35
5.7	Ví dụ, cho thấy đầu ra yêu cầu của dialog-state tracker sau mỗi lượt	35
5.8	Dạng declarative surface của Checks	36
5.9	Một số đặc trưng tiêu chuẩn cho việc phát hiện các correction acts	37
5.10	Ví dụ về chiến lược progressive prompting	38
5.11	An input frame to NLG and a resulting output sentence, in the Communicator system	38

Chương 1

Information Extraction (IE) (Chapter 21)[1]

Quá trình trích rút thông tin là quá trình biến thông tin phi cấu trúc được nhúng trong các văn bản thành dữ liệu có cấu trúc.

Bước đầu tiên của hầu hết các nhiệm vụ IE là để tìm các tên riêng hoặc các thực thể có tên được đề cập trong văn bản. Những gì cấu thành một loại tên thực thể thường bao gồm: con người, các địa điểm, và các tổ chức nhưng cũng có các thực thể cụ thể hơn từ tên của các loại gen và các protein cho đến tên của các khóa học.

Nhiệm vụ của việc trích rút quan hệ là để tìm và phân loại các quan hệ ngữ nghĩa giữa các thực thể trong văn bản, thường là các quan hệ nhị phân như “vợ của” (spouse-of), “child-of”,... và các quan hệ về không gian địa lý. Trích xuất quan hệ có kết nối chặt chẽ để điền vào Cơ sở dữ liệu (CSDL).

Nhiệm vụ của trích xuất sự kiện (event extraction) là để tìm các sự kiện trong đó các thực thể này tham gia. Chúng ta cũng cần phải thực hiện sự kiện cốt lõi (event coreference) để tìm ra cái nào trong số nhiều sự kiện trong một văn bản đề cập đến cùng một sự kiện.

Để biết khi nào các sự kiện trong văn bản xảy ra, chúng ta sẽ nhận diện các biểu thức về thời gian (temporal expressions) như các ngày trong tuần (thứ 5 và thứ 6), các tháng, các ngày lễ,... các biểu thức tương đương như “two days from now” hoặc “next year” và các thời gian như là: 3:30pm hoặc “noon”. Vấn đề của chuẩn hóa biểu thức thời gian (temporal expression normalization) là ánh xạ những biểu thức thời gian này vào các ngày hoặc thời gian cụ thể trong ngày để xác định các sự kiện theo thời gian.

Cuối cùng, nhiều văn bản mô tả các tình huống rập khuôn định kỳ, nhiệm vụ của “template filling” là để tìm các tình huống như vậy trong các tài liệu và điền vào các vị trí mẫu bằng thông tin phù hợp. Các “slot-filler” có thể bao gồm các đoạn văn bản được trích xuất trực tiếp từ văn bản.

1.1 Named Entity Recognition (NER) (Nhận diện thực thể có tên)

Một thực thể có tên, nói một cách đại khái là bất cứ cái gì có thể gọi bằng một tên thích hợp như: một người, một địa điểm, một tổ chức. Thuật ngữ này thường được mở rộng để bao gồm: ngày, tháng, thời gian, và các loại biểu thức thời gian khác, hoặc thậm chí bao gồm các biểu thức số học như: giá cả,...

Ngoài việc sử dụng NER trong việc trích xuất các sự kiện và mối quan hệ giữa chúng, các thực thể có tên còn hữu ích cho nhiều tác vụ NLP khác. NER có nghĩa là tìm các khoảng văn bản mà tạo thành tên riêng và sau đó phân loại nó vào loại thực thể tương ứng. Nhận diện là một phần khó khăn vì sự mơ hồ của đoạn (chúng ta phải quyết định cái gì là một thực thể, cái gì không và ranh giới của chúng ở đâu). Khó khăn nữa được gây ra bởi sự mơ hồ về loại thực thể.

1.1.1 NER as Sequence Labeling (Nhận diện thực thể có tên như việc gán nhãn tuần tự)

Thuật toán tiêu chuẩn để nhận dạng thực thể có tên là một tác vụ ghi nhãn theo thứ tự “word-by-word”. Trong đó, các thực thể được gán bắt được cả ranh giới và loại thực thể. Một trình phân loại tuần tự được huấn luyện để gán nhãn trong một văn bản với các thẻ mà chỉ ra sự hiện diện của các loại thực thể được đặt tên cụ thể.

Mã hóa dữ liệu huấn luyện bằng các thẻ IOB (Inside-Outside-Begin), ví dụ: [ORG (tag) American (B-ORG) Airlines (I-ORG) is (O)]. Bước tiếp theo là cho một bộ các đặc trưng để liên kết với từng token đầu vào.

identity of w_i
identity of neighboring words
part of speech of w_i
part of speech of neighboring words
base-phrase syntactic chunk label of w_i and neighboring words
presence of w_i in a gazetteer
w_i contains a particular prefix (from all prefixes of length ≤ 4)
w_i contains a particular suffix (from all suffixes of length ≤ 4)
w_i is all upper case
word shape of w_i
word shape of neighboring words
short word shape of w_i
short word shape of neighboring words
presence of hyphen

Hình 1.1: Các đặc trưng được sử dụng phổ biến trong việc huấn luyện các hệ thống nhận dạng thực thể có tên

Tính hữu dụng tương đối của bất kỳ đặc trưng hoặc sự kết hợp các đặc trưng này phụ thuộc rất nhiều vào mỗi ứng dụng, thể loại, ngôn ngữ và cách mã hóa văn bản.

1.1.2 Evaluation of Named Entity Recognition (Đánh giá hệ thống nhận diện thực thể có tên)

Để đánh giá một hệ thống NER, các độ đo quen thuộc được sử dụng: Recall, Precision, F1.

1.1.3 Practical NER Architectures (Các kiến trúc NER thực tế)

Trong khi các mô hình tuần tự thống kê thuần túy là tiêu chuẩn trong nghiên cứu học thuật thì các phương pháp mang tính thương mại cho bài toán NER thường dựa trên

sự kết hợp thực tế của các danh sách các luật và học máy có giám sát. Một hướng tiếp cận chung là thực hiện nhiều lần truyền qua một văn bản, cho phép kết quả của một lần truyền qua ảnh hưởng đến lần tiếp theo:

- Đầu tiên, sử dụng các luật với “precision” cao để gán nhãn cho các thực thể không rõ ràng.
- Tìm kiếm các “substring” trùng khớp của các “tên” đã được phát hiện trước đó.
- Tham khảo danh sách “tên” dành riêng cho ứng dụng để xác định các thực thể có khả năng từ “domain” đã cho.
- Áp dụng các kỹ thuật gán nhãn tuần tự theo xác suất mà sử dụng các thẻ từ giai đoạn trước làm các đặc trưng bổ sung.

1.2 Relation Extraction (Trích xuất quan hệ)

Có bốn thuật toán chính để trích xuất quan hệ: các mẫu viết tay (hand-written patterns); học có giám sát (supervised learning); học bán giám sát (semi-supervised learning); và học không giám sát (unsupervised learning).

1.2.1 Using Patterns to extract relations (Việc sử dụng các mẫu để trích xuất các quan hệ)

Thuật toán phổ biến nhất cho việc trích xuất quan hệ là việc sử dụng các mẫu cú pháp từ vựng (lexico-syntactic patterns):

$$NP_0 \text{ such as } NP_1\{, NP_2 \dots, (\text{and}|\text{or})NP_i\}, i \geq 1$$

implies the following semantics

$$\forall NP_i, i \geq 1, \text{hyponym}(NP_i, NP_0)$$

allowing us to infer

$$\text{hyponym}(\text{Gelidium, red algae})$$

Hình 1.2: Ví dụ về lexico-syntactic patterns

Trong đó, hyponym là một thuật ngữ dùng để chỉ một thành viên đặc biệt của một lớp rộng lớn hơn. Ví dụ, hoa cúc và hoa hồng là hyponyms của hoa.

Hình 1.3 mô tả chi tiết các mẫu lexico-syntactic được xây dựng bằng tay cho việc tìm các hypernym.

$NP \{, NP\}^* \{,\}$ (and or) other NP_H	temples, treasures, and other important civic buildings
NP_H such as $\{NP,\}^* \{(or and)\} NP$	red algae such as <i>Gelidium</i>
such NP_H as $\{NP,\}^* \{(or and)\} NP$	such authors as <i>Herrick, Goldsmith, and Shakespeare</i>
$NP_H \{,\}$ including $\{NP,\}^* \{(or and)\} NP$	common-law countries , including Canada and England
$NP_H \{,\}$ especially $\{NP,\}^* \{(or and)\} NP$	European countries , especially France, England, and Spain

Hình 1.3: Các mẫu lexico-syntactic được xây dựng bằng tay cho việc tìm kiếm hypernyms

Các phiên bản hiện đại của cách tiếp cận dựa trên mẫu được mở rộng bằng cách thêm các ràng buộc về thực thể có tên. Ví dụ: PER(named|appointed|chose|etc.)PER Prep?POSITION => Truman appointed Marshall Secretary of State.

Các mẫu được xây dựng bằng tay có lợi thế về “precision” cao và chúng có “recall” thấp và chúng có thể được điều chỉnh theo các miền cụ thể. Mặc khác, chúng có recall thấp nên sẽ mất nhiều công sức để tạo ra tất cả các mẫu có thể.

1.2.2 Relation Extraction via Supervised Learning (Trích xuất quan hệ thông qua việc học có giám sát)

Một tập hợp quan hệ và thực thể cố định được chọn, một tập huấn luyện được chú thích bằng tay với các quan hệ, thực thể và các văn bản chú thích sau đó được sử dụng để huấn luyện các trình phân loại để sau đó chú thích một tập kiểm thử.

Cách tiếp cận đơn giản nhất có 3 bước. Bước 1 là tìm các cặp thực thể có tên (thường trong cùng 1 câu). Trong bước 2, một bộ phân loại “filtering” được huấn luyện đưa ra các quyết định nhị phân về việc một cặp thực thể có tên nhất định có liên quan đến nhau hay không (bởi bất kỳ mối quan hệ nào). Các mẫu tích cực được trích xuất trực tiếp từ tất cả các quan hệ trong kho văn bản có chú thích, và các mẫu tiêu cực được tạo ra từ các cặp thực thể trong câu mà không được chú thích với bất kỳ một quan hệ nào. Trong bước 3, 1 lớp phân loại được huấn luyện để gán 1 nhãn tới quan hệ mà đã tìm được trong bước 2. Việc sử dụng bộ phân loại “filtering” có thể tăng tốc độ phân loại cuối cùng và cũng cho phép sử dụng các bộ đặc trưng riêng biệt phù hợp cho từng tác vụ. Đối với mỗi loại trong 2 bộ phân loại, chúng ta có thể sử dụng bất kỳ các kỹ thuật phân loại tiêu chuẩn nào (Logistic Regression, Support Vector Machine,...).

Như với NER, bước quan trọng nhất trong quá trình này là để xây dựng các đặc trưng bề mặt hữu ích cho phân loại quan hệ:

M1 headword	<i>airlines</i>
M2 headword	<i>Wagner</i>
Word(s) before M1	NONE
Word(s) after M2	<i>said</i>
Bag of words between	{ <i>a, unit, of, AMR, Inc., immediately, matched, the, move, spokesman</i> }
M1 type	ORG
M2 type	PERS
Concatenated types	ORG-PERS
Constituent path	$NP \uparrow NP \uparrow S \uparrow S \downarrow NP$
Base phrase path	$NP \rightarrow NP \rightarrow PP \rightarrow NP \rightarrow VP \rightarrow NP \rightarrow NP$
Typed-dependency path	<i>Airlines</i> \leftarrow_{subj} <i>matched</i> \leftarrow_{comp} <i>said</i> \rightarrow_{subj} <i>Wagner</i>

Hình 1.4: Mẫu của các đặc trưng đã trích rút trong quá trình phân loại của <American Airlines, <Tim Wager> tuple

Cuối cùng, cấu trúc ngữ pháp của một câu có thể báo hiệu nhiều mối quan hệ giữa các thực thể của nó. 1 cách đơn giản và hiệu quả để thực hiện cấu trúc là sử dụng các chuỗi biểu thị các đường cú pháp (syntactic paths).

Các hệ thống có giám sát có thể có được độ chính xác cao với đủ dữ liệu huấn luyện được gán nhãn bằng tay, nếu bộ kiểm thử đủ tương tự với tập huấn luyện. Nhưng việc gán nhãn cho một bộ dữ liệu huấn luyện lớn thì cực kỳ tốn kém và các mô hình giám sát rất dễ vỡ, chúng không thể khai quát tốt cho các thể loại khác nhau.

1.2.3 Semisupervised Relation Extraction via Bootstrapping (Trích xuất quan hệ bán giám sát thông qua Bootstrapping)

(Bootstrapping thường đề cập đến một quá trình tự khởi động, tiến hành mà không cần đầu vào bên ngoài)

Giả sử ta chỉ có một vài “seed patterns” có độ chính xác cao, hoặc có lẽ một vài “seed tuples”. Điều đó là đủ để khởi động trình phân loại. Bootstrapping tiến hành bằng cách lấy các thực thể trong “seed pair”, sau đó tìm các câu (tìm trên web hoặc bất kỳ tập dữ liệu nào đang sử dụng) có chứa cả 2 thực thể. Từ tất cả các câu như vậy, chúng ta trích rút và khai quát hóa ngữ cảnh xung quanh các thực thể để tìm các pattern mới.

```

function BOOTSTRAP(Relation R) returns new relation tuples
    tuples  $\leftarrow$  Gather a set of seed tuples that have relation R
    iterate
        sentences  $\leftarrow$  find sentences that contain entities in seeds
        patterns  $\leftarrow$  generalize the context between and around entities in sentences
        newpairs  $\leftarrow$  use patterns to grep for more tuples
        newpairs  $\leftarrow$  newpairs with high confidence
        tuples  $\leftarrow$  tuples + newpairs
    return tuples

```

Hình 1.5: Bootstrapping from seed entity pairs to learn relations

Các hệ thống Bootstrapping cũng gán các “confidence value” đến các tuple mới để tránh “semantic drift”. Trong “semantic”, một mâu sai lầm sẽ dẫn đến sự ra đời của các mâu có vấn đề và ý nghĩa của các mối quan hệ được trích rút “drift”.

Các giá trị độ tin cậy cho các mâu dựa trên việc cân bằng 2 yếu tố: Hiệu suất của mâu với bộ dữ liệu hiện tại và năng suất của mâu về mặt số lượng so khớp mà nó tạo ra trong bộ sưu tập tài liệu. Chính thức hơn, cho một bộ sưu tập tài liệu D, một bộ tuples T hiện tại và một mâu P được đề xuất, chúng ta cần theo dõi hai yếu tố: “hits” - bộ tuples trong T khớp với khi tìm kiếm trong D; “finds” - tổng số tuple mà P tìm thấy trong D.

$$Conf_{RlogF}(p) = \frac{hits_p}{finds_p} \times log(finds_p)$$

Hình 1.6: Công thức tính độ tin cậy

Đặt ngưỡng tin cậy cho việc chấp nhận các mâu và bộ dữ liệu mới trong quá trình Bootstrapping giúp ngăn hệ thống tránh được “drift” xa khỏi mối quan hệ mục tiêu.

