CSC485/2501 Homework Assignment 1: Write-up

Edwin Chacko

Student Number: `1009149716` *UTORid:* `chackoed`
`edwin.chacko@mail.utoronto.ca`

1 Transition-based dependency parsing

(a) Complete the sequence of transitions. Please complete Table 1.

Step	Stack	Buffer	New dep	Transition
0	[ROOT]	[To, raise, those, doubts, is, to, resolve, them]		
1	[ROOT, To]	[raise, those, doubts, is, to, resolve, them]		SHIFT
2	[ROOT, To, raise]	[those, doubts, is, to, resolve, them]		SHIFT
3	[ROOT, raise]	[those, doubts, is, to, resolve, them]	To $\overset{mark}{\leftarrow}$ raise	LEFT-ARC
4	[ROOT, raise, those]	[doubts, is, to, resolve, them]		SHIFT
5	[ROOT, raise, those, doubts]	[is, to, resolve, them]		SHIFT
6	[ROOT, raise, doubts]	[is, to, resolve, them]	those $\overset{det}{\leftarrow}$ doubts	LEFT-ARC
7	[ROOT, raise]	[is, to, resolve, them]	raise $\overset{dobj}{\rightarrow}$ doubts	RIGHT-ARC
8	[ROOT, raise, is]	[is, to, resolve, them]		SHIFT
8	[ROOT, is]	[to, resolve, them]	raise $\overset{csub}{\leftarrow}$ is	LEFT-ARC
10	[ROOT, is, to]	[resolve, them]		SHIFT
11	[ROOT, is, to, resolve]	[them]		SHIFT
12	[ROOT, is, resolve]	[them]	to $\overset{mark}{\leftarrow}$ resolve	LEFT-ARC
13	[ROOT, is, resolve, them]	[]		SHIFT
14	[ROOT, is, resolve]	[]	resolve $\overset{dobj}{\rightarrow}$ them	RIGHT-ARC
15	[ROOT, is]	[]	is $\overset{xcomp}{\rightarrow}$ resolve	RIGHT-ARC
16	[ROOT]	[]	ROOT $\overset{root}{\rightarrow}$ is	RIGHT-ARC
17	[ROOT]	[]		DONE

Table 1: Sequence of transitions.

(b) Number of Steps.

The buffer has n words which requires n shift operations to transfer them into the stack – one operation per word. The stack must only contain the ROOT at the end which means all n words entering the stack must exit – we require each word either give or receive a dependency. This means $n + n$ operations $= 2n$ steps.

Therefore a sentence conting n words requires $2n$ steps to parse.

(g) Comparison with ChatGPT.

I used the below prompt:

```
You are tasked with dependency parsing. You are to parse the word dependencies of
 the below statment includeing the dependent, the head, and the relationship.

  Sentence to parse:
  “To ask those questions is to answer them.”
```

And got the below ouput:

```
Here’s the dependency parsing of the sentence "To ask those questions is to answer
  them." with the dependent, head, and relationship:

  1. To → (Head: ask, Relationship: mark)
  2. ask → (Head: is, Relationship: csubj)
  3. those → (Head: questions, Relationship: det)
  4. questions → (Head: ask, Relationship: obj)
  5. is → (Head: ROOT, Relationship: ROOT)
  6. to → (Head: answer, Relationship: mark)
  7. answer → (Head: is, Relationship: xcomp)
  8. them → (Head: answer, Relationship: obj)

This structure represents the dependencies between the words, identifying how each
word (dependent) connects to a head word with a specific syntactic relationship.
```

ChatGPT’s parsing is not entirely accurate given its zero-shot attempt however the extent to which it has been trained on parsing and what impact this has is remains largely unknown. It is highly probable that the training data includes textbooks, examples, or other excerpts about dependency parsing which provide it some insight towards the task. In addition, as a language model, it is likely that there is underlying dependency parsing occurring within the model itself. The relationship labels are off for many cases, however the head-dependent relationships are all correct. This can be attributed to both err and potentially even ChatGPT parsing the text differently than humans.

Test LAS: 81.2% UAS: 83.6%. This model perfroms better overall. The model perfectly parsed the given sentence in this question. It was much quicker than ChatGPT but it was just an answer with no justification. ChatGPT was able to provide some justification and is open to additional prompts regarding why it made its choice.

ChatGPT 4o likely has more parameters than GPT-3 which has 175 billion so immediately there is this disparity in model size raises differences in appropriate use. GPT-4o is far more computationally expensive and is not trained to parse dependencies, so it is not something to be trusted for the task – unless you can validate it. Even so, if the goal is to parse dependencies as part of a larger program, speed becomes a great priority and using out quanitzed model presents an agreeable compromise between performance and accuracy. However, if the goal is to learn parsing, GPT-4o far exceeds our model as it is not just parsing the text but is a LLM and will attempt to justify its answers and is open to discussion surrounding the task given. Our model on the other hand simply makes a prediction and would force a learner to learn from data them self, albeit data that is not always correct.

