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

Phân tích cú pháp phụ thuộc

Lê Thanh Hương
Bộ môn Hệ thống Thông tin
Viện CNTT &TT – Trường ĐHBKHN
Email: huonglt@soict.hust.edu.vn

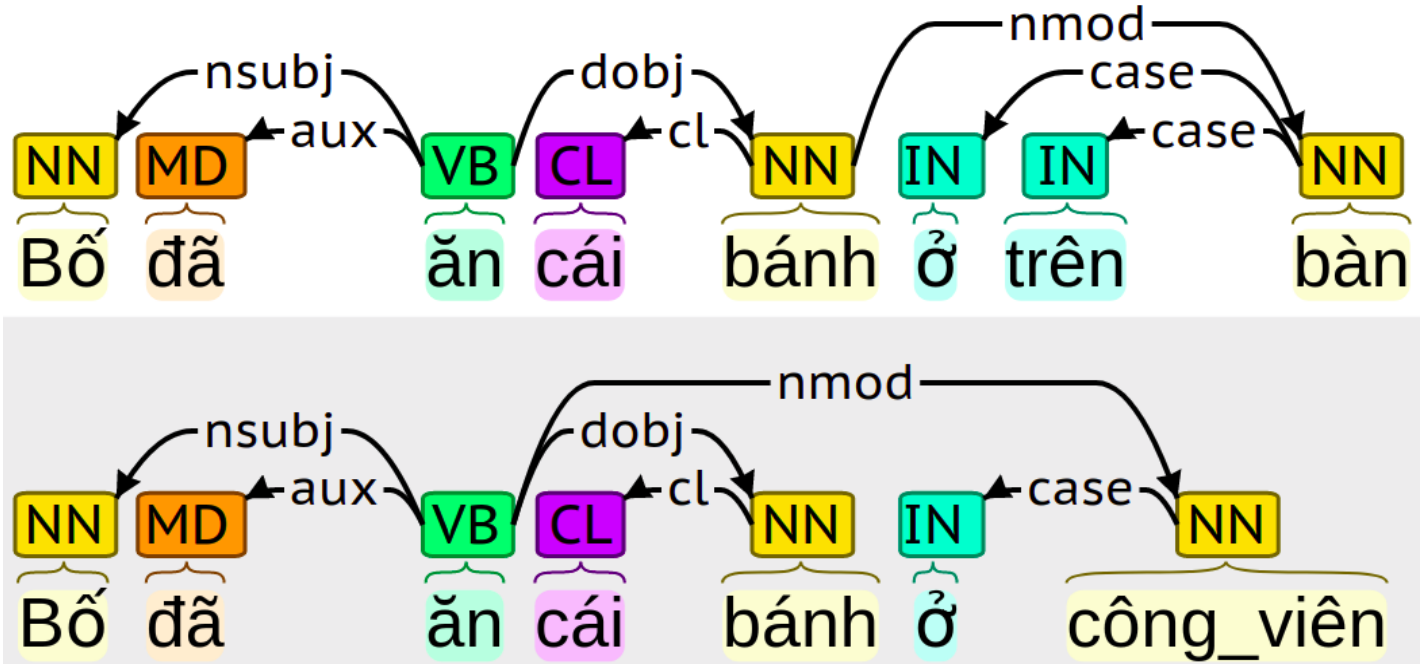
Các nội dung chính

1. **Tổng quan về Dependency Parsing.**
   a. **Định nghĩa**
   b. Ứng dụng
   c. Tính chất
2. Các phương pháp giải quyết bài toán.
   a. Transition-based
   b. Graph-based
   c. Các cách tiếp cận hiện nay
3. Các kết quả cài đặt.

Dependency Parsing là gì

- Phân tích cú pháp phụ thuộc
- Không phân tích chủ ngữ, vị ngữ, các cụm danh từ, cụm động từ,… thay vì đó, phân tích quan hệ phụ thuộc giữa các từ trong câu với nhau.
- Thường liên quan chặt chẽ đến bài toán Gán nhãn từ loại (Part Of Speech Tagging)
- Được bắt đầu quan tâm nhiều từ thập kỷ trước do sự giàu thông tin mà kiểu phân tích này mang lại.

Dependency Parsing là gì



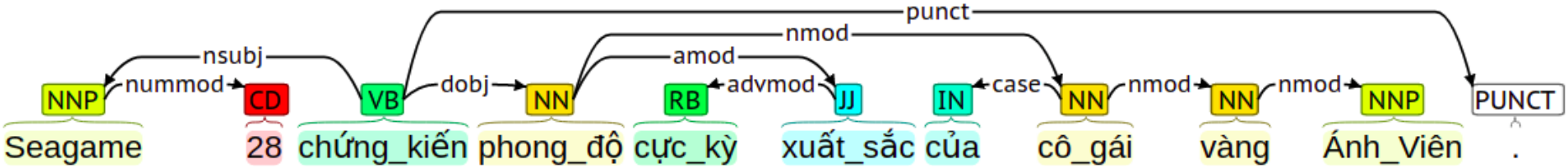
Dependency Parsing là gì

Một quan hệ phụ thuộc thể hiện bằng 1 mũi tên có hướng, trong đó:

- **head**: đầu không có mũi tên, là từ được bổ nghĩa
- **dependent**: đầu có mũi tên, là từ bổ nghĩa
- **label**: quan hệ phụ thuộc giữa 2 từ này.

Các nhãn phụ thuộc

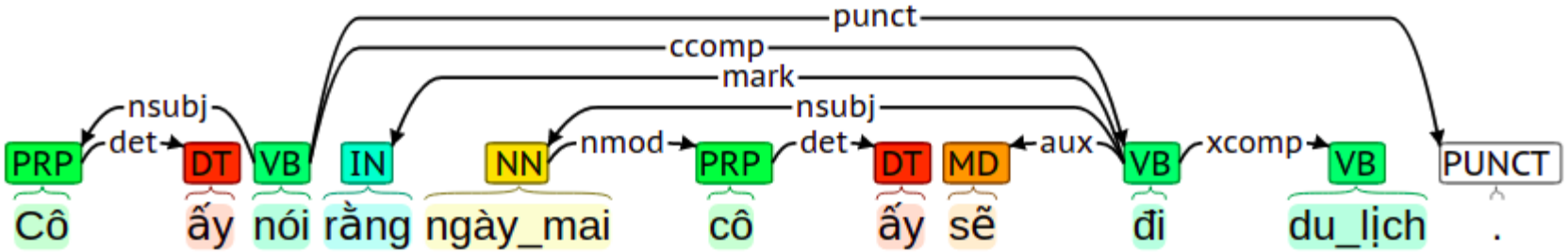
- Một số nhãn phụ thuộc:
  - nsubj (Nominal subject): chủ ngữ, chủ thể
  - dobj (Direct object): tân ngữ trực tiếp
  - nmod (Nominal modifier): danh từ bổ nghĩa
  - amod (Adjectival modifier): tính từ bổ nghĩa
  - nummod (Numeric modifier): số từ bổ nghĩa



Các nhãn phụ thuộc

- Một số nhãn phụ thuộc:
  - ccomp (Clausal component): Mệnh đề thành phần
  - xcomp (Open clausal component): Mệnh đề thành phần mở rộng
  - aux (Auxiliary): phụ từ, trợ động từ