1.2.4 Distant Supervision for Relation Extraction

Phương thức “distant supervision” kết hợp lợi thế của Bootstrapping với việc học có giám sát. Thay vì chỉ một số ít “seeds”, “distant supervision” sử dụng một CSDL lớn để thu được một số lượng lớn các ví dụ “seed”, tạo ra nhiều đặc trưng từ tất cả các ví dụ này và sau đó kết hợp chúng trong một bộ phân loại có giám sát.

```
function DISTANT SUPERVISION(Database D, Text T) returns relation classifier C
foreach relation R
    foreach tuple (e1, e2) of entities with relation R in D
        sentences  $\leftarrow$  Sentences in T that contain e1 and e2
        f  $\leftarrow$  Frequent features in sentences
        observations  $\leftarrow$  observations + new training tuple (e1, e2, f, R)
    C  $\leftarrow$  Train supervised classifier on observations
    return C
```

Hình 1.7: The distant supervision algorithm for relation extraction

Thuật toán chia sẻ lợi thế với từng phương pháp mà chúng ta đã xem xét. Giống như phân loại có giám sát, “distant supervision” sử dụng một bộ phân loại có nhiều đặc trưng và được giám sát bởi tri thức được tạo bằng tay. Nhưng “distant supervision” chỉ có thể giúp trích xuất các mối quan hệ đã tồn tại trong một CSDL đủ lớn.

1.2.5 Unsupervised Relation Extraction (Trích xuất quan hệ không giám sát)

Còn được gọi là “Open Information Extraction” hoặc OpenIE. Trong OpenIE, các quan hệ là các chuỗi đơn giản của các từ (thường bắt đầu bằng một động từ).

Hệ thống Reverb: Trích xuất một quan hệ từ một câu S theo 4 bước:

- Chạy bộ gán nhãn POS và “entity chunker” trên S.
- Cho mỗi động từ trong S, tìm chuỗi từ dài nhất w mà bắt đầu với một động từ và thỏa mãn các ràng buộc về ngữ pháp và từ vựng, hợp nhất các “adjacent (liền kề) matches”.
- Đối với mỗi cụm w, tìm cụm danh từ gần nhất x ở bên trái mà không phải là đại từ quan hệ, wh_word hoặc “existential” (there). Tìm các cụm danh từ gần nhất y từ bên phải.
- Gán độ tin cậy c cho quan hệ $r = (x, w, y)$ bằng cách sử dụng trình phân loại độ tin cậy và trả về kết quả.

Một quan hệ chỉ được chấp nhận nếu nó đáp ứng được các ràng buộc về cú pháp và từ vựng. Các ràng buộc về cú pháp đảm bảo rằng đó là một chuỗi động từ ban đầu cũng có thể bao gồm các danh từ (quan hệ bắt đầu bằng động từ nhẹ như: make, have hoặc thường thể hiện cốt lõi của mối quan hệ với một danh rừ như “have a hub in”).

Các ràng buộc từ vựng được dựa trên một từ điển mà được sử dụng để cắt tỉa các mối quan hệ dài và hiếm.

Cuối cùng, một giá trị độ tin cậy được tính toán cho từng quan hệ bằng cách sử dụng phân loại hồi quy logic.

1.2.6 Evaluation of Relation Extraction (Đánh giá hệ thống trích xuất quan hệ)

Các hệ thống trích xuất quan hệ có giám sát được đánh giá bằng việc sử dụng các tập “test” với chú thích của con người, các quan hệ chuẩn và tính toán “precision”, “recall” và “F-measure”.

Các phương thức học không giám sát và bán giám sát thì khó khăn hơn để đánh giá, vì chúng ta trích xuất những quan hệ mới từ web hoặc một text lớn. Bởi vì những phương thức này sử dụng lượng text rất lớn, nhìn chung không thể chạy chúng chỉ trên một bộ kiểm thử có nhãn nhỏ và do đó, nó không thể chú thích trước một tập hợp vàng các quan hệ chính xác.

Đối với phương thức này, có thể “precision” xấp xỉ bằng cách vẽ một mẫu ngẫu nhiên các quan hệ đầu ra và nhờ con người kiểm tra “accuracy” của mỗi quan hệ trong các quan hệ này. Thông thường, cách tiếp cận này tập trung vào các tuple được trích xuất từ “body” của text thay vì “relation mentions”, các hệ thống cần phải phát hiện mọi đề cập đến mối quan hệ để được ghi điểm một cách chính xác:

$$\hat{P} = \frac{\text{\# of correctly extracted relation tuples in the sample}}{\text{total \# of extracted relation tuples in the sample.}}$$

Hình 1.8: Công thức tính Precision

1.3 Extracting Times

1.3.1 Temporal Expression Extraction

Biểu thức thời gian (Temporal Expression) là những biểu thức để cập đến các điểm tuyệt đối về thời gian (Absolute temporal expressions), thời gian tương đối (Relative temporal expressions),...

“Absolute temporal expressions” là những thứ mà có thể được ánh xạ trực tiếp đến “calendar dates”, “times of day”, hoặc cả hai. “Relative temporal expressions” ánh xạ đến những thời gian cụ thể thông qua điểm tham chiếu khác (as in a week from last Tuesday). Cuối cùng, “durations” biểu thị các khoảng thời gian ở mức độ chi tiết khác nhau (seconds, minutes, days, weeks, centuries,...).

Absolute	Relative	Durations
April 24, 1916	yesterday	four hours
The summer of '77	next semester	three weeks
10:15 AM	two weeks from yesterday	six days
The 3rd quarter of 2006	last quarter	the last three quarters

Hình 1.9: Examples of absolute, relational and durational temporal expressions

“Temporal Expression” là các cấu trúc ngữ pháp mà có “lexical triggers” như “their head”. “Lexical triggers” phải là các danh từ, các danh từ riêng, tính từ, trạng từ; biểu thức thời gian cụ thể bao gồm các dự đoán cụm: các cụm danh từ, tính từ, trạng từ.

Nhiệm vụ nhận diện biểu thức thời gian bao gồm tìm điểm bắt đầu và kết thúc của tất cả các vùng text mà tương ứng như các biểu thức thời gian. Hướng tiếp cận dựa trên luật để nhận diện biểu thức thời gian sử dụng các tầng của “automata” để nhận ra các mẫu ở mức độ phức tạp tăng dần. Các token được gán nhãn POS, sau đó các đoạn lớn hơn được nhận diện từ các kết quả “trigger words” hoặc các lớp.

Sequence-labeling approaches theo lược đồ tương tự IBO được sử dụng cho các nhãn thực thể có tên, đánh dấu các từ bên trong, bên ngoài và bắt đầu.

A fare increase initiated last week by UAL Corp's...
 O O O O B I O O O

Hình 1.10: Ví dụ về cách đánh nhãn IBO

Feature	Explanation
Token	The target token to be labeled
Tokens in window	Bag of tokens in the window around a target
Shape	Character shape features
POS	Parts of speech of target and window words
Chunk tags	Base-phrase chunk tag for target and words in a window
Lexical triggers	Presence in a list of temporal terms

Hình 1.11: Typical features used to train IBO-style temporal expression taggers

Các đặc trưng được trích xuất từ token và ngữ cảnh của nó và một trình gán nhãn tuân tự thống kê được huấn luyện.

Temporal Expression Recognizers được đánh giá với “recall”, “precision”, và “F-measures” thông thường. Một khó khăn lớn cho tất cả các phương pháp tiếp cận từ vựng này là tránh các biểu thức mà “trigger false positives”.

1.3.2 Temporal Normalization

Temporal Normalization là quá trình ánh xạ biểu thức thời gian sang một trong hai điểm đặc biệt về thời gian hoặc một “duration”. Các điểm thời gian tương ứng với “calender dates”, “time of day” hoặc cả hai. “Duration” bao gồm chủ yếu là các khoảng thời gian nhưng cũng có thể bao gồm thông tin về điểm bắt đầu và điểm kết thúc.

Hầu hết các hướng tiếp cận hiện tại để chuẩn hóa thời gian là dựa trên luật. Các mẫu và “match” với biểu thức thời gian được kết hợp với các thủ tục phân tích ngữ nghĩa.

Fully qualified date expressions (FQDE) chứa năm, tháng và ngày trong một số dạng thông thường:

FQDE -> Month Date, Year {Year.val - Month.val – Date.val}

FQDE là khá hiếm trong các văn bản thực sự. Hầu hết các biểu thức thời gian trong các bài báo đều không đầy đủ và chỉ được neo hoàn toàn, thường liên quan đến dòng thời gian của bài viết, mà gọi là neo thời gian. Các giá trị của biểu thức thời gian như “today”, “yesterday”, “tomorrow” có thể được tính toán đối với neo thời gian này.

1.4 Extracting Events and their Times

Nhiệm vụ của “event extraction” là xác định sự đề cập (mention) của các sự kiện trong văn bản. Đối với mục đích của tác vụ này, một “event mention” là bất kỳ biểu thức nào biểu thị một sự kiện hoặc một trạng thái mà có thể được gán cho một điểm hoặc khoảng thời gian cụ thể.

[EVENT Citing] high fuel prices, United Airlines [EVENT said] Friday it has [EVENT increased] fares by \$6 per round trip on flights to

Hình 1.12: Ví dụ về event extraction

Trong tiếng anh, hầu hết các “mentions” tương ứng với các động từ và hầu hết các động từ giới thiệu các sự kiện. Tuy nhiên, điều này không phải lúc nào cũng đúng, các

sự kiện có thể được giới thiệu bằng các cụm danh từ (the move, the increase) và một số động từ không giới thiệu các sự kiện, như trong các cụm động từ “took effect” mà suy luận đến khi sự kiện bắt đầu hơn thay vì chính sự kiện. Tương tự, các danh từ nhẹ như “Make”, “Take” thường không biểu thị các sự kiện.

Các sự kiện sẽ được phân loại thành các hành động, các trạng thái, “reporting events” (say, report, tell, explain), perception events, ... Khía cạnh, thì, “modality” của mỗi sự kiện cũng cần được trích rút. Do đó, ví dụ, các sự kiện “said” trong một văn bản sẽ được chú thích như “class=Reporting; tense=past; aspect=Perfective”.

Event extraction thường được mô hình hóa thông qua học máy, việc phát hiện các sự kiện thông qua các mô hình tuần tự với IOB tagging, gán các lớp và thuộc tính sự kiện với các trình phân loại nhiều lớp. Các đặc trưng bề mặt phổ biến bao gồm POS, lexical items, và thông tin thì của từ.

1.4.1 Temporal Ordering of Events

Với các sự kiện và biểu thức thời gian trong văn bản đã được phát hiện, nhiệm vụ tiếp theo là sử dụng thông tin này để điều chỉnh các sự kiện thành một dòng thời gian hoàn chỉnh. 1 dòng thời gian như vậy sẽ hữu ích cho các ứng dụng như QA và “summarization”.

Một nhiệm vụ có phần đơn giản hơn nhưng vẫn hữu ích là áp đặt một thứ tự cho các sự kiện và biểu thức thời gian được đề cập trong một văn bản. Một thứ tự như vậy có thể cung cấp nhiều lợi ích như một dòng thời gian thực sự.

Việc xây dựng thứ tự như vậy có thể được xem xét như một nhiệm vụ phân loại và phát hiện quan hệ nhị phân tương tự như các nhiệm vụ được mô tả trong phần trước. Một cách tiếp cận phổ biến cho vấn đề này là vận hành nó bằng cách cố gắng xác định mối quan hệ thời gian.

Tập dữ liệu TimeBank bao gồm văn bản được chú thích với nhiều thông tin mà đã được trình bày trong phần này. TimeBank 1.2 bao gồm 183 news articles được lựa chọn từ nhiều nguồn, bao gồm các bộ sưu tập của Penn TreeBank và PropBank. Mỗi bài viết trong kho dữ liệu TimeBank đều có các biểu thức thời gian và đề cập đến sự kiện.

1.5 Template Filling

Nhiệm vụ của Template Filling là tìm các tài liệu mà dẫn ra các “script” cụ thể và sau đó điền vào các vị trí trong các mẫu được liên kết với các “filler” được trích rút từ văn bản. Các “slot-filler” này bao gồm các đoạn văn bản được trích rút trực tiếp từ text, hoặc chúng có thể bao gồm các khái niệm đã được suy luận từ các yếu tố văn bản thông qua một số lý bối cảnh.

1.5.1 Statistical Approaches to Template Filling

Mô hình tiêu chuẩn cho Template Filling giả định rằng chúng ta cố gắng điền vào các mẫu đã biết cố định với các “slot” đã biết, và cũng giả định rằng chúng ta được cung cấp các tài liệu huấn luyện được dán nhãn với các ví dụ của từng mẫu với các “filler” của từng sự kiện được đánh dấu trong văn bản.

Sau đó Template Filling sẽ tạo một mẫu cho mỗi sự kiện trong tài liệu đầu vào với các “slot” chứa đầy văn bản từ tài liệu.

Nhiệm vụ thường được mô hình hóa bằng cách huấn luyện hai mô hình giám sát riêng biệt. Hệ thống đầu tiên quyết định xem liệu mẫu có trong một câu cụ thể hay

không? Hệ thống này được gọi là “Template Recognition” hoặc thỉnh thoảng gọi là “Event Recognition”. Template Recognition có thể được coi là một hệ thống phân loại với các đặc trưng được trích xuất từ mỗi chuỗi từ được gán nhãn trong tài liệu đào tạo khi điền vào bất kỳ vị trí nào từ mẫu được phát hiện. Hệ thống thứ hai là “role-filler extraction”. Một bộ phân loại riêng biệt được huấn luyện để phát hiện từng vai trò.

1.5.2 Earlier Fine-State Template-Filling Systems

Các mẫu ở trên tương đối đơn giản, vấn đề với việc áp dụng cho các mẫu phức tạp hơn.

Bốn bước đầu tiên sử dụng biểu thức chính quy được viết bằng tay và các luật ngữ pháp để tách từ, chunking, parsing cơ bản. Bước thứ 5, nhận diện thực thể và các sự kiện với một bộ nhận diện dựa trên FST và chèn các “object” đã nhận diện vào các “slot” thích hợp trong các mẫu. Bộ nhận diện FST này được dựa trên các biểu thức chính quy và được xây dựng bằng tay.

Chương 2

Sematic Role Labeling (Chapter 22)[1]

Việc hiểu những sự kiện là một phần quan trọng để hiểu ngôn ngữ tự nhiên. Trong chương này, giới thiệu một mức độ đại diện mà cho phép nắm bắt được điểm chung giữa các câu.

