Shelly Gamlielly
Yerus Mandfro

NLP EX5

Implementation Explanation :

- Feature Function :
  The features function operates on two nodes. The features could be indicative or contra indicative that there is an arc between those two nodes. The features function also considers the label of the arc. It returns a vector of which features are relevant for the two nodes.
  To save time and space we used indexe to represent a word , and index to represent a tag. Those indexes were used later to update the weights vector, and the score. W - the weight vector has n^2 + t^2 entries, each entry from 0 to n^2 -1 defines an arc between two words , and each entry between n^2 to t^2 -1 defines an arc between two tags. Where n represents the number of different words in the vocabulary , and t represents the number of different tags.

- Calculate Score:
  Instead of using a dot product between a sparse binary features vector and the weight vector, all we need to do is to sum only two entries from the weights vector.
  The two relevant entries are calculated by :
  1. first word index * number of words + second word index
  2. number of words^2 + first word's tag index * number of tags + second word's tag index

  The maximal spanning tree will be the one with the highest sum of scores of it's arcs
  $T' \leftarrow argmax_T \sum_{(v_1,v_2,l)\in T} score_\theta(T)$
  :

- Update the weights vector:
  $\theta^{((r-1)N+i)} \leftarrow \theta^{((r-1)N+i-1)} + \eta \cdot \left(\sum_{(v_1,v_2,l)\in T_i} \Phi(v_1,v_2,l) - \sum_{(v_1,v_2,l)\in T'} \Phi(v_1,v_2,l) \right)$
  In order to update the weights vector we need to subtract the sums of the feature function of an arc from the gold standard tree, from the sums of the feature function of an arc from the maximal spanning tree. To avoid building very large binary vectors we check if the mapped arcs are identical, if not we add 1 or -1 to the relevant entries, where finding the entries is done as explained above.

Results

Test Set Evaluation

Test Set Average Accuracy: 0.301

Where the accuracy of each sentence is calculated by :

$$ACC = \frac{\#CORRECT\ EDGES}{\#NUMBER\ OF\ WORDS}$$

and the average accuracy is the sum of sentences accuracy divided by the number of sentences.

We tried different ways to deal with the root node from the gold standard tree.
We chose the most reasonable way that also had the highest accuracy of 0.301.

We chose to map root nodes of arcs to an index by checking if the head node is None or 0.
According to the data set:

- If the head node is None than this arc is the one that represents the root. i.e (tree['head'] = None,tree['address']= 0), where tail = tree['deps']['ROOT'] is the index of the tail word in the sentence.
  We map the head to index 0, and the tail index on the sentence is found by the 'deps' value of the tree's root. Then, we map it to a word, and then to the index according to the location on the words set, i.e (head = 0, tail = index on words set). In this case the tags are (head_tag = 0,tail_tag = index on tags set).

- If the head node is 0 then this arc is from the root to the first word, i.e (tree['head'] = 0,tree['address']= tail)
  We find the tail index with dictionaries that maps the index of a word from a sentence to a word, and a word from the words set of training set to an index that will be updated in W , i.e (head = 0, tail = index on words set). In this case the tags are (head_tag = 0,tail_tag = index on tags set).

- Else, we find the head index and the tag index with dictionaries that maps the index of a word from a sentence to a word, and a word from the words set of training set to an index that will be updated in W, i.e (head = index on words set, tail = index on words set). . In this case the tags are (head_tag = index on words set ,tail_tag = index on tags set).

Accuracies :

1. When root is defined by index 0 , and the word it's connected to get's the relevant index, the accuracy is **0.301.**
   When adding an arc of the root (head= None, tail = 0) that is mapped to (0, first word index) to the gold standard arcs list the accuracy is 0.284

2. When root's head and tail are defined by indexes 0 (ignoring the word that the root is connected to) the accuracy is 0.267
   Check root : 0.251

3. When only root's head is defined by indexes 0, without checking if head is None the accuracy is 0.2799
   Check root : 0.263

Evaluate 100 sentences from training set results :

Training Set Average Accuracy: 0.519

Example of a sentence from training set:

Predicted arcs of the maximal spanning tree :

ROOT -> underscore
Plans -> that
that -> give
that -> have
give -> maintaining
give -> .
advertisers -> ad
advertisers -> competition
discounts -> for
maintaining -> or
maintaining -> increasing
maintaining -> spending
or -> discounts
increasing -> at
ad -> Plans
ad -> the
ad -> B.
ad -> Zuckerman
ad -> World
have -> News
become -> fixtures
become -> between
become -> advertisers
permanent -> become
at -> weeklies
at -> Inc.
weeklies -> news
underscore -> permanent
competition -> fierce
between -> Report
Inc. -> ,
Inc. -> &
's -> Newsweek
's -> Warner
's -> magazine
's -> Mortimer
's -> U.S.
magazine -> Time

Report -> and
Report -> 's

Gold standard tree arcs:

ROOT -> have
have -> Plans
Plans -> that
that -> give
give -> advertisers
give -> discounts
discounts -> for
for -> maintaining
maintaining -> or
maintaining -> increasing
spending -> ad
maintaining -> spending
have -> become
fixtures -> permanent
become -> fixtures
fixtures -> at
weeklies -> the
at -> weeklies
weeklies -> news
have -> and
have -> underscore
competition -> the
underscore -> competition
competition -> fierce
competition -> between
Report -> Newsweek
between -> Report
Report -> ,
's -> Time
magazine -> 's
's -> Warner
's -> Inc.
Report -> magazine
magazine -> Time
Report -> ,
Report -> and
's -> Mortimer
Report -> 's
's -> B.
's -> Zuckerman
Report -> U.S.
Report -> News
Report -> &
Report -> World
have -> .

Shared arcs of gold standard tree and predicted tree:

Plans -> that
that -> give
discounts -> for
maintaining -> or
maintaining -> increasing
maintaining -> spending
become -> fixtures
at -> weeklies
weeklies -> news
competition -> fierce
between -> Report
's -> Warner
magazine -> Time
Report -> and
's -> Mortimer
Report -> 's

Example Accuracy: 0.3478

Σ : stack : root
B buffer : John likes balloons that pop and eggs
A : John likes balloons that pop and eggs

Shift

Σ : root John likes
B : likes balloons that pop and eggs
A : John likes balloons that pop and eggs

Shift

Σ : root John likes
B : balloons that pop and eggs
A : John likes balloons that pop and eggs

left arc - nsubj

Σ : root likes
B : balloons that pop and eggs
A :

Shift

nsubj
John likes balloons that pop and eggs

Σ : root likes balloons
B : that pop and eggs

Shift Σ : root likes balloons that
B : pop and eggs

Shift

Σ : root likes balloons that pop
B : and eggs

left arc - mark

Σ : root likes balloons pop
B : and eggs

A

nsubj mark
John likes balloons that pop and eggs

right arc - acl

Σ : root likes balloons
B : and eggs

A

acl
nsubj mark
John likes balloon that pop and eggs

Shift

Σ : root likes balloons and
B : eggs

Shift Σ : root likes balloon and eggs
B :

left arc - cc

Σ : root likes balloon eggs
B :

