Mục Lục

[I. Đặt vấn đề: 1](#_Toc461142522)

[1. Thực tế: 1](#_Toc461142523)

[2. Tạo ra mạng neural học từ các vector thể hiện của từ, part-of-speech tag (POS) và các nhãn phụ thuộc. 1](#_Toc461142524)

[II. Transition-based Dependency Parsing 1](#_Toc461142525)

[III. Neural Network Based Parser 3](#_Toc461142526)

[1. Model 3](#_Toc461142527)

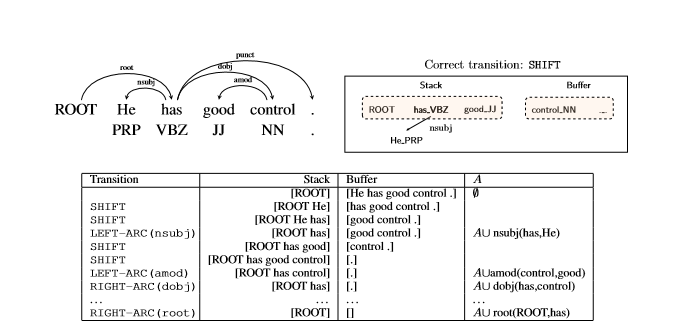
[2. POS and label embeddings 5](#_Toc461142528)

[3. Cube activation function 5](#_Toc461142529)

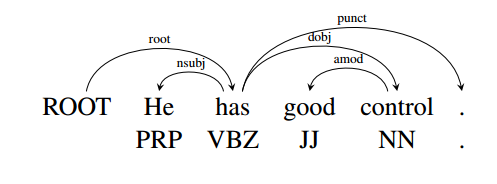
**A Fast and Accurate Dependency Parserusing Neural Networks**

1. Đặt vấn đề:
   1. Thực tế:
      * Hầu hết các lớp phân tích cú pháp phụ thuộc hiện nay dựa trên rất nhiều tính năng thưa thớt.
      * Dẫn tới các kết quả không tốt, chi phí cho việc thực hiện lại rất lớn, nghiêm trọng.
      * Những năm gần đây, các mô hình phân tích đạt được những thành tựu dựa trên việc sử dụng các phân tích cú pháp phụ thuộc dựa trên sự phân biệt các tính năng. Tuy nhiên những mô hình này vẫn chưa thực sự hoàn hảo do:
        + Chịu ảnh hưởng bởi việc sử dụng hàng triệu các các tính năng ước lượng trọng số không tốt trong khi việc tổng hợp cả tính năng linh hoạt và tính năng sắp thứ tự tương tác ở mức cao là rất quan trọng.
        + Hầu hết các mô hình phân tích đều dựa trên các tính năng mẫu được thiết kế một cách thủ công.
        + Việc sử nhiều các tính năng mẫu gây ra vấn đề cho việc học.
   2. Tạo ra mạng neural học từ các vector thể hiện của từ, part-of-speech tag (POS) và các nhãn phụ thuộc.
2. Transition-based Dependency Parsing

* Mục đích của mô hình này là có thể đưa ra được dự đoán một chuỗi các chuyển tiếp từ cài đặt ban đầu tới một số các cài đặt cuối dựa trên cây phân tích phụ thuộc mong muốn.
* Các thành phần trong mô hình:
  + A Configuration **c = (s,b,A)** bao gồm:
    - Stack s
    - Buffer b
    - Tập các cung phụ thuộc A
  + Configuration khởi tạo cho 1 câu w1,…,wn là **s = [ROOT]**,   
    **b = [w1, …, wn]**, **A = rỗng**.
  + Configuration c là 1 terminal configuration nếu buffer b trở thành rỗng , stack chứa 1 nốt ROOT và cây phụ thuộc được đưa ra bởi Ac.
  + Định nghĩa:
    - **Si (i = 1, 2, …)** như là i phần tử trên đầu của stack.
    - **Bi (i = 1,2, …)** như là i phần tử trong buffer.
* Mô hình định nghĩa 3 kiểu chuyển tiếp:
  + **LEFT-ARC(l)** : thêm một cung s1 -> s2 với nhãn l và xóa s2 ra khỏi stack. Điều kiện |s| >= 2.
  + **RIGHT-ARC(l)** : thêm một cung s2 -> s1 với nhãn l và xóa s1 ra khỏi stack. Điều kiến |s| >= 2.
  + **SHIFT** : di chuyển b1 từ buffer sang stack, điều kiện |b| >= 1.
* Tổng cộng sẽ có **|T| = 2Nl + 1** transitions với Nl là số các nhãn cung phụ thuộc khác nhau.
* Hoạt động:

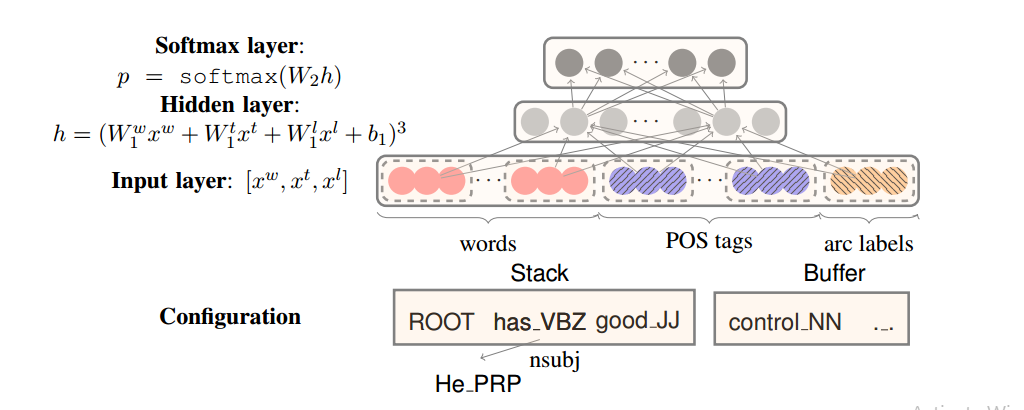


* Cây phụ thuộc phân tích được chứa các từ và Pos tag tương ứng ví dụ như has / VZB, các head of a word và nhãn của nó như nsubj, dobj.
* Việc phân tích hoạt động như sau:
  + Ta khởi tạo một cấu hình c ban đầu với **c = (s,b,A)** với **s = [ROOT], b = [he, has, good, control, . ]** và **A = rỗng.** Ta tiến hành các bước transition để được cây phân tích phụ thuộc mong muốn.
  + Việc tìm các phụ thuộc dựa vào từ khóa là động từ trong câu tiến hành tìm các phụ thuộc của câu.
  + Dùng transition SHIFT để chuyển phần tử từ buffer vào stack, cụ thể :
    - Đầu tiên dùng **SHIFT** đưa một từ đầu tiên trong bộ nhớ đệm vào stack s là “he”.
    - Tiếp tục dung **SHIFT** đưa từ tiếp theo từ b vào s là “has”.
  + Chưa tìm thấy sự phụ thuộc của các phần tử trong stack thì tiếp tục công việc bước 2 ngược lại dùng transition tương ứng tạo các cung phụ thuộc tương ứng trong A:
    - Khi này trong s đã tạo nên một phụ thuộc , ta tiến hành dùng transition **LEFT-ARC** với label đặt là “nsubj” ta đc **LEFT-ARC(nsubj)** tạo ra một arc nsubj(has, He) được đưa và **A** và tiến hành xóa bỏ He trong has - > He (s1 -> s2) khỏi stack s. Tiếp tục quá trình như vậy ta đưa các từ trong b vào s với transition SHIFT để tạo thành các arc đưa vào trong A.
  + Kết quả cuối cùng ta được cây phân tích phụ thuộc với **A= [nsubj(has, He), amod(control, good), dobj(has, control), punct(has, . ), root(Root, has)]** là:



* Hạn chế
  + Phụ thuộc vào các vấn đề:
    - Sparsity
    - Incompleteness
    - Expensive feature computation

1. Neural Network Based Parser
   1. Model
      * Như việc nhúng từ thông thường, mỗi từ sẽ được thể hiện như một véc tơ có d chiều thuộc không gian R^d và một ma trận nhúng đầy đủ là E(w) thuộc không gian R^(d\*Nw), trong đó Nw là kích thước của từ điển. Trong khi đó, chúng ta cũng ánh xạ POS tag và nhãn arc tới một không gian vector d chiều.
      * Việc chọn tập các phần tử dựa trên ví trí trong ngăn xếp và bộ nhớ đệm của mỗi một kiểu thông tin (word, POS hay label) có thể hữu ích cho việc dự đoán.
      * Ví dụ:



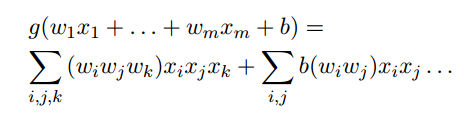
* + - Chúng ta định nghĩa các tập như Sw, St, Sl một cách tương ứng với word , POS , lable.
    - Chúng ta sẽ xây dựng lên mạng neural với một lớp hidden nơi nhúng tương ứng các thành phần được chọn từ S(w), S(t) S(l) sẽ được thêm vào input layer.
    - Định nghĩa nw, nt, nl như là tập số lượng thành phần được chọn của mỗi loại.
    - Chúng ta thêm xw vào input layer , tương tự chúng ta cũng thêm POS tags feature x(t) và arc label feature (xl) vào input layer.
    - Tiếp đến chúng ta ánh xạ từ input layer vào một hidden layer với d(h) nodes thông qua một hàm kích hoạt lập phương (**cube activation function**).



* + - Cuối cùng là một softmax layer sẽ được thêm vào trên cùng của hidden layer cho mô hình hóa xác xuất của nhiều lớp

.

* 1. POS and label embeddings
     + Là tập bao gồm các tương đồng ngữ nghĩa như từ ví dụ như
       - POS tags P = {NN, NNP, NNS, DT, JJ, …} cho tiếng Anh.
       - Arc labels L = {amod, tmod, nsubj, csubj, dobj, …}
     + Trong đó ví dụ NN (singular noun), NNS (plural noun), DT (deteminer), amod(adjective modifier), num(numeric modifier) , nsubj (nonminal subject). Những POS tags và arc labels này có tác dụng rất lớn cho việc thể hiện các phụ thuộc trên cây phân tích phụ thuộc.
  2. Cube activation function
* Tất cả các đơn vị hidden được tính toán bở việc ánh xạ một trọng số tổng của một đơn vị input cộng với sai số. sử dụng g(x) = x^3 có thể mô hình được sản phẩn của x(i), x(j), x(k) cho bất cứ một trong 3 thành phần của input layer:



* + - Trong trường hợp này, xi, xj, xk có thể đến từ một trong 3 chiều khác nhau của không gian nhúng.