KU Leuven - NLP 2021-2022
23 November 2021

Exercise Session 3
Syntactic Parsing and Semantic Role Labelling with GNN

For questions contact:
Toledo Forum for Questions or
nlp@ls.kuleuven.be

In this session we discuss syntactic constituency parsing and dependency parsing as well as Semantic Role Labelling(SRL). Background can be found in chapters 12, 13, and 14 of "Speech and Language Processing" by Dan Jurafsky and James H. Martin (2019, 3rd edition) of which a draft can be found online at: https://web.stanford.edu/~jurafsky/slp3/
For the GNN and more specifically GCNs, you can look at the original blog post by the author of the GCN paper: https://tkipf.github.io/graph-convolutional-networks/.
Here is also a short youtube video describing and visualizing the basics of GCN https://youtu.be/2KRAOZIULzw. Note that his aggration and update of the layer is a bit different. But the visualization is great for understanding the concept of Graph Neural Networks.

1 Parsing Context-Free Grammars

In this exercise we cover the classical CKY (Cocke-Younger-Kasami, sometimes also CYK) algorithm for parsing context-free grammars (CFG). The CFG is discussed in chapter 12.2 of the book, and the CKY algorithm is discussed in chapter 13.2 of the book, which can be used as a reference.

1.1 Question 1.A

We consider the following CFG:

S	$\rightarrow$	NP	VP					
NP	$\rightarrow$	D	N	$\mid$	NP	PP	$\mid$	Pro
VP	$\rightarrow$	V	NP	$\mid$	V	NP	PP	
PP	$\rightarrow$	P	NP					
D	$\rightarrow$	*the*	$\mid$	*a*				
P	$\rightarrow$	*with*						
Pro	$\rightarrow$	*john*						
N	$\rightarrow$	*cat*	$\mid$	*tree*	$\mid$	*binoculars*		
V	$\rightarrow$	*sees*	$\mid$	*climbs*				

Now answer the solve the following open questions:

1. Think of two sentences that would be grammatical according to this grammar.

1.2 Question 1.B

We would like to use this grammar to parse sentences using the CKY algorithm. However, for CKY the grammar needs to be in Chomsky Normal Form (CNF). Convert the grammar given above into CNF.

1.3 Question 1.C

Now that we have our grammar in CNF, we could parse sentences using CKY. Find **all the trees** for the following sentences:

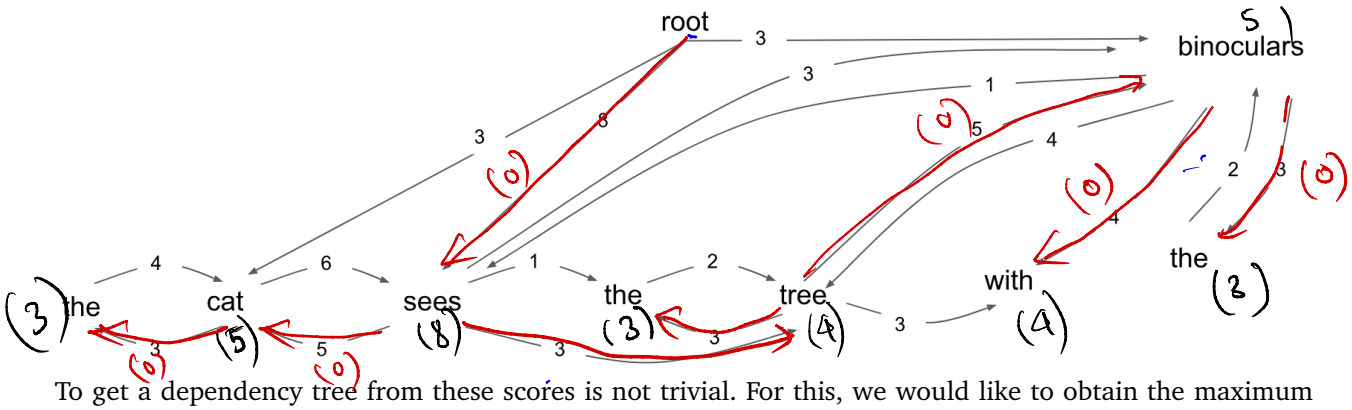
1. John climbs the tree.
2. The cat sees the tree with the binoculars.

1.4 Question 1.D

If everything went correct, you found two possible trees for sentence 2.
This type of ambiguity is called PP-attachment ambiguity. Make use of the difference in trees to describe what this ambiguity is and when it occurs.