Sematic roles là các biểu diễn mà thể hiện vai trò trừu tượng mà các đối số của một vị ngữ có thể đảm nhận trong sự kiện; Những thứ này có thể rất cụ thể như “buyer”; trừu tượng như “Agent” hoặc siêu trừu tượng (the proto-agent). Các “roles” này có thể vừa thể hiện các thuộc tính ngữ nghĩa chung của các đối số, vừa thể hiện mối quan hệ có khả năng của chúng với vai trò cú pháp của đối số trong câu. “Agents” có xu hướng là chủ đề của một câu chủ động, “Themes” đối tượng trực tiếp,... Những quan hệ này được mã hóa trong CSDL như PropBank và FrameNet.

2.1 Sematic Roles

Một số “thematic roles” thường được sử dụng với các định nghĩa như sau:

Thematic Role	Definition
AGENT	The volitional cause of an event
EXPERIENCER	The experiencer of an event
FORCE	The non-volitional cause of the event
THEME	The participant most directly affected by an event
RESULT	The end product of an event
CONTENT	The proposition or content of a propositional event
INSTRUMENT	An instrument used in an event
BENEFICIARY	The beneficiary of an event
SOURCE	The origin of the object of a transfer event
GOAL	The destination of an object of a transfer event

Hình 2.1: Some commonly used thematic roles with their definitions

Ví dụ: "Sasha broke the window" và "Pat opened the door"

Một đại diện sự kiện “neo-Daviddsonian” của 2 câu này sẽ là:

$$\begin{aligned}
 & \exists e, x, y \text{ } \textit{Breaking}(e) \wedge \textit{Breaker}(e, \textit{Sasha}) \\
 & \quad \wedge \textit{BrokenThing}(e, y) \wedge \textit{Window}(y) \\
 & \exists e, x, y \text{ } \textit{Opening}(e) \wedge \textit{Opener}(e, \textit{Pat}) \\
 & \quad \wedge \textit{OpenedThing}(e, y) \wedge \textit{Door}(y)
 \end{aligned}$$

Hình 2.2: Ví dụ về một đại diện sự kiện

Trong biểu diễn này, các “roles” của các chủ đề cho các danh từ “break” và “open” là “Breaker” và “Opener” tương ứng. Những “deep roles” này là đặc biệt cho mỗi sự kiện. Các sự kiện “Breaking” có Breaker, các sự kiện “Opening” có “Openers”,...

Nếu chúng ta có thể trả lời các câu hỏi, thực hiện các suy luận, hoặc làm bất kỳ loại để hiểu ngôn ngữ tự nhiên nào khác về các sự kiện này, chúng ta sẽ cần biết thêm một chút về nghĩa ngữ của các đối số này.

Chúng ta nói rằng các chủ ngữ của cả hai động từ này là “agents” (các tác nhân), do đó, “agents” là “thematic roles” mà đại diện cho một ý tưởng trừu tượng như quan hệ nhân quả. “Sematic role” cho các “participant” này là “theme” (chủ đề). “Thematic roles” là một trong những mô hình cũ nhất.

Hầu hết các bộ “thematic role” có một tập các “role”, nhưng chúng ta sẽ thấy các bộ có số lượng nhỏ hơn với ý nghĩa trừu tượng hơn và các bộ có số lượng “role” rất lớn dành riêng cho các tình huống khác nhau. (Thuật ngữ “sematic roles” được sử dụng cho tất cả các bộ “role”, dù lớn hay nhỏ).

2.2 Diathesis Alternations

Lý do chính khiến các hệ thống tính toán sử dụng “sematic roles” là để hoạt động như một sự đại diện ngữ nghĩa nông mà có thể cho phép ta đưa ra những suy luận đơn giản mà không thể tạo ra từ chuỗi bề mặt thuần túy của từ, hoặc thậm chí cây phân tích cú pháp. Ví dụ, nếu một tài liệu nói rằng “Company A acquired Company B”, ta muốn biết rằng câu trả lời này là từ câu hỏi “Was Company B acquired?” mặc dù sự thật rằng hai câu có cú pháp bề mặt rất khác nhau. Tương tự ngữ nghĩa nông này có thể hoạt động như một ngôn ngữ trung gian hữu ích.

Do đó, “sematic roles” giúp khai quát hóa trên các bề mặt khác nhau của cái đối số vị ngữ. Ví dụ, trong khi “agent” thường được nhận ra như các chủ đề của câu, trong các trường hợp khác “theme” có thể là chủ đề. Xem xét những thực tế có thể của các đối số “thematic” của động từ “break”.

Ví dụ:

- (22.3) *John broke the window.*
 AGENT THEME
- (22.4) *John broke the window with a rock.*
 AGENT THEME INSTRUMENT
- (22.5) *The rock broke the window.*
 INSTRUMENT THEME
- (22.6) *The window broke.*
 THEME
- (22.7) *The window was broken by John.*
 THEME AGENT

Hình 2.3: Ví dụ 1

Những ví dụ này gợi ý rằng “break” có (ít nhất) các đối số có thẻ “Agent”, “Theme”, và “Instrument”. Tập các đối số “thematic role” được thực hiện bởi một động từ thường được gọi là “thematic grid” hoặc “case frame”. Ta có thể thấy rằng có (trong số các trường hợp khác) các khả năng sau đây để thực hiện các đối số của “break”.

AGENT/Subject, THEME/Object
 AGENT/Subject, THEME/Object, INSTRUMENT/PP_{with}
 INSTRUMENT/Subject, THEME/Object
 THEME/Subject

Hình 2.4: Ví dụ 2

Nó chỉ ra rằng những động từ cho phép “thematic roles” của chúng được thực hiện ở các vị trí cú pháp khác nhau. Ví dụ, các động từ như “give” có thể nhận ra các đối số “Theme” và “Goal” theo hai cách khác nhau:

Ví dụ:

- a. *Doris gave the book to Cary.*
 AGENT THEME GOAL
- b. *Doris gave Cary the book.*
 AGENT GOAL THEME

Hình 2.5: Ví dụ 3

Những “realizations” cấu trúc nhiều đối số này (thực tế là “break” có thể nhận “Agent”, “Instrument” hoặc “Theme” làm chủ đề; và “give” có thể nhận “Theme” và “Goal”) được gọi là “verb alternation” hoặc “diathesis alternations”. Sự xen kẽ này chúng ta đã thấy bên trên với động từ “give” (dative alternation), dường như xảy ra với các lớp động từ ngữ nghĩa cụ thể, bao gồm “verbs of future having” (advance, allocate, offer, owe); “send verbs” (kick, pass, throw),...

2.3 Sematic Roles: Problems with Thematic Roles

Đại diện cho ý nghĩa ở cấp độ “thematic roles” dường như nó sẽ hữu ích trong việc xử lý các biến chứng (complications) như “diathesis alternations”. Tuy nhiên nó tỏ ra khá khác biệt khi đưa ra một bộ “roles” tiêu chuẩn, và khó có thể đưa ra một định nghĩa chính thức về các “roles” như “Agent”, “Theme” hoặc “Instrument”.

Có những trường hợp, ta muốn lập luận và khái quát hóa qua “sematic roles”, nhưng danh sách rời rạc hữu hạn của “role” không cho phép làm điều này. Cuối cùng, nó đã tỏ ra khó khăn khi chính thức xác định “thematic roles”.

2.4 The Proposition Bank

The Proposition Bank thường được gọi là PropBank, là một nguồn các câu đã chú thích với “sematic roles”. Do khó khăn trong việc xây dựng một tổng bộ “thematic roles” tổng quát, các “sematic roles” trong ProBank được xác định theo nghĩa của một động từ riêng lẻ. Mỗi ý nghĩa của mỗi động từ do đó có một bộ “roles” cụ thể, chỉ được cung cấp các con số chứ không phải các tên: Arg0, Arg1, Arg2,.... Trong đó, Arg0 đại diện cho “Proto-Agent”, Arg1 là “Proto-Patient”,...

Việc gán nhãn “sematic roles” của PropBank sẽ cho phép ta suy luận ra tính phổ biến trong cấu trúc sự kiện.

2.5 FrameNet

Mặc dù việc suy luận về sự tương đồng về mặt ngữ nghĩa giữa các câu khác nhau với “increase” là hữu ích, nó sẽ còn hữu ích hơn nếu chúng ta có thể đưa ra những suy luận như vậy trong nhiều tình huống, qua các động từ khác nhau và giữa các động từ và danh từ.

Ví dụ:

- (22.17) [Arg1 The price of bananas] increased [Arg2 5%].
- (22.18) [Arg1 The price of bananas] rose [Arg2 5%].
- (22.19) There has been a [Arg2 5%] rise [Arg1 in the price of bananas].

Hình 2.6: Ví dụ 4

Chú ý rằng ví dụ này sử dụng động từ “tăng” khác nhau. Ta muốn một hệ thống nhận diện rằng “the price of bananas” là cái đã tăng và “5%” là số tiền tăng lên, bất kể “5%” xuất hiện khi đối tượng của động từ là “increase” hay “rise”.

FrameNet là một dự án cố gắng giải quyết vấn đề này, ý tưởng rằng nhóm các từ đồng nghĩa liên quan đến một số thông tin cơ bản là phổ biến trong trí tuệ nhân tạo và khoa học nhận thức, sau đó các từ này được nhóm lại thành một “frame”.

Một “frame” trong FrameNet là một cấu trúc tri thức nền tảng xác định một tập “frame-specific semantic roles”, được gọi là “frame elements” và bao gồm một tập các “predicates” (vị ngữ) mà sử dụng các role này. Mỗi từ gợi lên một “frame” và cấu hình một số khía cạnh của “frame” và các phần tử của nó. Bộ dữ liệu FrameNet bao gồm 1 tập hợp các “frame” và các “frame elements”, các đơn vị từ vựng được liên kết với mỗi “frame” và một tập hợp các câu ví dụ được gán nhãn.

Core Roles	
ATTRIBUTE	The ATTRIBUTE is a scalar property that the ITEM possesses.
DIFFERENCE	The distance by which an ITEM changes its position on the scale.
FINAL_STATE	A description that presents the ITEM's state after the change in the ATTRIBUTE's value as an independent predication.
FINAL_VALUE	The position on the scale where the ITEM ends up.
INITIAL_STATE	A description that presents the ITEM's state before the change in the ATTRIBUTE's value as an independent predication.
INITIAL_VALUE	The initial position on the scale from which the ITEM moves away.
ITEM	The entity that has a position on the scale.
VALUE_RANGE	A portion of the scale, typically identified by its end points, along which the values of the ATTRIBUTE fluctuate.
Some Non-Core Roles	
DURATION	The length of time over which the change takes place.
SPEED	The rate of change of the VALUE.
GROUP	The GROUP in which an ITEM changes the value of an ATTRIBUTE in a specified way.

Hình 2.7: The frame elements in the **change-position-on-a-scale** frame from the FrameNet Labelers Guide

FrameNet cũng mã hóa mối quan hệ giữa các “frame”, cho phép các “frame” kế thừa lẫn nhau hoặc biểu thị các mối quan hệ giữa các “frame” như các quan hệ nhân quả (và khái quát hóa giữa các “frame elements” trong các “frame” khác nhau cũng có thể được biểu thị bằng kế thừa).

2.6 Semantic Role Labeling (SRL)

Là một nhiệm vụ tự động tìm kiếm các “semantic role” của mỗi đối số của mỗi “predicate” trong câu. Các hướng tiếp cận hiện tại để SRL được dựa trên học máy có giám sát, thường sử dụng nguồn FrameNet và PropBank để chỉ định những gì được coi là vị ngữ, xác định bộ “roles” được sử dụng trong nhiệm vụ và cung cấp các tập “train” và “test”.

Một thuật toán SRL đơn giản được mô tả trong hình dưới đây:

```

function SEMANTICROLELABEL(words) returns labeled tree
    parse  $\leftarrow$  PARSE(words)
    for each predicate in parse do
        for each node in parse do
            featurevector  $\leftarrow$  EXTRACTFEATURES(node, predicate, parse)
            CLASSIFYNODE(node, featurevector, parse)

```

Hình 2.8: A generic semantic-role-labeling algorithm

Mặc dù có một số lượng các thuật toán, nhiều trong số chúng sử dụng một số phiên bản của các bước trong thuật toán này.

Hầu hết các thuật toán bắt đầu với các trình phân tích “semantic roles”, bắt đầu bằng cách phân tích cú pháp, sử dụng trình phân tích cú pháp “broad-coverage” để gán nhãn một “parse” tới chuỗi đầu vào. “Parse” sau đó được duyệt để tìm tất cả các từ là “predicates”.

Đối với mỗi “predicates” này, thuật toán kiểm tra từng “node” trong cây phân tích cú pháp và quyết định “semantic role” (nếu có) mà nó đóng vai trò cho “predicates” này.

Điều này thường được thực hiện bằng phân loại có giám sát. Dựa ra một tập huấn luyện có nhãn như PropBank hoặc FrameNet, một vector đặc trưng được trích xuất cho mỗi node.

Sau đó, một trình phân loại 1-N được huấn luyện để dự đoán “semantic roles” cho từng thành phần được cung cấp các đặc trưng này. Trong đó N là số lượng các “semantic roles” tiềm năng cộng với một “role” (NONE) bổ sung cho các thành phần không có “role”. Hầu hết các thuật toán phân loại tiêu chuẩn đều có thể sử dụng (Hồi quy logic, SVM, . . .).

Further Issues in Semantic Role Labeling

Thay vì việc huấn luyện một bộ phân loại một bước duy nhất, một số thuật toán gán nhãn “role” chia nhỏ nhiệm vụ phân loại cho các đối số của một vị ngữ thành nhiều bước:

- Pruning (cắt tỉa): vì chỉ có một số lượng nhỏ các “constituent” (là một từ hoặc một nhóm các từ có chức năng như một đơn vị trong cấu trúc phân cấp) trong một câu là các đối số của bất kỳ vị ngữ đã cho nào nên nhiều hệ thống sử dụng phương pháp phỏng đoán đơn giản để cắt tỉa các “constituent” không chắc chắn.
- Identification: 1 bộ phân loại nhị phân của mỗi node làm đối số được gán nhãn hoặc NONE.
- Classification: phân loại 1-N tất cả các “constituent” được gán nhãn như các đối số của bước trước.

Việc phân tách thành Identification và Classification có thể dẫn đến các việc sử dụng các đặc trưng tốt hơn (những đặc trưng khác nhau có thể hữu ích cho các nhiệm vụ khác nhau) hoặc hiệu quả tính toán hơn.

Thuật toán Classification ở trên mô tả phân loại mỗi đối số một cách riêng biệt (cục bộ), đưa ra giả định đơn giản hóa rằng mỗi đối số của một vị ngữ có thể được gán nhãn độc lập. Nhưng tất nhiên điều này là không đúng; có nhiều loại tương tác giữa các đối số đối với một sự gán nhãn toàn cầu hơn cho các “constituents”.