Xem thêm: http://universaldependencies.org/u/dep/

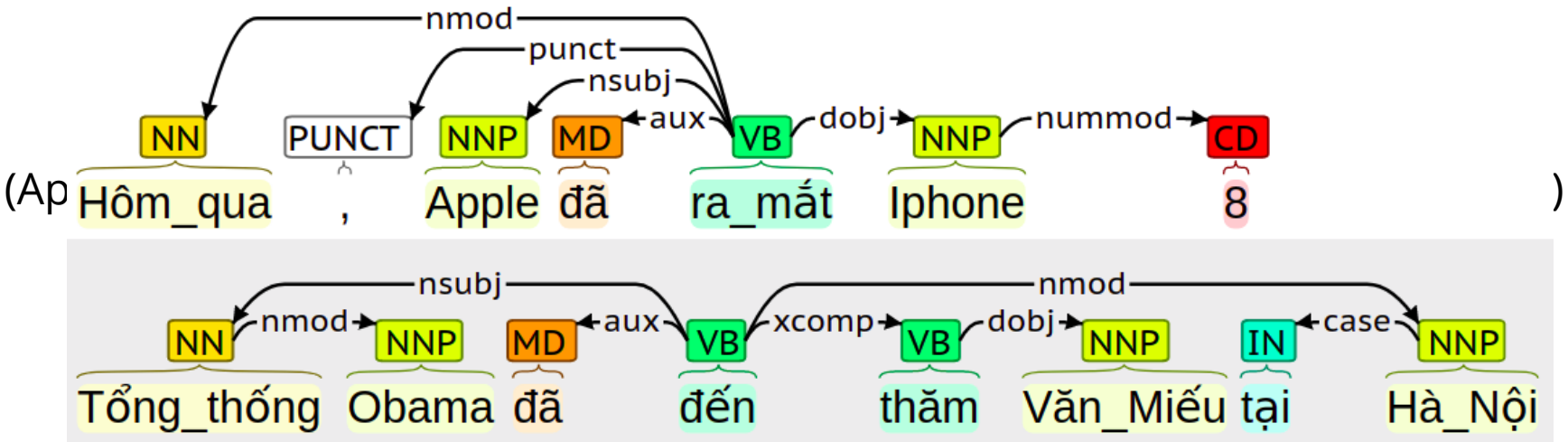


Tổng quan về Dependency Parsing

- Định nghĩa
- **Các ứng dụng của Dependency Parsing**
- Các tính chất của cây cú pháp phụ thuộc

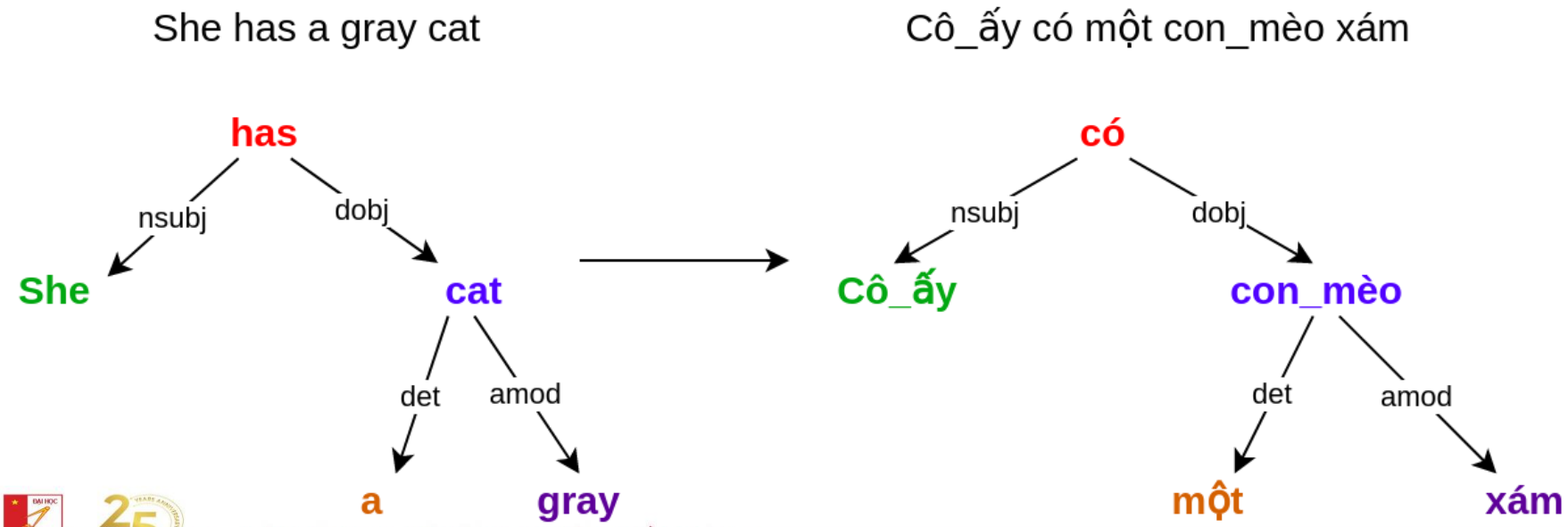
Ứng dụng.

Xây dựng cơ sở tri thức dựa trên Trích rút quan hệ (Relation Extraction)



Ứng dụng.

Dịch máy (Machine Translation)

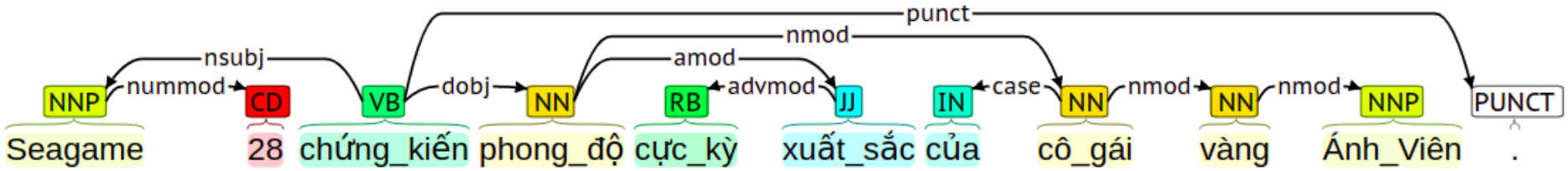


Các tính chất của cây cú pháp phụ thuộc

- Xét cây cú pháp là 1 đồ thị với các từ là các đỉnh (node), các quan hệ là các cạnh (arc)
- Đồ thị cú pháp phụ thuộc này có 4 tính chất:
  - Weakly Connected (Kết nối yếu)
  - Acyclic (Không có chu trình)
  - Single head (1 từ chỉ có duy nhất 1 head)
  - Projective

Các tính chất của cây cú pháp phụ thuộc

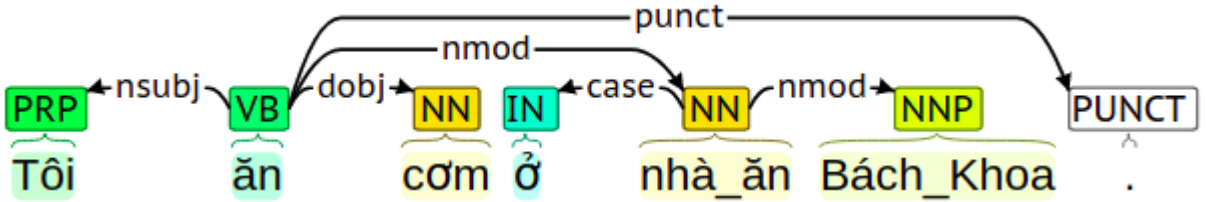
- Weakly Connected:
  - Với mọi node i, luôn tồn tại 1 node j sao cho có 1 cạnh nối i -> j hoặc j -> i
- Acyclic:
  - Nếu tồn tại cạnh i -> j, thì không thể tồn tại 1 đường đi j ->* i
- Single head:
  - Nếu có cạnh i -> j, thì sẽ không có cạnh k -> j, với k != i



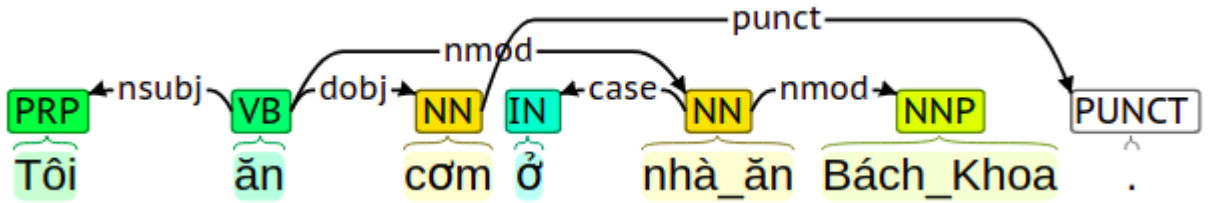
Các tính chất của cây cú pháp phụ thuộc

- Projective: (tính chất này không bắt buộc)
  - Nếu tồn tại cạnh i -> j, thì với mọi k nằm giữa i và j, luôn có đường đi i ->* k
  - Một cách trực quan, không có cạnh chéo nhau khi vẽ cây cú pháp tuần tự theo câu

Projective



Non-Projective



Các nội dung chính

1. Tổng quan về Dependency Parsing.
   a. Định nghĩa
   b. Ứng dụng
   c. Tính chất
2. **Các phương pháp giải quyết bài toán.**
   a. Transition-based
   b. Graph-based
   c. Các cách tiếp cận hiện nay
3. Các kết quả cài đặt.