A

acl
nsubj mark cc
John likes balloons that pop and eggs

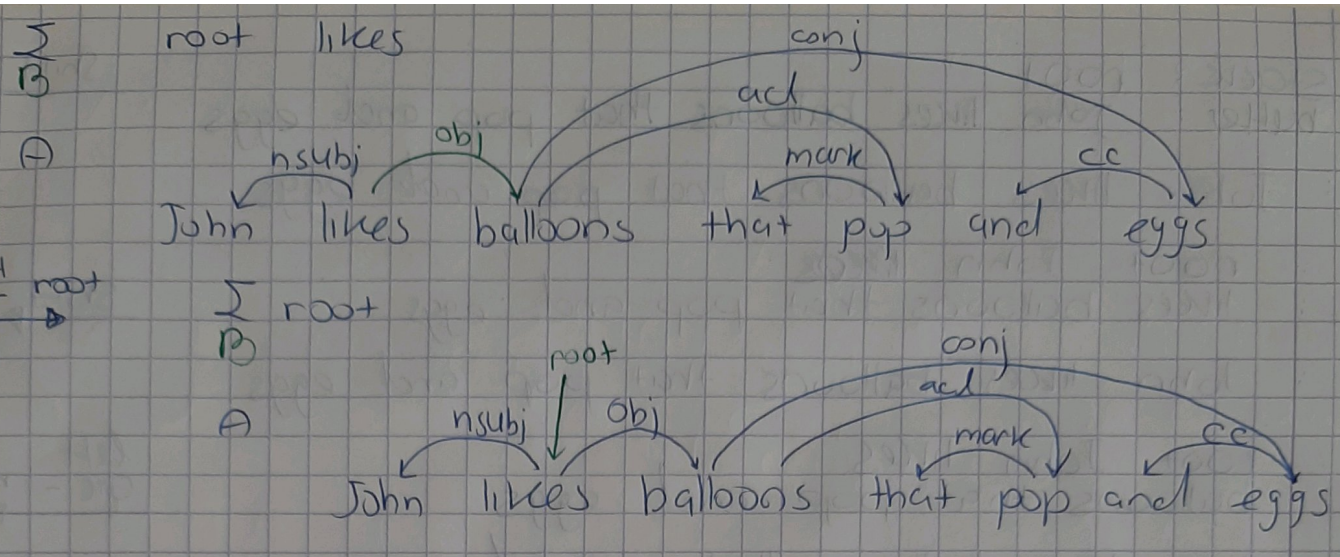
right arc - conj

Σ root likes balloons
B

A

conj
acl cc
nsubj mark
John likes balloons that pop and eggs

right arc - obj



shift, shift, left arc, shift, shift, shift, left arc, right arc, shift, shift, left arc, right arc, right arc, right arc

arc = eager

1) S root
B John likes balloons that pop and eggs — shift →
A John likes balloons that pop and eggs

2) S root John
B ~~John~~ likes balloons that pop and eggs — left arc - nsubj → head pc John - r
A John likes balloons that pop and eggs

3) S root — right arc - root →
B likes balloons that pop and eggs
A John (n-subj) likes balloons that pop and eggs.

4) S root likes — right arc - obj →
B balloons that pop and eggs
A John (nsubj) likes (root)

5) S root likes balloons ~~that pop and eggs~~ — shift →
B that pop and eggs
A John (nsubj) likes (root) (obj) balloons that pop and eggs

6) S root likes balloons that — left arc - mark → head pc that - r
B pop and eggs
A John (nsubj) likes (root) (obj) balloons that pop and eggs

7) S root likes balloons — right arc - acl →
B pop and eggs
A John (nsubj) likes (root) (obj) balloons (mark) that pop and eggs

8) S root likes balloons pop — reduce → head pc pop - r
B and eggs
A John (nsubj) likes (root) (obj) balloons (acl) that (mark) pop and eggs

9) S root likes balloons — shift →
B and eggs
A John (nsubj) likes (root) (obj) balloons (acl) that (mark) pop and eggs — left arc - cc → head pc and - r

10) S root likes balloons and
B eggs
shift - w nothing left A

11) Σ root likes balloons

B eggs

A John likes balloons that pop and eggs

(arcs: n-subj, root, obj, acl, mark, cc; right arc-conj)

12 Σ root likes balloons eggs

B

A John likes balloons that pop and eggs

(arcs: n-subj, root, obj, acl, mark, cc, conj)

ה- buffer ריק ולכן סיימנו

נשים לב שברגע שמתחת לכל מילית ההפרדה הכי שהאיבר שמעליו במחסנית
אינו שורש וקיבלנו לו head
head
כמו כן, נעשה reduce רק, שהאיבר שמעליו משמש כ head פ.ש.

בסה״כ 11 מעברים :

Shift, left arc-n subj, right arc-root, right arc-obj,
shift, left arc-mark, right arc-acl, reduce, shift,
left arc-cc, right arc-conj