Chương 3

Question Answering (QA) (Chapter 27)[1]

Đến đầu những năm 1960, đã có nhiều hệ thống thực hiện hai mô hình chính cho bài toán QA: IR-based QA (hoặc text-based QA) và Knowledge-based QA để trả lời những câu hỏi.

Các hệ thống QA gần đây nhất tập trung vào “factoid questions”. Factoid question là những câu hỏi mà có thể được trả lời với các sự kiện đơn giản được thể hiện trong câu trả lời ngắn.

+ IR-based QA: dựa vào lượng thông tin khổng lồ có sẵn dưới dạng văn bản trên web. Dựa ra một câu hỏi người dùng, các kỹ thuật truy xuất thông tin trích xuất các đoạn trực tiếp từ các tài liệu này, được hướng dẫn bởi “text” của câu hỏi.

Phương thức này xử lý câu hỏi để xác định loại câu hỏi có khả năng (thường là một thực thể có tên như người, địa điểm hoặc thời gian) và tạo các truy vấn để gửi đến các công cụ tìm kiếm. Công cụ này trả về các tài liệu được xếp hạng mà được chia thành các đoạn phù hợp và được sắp xếp lại. Cuối cùng, chuỗi câu trả lời ứng viên được trích xuất từ các đoạn và được xếp hạng.

+ Knowledge-based QA, xây dựng một biểu diễn ngữ nghĩa của câu truy vấn. Ý nghĩa của một câu truy vấn có thể là một câu lệnh tính toán với đầy đủ vị ngữ (a full predicate calculus statement). Ví dụ, What states border Texas?

=> $\lambda x. state(x) \wedge borders(x, texas)$

Mặt khác, ý nghĩa của một câu hỏi có thể là một quan hệ duy nhất giữa một thực thể đã biết và một thực thể chưa biết.

3.1 IR-based Factoid QA

Mục tiêu của IR-based QA là để trả lời một câu hỏi của người dùng bằng cách tìm kiếm những đoạn “text” ngắn trên web hoặc một vài bộ sưu tập tài liệu khác.

3.1.1 Question Processing

Mục tiêu của pha xử lý câu hỏi (Question Processing) là để trích xuất một số thông tin từ câu hỏi. “Loại câu trả lời” chỉ định loại thực thể mà câu trả lời bao gồm (Người, địa điểm, thời gian,...). Câu truy vấn chỉ định các từ khóa nên được sử dụng cho hệ thống IR để sử dụng cho việc tìm kiếm các tài liệu. Một vài hệ thống cũng trích xuất một “focus” (trọng tâm), đó là chuỗi các từ trong câu hỏi có khả năng được thay thế bằng câu trả lời

trong bất kỳ chuỗi câu trả lời nào được tìm thấy. Một vài hệ thống cũng phân loại “loại câu hỏi”: đây có phải là câu hỏi định nghĩa (definition question), câu hỏi toán học, câu hỏi danh sách không?

3.1.2 Answer type detection (Question Classification)

Nhiệm vụ phân loại câu hỏi (question classification) hoặc “answer type recognition” là để xác định “answer type”. Ví dụ, một câu hỏi “Who founded Virgin Airlines” mong đợi một câu trả lời thuộc loại PERSON.

Nếu chúng ta biết “loại câu trả lời” cho một câu hỏi, chúng ta có thể tránh nhầm vào tất cả các câu hoặc cụm danh từ trong toàn bộ tài liệu, thay vào đó chỉ tập trung vào người hoặc thành phố.

Các bộ phân loại câu hỏi có thể được xác định bằng các luật viết tay, bằng học máy có giám sát hoặc với một vài kết hợp nào đó.

Tuy nhiên, hầu hết các bộ phân loại câu hỏi hiện đại đều dựa trên học máy có giám sát và được huấn luyện với một CSDL các câu hỏi được gán nhãn bằng một loạt các loại câu trả lời. Các đặc trưng tiêu biểu được sử dụng để phân loại bao gồm các từ trong câu hỏi, POS của từng từ và thực thể có tên trong câu hỏi.

Nhìn chung, độ chính xác cho phân loại câu hỏi tương đối cao với các nhãn câu hỏi dễ như người, địa điểm và thời gian. Khó hơn cho các nhãn REASON hoặc DESCRIPTION.

3.1.3 Query Formulation

Query Formulation là nhiệm vụ tạo ra từ các câu hỏi một danh sách các từ khóa mà tạo thành một truy vấn có thể được gửi đến một hệ thống truy xuất thông tin (IR). Một cách chính xác những truy vấn phụ thuộc vào ứng dụng. Nếu việc trả lời câu hỏi được áp dụng cho web thì có thể chỉ cần tệp một từ khóa từ mỗi từ trong câu hỏi, để công cụ tìm kiếm web tự động xóa bất kỳ ký tự nào. Thông thường, ta bỏ qua các từ để hỏi (Who, When,...). Ngoài ra, từ khóa chỉ có thể được hình thành từ các cụm được tìm thấy trong các cụm danh từ trong câu hỏi, áp dụng danh sách “stopwords” để bỏ qua các từ chức năng và các động từ có tần số xuất hiện cao và có ít nội dung.

3.1.4 Passage Retrieval

Truy vấn đã được tạo trong pha xử lý câu hỏi, sau đó được sử dụng để truy vấn với hệ thống IR. Kết quả của giai đoạn trích xuất tài liệu nào là một tập các tài liệu.

Mặc dù tập các tài liệu được xếp hạng theo mức độ liên quan, tài liệu được xếp hạng hàng đầu có thể không trả lời cho câu hỏi. Điều này là do các tài liệu không phải là một đơn vị thích hợp cho việc xếp hạng độ liên quan đến mục tiêu của hệ thống QA. Một tài liệu lớn và có liên quan cao mà không trả lời cho câu hỏi thì không phải là một ứng viên lý tưởng cho việc xử lý sâu hơn.

Vì vậy, giai đoạn tiếp theo là trích xuất một tập các đoạn trả lời tiềm năng từ tập các tài liệu được trích xuất. Việc định nghĩa một đoạn nhất thiết phải phụ thuộc vào hệ thống, nhưng các đơn vị điển hình bao gồm các “sections”, “paragraphs” và “sentences”. Ta có thể chạy thuật toán phân đoạn (paragraph segmentation algorithm) trên tất cả các tài liệu trả về và coi mỗi “paragraph” như một “segment”.

Trong passage retrieval, trước tiên ta lọc ra các đoạn trong các tài liệu được trả về mà không chứa câu trả lời tiềm năng và sau đó xếp hạng phần còn lại tùy theo khả năng

chúng có chứa câu trả lời cho câu hỏi. Bước đầu tiên trong quá trình này là tìm thực thể có tên hoặc phân loại câu trả lời trên các đoạn được trích xuất. Loại câu trả lời mà chúng ta xác định từ câu hỏi cho chúng ta biết các loại câu trả lời có thể có mà ta mong đợi sẽ thấy trong câu trả lời. Do đó, ta có thể lọc ra các tài liệu mà không chứa bất kỳ thực thể nào đúng loại.

Các đoạn còn lại sau đó được xếp hạng, thường là bằng học máy có giám sát, dựa vào một tập nhỏ các đặc trưng có thể dễ dàng trích rút từ một số lượng lớn các câu trả lời có khả năng, chẳng hạn như:

- Số lượng thực thể có tên được đặt tên đúng trong đoạn.
- Số lượng các từ khóa từ câu hỏi có trong đoạn.
- Chuỗi chính xác dài nhất của các từ khóa trong câu hỏi mà xảy ra trong đoạn văn.
- Thứ hạng của tài liệu của đoạn văn được trích rút.
- N-gram chồng chéo giữa đoạn và câu hỏi.

3.1.5 Answer Processing

Giai đoạn cuối cùng của QA là để trích xuất một câu trả lời cụ thể từ đoạn văn.

Hai lớp thuật toán đã được áp dụng cho tác vụ trích xuất câu trả lời là: Answer-type pattern extraction và N-gram tiling.

Trong phương thức pattern-extraction cho xử lý câu trả lời, ta sử dụng thông tin về loại câu trả lời dự kiến cùng với các mẫu biểu thức chính quy. Ví dụ, cho các câu hỏi với loại câu trả lời là HUMAN, ta chạy “answer type” hoặc “named entity tagger” trên đoạn (câu) ứng viên và trả về bất cứ thực thể nào được gán nhãn HUMAN.

Thật không may, câu trả lời của một vài câu hỏi như các câu hỏi DEFINITION không có xu hướng thuộc một loại thực thể có tên cụ thể. Đối với một số câu hỏi, thay vì sử dụng các “loại câu trả lời”, ta sử dụng các mẫu biểu thức chính quy được viết bằng tay để giúp trích xuất câu trả lời. Các mẫu này cũng hữu ích trong trường hợp một đoạn văn chứa nhiều ví dụ cùng một loại thực thể có tên.

Các mẫu cụ thể cho từng loại câu hỏi có thể được viết bằng tay hoặc học tự động mà sử dụng các phương thức trích rút quan hệ. Các mẫu sau đó có thể được sử dụng với các thông tin khác như các đặc trưng trong một bộ phân loại xếp hạng các câu trả lời ứng viên. Ta trích rút các câu trả lời tiềm năng bằng cách sử dụng các thực thể có tên được trả về từ truy xuất đoạn (passage retrieval) và xếp hạng chúng bằng cách sử dụng trình phân loại với các đặc trưng chẳng hạn như:

- Answer-type match: True nếu câu trả lời ứng viên chứa một cụm với “loại câu trả lời” chính xác.
- Number of matched question keywords: Có bao nhiêu từ khóa của câu hỏi xuất hiện trong câu trả lời ứng viên.
- Keyword distance: Khoảng cách giữa câu trả lời ứng viên và các từ khóa truy vấn.

3.2 Knowledge-based Question Answering

Trong khi một lượng lớn thông tin được mã hóa trong số lượng lớn văn bản trên web, thông tin rõ ràng cũng tồn tại ở dạng có cấu trúc chặt chẽ hơn. Ta sử dụng Knowledge-based question answer cho ý tưởng về việc trả lời các câu hỏi tự nhiên bằng việc ánh xạ nó tới một câu truy vấn trên một CSDL có cấu trúc.

Các hệ thống để ánh xạ từ một chuỗi văn bản sáng bất kỳ dạng logic nào được gọi là trình phân tích ngữ nghĩa. Trình phân tích ngữ nghĩa cho QA thường ánh xạ tới một số phiên bản của phép tính vị ngữ hoặc một ngôn ngữ truy vấn như SQL hoặc SPARQL.

Do đó, dạng logic của câu hỏi là ở dạng truy vấn hoặc có thể dễ dàng chuyển đổi thành một câu hỏi.

3.2.1 Rule-based Methods

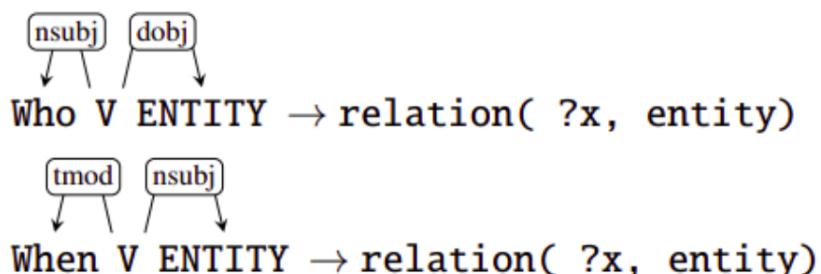
Đối với các quan hệ rất thường xuyên, có thể đáng để viết các quy tắc để trích xuất các mối quan hệ từ câu hỏi.

3.2.2 Supervised Methods

Trong một số trường hợp, ta có dữ liệu có giám sát, bao gồm tập các câu hỏi được ghép nối với dạng logic chính xác. Nhiệm vụ sau đó là đưa các cặp “tuples” huấn luyện và tạo ra một hệ thống ánh xạ từ các câu hỏi mới sang dạng logic của chúng.

Hầu hết các thuật toán học không giám sát cho việc học để trả lời các câu hỏi đơn giản về mối quan hệ, trước tiên phân tích các câu hỏi và sau đó cẩn chỉnh các cây phân tích thành dạng logic. Nói chung, các hệ thống này tự khởi động bằng cách có một tập luật nhỏ để xây dựng ánh xạ này và cả những từ vựng khởi tạo ban đầu nữa.

Ví dụ, hệ thống có thể có các chuỗi cho từng thực thể trong hệ thống (Texas, Ada Lovelace) và sau đó có các quy tắc mặc định đơn giản ánh xạ các đoạn của cây phân tích câu hỏi thành mỗi quan hệ đặc biệt.



Hình 3.1: Ví dụ, hệ thống có thể có các chuỗi cho từng thực thể trong hệ thống (Texas, Ada Lovelace)

=> When was Ada Lovelace born? birth-year (Ada Lovelace, ?x)

Một hệ thống có giám sát sẽ phân tích từng “tuple” trong tập huấn luyện và tạo ra một tập lớn hơn các quy tắc cụ thể, cho phép nó ánh xạ vào các ví dụ chưa từng thấy.

3.2.3 Dealing with variation: Semi-Supervised Methods

Bởi vì rất khó để tạo ra các bộ huấn luyện với các câu hỏi được gán nhãn đại diện cho ý nghĩa của chúng, các bộ dữ liệu được giám sát có thể bao gồm nhiều dạng khác nhau mà ngay cả các câu hỏi thực tế đơn giản cũng có thể có.

Trích xuất thông tin mở REVERB trích xuất hàng tá bộ “triplets” (subject, relation, object) từ web như (Ada Lovelace, was born in, 1815). Bằng cách sắp xếp các chuỗi này với nguồn tri thức chính xác như Wikipedia, ta tạo các mối quan hệ mới mà có thể được truy vấn trong khi đồng thời học cách ánh xạ giữa các từ trong câu hỏi và các mối quan hệ chính xác.

3.3 Evaluation of Factoid Answer

Một số liệu đánh giá phổ biến để trả lời cho câu hỏi thực tế là MRR (Mean Reciprocal Rank). MRR giả định một tập test của các câu hỏi mà đã được con người đánh nhãn với câu trả lời chính xác. MRR cũng giả định rằng các hệ thống đang trả về một danh sách ngắn các câu trả lời hoặc các đoạn văn có chứa câu trả lời. Mỗi câu hỏi sau đó được tính theo tỷ lệ “reciprocal” của thứ hạng câu trả lời đúng đầu tiên.