Các phương pháp giải quyết.

- **Transition-based**
  - Thuật toán Nivre
- Graph-based
- Các cách tiếp cận hiện nay
  - End to end learning
  - Joint learning

Transition-based

- Ý tưởng cơ bản của nó là dựa trên các Transition (SHIFT, REDUCE, LEFT-ARC, RIGHT-ARC)
- Khi đọc câu từ trái sang phải, mô hình học máy của hệ thống sẽ quyết định thực hiện transition nào, dãy các transition này giúp xác định được quan hệ phụ thuộc giữa các từ trong câu
- Việc quyết định transition nào sẽ do 1 mô hình học máy (classifier) đảm nhận, cần huấn luyện mô hình này

Transition-based

- Thuật toán phân tích cú pháp: Nivre, Covington, ...
- Phương pháp huấn luyện Classifier: SVM, Neural network, ...

Thuật toán Nivre

- Là thuật toán phân tích cú pháp cho phương pháp Transition-based.
- Được Joakim Nivre đề xuất từ những năm 2003-2004
- Là thuật toán phân tích phổ biến nhất hiện tại trong hướng Transition-based, và cũng có nhiều biến thể khác nhau.
- Độ phức tạp O(n)

Thuật toán Nivre

- Cấu hình c: c = (**Σ**|s, b|**B**, **A**), gồm có 3 phần
  - 1 Stack **Σ** chứa các từ từng được xét, và sẽ còn được xét tiếp.
  - 1 Buffer **B** chứa các từ chưa được xét, hoặc mới xét đến.
  - 1 tập **A** chứa các quan hệ phụ thuộc đã tìm ra
  - Thể hiện trạng thái hiện tại của hệ thống
- Các transition sẽ dựa trên cấu hình hiện tại để đi đến cấu hình mới, cũng sẽ gồm 3 thành phần này.

Thuật toán Nivre

- 4 transition được định nghĩa như sau:
  - SHIFT [(**Σ**, b|**B**, **A**)] = (**Σ**|b, **B**, **A**)
  - RIGHT_{lb} [(**Σ**|s, b|**B**, **A**)] = (**Σ**|s|b, **B**, **A** ∪ {s, lb, b})
  - LEFT_{lb} [(**Σ**|s, b|**B**, **A**)] = (**Σ**, b|**B**, **A** ∪ {b, lb, s})
  - REDUCE [(**Σ**|s, **B**, **A**)] = (**Σ**, **B**, **A**)
- Có thể mô tả tương ứng như sau:
  - **SHIFT**: chuyển từ ở đầu buffer lên đỉnh của stack, không thêm quan hệ
  - **RIGHT**: thêm từ ở đầu buffer vào đỉnh stack, thêm quan hệ phụ thuộc (s, lb, b)
  - **LEFT**: bỏ từ ở đỉnh stack ra, giữ nguyên buffer, thêm quan hệ phụ thuộc (b, lb, s)
  - **REDUCE**: bỏ từ ở đỉnh Stack đi, không thêm quan hệ nào

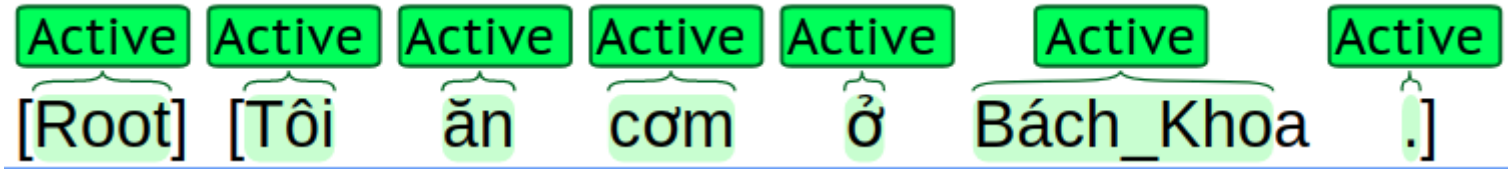
Thuật toán Nivre

- Câu đầu vào là W = w_1, w_2, ..., w_n. (w_i là từ thứ i trong câu)
- Cấu hình khởi tạo: c_{init} = (**Σ**, **B**, **A**)
  - **Σ**: chỉ chứa ROOT
  - **B**: **B** = w_1, w_2, ..., w_n
  - **A**: tập rỗng
- Cấu hình kết thúc: $c_{terminal}$ = (**Σ**, **B**, **A**)
  - **Σ**: chỉ chứa ROOT
  - **B**: rỗng
  - **A**: tập chứa quan hệ phụ thuộc.

Thuật toán Nivre

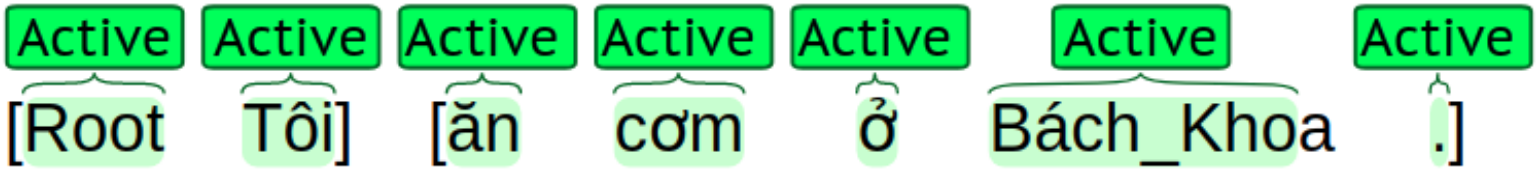
1: **Input**: sentence W , parameter-vector w
2: c ← **INITIAL** (W)
3: **while** not **TERMINAL** (c) do
4: $t_p \leftarrow \arg\max_{t \in LEGAL(c)} w \cdot \phi(c, t)$
5: c ← t_p (c)
6: **return** A_c

Minh họa Nivre.



- Câu đầu vào: **Tôi ăn cơm ở Bách_Khoa** .
- Stack: Ngoặc vuông bên trái
- Buffer: Ngoặc vuông bên phải.
- A : tập quan hệ phụ thuộc đang rỗng
- Active: Nút vẫn đang còn được xét.
- Deleted: Nút đã xét xong, loại bỏ khỏi Stack

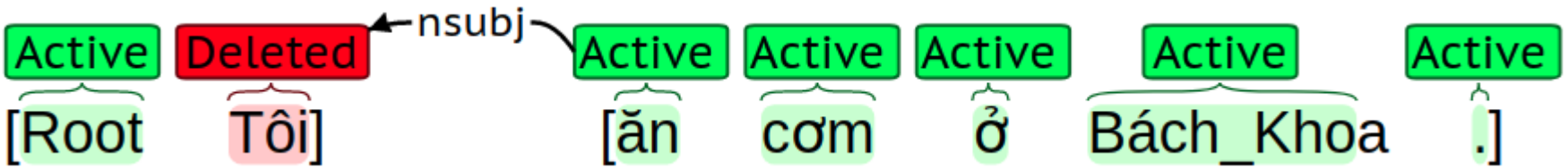
Minh họa Nivre.



