Document Analysis Assignment3

u5782545 Zhaolian Zhou

Q2

<s>	I	am	Sam	do	not	like	green	apple	and	</s>
4	4	3	4	1	1	1	1	1	1	4

$x_{i-1} = "am"$

$x_i = "Sam"$

$count(x_{i-1}) = 3$

$count(x_i, x_{i-1}) = count("Sam", "am") = 0$

$count(x_{i-1}, x_i) = count("am", "Sam") = 2$

$d = 0.75$

$$\lambda(x_{i-1}) = \frac{d}{count(x_{i-1})} \left|\{x | count(x_{i-1}, x_i) > 0\}\right| = \frac{0.75}{3} \left|\{\text{Sam}, \langle/s\rangle\}\right| = \frac{0.75}{3} * 2$$

$$P_{countinuation}(x_i) = \frac{\left|\{x_{i-1} | count(x_{i-1}, x_i) > 0\}\right|}{\left|\{(x_{j-1}, x_j) | count(x_{j-1}, x_j) > 0\}\right|} = \frac{\left|\{\text{am}, \langle s\rangle, and\}\right|}{\left|\left\{\begin{matrix}(\langle s\rangle, I), (I, am), (am, Sam), (Sam, \langle/s\rangle), (\langle s\rangle, Sam), (Sam, I), \\ (am, \langle/s\rangle), (I, do), (do, not), (not, like), (like, green), (green, apples), (apples, and), (and, Sam)\end{matrix}\right\}\right|} = 3/14$$

$$P_{NK}(x_i | x_{i-1}) = \frac{\max(count(x_{i-1}, x_i) - d, 0)}{count(x_{i-1})} + \lambda(x_{i-1}) P_{countinuation}(x_i) = \frac{\max(2 - 0.75, 0)}{3} + \frac{0.75}{3} * 2 * \frac{3}{14} = 52.38\%$$

Q3 Context-Free Grammars

The given context-free grammar rules can be represented as:

$$\text{Nominal} \rightarrow \text{PNP} \mid \text{PRP\$ Nominal} \mid Nominal\ Noun \mid Noun$$

$$\text{PRP\$} \rightarrow \text{my} \mid \text{his} \mid her \mid its$$

$$\text{PNP} \rightarrow \text{nounEndWithS}'$$

$$\text{Det Nominal} \rightarrow \text{Det Noun}$$

Apply the rules to the given cases:

- my uncle's bicycle:
    - my : PRP$, uncle's : PNP, bicycle : Noun
    - So the final result should be:
      $\text{Nominal} \rightarrow \text{PRP\$ PNP Noun}$

- Start from PRP$:
  $\mathrm{Nominal} \rightarrow \mathrm{PRP\$\ Nominal}$
- Then looking for PNP:
  $\mathrm{Nominal} \rightarrow \mathrm{PNP}$
  However, the rules end here, we cannot reach “bicycle”.

So we should **add another rule** in the rule set:

$\mathbf{Nominal \rightarrow PNP\ Nominal}$

And for apply for PNP, we only have: $\mathrm{PNP} \rightarrow \mathrm{nounEndWith}S'$, which cannot represent “uncle’s”. So we should add another rule: $\mathbf{PNP \rightarrow nounEndWith'S}$

Then, apply $\mathrm{Nominal} \rightarrow \ Noun$

- Companies’ workers
  - Companies’ -> PNP, workers -> Noun
  - So the final result should be:
    $\mathrm{Nominal} \rightarrow \mathrm{PRP\$\ PNP\ Noun}$
  - Apply **the new added** rule:
    $\mathrm{Nominal} \rightarrow \mathrm{PNP\ Nominal}$
  - Then apply $\mathrm{Nominal} \rightarrow \ Noun$
- a car
  - a -> Det, car -> Noun
  - We cannot find a rule from Nominal to Det.
    So we should **add another rule** instead of rule $\mathrm{Det\ Nominal} \rightarrow \mathrm{Det\ Noun}$ in the rule set:
    $\mathbf{NP \rightarrow Det\ Nominal}$
  - Apply $\mathrm{Nominal} \rightarrow \ \mathrm{Noun}$
  - We cannot find a rule from Det to “a”.
    So we should **add another rule** in the rule set:
    $\mathbf{Det \rightarrow a\ |\ an\ |the}$
- his books
  - his -> PRP$, books -> Noun
  - First, apply $\mathrm{Nominal} \rightarrow \ \mathrm{PRP\$\ Nominal}$
  - Then apply $\mathrm{Nominal} \rightarrow \ Noun$
- the bus stop
  - the -> Det, bus -> Noun, stop -> Noun
  - Apply **the new added** rule:
    $\mathbf{Nominal \rightarrow Det\ Nominal}$
  - Then apply $\mathrm{Nominal} \rightarrow \ Nominal\ Noun$
  - Then apply $\mathrm{Nominal} \rightarrow \ Noun$

Then set a start symbol, NP, and add a new to $\mathbf{NP \rightarrow Nominal}$

To be conclusion: the **modified grammatical rules** that cover the given case are:

$\mathbf{NP \rightarrow Nominal\ |\ Det\ Nominal}$

$\text{Nominal} \rightarrow \text{PNP} \mid \text{PRP\$ Nominal} \mid Nominal\ Noun \mid Noun \mid \textbf{PNP Nominal}$