Ví dụ, nếu hệ thống trả về 5 câu trả lời nhưng 3 câu đầu bị sai vì thế câu trả lời đúng được xếp hạng cao nhất là câu thứ 4. Do đó, điểm số sẽ là $\frac{1}{4}$. Các câu hỏi với tập trả về không chứa bất kỳ câu trả lời đúng nào sẽ bằng 0. Điểm của một hệ thống sau đó là điểm trung bình của mỗi câu hỏi trong một tập hợp.

$$\text{MRR} = \frac{1}{N} \sum_{\substack{i=1 \\ \text{s.t. } rank_i \neq 0}}^N \frac{1}{rank_i}$$

Hình 3.2: Công thức tính MRR

Chương 4

Dialog Systems and Chatbots (Chater 28)[1]

Chương này giới thiệu những thuật toán cơ bản của “conversational agents” (các tác nhân trò chuyện) hoặc dialog system. Những chương trình này giao tiếp với người dùng bằng ngôn ngữ tự nhiên (text, speech, hoặc cả 2) và thường rơi vào hai lớp.

Task-oriented dialog agents (các tác nhân hộp thoại hướng tác vụ) được thiết kế cho một nhiệm vụ cụ thể và được thiết lập để có những đoạn hội thoại ngắn (từ một tương tác đơn lẻ đến các tương tác có lẽ như nửa quảng cáo) để lấy thông tin từ người dùng để giúp hoàn thành nhiệm vụ. Chúng bao gồm các digital assistants (trợ lý kỹ thuật số) mà giờ hiện có trên mọi điện thoại di động hoặc trên các bộ điều khiển gia đình (siri, cortana, alexa,...) mà các dialog agents có thể đưa ra các chỉ dẫn về du lịch, điều khiển thiết bị gia đình, tìm nhà hàng hoặc giúp gọi điện thoại.

Chatbots là các hệ thống được thiết kế cho các cuộc hội thoại mở rộng. Được thiết lập để mô phỏng tính năng trò chuyện và tương tác giữa người với người thay vì tập trung vào một nhiệm vụ cụ thể như đặt lịch máy bay. Các ứng dụng này thường có giá trị giải trí. Chatbots cũng hữu ích cho các mục đích thực tế, chẳng hạn như kiểm tra các lý thuyết về tư vấn tâm lý.

4.1 Chatbots

Chatbots là các hệ thống có thể thực hiện các cuộc hội thoại mở rộng với mục tiêu bắt chước các cuộc hội thoại không có cấu trúc hoặc trò chuyện đặc trưng về tương tác giữa người với người.

Kiến trúc của chatbot rơi vào hai lớp: Hệ thống dựa trên quy tắc và hệ thống dựa trên “corpus”. Các hệ thống dựa trên quy tắc bao gồm các hệ thống như ELIZA có ảnh hưởng sớm (ELIZA được sử dụng cho các mục đích thực tế, như kiểm tra lý thuyết về tư vấn tâm lý). Các hệ thống dựa trên “corpus” khai thác các bộ dữ liệu lớn về các cuộc hội thoại giữa người với người, có thể được hoàn thành bằng cách sử dụng truy xuất thông tin (các hệ thống dựa trên IR đơn giản chỉ cần sao chép phản hồi của con người từ cuộc hội thoại trước đó) hoặc bằng các sử dụng mô hình dịch máy như các mạng nơ ron “Seq2Seq” để tìm hiểu cách ánh xạ từ cách nói của người dùng đến phản hồi hệ thống.

4.1.1 Rule-based chatbots: ELIZA và PARRY

ELIZA (Weizenbaum, 1966) là hệ thống dialog chatbot quan trọng nhất trong lịch sử của lĩnh vực này, do đó ta sẽ dành thời gian để mô tả chi tiết thuật toán của nó. ELIZA được thiết kế để mô phỏng một nhà tâm lý học Rogerian, dựa trên một nhánh của tâm lý học lâm sàng có phương pháp liên quan đến việc phác họa bệnh nhân bằng cách phản ánh các “statements” ngược lại họ.

```
function ELIZA GENERATOR(user sentence) returns response
    Find the word w in sentence that has the highest keyword rank
    if w exists
        Choose the highest ranked rule r for w that matches sentence
        response ← Apply the transform in r to sentence
        if w = 'my'
            future ← Apply a transformation from the 'memory' rule list to sentence
            Push future onto memory stack
        else (no keyword applies)
            either
                response ← Apply the transform for the NONE keyword to sentence
            or
                response ← Pop the top response from the memory stack
    return(response)
```

Hình 4.1: A simplified sketch of the ELIZA algorithm. The power of the algorithm come from the particular transforms associated with each keyword

Trong thuật toán của ELIZA, từ khóa được liên kết với một thứ hạng, với các từ cụ thể được xếp hạng cao hơn và các từ tổng quát được xếp hạng thấp hơn.

Nếu không có từ khóa khớp, ELIZA chọn một câu trả lời như “Please go on”, “That's very interesting” hoặc “I see”.

Cuối cùng ELIZA có một thủ thuật bộ nhớ thông minh chiếm phần cuối cùng của cuộc trò chuyện. Bất cứ khi nào từ “my” được xếp hạng cao nhất thì ELIZA sẽ ngẫu nhiên biến đổi trong danh sách nhớ, áp dụng nó vào câu và lưu nó trong ngăn xếp (Stack). Sau đó, nếu không có từ nào khớp thì nó trả về câu đầu hàng đợi.

PARRY đã được sử dụng để nghiên cứu tâm thần phân liệt, ngoài các biểu thức thông thường giống ELIZA, hệ thống PARRY bao gồm một mô hình về trạng thái tinh thần của chính nó. Với các biến số ảnh hưởng đến mức độ sợ hãi và tức giận của agents. Một số chủ đề của cuộc trò chuyện có thể khiến PARRY trở nên tức giận hoặc không tin tưởng hơn. Nếu biến số tức giận của PARRY cao thì nó chọn một tập hợp các kết quả đầu ra mang tính thù địch và tương tự cho tính ảo tưởng. PARRY là hệ thống đầu tiên biết đến vượt qua bài kiểm tra Turing.

4.1.2 Corpus-based chatbots

Các chatbot dựa trên “corpus” thay vì sử dụng các luật được xây dựng bằng tay, nó khai thác các cuộc hội thoại của con người với con người hoặc đôi khi khai thác các phản

ứng của con người từ các cuộc trò chuyện của con người. Các phản hồi của chatbot thậm chí có thể được trích xuất từ các câu trong “corpus” của văn bản.

Có hai loại chatbot dựa trên “corpus”: hệ thống dựa trên truy xuất thông tin và các hệ thống dựa trên học máy có giám sát mà dựa trên “Sequence transduction”.

Giống như chatbot dựa trên luật (không giống các hệ thống dialog dựa trên frame), hầu hết các chatbot dựa trên “corpus” hướng đến việc tập trung vào việc tạo ra câu trả lời duy nhất phù hợp với người dùng ngay lập tức thay vì việc mô hình hóa bối cảnh của cuộc hội thoại. Vì lý do đó, họ thường gọi là các hệ thống tạo phản ứng (response generation). Do đó, các hệ thống chatbot dựa trên “corpus” có một số điểm tương đồng với các hệ thống QA, mà tập trung vào những câu trả lời đơn trong khi bỏ qua bối cảnh hoặc các mục tiêu hội thoại lớn hơn.

IR-based chatbots

Nguyên tắc đầu tiên sau các chatbot dựa trên IR là phản hồi lại lượt của người dùng X bằng cách lặp lại 1 vài lượt Y thích hợp từ 1 corpus tự nhiên của con người. Sự khác biệt giữa các hệ thống như vậy nằm ở cách chúng ta chọn corpus và cách chúng ta quyết định các gì được coi là thích hợp của con người để sao chép.

Dựa vào corpus và câu của người dùng, hệ thống dựa trên IR có thể sử dụng bất cứ thuật toán trích xuất nào để lựa chọn 1 câu trả lời thích hợp từ corpus. Hai phương pháp đơn giản nhất như sau:

- Return the response to the most similar turn: một truy vấn người dùng đã cho q và một tập corpus trò chuyện C, tìm lượt t trong C mà tương tự với q nhất (ví dụ như có cosine cao nhất với q) và trả về lượt phía sau, tức là câu trả lời của con người với t trong C:

$$r = \text{response} \left(\underset{t \in C}{\operatorname{argmax}} \frac{q^T t}{\|q\| \|t\|} \right)$$

- Return the most similar turn: Cho 1 câu truy vấn người dùng q và 1 corpus trò chuyện C, trả về lượt t trong C mà tương tự nhất với q (ví dụ có cosine cao nhất với q):

$$r = \underset{t \in C}{\operatorname{argmax}} \frac{q^T t}{\|q\| \|t\|}$$

Ý tưởng là khớp trực tiếp q với câu trả lời trong C (thực tế chỉ ra rằng việc này tốt hơn việc trả về lượt phía sau).

Sequence to sequence chatbots

Một cách thay thế khác để sử dụng corpus để tạo dialog là nghĩ về việc tạo câu trả lời như nhiệm vụ của “transducing” (truyền tải) từ lượt người dùng trước khi chuyển sang lượt của hệ thống. Về cơ bản đây là phiên bản học máy của ELIZA. Học máy từ một corpus để transduce (chuyển) câu hỏi thành câu trả lời.

Ý tưởng này được phát triển đầu tiên bằng cách sử dụng dịch máy dựa trên cụm từ để dịch một lượt của người dùng sang một câu trả lời của hệ thống. Tuy nhiên nó nhanh chóng trở nên rõ ràng rằng nhiệm vụ tạo câu trả lời khác so với dịch máy. Trong dịch máy,

các từ hoặc cụm từ trong nguồn và câu đích có xu hướng liên kết tốt với nhau; nhưng trong cuộc trò chuyện thì các nói của người dùng có thể không chia sẻ từ hoặc cụm từ nào với câu trả lời mạch lạc.

Thay vào đó, các mô hình transduction cho tạo câu trả lời được mô hình hóa thay vì sử dụng Seq2Seq. Một sửa đổi được yêu cầu cho mô hình Seq2Seq cơ bản để điều chỉnh nó cho nhiệm vụ tạo câu trả lời. Ví dụ, các mô hình Seq2Seq cơ bản có xu hướng tạo ra các câu trả lời có thể dự đoán được nhưng lặp đi lặp lại và do đó các câu trả lời buồn tẻ như “I’m OK” hoặc “I don’t know” mà làm chấm dứt cuộc trò chuyện. Điều này có thể được giải quyết bằng cách thay đổi hàm mục tiêu cho mô hình Seq2Seq thành “mutual information objective” hoặc bằng cách sửa đổi một “beam decoder” để giữ nhiều câu trả lời đa dạng hơn trong beam.

Một vấn đề khác với kiến trúc tạo câu trả lời Seq2Seq đơn giản là nó không có khả năng mô hình hóa bối cảnh trước đó của cuộc trò chuyện. Điều này có thể được thực hiện bằng cách cho phép mô hình xem các lượt trước đó, chẳng hạn bằng cách sử dụng mô hình phân cấp mà tóm tắt thông tin qua nhiều lượt trước đó.

Cuối cùng, bộ tạo câu trả lời Seq2Seq tập trung tạo các câu trả lời đơn và vì vậy không có xu hướng làm tốt công việc về tạo liên tiếp các câu trả lời mà kết hợp qua nhiều lượt. Điều này có thể giải quyết bằng cách sử dụng học tăng cường, cũng như các kỹ thuật như “adversarial networks”, để học cách chọn các câu trả lời làm cho cuộc trò chuyện trở nên tự nhiên hơn.

Evaluating chatbots

Chatbot thường được đánh giá bởi con người.

ADEM classifier được huấn luyện trên một tập các câu trả lời được đánh nhãn bởi con người với mức độ phù hợp của chúng và học cách dự đoán nhãn này của từ ngữ cảnh đối thoại và các từ trong câu trả lời của hệ thống.

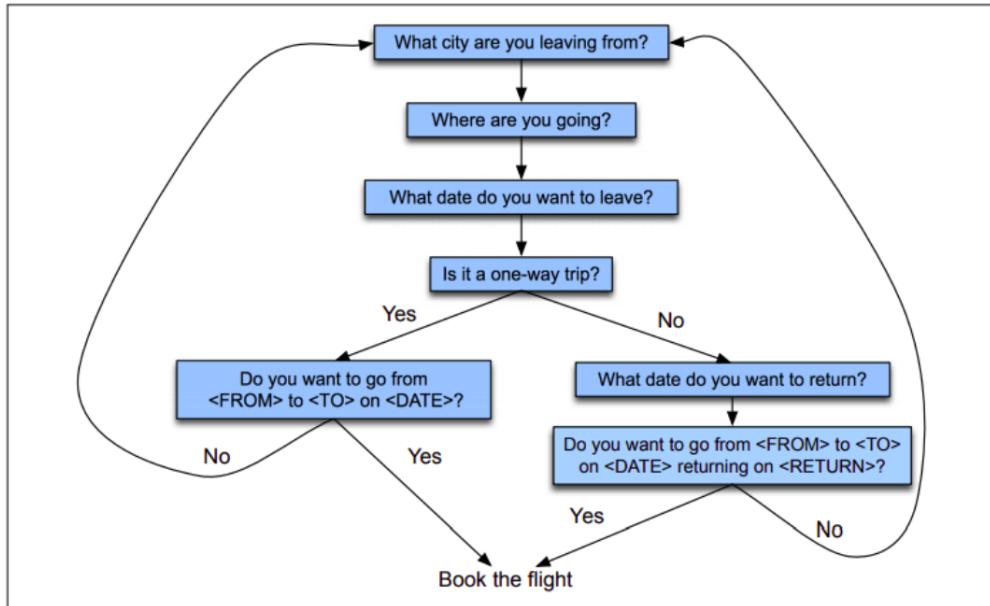
Adversarial evaluation: ý tưởng là đào tạo một trình phân loại đánh giá giống như Turing để phân biệt giữa câu trả lời được tạo bởi con người và của máy. Hệ thống tạo phản hồi càng thành công trong việc đánh lừa bộ phân loại này thì hệ thống càng tốt.

4.2 Frame Based Dialog Agents

Các hệ thống dialog dựa trên task hiện đại được dựa trên một domain ontology, một cấu trúc tri thức mà đại diện cho các loại ý định mà hệ thống có thể rút ra từ các câu của người dùng. Ontology định nghĩa một hoặc nhiều frames, mỗi bộ các slots và định nghĩa các values mà mỗi slot có thể lấy.

4.2.1 Control structure for frame-based dialog

Kiến trúc điều khiển của các hệ thống dialog dựa trên frame được thiết kế xung quanh frame. Mục tiêu là lắp đầy các slots trong frame với các filler ý định của người dùng và sau đó thực hiện hành động liên quan cho người dùng (trả lời 1 câu hỏi, hoặc đặt một chuyến bay). Hầu hết các hệ thống dialog dựa trên frame được dựa trên FSA mà được thiết kế bằng tay cho các nhiệm vụ bở một nhà thiết kế dialog.