2 Graph-based dependency parsing

(a) Insufficiency for non-projective dependency trees.

In non-projective trees, relationships are non linear and cross one another. There exists a head and set of dependents such that they do not form a contiguous block of words. This means crossed relationships will not always occur adjacent in the stack. Operations LEFT-ARC and RIGHT-ARC build a relationship between adjacent elements. Since non-projective dependencies involve non-adjacent elements due to crossing, these relationships cannot be handled by transition-based parsing operations, which rely on dependencies being adjacent on the stack.

(b) Projective counting results.

```
$ SOLN=1 python3 run_test.py q2
Reading data... Done.
100%|                   | 12543/12543 [00:00<00:00, 1225149.74it/s]
train: 12081/12543 (96.3%)
100%|                   | 2001/2001 [00:00<00:00, 1428075.94it/s]
dev: 1957/2001 (97.8%)
100%|                   | 2077/2077 [00:00<00:00, 690025.30it/s]
test: 2036/2077 (98.0%)
```

(c) Gap degree.

Lets start with the dependency tree from question 1. We can break each word into the relevant subsequence and examine it from there. We can determine k for each word, using the fact that the degree of a word is the minimum k where the word and its descendants comprise $k+1$ maximally contiguous substrings, and pick the largest k as the gap degree of the dependence tree.

“To” has no children so it is therefore it is its own maximally contiguous subsequence of size 1. $k_{To} = 0$

“raise” has three descendants, “To”, “those”, and “doubts”. The also form a singular maximally contiguous subsequence of size 4 containing ’To, raise, those, doubts’. Likewise, $k_{raise} = 0$

“those” has no descendants, $k_{those} = 0$.

“doubts” has a singular adjacent descendant. $k_{doubts} = 0$

“is” has 2 children, “raise” and “resolve”. “resolve” has 2 children, “to” and “them”, who have no children themselves, $k_{to} = k_{them} = 0$. ’resolve’ is in between both, creating a singular maximally contiguous subsequence. $k_{resolve} = 0$. Going back to “is”, we see that it is in between “raise” and “resolve”, both of which have $k = 0$. This forms a singular maximally contiguous subsequence, of size 8. Thus $k_{is} = 0$.

Since all the values of k for all words in the first tree are 0, this means the gap degree of the dependency tree is also 0.

We can perform similar analysis of the second sequence.

Note that all independent words (words that are not head to any other words) have $k = 0$ as their contiguous sequence is the word itself. ($k + 1 = 1 \rightarrow k = 0$).

Thus, $k_{sam} = k_{today} = k_{who} = k_{alinguisticsdegree} = 0$.

“met” has 3 children, two of which have $k = 0$. The third, “a student”, has 3 descendants and is not contiguous with them. “a student” forms its own subsequnce and the descendants of its dependant, “has” forms another subsequence. $k_{astudent} = 1$. “has” is adjacent to its descendants. $k_{has} = 0$. Returning to “met”, it’s three children sets form a singular contiguous subsequence. $k_{met} = 0$

Taking the maximum of k reveals that the gap degree of the second tree is 1.

This analysis raises the question “Do projective dependency trees always have a gap degree of 0?”. Intuitively, it seems logical as all dependencies are required to be adjacent for the arc-standard operations can parse them, however this remains to be proven.

(e) Weight matrices initialization.

$$\mu = \tfrac{a+b}{2} \qquad \sigma^2 = \tfrac{(b-a)^2}{12}$$
$$0 = \tfrac{a+b}{2} \qquad \tfrac{2}{m} = \tfrac{(b-a)^2}{12}$$
$$a = -b \quad \rightarrow \quad \tfrac{2}{m} = \tfrac{(-a-a)^2}{12}$$
$$\tfrac{2}{m} = \tfrac{(2a)^2}{12}$$
$$\tfrac{2}{m} = \tfrac{4a^2}{12} = \tfrac{a^2}{3}$$
$$\tfrac{2}{m} = \tfrac{a^2}{3}$$
$$a = \sqrt{\tfrac{6}{m}}$$
$$b = -a \rightarrow b = -\sqrt{\tfrac{6}{m}}$$

(g) Term selection.

In the arc scorer, we are interested in finding the best head for a given dependent word. This relies on H_A, and D_A simultaneously, as seen in the first term which accounts for parsing the combined best head given a specific dependent word. The bias term multiplied by H_A gives some consideration to the head itself. A term of just the dependent is not as here since we are looking for the best head for the word, so the dependent is already known and its effect on choosing the head is accounted for in the first term. This is why we dont need D_A in the arc scorer.