SHIFT: chuyển 'Tôi' từ Buffer sang Stack

A = {}

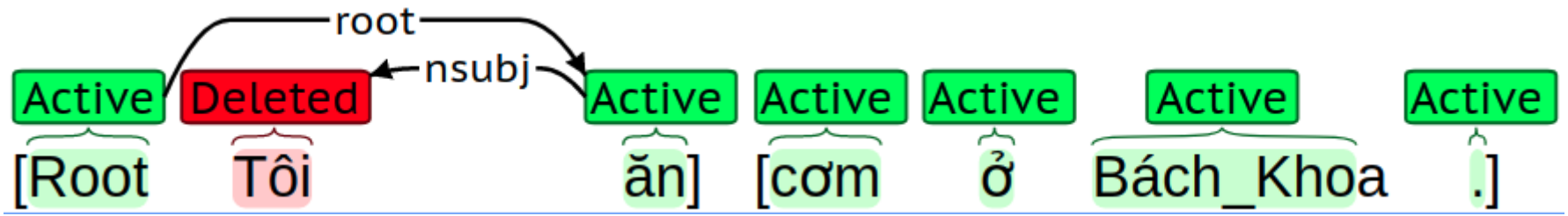
Minh họa Nivre.



$LEFT_{nsubj}$: Xóa 'Tôi' khỏi Stack, thêm (ăn, nsubj, Tôi) vào tập A

A= {(ăn, nsubj, Tôi)}

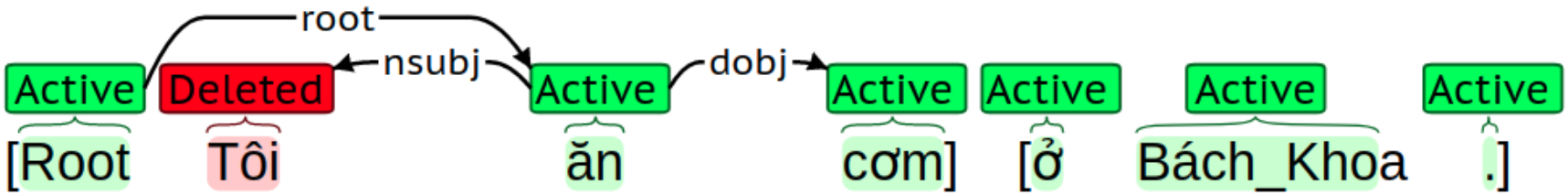
Minh họa Nivre.



RIGHT$_{root}$: Thêm 'ăn' từ bufer vào stack, thêm (Root, root, ăn) vào tập A

A= {(ăn, nsubj, Tôi), (Root, root, ăn)}

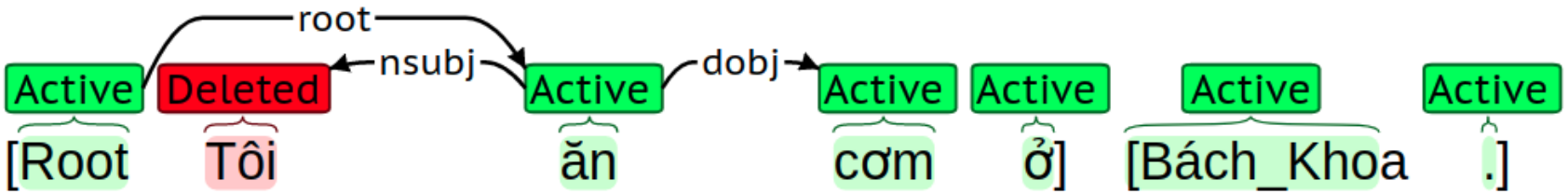
Minh họa Nivre.



RIGHT$_{dobj}$: Thêm 'cơm' từ buffer vào stack, thêm (ăn, dobj, cơm) vào tập A

A= {(ăn, nsubj, Tôi), (Root, root, ăn), (ăn, dobj, cơm) }

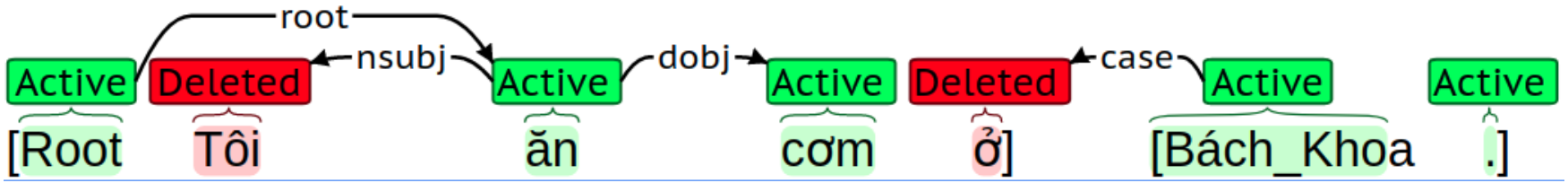
Minh họa Nivre.



SHIFT: chuyển 'ở' từ buffer sang stack

A= {(ăn, nsubj, Tôi), (Root, root, ăn), (ăn, dobj, cơm) }

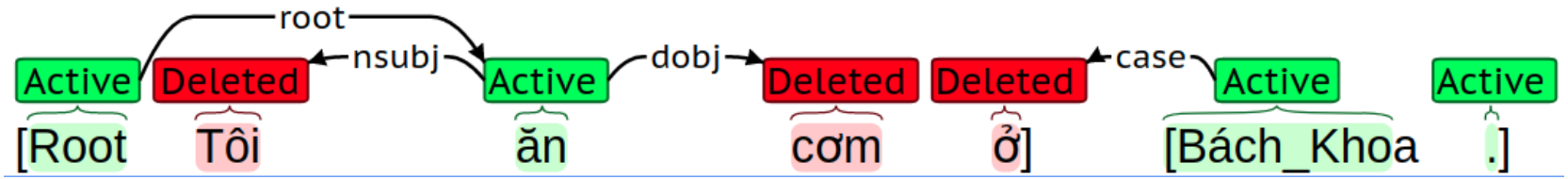
Minh họa Nivre.



$LEFT_{case}$: Xóa 'ở' khỏi Stack, thêm (Bách_Khoa, case, ở) vào tập A

A= {(ăn, nsubj, Tôi), (Root, root, ăn), (ăn, dobj, cơm), (Bách_Khoa, case, ở) }

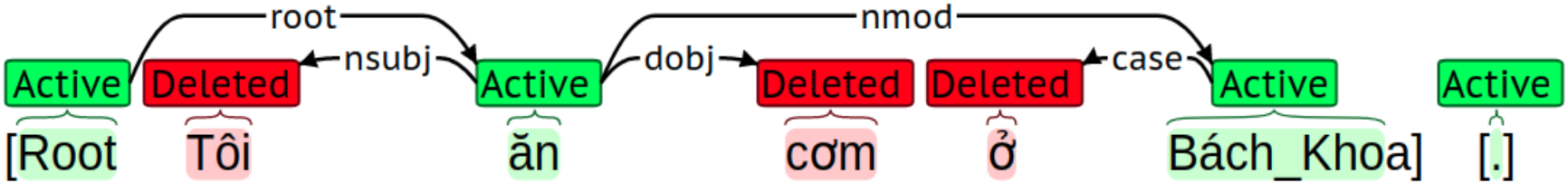
Minh họa Nivre.