Hình 4.2: A simple finite-state automaton architecture for frame-based dialog

Các trạng thái của FSA tương ứng với các slot question, người dùng và các vòng cung tương ứng với các hành động cần thực hiện phụ thuộc vào những gì người dùng phản hồi. Hệ thống này hoàn toàn kiểm soát cuộc trò chuyện với người dùng. Nó hỏi người dùng một loạt các câu hỏi, bỏ qua (hoặc giải thích sai) bất cứ điều gì không phải câu trả lời trực tiếp cho câu hỏi và sau đó tiếp tục chuyển sang câu hỏi tiếp theo.

Người nói kiểm soát bất kỳ cuộc trò chuyện nào được cho là có “initiative” (có sáng kiến) trong cuộc trò chuyện. Do đó, các hệ thống kiểm soát hoàn toàn cuộc trò chuyện theo cách này được gọi là system-initiative. Ngược lại, trong cuộc hội thoại bình thường giữa người với người, initiative chuyển đổi qua lại giữa những người tham gia.

Kiến trúc single-initiative finite-state dialog có lợi thế là hệ thống luôn biết câu hỏi mà người dùng trả lời. Điều này có nghĩa là hệ thống có thể chuẩn bị bộ nhận dạng giọng nói với một mô hình ngôn ngữ được điều chỉnh để trả lời cho câu hỏi này và cũng làm cho việc hiểu ngôn ngữ tự nhiên dễ dàng hơn. Hầu hết các hệ thống “Finite state” cũng cho phép các lệnh (command) phổ quát (universal) có thể được nói ở bất cứ đâu trong hộp thoại, như trợ giúp để đưa ra một thông báo trợ giúp và start over (hoặc main menu) để người dùng về trạng thái ban đầu chính được chỉ định. Tuy nhiên, kiến trúc trạng thái hữu hạn đơn giản như vậy thường chỉ được áp dụng cho các tác vụ đơn giản như nhập số thẻ tín dụng hoặc tên hoặc mật khẩu.

Cho hầu hết các ứng dụng, người dùng câu hình linh hoạt hơn một chút. Ví dụ, trong tình huống lập kế hoạch du lịch, người dùng có thể nói 1 câu mà điền vào nhiều slot. Hoặc trong những trường hợp có nhiều frame, một người dùng có thể nói điều gì đó để thay đổi frame. Ví dụ, từ việc đặt chỗ hàng không sai đặt xe cho thuê.

Khi các hệ thống có đầy đủ thông tin, nó sẽ thực hiện các hành động cần thiết (như truy vấn CSDL về các chuyến bay) và trả về kết quả cho người dùng.

4.2.2 Natural language understanding for filling slots

Mục tiêu của thành phần hiểu ngôn ngữ tự nhiên là để trích xuất ba thứ từ cách nói của người dùng. Nhiệm vụ đầu tiên là domain classification, ví dụ như người dùng

này đang nói về hàng hàng không, lập trình một đồng hồ báo thức, hoặc xử lý lịch của họ không? Tất nhiên các nhiệm vụ phân loại 1-n này là không cần thiết đối với các hệ thống có 1 domain tập trung vào. Thứ 2 là xác định mục đích của người dùng (intent determination): người dùng đang cố gắng hoàn thành nhiệm vụ hay mục tiêu chung nào? (Tìm phim hoặc hiện thị chuyến bay hoặc xóa lịch hẹn). Cuối cùng chúng ta cần làm slot filling: trích xuất các slot và filler cụ thể mà người dùng muốn hiểu từ cách nói của họ đối với mục đích của họ. Ví dụ:

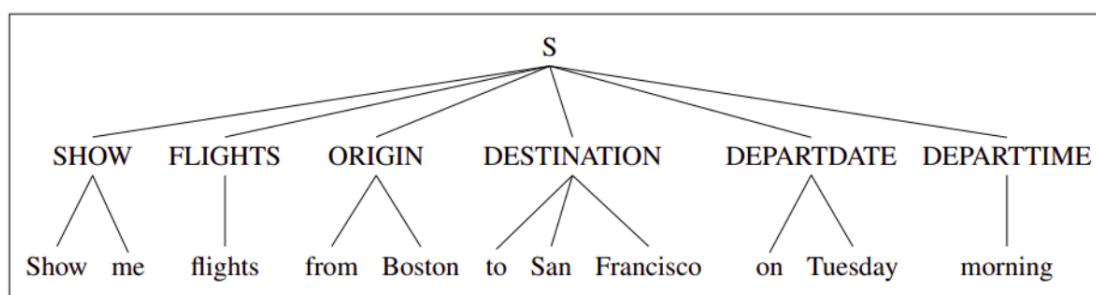
Sử dụng các quy tắc viết tay vẫn còn khá phổ biến trong các ứng dụng công nghiệp, thường là một phần của quy tắc hành động điều kiện gắn liền với các vị trí hoặc khái niệm. Ví dụ: wake me (up) | set (the|an) alarm | get me up.

Các hệ thống dựa trên luật thậm chí có thể được thực hiện với các ngữ pháp đầy đủ. Một ngữ pháp ngữ nghĩa là một ngữ pháp không ngữ cảnh, trong đó phía bên trái của mỗi luật tương ứng với các thực thể ngữ nghĩa được thể hiện như:

SHOW	→ show me i want can i see ...
DEPART_TIME_RANGE	→ (after around before) HOUR morning afternoon evening
HOUR	→ one two three four... twelve (AMPM)
FLIGHTS	→ (a) flight flights
AMPM	→ am pm
ORIGIN	→ from CITY
DESTINATION	→ to CITY
CITY	→ Boston San Francisco Denver Washington

Hình 4.3: Phía bên trái của mỗi luật tương ứng với các thực thể ngữ nghĩa

Các ngữ pháp ngữ nghĩa có thể được phân tích cú pháp bằng bất kỳ thuật toán phân tích ngữ pháp CFG nào, dẫn đến việc ghi nhãn phân cấp của chuỗi đầu vào với nhãn ngữ nghĩa.



Hình 4.4: A semantic grammar parse for a user sentence, using slot names as the internal parse tree nodes

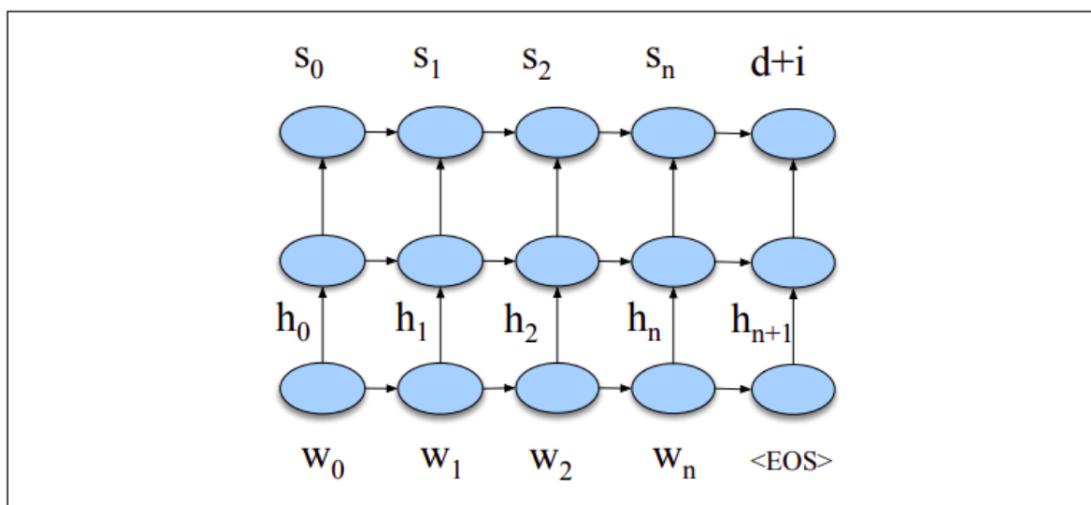
Một số vấn đề khó khăn phải được xử lý. Một vấn đề quan trọng là phủ định; nếu một người dùng chỉ định rằng họ có thể bay vào sáng thứ 3, hoặc muốn gặp một người bất kỳ lúc nào ngoại trừ sáng thứ 3 thì hệ thống đơn giản sẽ trích xuất không chính xác vào sáng thứ 3, làm mục tiêu người dùng thay vì ràng buộc phủ định. Lỗi về nhận dạng giọng nói cũng phải được xử lý. Một mẹo phổ biến được sử dụng trong thực tế là các máy

nhận dạng giọng nói thường trả về một danh sách N-best được xếp hạng của các phiên âm (transcription) giả định thay vì chỉ có 1 transcription ứng viên. Các RE hoặc trình phân tích cú pháp thông thường có thể được chạy đơn giản trên mỗi câu trong danh sách này và mọi mẫu được trích xuất từ bất kỳ giả thuyết nào được sử dụng.

Bên cạnh việc slot filling, đầu tiên ta phải áp dụng 1 bộ phân loại mà sử dụng các đặc trưng phổ biến của các câu để họ dự đoán slot nào mà người dùng muốn fill.

Ngoài các đặc trưng word unigram, bigram và trigram ta phải sử dụng các đặc trưng thực thể có tên hoặc các đặc trưng mà chỉ ra rằng 1 từ là 1 từ vựng đặc biệt (như danh sách các thành phố, các sân bay, hoặc các ngày trong tuần) và bộ phân loại có thể trả về một slot name. Một bộ phân loại thứ 2 sau đó có thể được sử dụng để quyết định filler của named slot.

Một mô hình thay thế là sử dụng mô hình trinh tự (MFMMS, CRFs, RNN) để gán trực tiếp nhãn slot cho mỗi từ trong chuỗi, theo phương pháp được sử dụng cho các mô hình trích xuất thông tin. Một phần nữa chúng ta sẽ cần học máy có giám sát với các câu được ghép với các nhãn IOB (inside-outside-begin).



Hình 4.5: An LSTM architecture for slot filling, mapping the words in the input (represented as 1-hot vectors or as embeddings) to a series of IBO tags plus a final state consisting of a domain concatenated with an intent

Một kiến trúc LSTM thông thường nhận đầu vào là một chuỗi các từ w_1, w_2, \dots, w_n (được đại diện như các vec tơ embedding hoặc như các vector onehot) và đầu ra là một chuỗi IOB.

4.2.3 Evaluating Slot Filling

Một số liệu lỗi intrinsic (nội tại) cho các hệ thống hiểu ngôn ngữ tự nhiên cho việc slot filling là slot error rate cho mỗi câu:

$$\text{Slot Error Rate for a Sentence} = \frac{\# \text{ of inserted/deleted/substituted slots}}{\# \text{ of total reference slots for sentence}}$$

Hình 4.6: Công thức tính Slot Error Rate for a Sentence

4.2.4 Other components of frame-based dialog

Thành phần ASR (tự động nhận diện giọng nói) lấy đầu vào từ một thiết bị và đầu ra là một chuỗi từ đã được transcribed.

Module language generation của bất kỳ hệ thống dialog nào mà tạo ra cách nói mà hệ thống nói với người dùng.

4.3 Evaluating Dialogue Systems

- Tính toán mức độ hạnh phúc của người dùng, từ đó để người dùng tương tác với hệ thống bằng cách yêu cầu họ hoàn thành bảng câu hỏi. Vấn đề của việc làm này là nó không khả thi về mặt kinh tế.
- Task completion success: thành công của nhiệm vụ có thể được đo lường bằng cách đánh giá tính đúng đắn của tổng các giải pháp. Đối với kiến trúc dựa trên frame, đây có thể là tỷ lệ phần trăm của các slot được điền với các giá trị chính xác hoặc tỷ lệ phần trăm của các nhiệm vụ được hoàn thành.
- Efficiency cost: Là các độ đo về hiệu quả của hệ thống trong việc giúp đỡ người dùng. Điều này có thể được đo bằng tổng thời gian hoặc tổng số lượt của hệ thống hoặc tổng số truy vấn.
- Quality cost: Đo lường các khía cạnh khác của các tương tác ảnh hưởng đến nhận thức của người dùng về hệ thống. Ví dụ, số lần hệ thống không trả về bất kỳ câu nào hoặc số lời nhắc từ chối.

Chương 5

Advanced Dialog Systems (Chapter 29)[1]

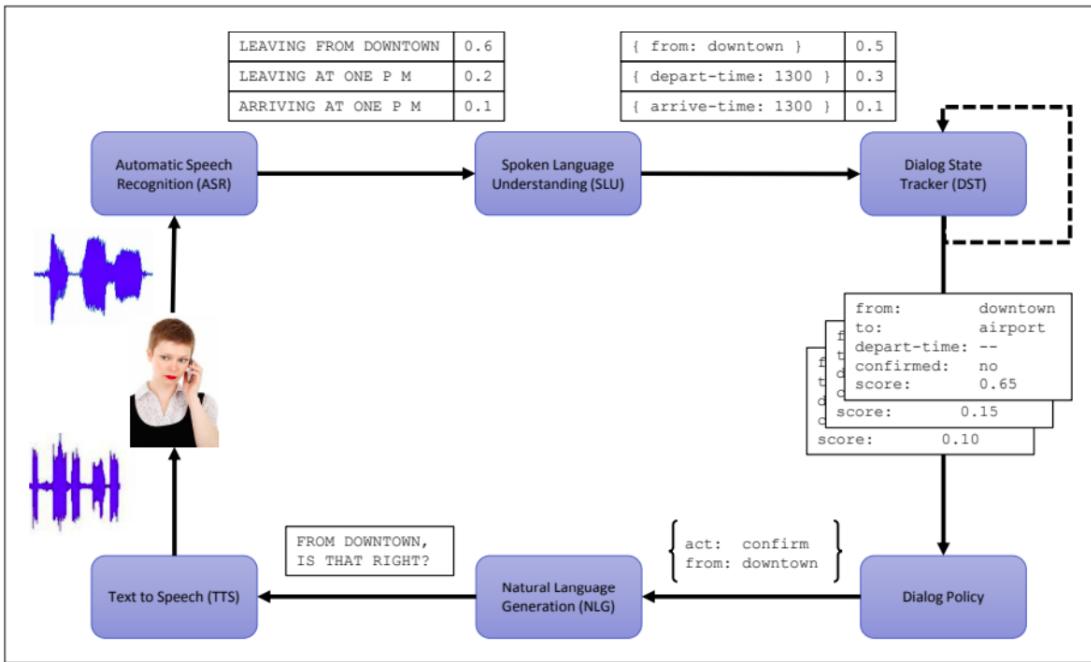
Đặt các câu hỏi, ra lệnh hoặc đưa ra tuyên bố thông tin là những điều mà mọi người làm trong cuộc trò chuyện nhưng xử lý các loại hành động như vậy trong dialog vẫn chưa được xử lý (được gọi là dialog acts). Dialog acts là những điều mà hệ thống dialog dựa trên frame trong chương 29 không có khả năng xử lý.