2 Dependency Parsing

In this section we cover a classical algorithm for edge-factored dependency parsing models. This topic is covered in Chapter 14 of the book (particularly 14.5).
We assume we already have an edge scoring model. Consider the following scores given by our model:



To get a dependency tree from these scores is not trivial. For this, we would like to obtain the maximum spanning tree based on these scores. Use the Chu-Liu Edmonds (CLE) algorithm to get the maximum spanning tree. To which of the two meanings does this dependency parse correspond?

3 Using the dependency tree in a Graph Neural Network

In this exercise we work with a simplified model from the "Graph Convolution over Pruned Dependency Trees Improves Relation Extraction"paper: https://aclanthology.org/D18-1244/.

We just extracted a dependency tree in Question 2 that allows us to get insight into the meaning of the sentence. In this question, we will see how we can make use of the dependency tree to make other predictions for the sentence.
We are going to do semantic role labelling (SRL). We define a very simplistic model. The definition of the model is:

$$\hat{\boldsymbol{y}} = \text{softmax}(F_{pred}([\boldsymbol{h}; p]))$$

Where ; means concatenation, $\boldsymbol{h} \in \mathbb{R}^e$ is the vector representation of a word, $\hat{\boldsymbol{y}} \in \mathbb{R}^l$ the predicted label probabilities, and p the boolean value indicating if the word is the predicate or not, and F_{pred} is the prediction function or model. Our model consists of a GCN and as final step we use a linear layer function with the weight matrix $\boldsymbol{W}_{pred} \in \mathbb{R}^{m \times l}$ and biases $\boldsymbol{b}_{pred} \in \mathbb{R}^m$. e is the embedding size, l is the number of labels, m is the number hidden dimension. The possible labels are a reduced version of the prob bank: `<ARG0, ARG1, O>` with ARG0 the agent, ARG1 the patient, and O not part of a role.

There are many options for $\boldsymbol{h}$, such as pretrained word embeddings (word2vec, BERT) or some model like a bi-LSTM. However, we want stronger features that indicate relations between words. So we use a GCN over the dependency relations, to update the hidden states before passing them through the prediction layer. In addition to the dependency relations, we also use bidirectional sequence relations: each word has an additional incoming edge from the previous word and from the following word.

We don't have edge labels, so we don't use them. We define the direction of the edge as two different labels (incoming/outgoing), so the model can learn a distinction between incoming dependencies and outgoing dependencies.

Here is the definition of the GCN (taken and adapted from Equation 1 from the paper mentioned above):

$$\boldsymbol{h}_i^{(t+1)} = \text{ReLU}\big(\boldsymbol{h}^{(t)} + \sum_{n \in \mathbb{N}(i)} (\boldsymbol{W}_{in}^T \boldsymbol{h}_n + \boldsymbol{b}_{in}) + \sum_{m \in \mathbb{M}(i)} (\boldsymbol{W}_{out}^T \boldsymbol{h}_m + \boldsymbol{b}_{out})\big)$$

With $\mathbb{N}$ and $\mathbb{M}$ the set of incoming and outgoing neighbouring nodes respectively. $\boldsymbol{W}_{in} \in \mathbb{R}^{m \times m}$ and $\boldsymbol{W}_{out} \in \mathbb{R}^{m \times m}$ the weight matrices for incoming and outgoing neighbours, and $\boldsymbol{b}_{in} \in \mathbb{R}^m$ and $\boldsymbol{b}_{out} \in \mathbb{R}^m$ the

biases for incoming and outgoing neighbours. m is the hidden size.
We chose the hidden size between timesteps or GCN layers to be the same. This makes it possible to reuse the same weight matrices for multiple layers. It is not mandatory to keep the hidden size equal. In fact, for convolutions it is common to reduce size after every layer. In literature you also see different weight matrices for every layer. In this question we simply use a single timestep (or single GCN layer).

> **NOTE:** In this exercise we have multiple types of connections, without distinction between them (so we don't use their labels and don't apply different weights). This can result in multiple identical connections between words.

> **NOTE2:** We did not use the default GCN computations with adjacency matrices and such, but we used the version that takes the sum over the neighbours. This simplifies the understanding and makes it interpretable as an message passing algorithm.

Question: Predict the labels of each word using SRL model with GCN for the sentence:

> *the cat sees the tree with the binoculars*

As dependency tree, use the prediction from Question 2. The predicate of this sentence is **sees**.