REDUCE: Xóa 'cơm' khỏi Stack

A= {(ăn, nsubj, Tôi), (Root, root, ăn), (ăn, dobj, cơm), (Bách_Khoa, case, ở) }

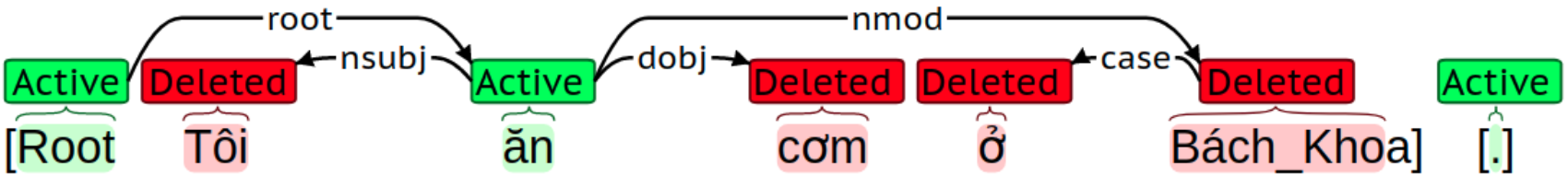
Minh họa Nivre.



RIGHT$_{nmod}$: Thêm 'Bách_Khoa' từ buffer vào stack, thêm (ăn, nmod, Bách_Khoa) vào tập A

A= {(ăn, nsubj, Tôi), (Root, root, ăn), (ăn, dobj, cơm), (Bách_Khoa, case, ở), (ăn, nmod, Bách_Khoa) }

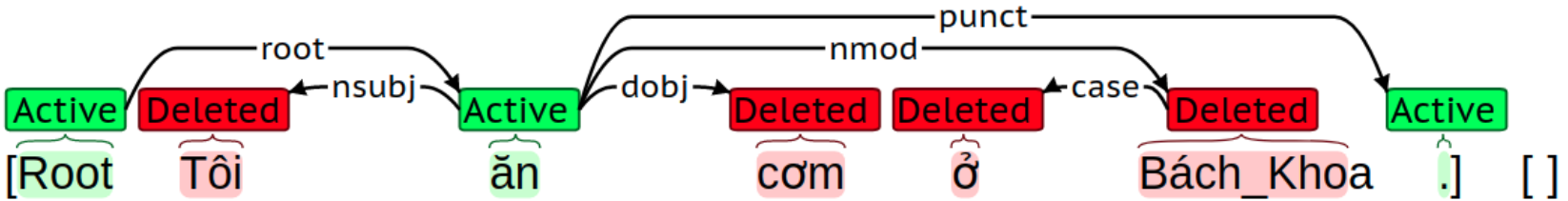
Minh họa Nivre.



REDUCE: Xóa 'Bách_Khoa' khỏi Stack

A= {(ăn, nsubj, Tôi), (Root, root, ăn), (ăn, dobj, cơm), (Bách_Khoa, case, ở), (ăn, nmod, Bách_Khoa) }

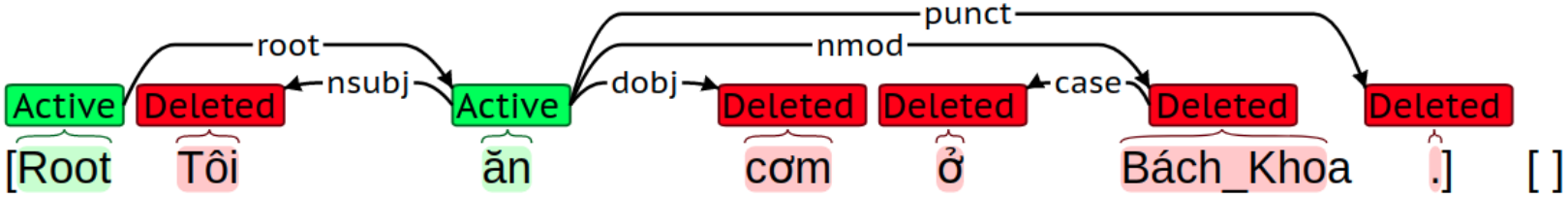
Minh họa Nivre.



RIGHT$_{punct}$: Thêm '.' từ buffer vào stack, thêm (ăn, punct, .) vào tập A

A= {(ăn, nsubj, Tôi), (Root, root, ăn), (ăn, dobj, cơm), (Bách_Khoa, case, ở), (ăn, nmod, Bách_Khoa), (ăn, punct, .) }

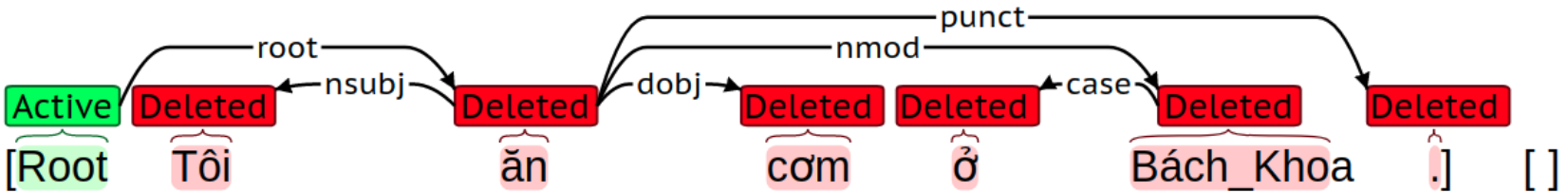
Minh họa Nivre.



REDUCE: Xóa '.' khỏi Stack

A= {(ăn, nsubj, Tôi), (Root, root, ăn), (ăn, dobj, cơm), (Bách_Khoa, case, ở), (ăn, nmod, Bách_Khoa), (ăn, punct, .) }

Minh họa Nivre.

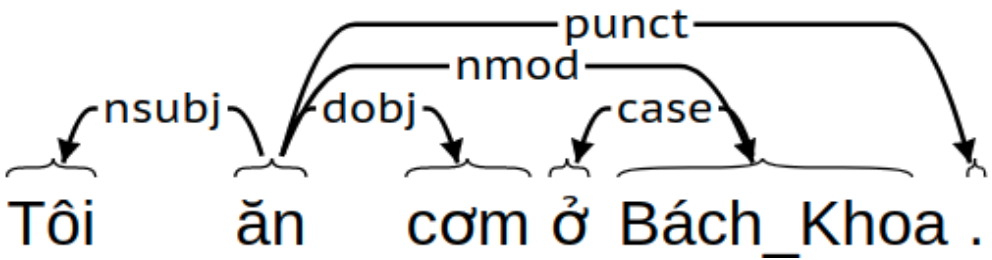


REDUCE: Xóa 'ăn' khỏi Stack

Hiện tại đã đến cấu hình cuối cùng, Stack chỉ có Root, Buffer rỗng. Hệ thống trả lại tập quan hệ phụ thuộc A.

Minh họa Nivre.

Kết quả cuối cùng



Các phương pháp giải quyết.

- Transition-based
  - Thuật toán Nivre
- **Graph-based**
- Các cách tiếp cận hiện nay
  - End to end learning
  - Joint learning

Graph-based

- Ý tưởng là dựa trên bài toán tìm cây khung lớn nhất trong đồ thị có hướng.
- Từ câu đầu vào, xây dựng 1 đồ thị, mà ở giữa 2 từ bất kỳ, luôn có các cạnh nối với nhau theo tất cả các nhãn phụ thuộc.
- Trọng số các cạnh này sẽ do 1 mô hình học máy quyết định.
- Tìm cây khung lớn nhất của đồ thị này.
- Một số ký hiệu:
  - L: tập nhãn phụ thuộc
  - n: độ dài câu, tính theo số lượng từ

Graph-based

Thuật toán phân tích:

- Chu-Liu-Edmond:
  - Cây tìm được có thể không thỏa mãn tính chất projective
  - Thuật toán là một kiểu tìm cây khung lớn nhất trên đồ thị đầy đủ có hướng
  - Độ phức tạp: $|L| * n^2$
- Eisner:
  - Cây tìm được luôn thỏa mãn tính chất projective
  - Thuật toán là 1 kiểu giải thuật quy hoạch động, theo hướng bottom up. Xây dựng cây nhỏ trước, rồi xây cây lớn hơn từ các cây nhỏ ở bước trước, cho đến khi thành cây hoàn chỉnh.
  - Độ phức tạp: n^3

Thuật toán phân tích phổ biến nhất hiện nay cho hướng Graph-based là Eisner

Graph-based

- Để tính trọng số cho các cạnh, ta cần học ra 1 mô hình tính điểm cho cạnh này
- Phương pháp học: SVM, Neural Network, …

Các phương pháp giải quyết.