Trong chương này chúng ta sẽ mô tả kiến trúc dialog-state, cũng được gọi là kiến trúc belief-state hoặc information-state. Giống như các hệ thống dựa trên frame, các agent điền vào các slot, nhưng chúng cũng có khả năng hiểu và tạo ra các dialog acts. Các hành động như việc đặt câu hỏi, đưa ra đề xuất, từ chối một đề nghị,... và chúng có thể kết hợp kiến thức nào vào 1 mô hình phong phú hơn về trạng thái của dialog bất cứ lúc nào.

Như các hệ thống dựa trên frame, kiến trúc dialog-state được dựa trên việc điền vào các slot của các frame và do đó các hệ thống dialog-state có 1 thành phần hiểu ngôn ngữ tự nhiên để quyết định các slot đặc biệt và các filler thể hiện trong một câu của người dùng. Các hệ thống phải quyết định thêm dialog gì mà người dùng đang thực hiện, ví dụ như để theo dõi xem người dùng có đặt câu hỏi không? Và hệ thống này phải tính đến bối cảnh của dialog (những gì hệ thống vừa nói và tất cả các ràng buộc mà người dùng thực hiện trong quá khứ).

Hơn nữa, kiến trúc dialog-state có một cách khác để quyết định nên nói gì tiếp theo so với các hệ thống dựa trên frame. Các hệ thống dựa trên frame đơn giản thường chỉ đặt các câu hỏi liên tục tương ứng đối với các slot chưa được điền và sau đó báo cáo lại kết quả của một số truy vấn CSDL. Nhưng trong cuộc trò chuyện tự nhiên, người dùng đôi khi chủ động, chẳng hạn như việc đặt các câu hỏi cho hệ thống, cách khác, hệ thống có thể không hiểu người dùng nói gì và có thể cần đặt câu hỏi để làm rõ. Hệ thống này cần 1 dialog policy để quyết định cái gì để nói (khi nào trả lời các câu hỏi của người dùng, khi nào nên hỏi người dùng câu hỏi để làm rõ thông tin, đưa ra gợi ý,...).

Hình dưới đây thể hiện một kiến trúc cơ bản cho một hệ thống dialog-state. Nó có 6 thành phần, như với các hệ thống dựa trên frame (các thành phần hiểu và nhận dạng giọng nói; trích xuất ngữ nghĩa từ đầu vào; và các thành phần generation và TTS (Text To Speech) ánh xạ từ ngữ sang lời nói).



Hình 5.1: Architecture of a dialog-state system for task-oriented dialog

5.1 Dialog Acts

Mỗi phát ngôn (utterance) trong một dialog là một loại hành động (action) được người nó thực hiện. Những hành động này được gọi một cách đơn giản là speech acts. Ở đây là một phân loại gồm 4 lớp chính (constative: cam kết; directives: chỉ thị; commissives: cam kết; acknowledgment):

Constatives:	committing the speaker to something's being the case (<i>answering, claiming, confirming, denying, disagreeing, stating</i>)
Directives:	attempts by the speaker to get the addressee to do something (<i>advising, asking, forbidding, inviting, ordering, requesting</i>)
Commissives:	committing the speaker to some future course of action (<i>promising, planning, vowed, betting, opposing</i>)
Acknowledgments:	express the speaker's attitude regarding the hearer with respect to some social action (<i>apologizing, greeting, thanking, accepting an acknowledgment</i>)

Hình 5.2: Dialog Acts

Một người dùng yêu cầu 1 hệ thống dialog để làm 1 cái gì đó (bật nhạc) là một directive. Một người dùng hỏi một câu hỏi mà hệ thống dự kiến sẽ trả lời cũng là directive: theo một nghĩa là người dùng đang ra lệnh cho hệ thống để trả lời. Ngược lại, 1 người dùng nói rõ một ràng buộc (I am flying on Tuesday) là một Assertive. Một người dùng cảm ơn hệ thống là một acknowledgment. Dialog act thể hiện một thành phần quan trọng trong ý định của người nói (hoặc người viết). Trong khi ý tưởng của speech acts là mạnh mẽ, các hệ thống hiện đại mở rộng các phân loại (taxonomies) của speech acts để mô tả tốt hơn các cuộc hội thoại thực tế. Khi thực hiện hành động chung này, người nói với người nghe phải thiết lập liên tục những điểm chung (common ground), tập hợp những điều được cả hai người tin tưởng lẫn nhau.

Sự cần thiết để đạt được điểm có nghĩa là người nghe phải hiểu những phát ngôn của người nói. “Ground” có nghĩa là những hiểu biết để làm rõ rằng người nghe đã hiểu ý nghĩa và ý định của người nói.

Grounding cũng quan trọng khi người nghe cần chỉ ra rằng người nói “chưa” thành công. Nếu người nghe có vấn đề trong việc hiểu, cô ấy phải chỉ ra những vấn đề này cho người nói, một lần nữa sự hiểu biết lẫn nhau cuối cùng có thể đạt được.

Clark Schefer (1989) chỉ ra sự liên tục của các phương thức mà người nghe B có thể sử dụng để nói lên cách nói của người nói, được sắp xếp từ yếu đến mạnh:

Continued attention:	B shows she is continuing to attend and therefore remains satisfied with A's presentation.
Next contribution:	B starts in on the next relevant contribution.
Acknowledgment:	B nods or says a continuer like <i>uh-huh, yeah, or the like</i> , or an assessment like <i>that's great</i> .
Demonstration:	B demonstrates all or part of what she has understood A to mean, for example, by reformulating (paraphrasing) A's utterance or by collaborative completion of A's utterance.
Display:	B displays verbatim all or part of A's presentation.

Hình 5.3: Sự liên tục của các phương thức mà người nghe B có thể sử dụng để nói lên cách nói của người nói, được sắp xếp từ yếu đến mạnh

Từ “uh-huh” là một backchannel, cũng được gọi là 1 “continuer” hoặc một acknowledgement token. 1 backchannel là 1 cách nói tùy chọn (ngắn) thừa nhận nội dung cách nói của người khác và điều đó đòi hỏi sự thừa nhận của người khác.

Speech acts là quan trọng cho các hệ thống dialog thực tế, cần phân biệt 1 statement từ 1 directive và phải phân biệt trong nhiều loại directive để thực hiện 1 điều gì đó từ 1 câu hỏi yêu cầu thông tin. Bên cạnh đó Grounding cũng quan trọng trong các hệ thống dialog.

Ý tưởng của speech acts và grounding đã kết hợp trong 1 loại hành động được gọi là dialog act, 1 thẻ đại diện cho chức năng tương tác của câu được gắn thẻ. Các loại hệ thống dialog khác nhau yêu cầu các loại hành động khác nhau và do đó tập tagset - định nghĩa 1 dialog act có xu hướng thiết lập cho các nhiệm vụ cụ thể.

Hình dưới đây cho thấy 1 bộ tagset cụ thể theo miền cho nhiệm vụ của 2 người lên lịch các cuộc họp:

Tag	Example
THANK	<i>Thanks</i>
GREET	<i>Hello Dan</i>
INTRODUCE	<i>It's me again</i>
BYE	<i>Alright bye</i>
REQUEST-COMMENT	<i>How does that look?</i>
SUGGEST	<i>from thirteenth through seventeenth June</i>
REJECT	<i>No Friday I'm booked all day</i>
ACCEPT	<i>Saturday sounds fine</i>
REQUEST-SUGGEST	<i>What is a good day of the week for you?</i>
INIT	<i>I wanted to make an appointment with you</i>
GIVE_REASON	<i>Because I have meetings all afternoon</i>
FEEDBACK	<i>Okay</i>
DELIBERATE	<i>Let me check my calendar here</i>
CONFIRM	<i>Okay, that would be wonderful</i>
CLARIFY	<i>Okay, do you mean Tuesday the 23rd?</i>
DIGRESS	<i>[we could meet for lunch] and eat lots of ice cream</i>
MOTIVATE	<i>We should go to visit our subsidiary in Munich</i>
GARBAGE	<i>Oops, I-</i>

Hình 5.4: The 18 high-level dialog acts for a meeting scheduling task

Hình 5.4 cho thấy một tập tagset cho hệ thống gọi ý nhà hàng. Hình 5.5 cho thấy các thẻ này ghi nhãn dialog từ hệ thống HIS. Ví dụ này cũng cho thấy nội dung của từng dialog acts, đó là các slot fillers được truyền đạt.

Tag	Sys	User	Description
HELLO($a = x, b = y, \dots$)	✓	✓	Open a dialog and give info $a = x, b = y, \dots$
INFORM($a = x, b = y, \dots$)	✓	✓	Give info $a = x, b = y, \dots$
REQUEST($a, b = x, \dots$)	✓	✓	Request value for a given $b = x, \dots$
REQALTS($a = x, \dots$)	✗	✓	Request alternative with $a = x, \dots$
CONFIRM($a = x, b = y, \dots$)	✓	✓	Explicitly confirm $a = x, b = y, \dots$
CONFREQ($a = x, \dots, d$)	✓	✗	Implicitly confirm $a = x, \dots$ and request value of d
SELECT($a = x, a = y$)	✓	✗	Implicitly confirm $a = x, \dots$ and request value of d
AFFIRM($a = x, b = y, \dots$)	✓	✓	Affirm and give further info $a = x, b = y, \dots$
NEGATE($a = x$)	✗	✓	Negate and give corrected value $a = x$
DENY($a = x$)	✗	✓	Deny that $a = x$
BYE()	✓	✓	Close a dialog

Hình 5.5: Dialogue acts used by the HIR restaurant recommendation system

Utterance	Dialogue act
U: Hi, I am looking for somewhere to eat.	hello(task = find, type=restaurant)
S: You are looking for a restaurant. What type of food do you like?	confreq(type = restaurant, food)
U: I'd like an Italian somewhere near the museum.	inform(food = Italian, near=museum)
S: Roma is a nice Italian restaurant near the museum.	inform(name = "Roma", type = restaurant, food = Italian, near = museum)
U: Is it reasonably priced?	confirm(pricerange = moderate)
S: Yes, Roma is in the moderate price range.	affirm(name = "Roma", pricerange = moderate)
U: What is the phone number?	request(phone)
S: The number of Roma is 385456.	inform(name = "Roma", phone = "385456")
U: Ok, thank you goodbye.	bye()

Hình 5.6: A sample dialog from the HIS system using the dialog acts in Fig. 5.4

Dialog acts không chỉ xuất hiện một cách riêng biệt và độc lập, các cuộc hội thoại có cấu trúc và dialog acts phản ánh một số cấu trúc đó. Một khía cạnh của cấu trúc này đến từ lĩnh vực conversational analysis (CA) mà tập trung vào các thuộc tính tương tác của cuộc trò chuyện của con người. CA định nghĩa các adjacency pairs như một cặp của 2 dialog acts (như Question and Answer, Proposal and Acceptance (hoặc Rejection)).

5.2 Dialog State: Interpreting Dialog Acts

Công việc của dialog-state tracker là để xác định cả trạng thái hiện tại của frame (các filler của mỗi slot) cũng như dialog act gần nhất của người dùng. Lưu ý rằng dialog-state bao gồm nhiều hơn chỉ các slot-filler được thể hiện trong câu hiện tại. Nó bao gồm toàn bộ trạng thái của frame tại thời điểm này, tóm tắt tất cả các ràng buộc của người dùng. Ví dụ, cho thấy đầu ra yêu cầu của dialog-state tracker sau mỗi lượt:

```
User: I'm looking for a cheaper restaurant
      inform(price=cheap)
System: Sure. What kind - and where?
User: Thai food, somewhere downtown
      inform(price=cheap, food=Thai, area=centre)
System: The House serves cheap Thai food
User: Where is it?
      inform(price=cheap, food=Thai, area=centre); request(address)
System: The House is at 106 Regent Street
```

Hình 5.7: Ví dụ, cho thấy đầu ra yêu cầu của dialog-state tracker sau mỗi lượt

Làm thế nào để chúng ta có thể diễn giải 1 dialog act, quyết định xem 1 đầu vào nhất định là “câu hỏi”, “statement” hay một “suggest”. Cú pháp bề mặt có vẻ như là một gợi ý hữu ích vì câu hỏi yes-no trong tiếng anh có trợ từ đảo ngược, các statement có cú pháp khai báo (declarative syntax) (không có đảo ngược) và các command không có chủ đề cú pháp (syntactic subject).

Việc ánh xạ từ dạng bề mặt sang dialog act là phức tạp. Ví dụ, các nói sau đây có vẻ như về mặt ngữ pháp giống như câu hỏi yes-no:

Can you give me a list of the flights from Atlanta to Boston?

Tuy nhiên, trên thực tế, người này không quan tâm đến việc hệ thống có khả năng đưa ra danh sách hay không. Các nói này là một hình thức lịch sự của 1 yêu cầu, có nghĩa là 1 cái gì đó như “Please give me a list of...”. Những gì thấy trên bề mặt như 1 Question thực sự có thể là 1 Request.

Ngược lại, những gì nhìn thấy trên bề mặt như 1 statement thực sự có thể là 1 Question. Câu hỏi Check rất phổ biến yêu cầu 1 người đối thoại để xác nhận 1 cái gì đó. Checks có declarative surface dạng:

A OPEN-OPTION	I was wanting to make some arrangements for a trip that I'm going to be taking uh to LA uh beginning of the week after next.
B HOLD	OK uh let me pull up your profile and I'll be right with you here. [pause]
B CHECK	And you said you wanted to travel next week?
A ACCEPT	Uh yes.

Hình 5.8: Dạng declarative surface của Checks

Những đặc trưng có thể giúp trong tác vụ này. Ví dụ, trong các hệ thống spoken-language, intonation (ngữ điệu) là tên của một tập hợp các khía cạnh âm vị học cụ thể của giọng nói báo hiệu giai điệu và các thay đổi trong cao độ (accent, stress, hoặc loudness và những thay đổi về thời lượng và tốc độ nói).

5.2.1 Sketching an algorithm for dialog act interpretation

Vì các dialog act đặt một vài ràng buộc trên các slot và value nên các nhiệm vụ phát hiện và slot-filling thường được thực hiện cùng nhau. Xem xét nhiệm vụ determining:

I'd like Cantonese food near the Mission District

có cấu trúc:

inform(food=cantonese, area=mission)

Thuật toán dialog act interpretation/slot filling kết hợp thường bắt đầu với một trình phân loại pass đầu tiên để quyết định dialog act cho các câu. Trong ví dụ trên, bộ phân loại này sẽ chọn inform trong số tập dialog acts có thể trong tag set cho nhiệm vụ cụ thể này. Dialog act interpretation thường được mô hình hóa như một nhiệm vụ phân loại có giám sát, được huấn luyện trên 1 kho văn bản, trong đó mỗi phát ngôn được dán nhãn bằng tay cho dialog act của mình và đưa vào những đặc trưng khác nhau, bao gồm cả unigram, bigram, parse features, dấu câu, dialog context,...