Use the given weights and biases below. We also provide the initial embeddings for the words in the sentence in matrix $\boldsymbol{E} \in \mathbb{R}^{|V| \times 2}$, where $|V|$ is the vocabulary size. The vocabulary is:

```
['binoculars', 'cat', 'sees','the', 'tree', 'with']
```

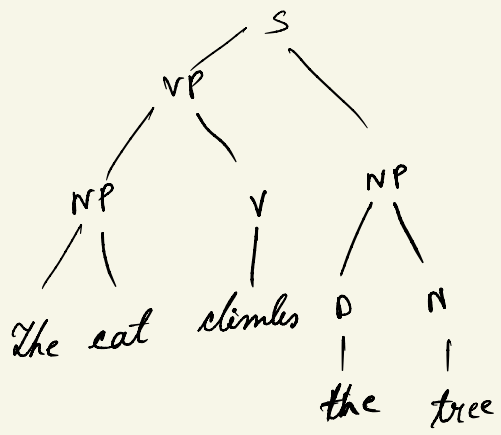
Use the following steps to solve the question:

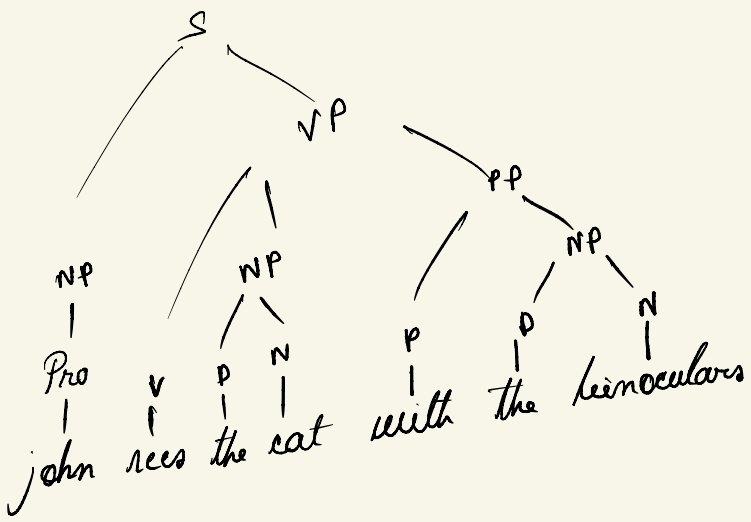
1. For each node, define the two sets of neighbours, incoming and outgoing. Duplicates are allowed.
2. Do one GCN step and update the hidden state for all the nodes in the graph to $\boldsymbol{h}^{(1)}$. Use the word embedding as $\boldsymbol{h}^{(0)}$, for each node. Do not forget to append the predicate boolean p to the hidden state before processing it through the GCN.
3. Use $\boldsymbol{h}^{(1)}$ to predict the role label for each of the words in the sentence.

$$\boldsymbol{E} = \begin{bmatrix} -0.5 & 0.7 \\ 1.4 & 1.1 \\ 0.8 & -0.5 \\ -0.3 & 0.1 \\ 1.3 & 0.8 \\ 0.8 & -0.8 \end{bmatrix} \qquad \boldsymbol{b}_{in} = \begin{bmatrix} 0.6 \\ 3.0 \\ -1.0 \end{bmatrix} \qquad \boldsymbol{b}_{pred} = \begin{bmatrix} 1.0 \\ -0.5 \\ -0.5 \end{bmatrix} \qquad \boldsymbol{b}_{out} = \begin{bmatrix} -0.7 \\ 1.0 \\ 0.2 \end{bmatrix}$$

$$\boldsymbol{W}_{in} = \begin{bmatrix} -0.5 & 0.0 & 0.6 \\ -2.0 & 0.5 & 1.0 \\ 2.0 & -0.7 & -0.8 \end{bmatrix} \qquad \boldsymbol{W}_{pred} = \begin{bmatrix} 3.0 & -2.0 & 0.2 \\ 0.1 & 2.0 & -3.0 \\ 0.7 & -2.0 & 2.0 \end{bmatrix} \qquad \boldsymbol{W}_{out} = \begin{bmatrix} 0.1 & -1.0 & 1.0 \\ -0.7 & -2.0 & 2.0 \\ 1.0 & 0.3 & -0.8 \end{bmatrix}$$