- Transition-based
  - Thuật toán Nivre
- Graph-based
- **Các cách tiếp cận hiện nay**
  - End to end learning
  - Joint learning

End-to-end Learning

- Dữ liệu học: CoNLL Format.
- Các thông tin được cho:
  - id
  - từ
  - nhãn từ loại
  - id của head
  - nhãn quan hệ phụ thuộc

```
1   Nhưng       _   CC  CC  _   8   cc      _   _
2   có_vẻ       _   RB  RB  _   8   advmod  _   _
3   như _       IN  IN  _   8   mark        _   _
4   rất _       RB  RB  _   5   advmod      _   _
5   nhiều       _   JJ  JJ  _   6   amod      _   _
6   người       _   NN  NN  _   8   nsubj     _   _
7   chưa        _   RB  RB  _   8   neg       _   _
8   biết        _   VB  VB  _   0   ROOT      _   _
9   về  _       IN  IN  _   10  case        _   _
10  nấm _       NN  NN  _   8   nmod        _   _
11  Agaricus        _   NNP NNP _   10  nmod      _   _
12  cùng        _   IN  IN  _   13  case      _   _
13  công_dụng       _   NN  NN  _   10  nmod      _   _
14  vượt_trội       _   JJ  JJ  _   13  amod      _   _
15  từ  _       IN  IN  _   16  case        _   _
16  nó  _       PRP PRP _   13  nmod        _   _
17  .   _       PUNCT   PUNCT   _   8   punct     _   _

1   Nhằm        _   TO  TO  _   2   mark      _   _
2   hưởng_ứng       _   VB  VB  _   0   ROOT      _   _
3   chương_trình        _   NN  NN  _   2   dobj      _   _
4   “       PUNCT   PUNCT   _   5   punct     _   _
5   Hành_trình      _   NN  NN  _   3   nmod      _   _
6   đỏ  _       JJ  JJ  _   5   amod        _   _
7   ”   _       PUNCT   PUNCT   _   5   punct     _   _
```

End-to-end Learning

Lựa chọn đặc trưng thủ công:

- Cần chuyên gia trong lĩnh vực này
- Số lượng giá trị feature template lớn do sự tổ hợp của các feature

=> Có thể xem là khâu tốn nguồn lực nhất khi giải quyết bài toán này

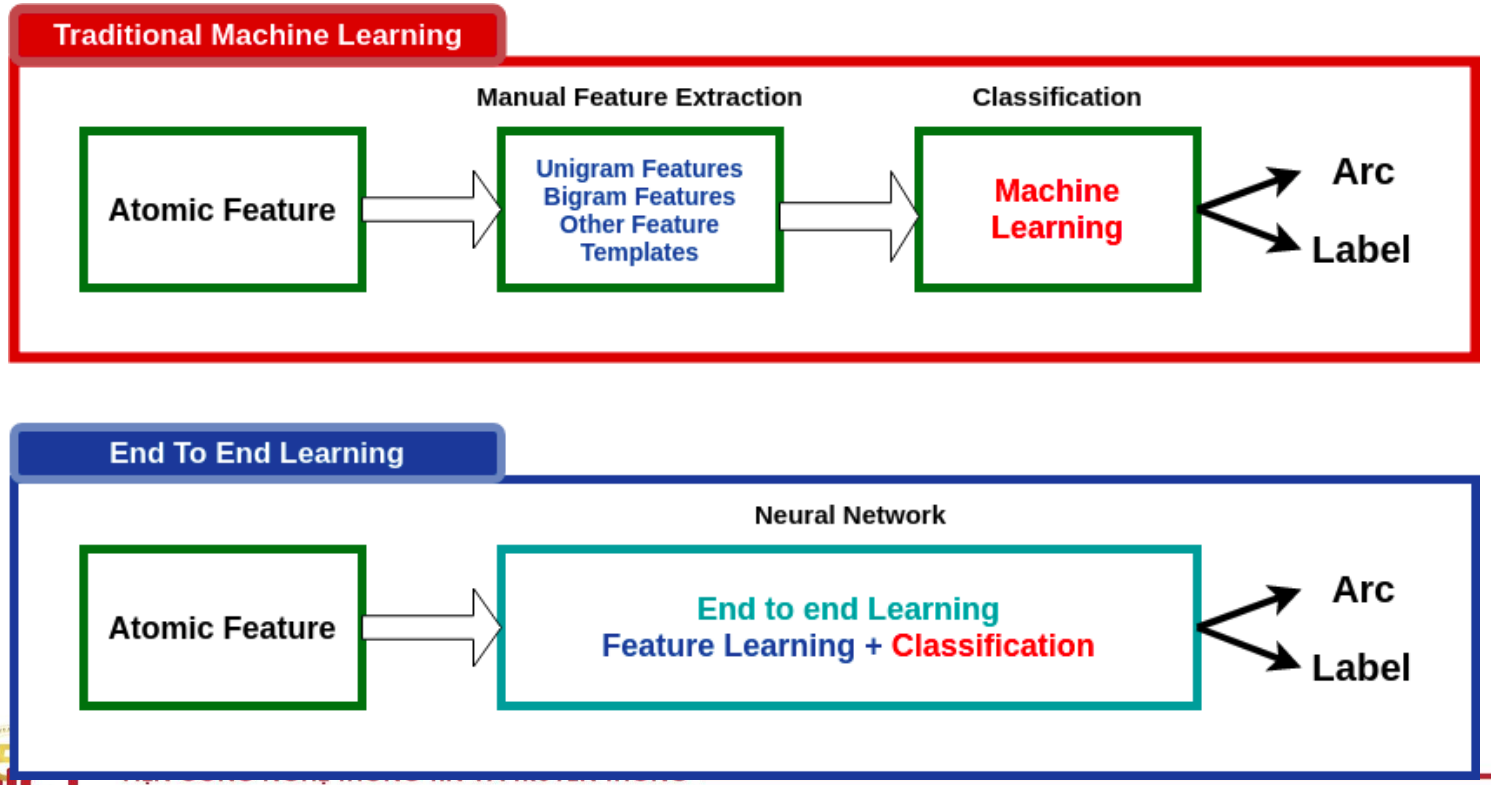
Basic Big-ram Features
p-word, p-pos, c-word, c-pos
p-pos, c-word, c-pos
p-word, c-word, c-pos
p-word, p-pos, c-pos
p-word, p-pos, c-word
p-word, c-word
p-pos, c-pos

feature template: 1 tổ hợp sinh ra bằng cách kết hợp các đặc trưng nguyên tố (từ, nhãn từ loại, ...)