$\text{PRP\$} \rightarrow \text{my} \mid \text{his} \mid her \mid its$

$\text{PNP} \rightarrow \text{nounEndWithS}' \mid \textbf{nounEndWith'S}$

$\textbf{Det} \rightarrow \textbf{a} \mid \textbf{an} \mid \textbf{the}$

Q4 Word Embedding

1. Add a pseudo-word <UNK> to the training dataset, all words occurred only in test dataset will be considered as <UNK>.
2. We can add 1 (or k, 0 < k <1) to all frequency counts. Which means, add 1 or k to the training dataset of each possible N-gram, and also add 1 or k to the training data of new occurred N-gram which contains the m unseen words in the test dataset. [1]
3. Use the frequency of all words occur only once in the training dataset to estimate the frequency of unseen words in the test dataset. [2]
4. For the N-gram in the test dataset that contains the word *w* which doesn't appear in the train dataset, we estimate the probability of (N-1)-gram, (N-2)-gram … that don't contain *w* and add them up with some coefficient $\lambda_i, 0 < \lambda_i < 1, \sum \lambda_i = 1$ [3]

Q5 Transition-based Dependency Parsing

1.

According to Joakim Nivre [4], a dependency graph $\mathrm{D} = (N_W, \mathrm{A})$ is well formed iff it satisfied the following conditions:

Single Head $(\forall\, \boldsymbol{n}\, \boldsymbol{n}' \boldsymbol{n}'')(\boldsymbol{n} \rightarrow \boldsymbol{n}' \wedge \boldsymbol{n}'' \rightarrow \boldsymbol{n}') \Rightarrow \boldsymbol{n} = \boldsymbol{n}''$

Acyclic $(\forall\, \boldsymbol{n}\, \boldsymbol{n}')\, \neg(\boldsymbol{n} \rightarrow \boldsymbol{n}' \wedge \boldsymbol{n}' \rightarrow^{*} \boldsymbol{n})$

Connected $(\forall\, \boldsymbol{n}\, \boldsymbol{n}')\,(\boldsymbol{n} \leftrightarrow^{*} \boldsymbol{n}')$

Projective $(\forall\, \boldsymbol{n}\, \boldsymbol{n}' \boldsymbol{n}'')(\boldsymbol{n} \leftrightarrow \boldsymbol{n}' \wedge \boldsymbol{n} < \boldsymbol{n}'' < \boldsymbol{n}') \Rightarrow (\boldsymbol{n} \rightarrow^{*} \boldsymbol{n}'' \vee \boldsymbol{n}' \rightarrow^{*} \boldsymbol{n}'')$

Note:

1. $\rightarrow^{*}$ denote the reflexive and transitive closure of the arc relation.
2. $\leftrightarrow$ and $\leftrightarrow^{*}$ represent undirected relations, i.e. $n \leftrightarrow n'$ iff $n \rightarrow n'$ or $n' \rightarrow n$

Back to the Nivre's parsing algorithm, the transition Left-Arc

$\langle \boldsymbol{n}|\boldsymbol{S}, \boldsymbol{n}'|\boldsymbol{I}, \boldsymbol{A} \rangle \rightarrow \langle \boldsymbol{S}, \boldsymbol{n}'|\boldsymbol{I}, \boldsymbol{A} \cup \{(\boldsymbol{n}', \boldsymbol{n})\} \rangle, \boldsymbol{n} \leftarrow \boldsymbol{n}' \in \boldsymbol{R}, \neg\exists \boldsymbol{n}''\,(\boldsymbol{n}'', \boldsymbol{n}) \in \boldsymbol{A}$

adds an arc $\boldsymbol{n}' \rightarrow \boldsymbol{n}$ from the next input token $\boldsymbol{n}'$ to the node n on top of the stack and reduces (pops) n from the stack.

The reason to remove the topmost element from the stack is to eliminate the possibility of adding an arc $\boldsymbol{n} \rightarrow \boldsymbol{n}'$,which would create a cycle in the graph, violate the acyclic condition.

With the same reason, the transition Right-Arc adds an arc $\boldsymbol{n} \rightarrow \ \boldsymbol{n}'$ from the node n on top of the stack to the next input token $\boldsymbol{n}'$. To prevent the create of cycles, the dependent node is immediately shifted.

2.

As described in Nivre's article [5], the time complexity of Nivre's algorithm is O(n), n is the length of the input sentence.

Parser configurations are represented by triples <S,I,A>, where S is the stack, I is the list of remaining input tokens, and A is the current arc relation for the dependency graph. During the whole parsing process, only one configuration c = <σ, β, A, I> needs to be stored at any given time.

Assuming that a single node can be stored in some constant space, the space needed to store σ and β is bounded by the number of nodes. And the same of A, number of arcs in dependency forest is bounded by the number of nodes, and each arc can be stored in constant space.

Hence, the worst-case space complexity is O(n).

References

[1] "Additive smoothing," Wikipedia, 19 Jun 2016. [Online]. Available: https://en.wikipedia.org/wiki/Additive_smoothing. [Accessed 22 09 2016].

[2] "Good–Turing frequency estimation," Wikipedia, 21 Aug 2016. [Online]. Available: https://en.wikipedia.org/wiki/Good%E2%80%93Turing_frequency_estimation. [Accessed 22 Sep 2016].

[3] "Linear interpolation," Wikipedia, 07 Jun 2016. [Online]. Available: https://en.wikipedia.org/wiki/Linear_interpolation. [Accessed 22 Sep 2016].

[4] J. Nivre, "An Efficient Algorithm For Projective Dependency Parsing".

[5] J. Nivre, "Algorithms for Deterministic Incremental Dependency Parsing," *Association for Computational Linguistics,* vol. 34, no. 4, pp. 513-553, 2008.