①





1.1

S → NP VP

NP → D N | NP PP | john

VP → V NP | X PP

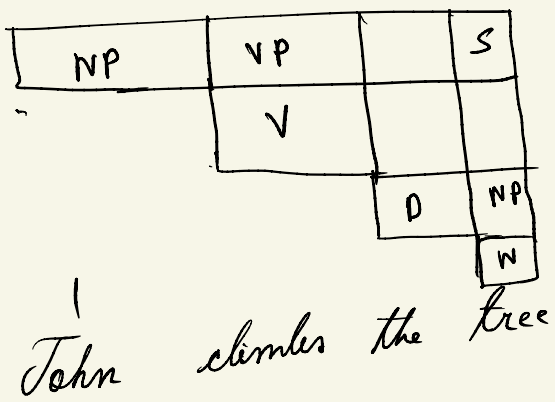
PP → P NP | X → V NP

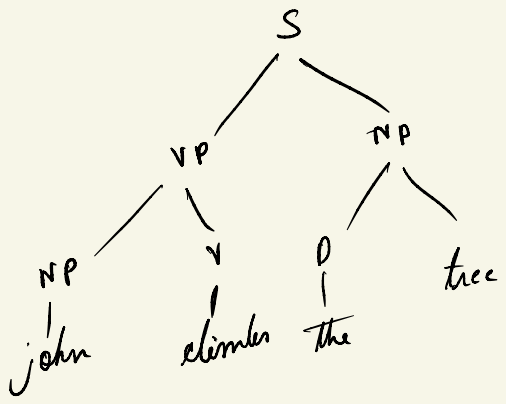
D → the | a

P → with

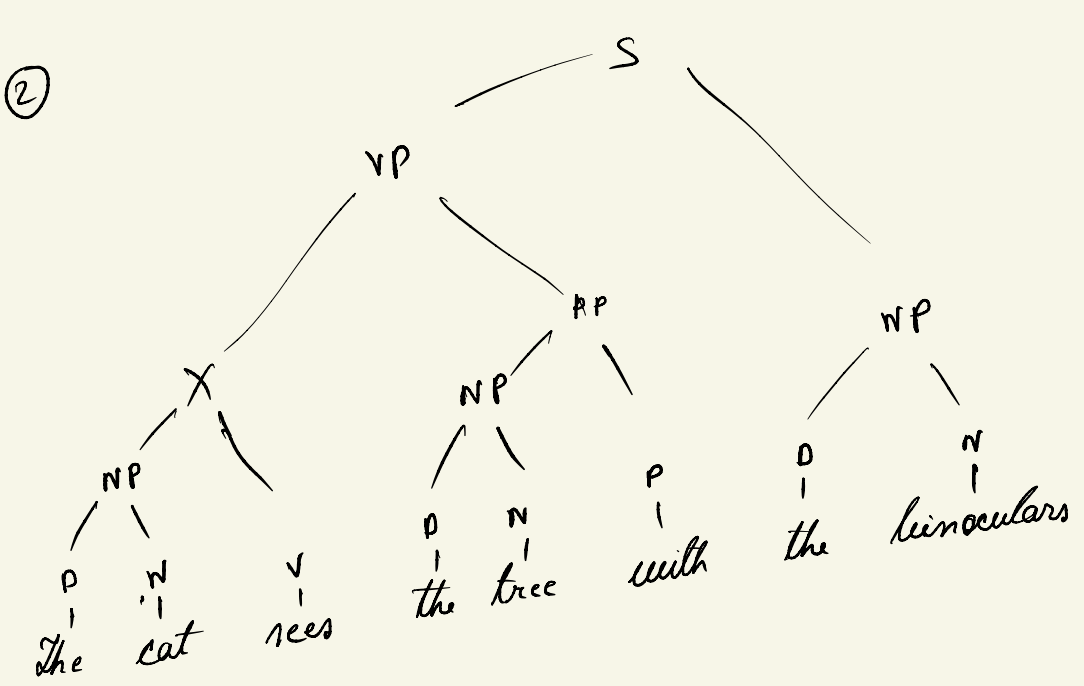
N → cat / tree / binoculars

V → sees / climbs





②



---

① $G=(V,E)$

$X \in \mathbb{R}^{N \times D}$, $A \in \mathbb{R}^{n \times n}$ AXW

$\therefore f(H^{(l)}, A) = O(A H^{(l)} W^{(l)})$

$\hat{y} = \text{softmax}(F_{feed}([h:p]))$

$$h_i^{t+1} = \text{RELU}\left(h^{(t)} + \sum_{n \in N(i)} \left(W_{in}^T h_n + b_{in}\right) + \sum_{m \in M(i)} \left(W_{out}^T h_m + b_{out}\right)\right)$$