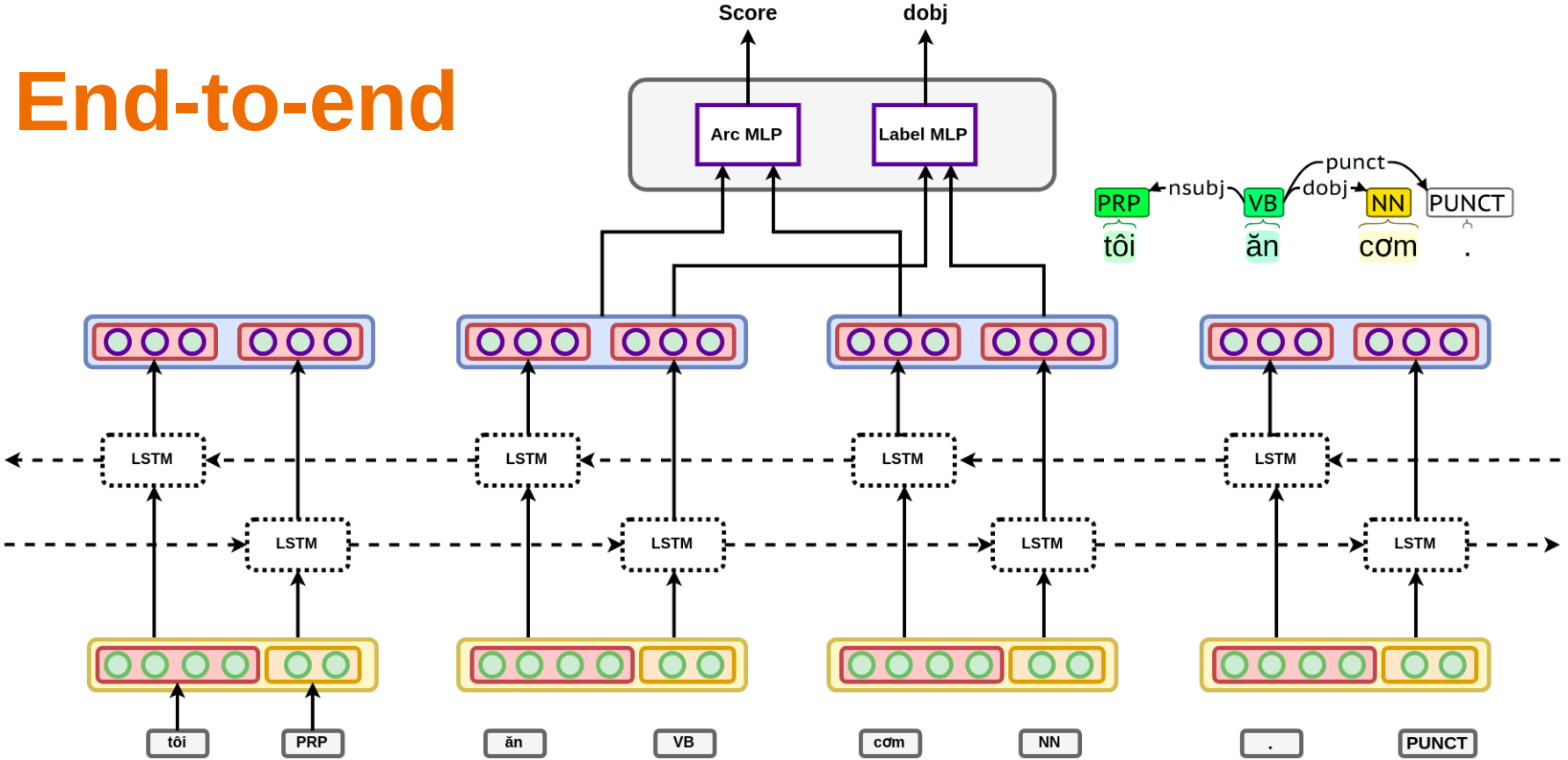
End-to-end Learning

- End to end learning giúp giải quyết vấn đề này:
- Tư tưởng của nó là huấn luyện đồng thời 2 khối là bộ trích chọn đặc trưng (feature extractor) và bộ phân loại (classifier)
- Như vậy, không cần quan tâm đến việc phải lựa chọn các feature template như thế nào nữa, chỉ cần đưa các đặc trưng nguyên tố (từ, nhãn từ loại), bộ trích chọn đặc trưng sẽ tự học ra cách biểu diễn tốt nhất từ các đặc trưng này.

End-to-end Learning



End-to-end

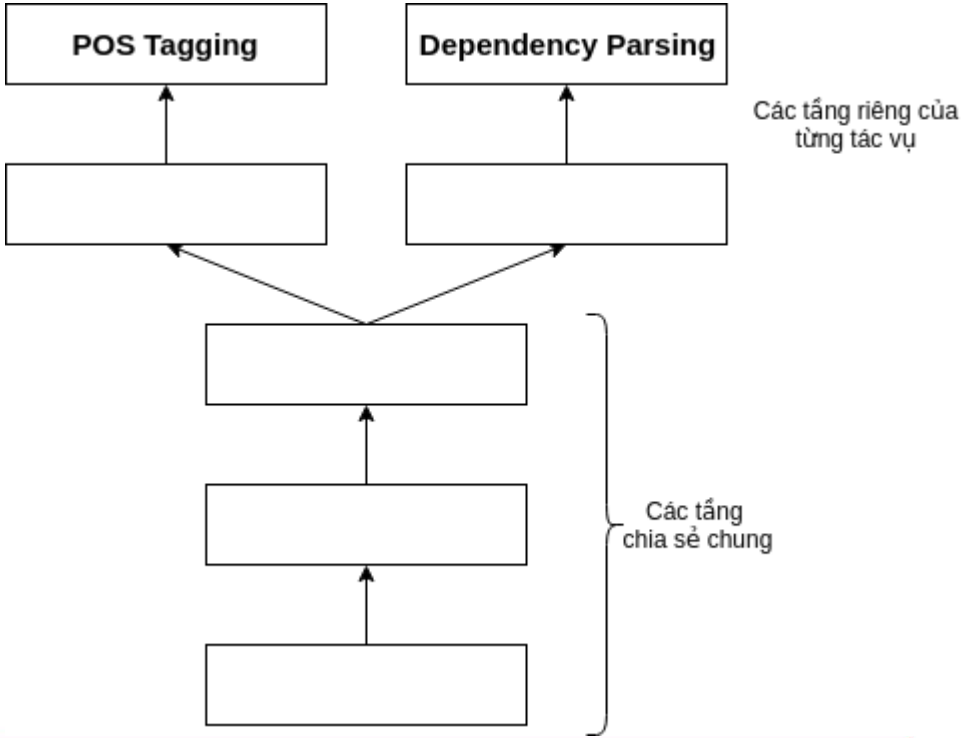

Score
dobj
Arc MLP
Label MLP
PRP
nsubj
VB
punct
dobj
NN
PUNCT
tôi
ăn
cơm
.
LSTM
tôi
PRP
ăn
VB
cơm
NN
.
PUNCT