In the label scorer, we are looking for the best label for a head-dependent relationship. This is dependent on both the head and the dependent as the same head can have multiple differently labelled dependents, and the same word can have a different relation to different heads. So the first term applies a simultaneous term considering both the words at the same time, while the second and third term account for the individual words impact on the relationship. This way the label scorer can base the score of the word on both the words and not just the head, since both have a role in defining the relationship.

(h) Double multiplication.

Standard classification problems typically have a singular standalone prediction value. This differs from our implementation as we are finding a prediction that has two (or 3) parts, the head, dependent, (and the relationship). This means we need to determine the best head-dependent relationship where the head can be any head in the sentence (except itself) and likewise for the dependent, leading to a choice of two values simultaneously, different from the singular standalone prediction. As mentioned in part (g), the first term computes the score for the simultaneous occurrence of head with dependent, as there is a likelihood for each pair of words to occur together. For example the pair $x = (head, dep)$ such that $x = (head, head)$ or $x = (dep, dep)$ has a likelihood of 0. This is something the first cumulative term calculates. Thus, we require this term to simultaneously classify the interaction between the two inputs – head and dependent.

(i) Masking constraint justification.

Constraint 1: No self-loops ($i == j$)

We require that there are no relationships where the head and the dependent are the same. This is by definition of dependency trees.

Constraint 2: No arcs where ROOT is the dependent ($i == 0$)

We require the ROOT no to be dependent to anything. Again this follows from the definition of dependency trees and the definition of the ROOT as being the only non dependent part of the tree.

Constraint 3: Only ROOT can have the "root" relation (label 0)

We require that the only relation that has the "root" label is also from ROOT to vertex any vertex i. This is by definition of the ROOT and "root" labels as the label "root" is reserved for the relationship from ROOT to the root of the dependency tree, of which there can only be one.

(j) MST *vs.* argmax.

MST is a necessary step because it ensures the validity of the dependency tree is maintained —- it is an acyclic, connected, and single-headed graph. Simply using argmax for arc prediction won't enforce these constraints and can result in invalid dependency trees with multiple heads, cycles, or disconnected components. This differs from predicting dependency relations, which is more like typical multiclass classification, where there is no structure constraints and any issues can be handled in post. But, to reiterate, with arc predictions, we need to ensure that the dependency tree generates is indeed a valid dependency tree.

It is true that the model can potentially learn to generate valid dependency trees as it sees this requirement from its training data, however this would require a deep and more complex model. By forcing it to make valid dependency tree predictions via MST, we enable a simpler model to be more powerful. If we just stick to argmax, the results space will include invalid dependency trees, leading to another avenue of error and a different path of error handling will be required for this.

(l) Parser mismatch.

Sentence 1: "The scientist explained the theory to students after conducting the experiment."

- **Transition-based parser output:**
  - Dependency: "explained" → "theory" (dobj), "explained" → "students" (prep)
  - Error: Incorrectly treats "theory" and "students" as separate arguments rather than linking them properly.
- **Graph-based parser output:**
  - Dependency: "scientist" → "explained" (nsubj), "explained" → "experiment" (prep)
  - Error: Fails to establish the correct prepositional attachment for "to students."
- **Preferred Parse:** I prefer the transition-based parser, as it better captures the object ("theory") and recipient ("students"), despite misinterpreting prepositional structure.

Sentence 2: "The complex issue was resolved by the committee during the meeting."

- **Transition-based parser output:**
  - Dependency: "resolved" → "issue" (nsubj), "resolved" → "committee" (agent)
  - Error: Treats "issue" as the subject rather than the object.
- **Graph-based parser output:**
  - Dependency: "issue" → "resolved" (root), "meeting" → "during" (prep)
  - Error: Fails to identify "committee" as the agent properly.
- **Preferred Parse:** I prefer the graph-based parser as it captures the main verb relations, even though it struggles with agent relations.

Sentence 3: "The government quickly responded to the emergency after the natural disaster."

- **Transition-based parser output:**
  - Dependency: "responded" $\rightarrow$ "government" (nsubj), "responded" $\rightarrow$ "disaster" (prep)
  - Error: Treats "disaster" as a prepositional object instead of attaching it to "emergency."
- **Graph-based parser output:**
  - Dependency: "government" $\rightarrow$ "responded" (nsubj), "emergency" $\rightarrow$ "responded" (dobj)
  - Error: Treats "emergency" and "disaster" as two separate objects of the same verb.
- **Preferred Parse:** I prefer the transition-based parser as it more accurately reflects the relationship between the government's response and the natural disaster, despite some prepositional missteps.