Trình phân loại pass thứ 2 có thể sử dụng bất kỳ thuật toán nào để trích xuất slot-filler trong chương 29. Ngoài ra, một trình phân loại multinomial có thể sử dụng để chọn giữa tất cả các cặp slot-value có thể, 1 lần nữa sử dụng bất kỳ chức năng đặc trưng nào trong chương 29.

5.2.2 A special case: detecting correction acts

1 số dialog act là quan trọng bởi vì ý nghĩa của chúng cho điều khiển dialog. Nếu 1 hệ thống dialog nhận dạng sai hoặc hiểu sai cách nói, người dùng sẽ sửa lỗi bằng cách lặp lại hoặc thay đổi cách nói. Do đó, việc phát hiện hành vi chỉnh sửa của người dùng (user correction acts) khá quan trọng. Trớ trêu thay, hóa ra các hình vi sửa đổi của người dùng thực sự khó nhận ra hơn các câu thông thường. Các correction trong các hệ thống dialog ban đầu đã tăng gấp đôi tỉ lệ lỗi “từ” ASR của các non-correction.

User corrections có xu hướng là sự lặp lại chính xác hoặc sự lặp lại với 1 hoặc nhiều từ bị bỏ qua, mặc dù chúng cũng có thể là cách diễn đạt khác của cách nói ban đầu. Việc phát hiện các correction acts có thể được làm bởi bất kỳ trình phân loại nào; một số đặc trưng tiêu chuẩn cho nhiệm vụ này là:

lexical features	words like “no”, “correction”, “I don’t”, or even swear words, utterance length
semantic features	overlap between the candidate correction act and the user’s prior utterance (computed by word overlap or via cosines over embedding vectors)
phonetic features	phonetic overlap between the candidate correction act and the user’s prior utterance (i.e. “WhatsApp” may be incorrectly recognized as “What’s up”)
prosodic features	hyperarticulation, increases in F0 range, pause duration, and word duration, generally normalized by the values for previous sentences
ASR features	ASR confidence, language model probability

Hình 5.9: Một số đặc trưng tiêu chuẩn cho việc phát hiện các correction acts

5.3 Dialogue Policy

Mục tiêu của dialogue policy là để quyết định hệ thống nào sẽ thực hiện hành động tiếp theo, nghĩa là, dialog act sẽ được tạo ra tiếp theo.

5.3.1 Generating Dialogue Acts: Confirmation and Rejection

Các hệ thống dialog hiện đại thường mắc những sai lầm. Do đó, điều quan trọng đối với các hệ thống dialog là đảm bảo rằng chúng đã đạt được sự giải thích chính xác về đầu vào của người dùng. Điều này thường được thực hiện bằng 2 phương pháp: xác nhận sự hiểu biết với người dùng và từ chối những phát ngôn mà hệ thống có thể đã hiểu sai.

Các chiến lược khác nhau có thể được sử dụng để xác nhận với người dùng. Khi sử dụng chiến lược xác nhận rõ ràng (explicit confirmation), 1 hệ thống yêu cầu người dùng 1 câu hỏi trực tiếp xác nhận việc hiểu biết của hệ thống.

Khi sử dụng chiến lược implicit (ngầm) confirmation, thay vào đó hệ thống sẽ sử dụng các chiến lược grounding display được mô tả bên trên, lặp lại sự hiểu biết của hệ thống như 1 phần của việc đặt câu hỏi tiếp theo.

Explicit và Implicit confirmation có sức mạnh bổ sung. Explicit confirmation giúp người dùng dễ dàng sửa chữa các nhận thức sai lầm của hệ thống vì người dùng có thể trả lời “no” cho câu hỏi xác nhận. Nhưng Explicit confirmation là khó triển khai và tăng thời lượng cuộc trò chuyện. Xác nhận chỉ là một loại dialog act mà theo đó 1 hệ thống có thể thể hiện sự thiếu hiểu biết. 1 tùy chọn khác là rejection, trong đó hệ thống cung cấp cho người dùng lời nhắc như “I’m sorry” hoặc “I didn’t understand”.

Đôi khi các utterance bị từ chối nhiều lần. Điều này có ý nghĩa là người dùng đang sử dụng sai ngôn ngữ mà hệ thống có thể theo dõi. Do đó, khi 1 phát ngôn bị từ chối, các

hệ thống thường tuân theo chiến lược progressive prompting (thay vì lặp lại câu hỏi thì sử dụng cách diễn đạt khác):

```
System: When would you like to leave?  
Caller: Well, um, I need to be in New York in time for the first World Series game.  
System: <reject>. Sorry, I didn't get that. Please say the month and day you'd like  
to leave.  
Caller: I wanna go on October fifteenth.
```

Hình 5.10: Ví dụ về chiến lược progressive prompting

Một chiến lược thay thế cho việc xử lý lỗi là rapid reprompting, trong đó hệ thống từ chối một phát ngôn chỉ bởi câu nói: “I'm sorry” hoặc “what was that”. Chỉ khi phát ngôn của người gọi bị từ chối lần thứ 2, hệ thống mới bắt đầu áp dụng progressive prompting (nhắc nhở tiến bộ).

Nhiều yếu tố có thể được sử dụng làm đặc trưng cho dialog policy trong việc quyết định nên sử dụng explicit confirmation, implicit confirmation hay rejection. Ví dụ, confidence mà hệ thống ASR gán tới một lời thoại có thể được sử dụng bởi explicitly confirmation cho các câu có độ tin cậy thấp.

5.4 Natural language generation in the dialog-state model

Khi 1 dialog act đã được quyết định, ta cần tạo text để trả lời cho người dùng. Nhiệm vụ tạo ngôn ngữ tự nhiên (NLG) trong kiến trúc information-state thường được mô hình hóa theo hai giai đoạn: content planning (phải nói gì) và sentence realization (nói như thế nào).

Ở đây, ta sẽ sử giả định việc content planning đã được thực hiện bởi dialog policy, đã chọn dialog act để tạo và có lẽ cũng chọn 1 số thuộc tính bổ sung (slots và values) mà trình lên kế hoạch (planner) muốn implicitly confirm tới người dùng.

Ví dụ về cấu trúc đầu vào của policy/content planner:

```
{  
    act query  
    content depart_time  
    depart_date {  
        year 2000  
        month 10  
        day 5  
    }  
    depart_airport BOS  
}  
=> What time on October fifth would you like to leave Boston?
```

Hình 5.11: An input frame to NLG and a resulting output sentence, in the Communicator system

rước tiên lưu ý rằng policy đã quyết định tạo ra dialog act query với đối số Depart_time. Frame đầu vào trong hình 5.11 cũng chỉ định 1 số vị trí filled slots mà nên được bao gồm trong câu cho người dùng (depart_airport BOS và depart_date).

Sentence realizer hoạt động với 2 bước. Đầu tiên nó sẽ tạo 1 delexicalized string như:

What time on [depart_date] would you like to leave [depart_airport]?

Delexicalization là quá trình thay thế các từ cụ thể với 1 generic representation các loại slot của chúng. 1 delexicalization sentence là dễ dàng hơn nhiều để tạo ra vì chúng ta có thể huấn luyện nhiều các câu nguồn khác nhau từ các date và airport cụ thể khác nhau. Sau đó khi ta tạo ra chuỗi delexicalized, ta có thể sử dụng đơn giản input frame từ content planner để relexicalize (điền ngày khởi hành và sân bay chính xác).

Để tạo delexicalized sentences, sentence realizer sử dụng 1 kho dữ liệu lớn các dialog du lịch giữa người với người mà được gán nhãn với các dialog act.

1 bộ ngữ pháp N-gram sau được huấn luyện cho mỗi dialog act. Với mỗi dialog query depart_time đã cho, hệ thống lấy mẫu ngẫu nhiên các câu từ mô hình ngôn ngữ.

Sau đó, mỗi câu được lấy mẫu ngẫu nhiên này sẽ được gán 1 trọng số dựa trên các quy tắc heuristic để mà phạt các câu quá ngắn hoặc quá dài, lặp lại các vị trí hoặc thiếu một số vị trí bắt buộc từ frame đầu vào. Câu có trọng số cao nhất sau đó sẽ được chọn.

Hầu hết các công việc gần đây đã thay thế phần N-gram đơn giản của generator với các mô hình nơ ron mà học cách ánh xạ từ các frame đầu vào sang các câu kết quả.

5.5 Advanced: Markow Decision Processes

Policy ta đã mô tả, quyết định những hành động nào mà hệ thống nên thực hiện chỉ dựa trên các filled slot hiện tại và câu nói cuối cùng của người dùng, nhưng có 1 vấn đề là nó chỉ nhìn vào quá khứ của dialog mà hoàn toàn bỏ qua rằng liệu hành động thực hiện có dẫn đến kết quả thành công hay không?

Nhưng ta không thể biết được liệu kết quả có thành công hay không cho đến lời thoại hiện tại. Reinforcement learning là 1 nhánh của học máy liên quan đến các mô hình mà học để tối đa hóa các phần thưởng trong tương lai.

1 Markov decision hoặc MDP được đặc trưng bởi 1 tập hợp các trạng thái S mà 1 tác nhân có thể tham gia, 1 tập hợp các hành động A mà tác nhân có thể thực hiện và 1 phần thưởng $r(a, s)$ mà tác nhân có thể nhận được khi thực hiện một hành động trong 1 trạng thái. Dựa trên các yếu tố này, chúng ta có thể tính toán 1 policy P_i mà chỉ định hành động mà tác nhân nên thực hiện khi ở trạng thái nhất định để nhận phần thưởng tốt nhất.

1 Markov decision hoặc MDP được đặc trưng bởi 1 tập hợp các trạng thái S mà 1 tác nhân có thể tham gia, 1 tập hợp các hành động A mà tác nhân có thể thực hiện và 1 phần thưởng $r(a, s)$ mà tác nhân có thể nhận được khi thực hiện một hành động trong 1 trạng thái. Dựa trên các yếu tố này, chúng ta có thể tính toán 1 policy P_i mà chỉ định hành động mà tác nhân nên thực hiện khi ở trạng thái nhất định để nhận phần thưởng tốt nhất.

Về nguyên tắc này, 1 trạng thái của 1 MDP có thể bao gồm mọi thông tin có thể về dialog, như lịch sử dialog hoàn chỉnh cho đến hiện tại. Sử mô hình trạng thái phong phú như vậy sẽ làm cho số lượng các trạng thái có thể cực kỳ lớn. Vì vậy 1 mô hình trạng thái thường được lựa chọn để mã hóa 1 tập hợp thông tin hạn chế hơn nhiều, chẳng hạn như

các giá trị của các vị trí trong frame hiện tại, câu hỏi gần đây nhất được hỏi cho người dùng, câu trả lời gần nhất của người dùng, độ tin cậy ASR,...

Về nguyên tắc này, 1 trạng thái của 1 MDP có thể bao gồm mọi thông tin có thể về dialog, như lịch sử dialog hoàn chỉnh cho đến hiện tại. Sử mô hình trạng thái phong phú như vậy sẽ làm cho số lượng các trạng thái có thể cực kỳ lớn. Vì vậy 1 mô hình trạng thái thường được lựa chọn để mã hóa 1 tập hợp thông tin hạn chế hơn nhiều, chẳng hạn như các giá trị của các vị trí trong frame hiện tại, câu hỏi gần đây nhất được hỏi cho người dùng, câu trả lời gần nhất của người dùng, độ tin cậy ASR,...

Các hành động của hệ thống dialog MDP phải bao gồm việc tạo các hành vi lời nói cụ thể hoặc thực hiện truy vấn CSDL để tìm thông tin.

Vì mục tiêu của hệ thống là có được câu trả lời chính xác với ít tương tác nhất, 1 function phần thưởng có thể cho hệ thống sẽ như sau:

$$R = - (w_i n_i + w_e n_e + w_f n_f)$$

Trong đó, n_i là số lượng tương tác với người dùng, n_e là số lượng lỗi, n_f là số lượng các slot được điền và các w là các trọng số.

Cuối cùng, dialog policy P_i chỉ định hành động nào sẽ được áp dụng ở trạng thái nào.

Nhìn chung, số lượng hành động, trạng thái và policy có thể có là khá lớn và do đó, vẫn đề tìm kiếm policy tối ưu P_i^* khó khăn hơn nhiều.

Lý thuyết Markov desicion cùng với việc học tăng cường cổ điển cho ta 1 cách suy nghĩ về vấn đề này.

Policy tốt nhất P_i^* là policy có phần thưởng được mong đợi lớn nhất. Phần thưởng dự kiến cho một chuỗi trạng thái nhất định là gì? 1 cách phổ biến nhất để gán phần thưởng cho 1 chuỗi là sử dụng discounted rewards. Chúng ta tính toán phần thưởng tích lũy dự kiến Q của 1 chuỗi dưới dạng tổng discounted của các phần thưởng trong từng trạng thái.

$$Q([s_0, a_0, s_1, a_1, s_2, a_2 \dots]) = R(s_0, a_0) + \gamma R(s_1, a_1) + \gamma^2 R(s_2, a_2) + \dots$$

Nhân tố discount Gamma là một số nằm trong khoảng 0-1. Điều này làm cho các tác nhân quan tâm nhiều hơn về phần thưởng hiện tại hơn là phần thưởng trong tương lai; phần thưởng trong tương lai càng nhiều thì giá trị của nó càng giảm.

Với mô hình này, có thể chỉ ra rằng phần thưởng tích lũy dự kiến $Q(s, a)$ khi thực hiện 1 trạng thái cụ thể là phương trình đệ quy được gọi là phương trình Bellaman:

$$Q(s, a) = R(s, a) + \gamma \sum_{s'} P(s'|s, a) \max_{a'} Q(s', a')$$

Phương trình Bellaman nói rằng phần thưởng tích lũy dự kiến cho 1 cặp trạng thái/hành động nhất định là phần thưởng ngay lập tức cho trạng thái hiện tại cộng với phần thưởng discounted dự kiến của tất cả các trạng thái tiếp theo s^t , được đánh trọng số bằng xác suất của việc di chuyển từ trạng thái s^t đó và giả sử rằng khi đí chúng ta thực hiện hành động tối ưu là a^t .

Trong phương trình Bellaman sử dụng 2 tham số, ta cần 1 mô hình $P(s^t|s, a)$ nghĩa là khả năng 1 cặp trạng thái/hành động nhất định (s, a) sẽ dẫn đến trạng thái s^t . Và chúng ta cũng cần 1 ước tính tốt về $R(s, a)$. Nếu ta có nhiều dữ liệu huấn luyện, ta có thể tính toán cả 2 tham số này từ dữ liệu.

Trong thực tế, MDP chỉ hữu ích trong các ví dụ nhỏ và không được sử dụng trong thực tế.

Tài liệu tham khảo

- [1] Daniel Jurafsky and James H. Martin. *Speech and Language Processing*. 2017.