Joint Learning

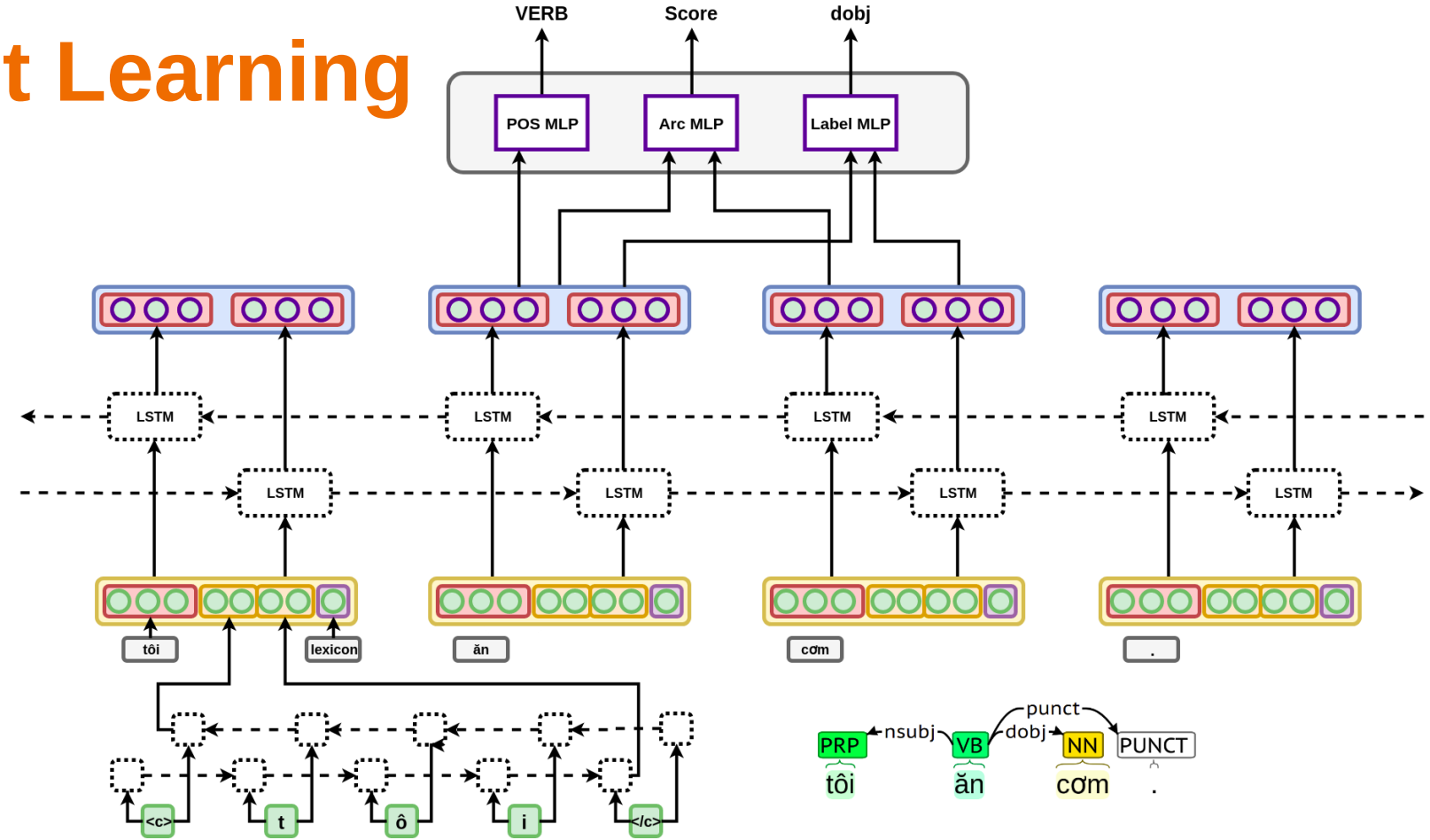
- Học đồng thời nhiều tác vụ với nhau:
  - Các tác vụ học chung phải liên hệ mật thiết với nhau.
  - Học chung mang lại nhiều lợi ích: phần chia sẻ chung chứa thông tin của nhiều tác vụ, giảm overfit mô hình.
- Trong bài toán Dependency Parsing, học chung 2 tác vụ là POS Tagging + Dependency Parsing.

Joint Learning

- Xem hình bên:
  - 2 tác vụ gán nhãn từ loại và phân tích cú pháp chia sẻ chung các tầng neural đầu vào
  - Sau đấy, đầu ra của các tầng chia sẻ chung này được đưa vào các tầng neural riêng của mỗi tác vụ.
- Với các nghiên cứu gần đây, tầng neural chia sẻ chung thường là các mạng BiLSTM biểu diễn đầu vào



Joint Learning



Joint Learning

Ở đây:

- Một mạng RNN đóng vai trò embedding ký tự
- Một mạng BiLSTM đóng vai trò tạo ra biểu diễn đầu vào cho các mạng MLP đầu ra của các tác vụ POS Tagging và Dependency Parsing (từ vector chứa thông tin các dữ liệu từ, ký tự, nhãn từ loại).

Joint Learning

Có 3 tác vụ được học chung ở đây:

- Gán nhãn từ loại.
- Tính trọng số của cạnh (quan hệ phụ thuộc) nối 2 từ.
- Xác định nhãn phụ thuộc của 2 từ.

2 tác vụ sau là cho bài toán phân tích cú pháp phụ thuộc.

Các nội dung chính

1. Tổng quan về Dependency Parsing.
   a. Định nghĩa
   b. Ứng dụng
   c. Tính chất
2. Các phương pháp giải quyết bài toán.
   a. Transition-based
   b. Graph-based
   c. Các cách tiếp cận hiện nay
3. **Các kết quả cài đặt.**

Các kết quả cài đặt.

- Phần POS Tagging
- Phần Dependency Parsing
- Dữ liệu và chỉ số đánh giá
- Kết quả đánh giá

Phần POS Tagging.

- CRFSuite: Công cụ học máy CRF, ở đây dùng trong bài toán gán nhãn từ loại.
- jPTDP: công cụ học joint learning, phương pháp Neural Network, học cả 2 phần POS Tagging và Dependency Parsing.

Phần Dependency Parsing.

- Malt Parser (Transition based):
  - Thuật toán phân tích: **Nivre**
  - Phương pháp học: **SVM**
- Yara Parser (Transition based):
  - Thuật toán phân tích: **Nivre**
  - Phương pháp học: **Neural Network**
  - Các cải tiến: **Error Exploration, Beam Search**
- BiLSTM Transition-based:
  - Thuật toán phân tích: **Nivre**
  - Phương pháp học: **Neural Network**
  - End to end learning

Phần Dependency Parsing.

- BiLSTM Graph-based:
  - Thuật toán phân tích: **Eisner**
  - Phương pháp học: **Neural Network**
  - End to end learning
- jPTDP (Graph-based):
  - Thuật toán phân tích: **Eisner**
  - Phương pháp học: **Neural Network**
  - End to end learning
  - Joint Learning POS Tagging + Dependency Parsing

Dữ liệu và chỉ số đánh giá

- Dữ liệu: BK Treebank.
  - 6908 câu được định dạng theo CoNLL-U Format
  - Chia làm 4505 câu cho tập train, 1134 câu cho tập development, 1269 câu cho tập test
- Các chỉ số đánh giá:
  - POS Tagging: Accuracy.
  - Dependency Parsing: UAS và LAS
    - UAS: chỉ cần đúng quan hệ mà không cần đúng nhãn
    - LAS: phải đúng cả quan hệ và cả nhãn của quan hệ đó

Kết quả đánh giá

Phương pháp	UAS	LAS
Malt Parser	84.4 %	81.4 %
Yara Parser	86.3 %	83.4 %
BiLSTM Transition	86.4 %	82.9 %
BiLSTM Graph	**87 %**	**84.2%**

Kết quả với bộ dữ liệu đã được gán nhãn từ loại chính xác

Kết quả đánh giá

Phương pháp	POS Accuracy	UAS	LAS
CRF + Malt Parser	90.66 %	76.7 %	70.2 %
CRF + Yara Parser	90.66 %	79.1 %	72.6 %
CRF + BiLSTM Transition	90.66 %	78.9 %	72.2 %
CRF + BiLSTM Graph	90.66 %	79.7 %	**73 %**
jPTDP	89.16 %	**80.4 %**	**73 %**

Kết quả với bộ dữ liệu chưa được gán nhãn từ loại.

Kết quả đánh giá

Phương pháp	POS Accuracy	UAS	LAS
jPTDP	89.16 %	80.4 %	73 %
jPTDP + Lexicon	**91.50 %**	**82.13 %**	**75.67 %**
jPTDP + Lexicon (Not Character Embed)	91.05%	81.46 %	75.23 %

Kết quả với bộ dữ liệu chưa được gán nhãn từ loại.

Tài liệu tham khảo.

[1] Kiem-Hieu Nguyen. BKTreebank: Building a Vietnamese Dependency Treebank
[2] Yue Zhang and Joakim Nivre. Training Deterministic Parsers with Non-Deterministic Oracles
[3] Eliyahu Kiperwasser and Yoav Goldberg. Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations
[4] Dat Quoc Nguyen, Mark Dras and Mark Johnson. A Novel Neural Network Model for Joint POS Tagging and Graph-based Dependency Parsing

Tài liệu tham khảo.

[5] Yuan Zhang and David Weiss. Stack-propagation: Improved Representation Learning for Syntax
[6] Barbara Plank, Anders Søgaard, Yoav Goldberg. Multilingual Part-of-Speech Tagging with Bidirectional Long Short-Term Memory Models and Auxiliary Loss
[7] Ryan McDonald et al. Online Large-Margin Training of Dependency Parsers
[8] Yoav Goldberg and Joakim Nivre. Training Deterministic Parsers with Non-Deterministic Oracles