NLP Assignment 4

Group 63: Ramon de Boer, Wesley van der Horst & Nando Beijaard

May 2024

1 Group Contract Reflection

1. **Ground Rules and Goals:** Our group established clear ground rules, which include regular Zoom meetings for discussing ideas and progress, dividing tasks, utilizing Overleaf for collaborative editing, and setting deadlines for task completion before the next meeting. Decisions are made through discussions where different viewpoints are considered, aiming for unanimous agreement. To improve our functioning, we could implement a clearer schedule for the meetings, making it evident when work needs to be completed and when we discuss progress.

2. **Conflict:** During Assignment 2, we encountered no significant conflicts as we were familiar with each other's working styles and expectations. However, there was a minor issue with a missing answers file from the submission. This was swiftly resolved through open discussion and mutual understanding. We treat each other respectfully and acknowledge human imperfections, enabling an accepting and collaborative environment. To further improve our ability to deal with conflicts, we could establish a more structured process for addressing issues as they arise, ensuring that all members feel comfortable voicing concerns promptly.

3. **Learning:** The most significant lesson from A2 was the importance of adaptability and open communication in addressing unforeseen challenges. Feeling comfortable reaching out to discuss problems is crucial. For A4, we aim to refine our planning and utilization of productive meetings to enhance our collaborative efficiency. This includes better time management and clearer task delegation, ensuring everyone is on the same page and contributing effectively to the group's goals.

2 Machine Learning & Neural Networks

a) Stochastic Gradient Descent (SGD)

SGD is an optimization method that is used to find the minimum of a function in steps or iterations, typically the loss function in machine learning models. Unlike regular gradient descent, which uses the entire dataset, SGD updates parameters using a random subset of data of constant but arbitrary size or minibatch, making it computationally efficient and suitable for large datasets, as common in NLP. SGD is therefore useful because it allows for quick adjustments to the model based on new data due to its effectiveness, enabling the model to learn complex patterns in text efficiently. Additionally, its ability to handle high-dimensional spaces common in NLP tasks, such as those involving word embeddings, makes it a practical choice for optimizing neural network-based NLP models.

b) Adam Optimizer

i.

The use of $\mathbf{m}$ in Adam stops the updates from varying as much as this factor of the rolling average of gradients focuses mostly on this general direction the gradients have been moving in instead of only the gradient from the previous iteration. This overcomes issues with noisy or fluctuating gradients, helps with escaping local minima and thus prevents getting stuck in suboptimal solutions, ideally enabling a more stable and efficient convergence towards the global minimum. In that sense momentum $\mathbf{m}$ acts as a weight that drags the optimization in the general direction of the gradients along a more consistent path towards the optimal solution.

ii.

The use of adaptive learning rates $\mathbf{v}$ allows for unique learning rates for each model parameter. As Adam uses $\sqrt{\mathbf{v}}$ to divide the update parameters, parameters that have more constant gradients (flat loss landscape) receive larger updates to promote faster convergence, whereas fluctuating or larger gradients receive smaller updates to avoid overshooting and provide more stability. The adoption of these adaptive learning rates overall enhances optimization efficiency by adjusting to the behaviour of each parameter gradient, promoting faster convergence for stable parameters and increased stability for unstable ones.

c) Dropout

i.

During the training of a neural network, it might recognise that activating a specific neuron consistently yields, at that moment, the best results. Consequently, backpropagation adjusts the model by increasing the weight corresponding to

this neuron and lowering the others. For example, in the first fully connected layer, the network might begin to depend predominantly on a single feature since it yields good results at that point, while simultaneously reducing the influence of other neurons by scaling their weights toward zero. This could mean that the model converges to a sub optimal local minimum.

To mitigate such overfitting, we can apply dropout, where neurons are randomly deactivated during training. This forces the network to not rely on any single neuron excessively all the time. Instead, it must distribute its power and learn to use multiple features and neurons to achieve accurate results, enhancing its ability to generalize and perform robustly on diverse data.

ii.

Firstly, dropout is designed to randomly deactivate neurons during training, which inherently causes those neurons to output 'wrong' or rather incomplete values on purpose, so it can learn from it. This is useful in training as it helps prevent over-reliance on any particular neuron, promoting a more robust and generalised model. However, during evaluation, the goal is to assess the true performance of the fully operational network. Using dropout at this stage would mean that the network is intentionally producing incorrect outputs, which is counterproductive to that goal.

Secondly, since dropout is random, different neurons may be deactivated in each pass through the network during evaluation. This leads to variability in the outputs for the same input for different evaluations. This inconsistency reduces the reliability of the network's performance metrics, since it becomes difficult to determine whether changes in output are due to the model's learning and adaptation or simply the result of random neuron deactivation.

3 Neural Transition-Based Dependency Parsing

a

Stack	**Buffer**	**New Dependency**	**Transition**
[ROOT]	[I, attended, lectures, in, the, NLP, class]		Initial Configuration
[ROOT, I]	[attended, lectures, in, the, NLP, class]		SHIFT
[ROOT, I, attended]	[lectures, in, the, NLP, class]		SHIFT
[ROOT, attended]	[lectures, in, the, NLP, class]	attended→I	LEFT-ARC
[ROOT, attended, lectures]	[in, the, NLP, class]		SHIFT
[ROOT, attended]	[in, the, NLP, class]	attended→lectures	RIGHT-ARC
[ROOT, attended, in]	[the, NLP, class]		SHIFT
[ROOT, attended, in, the]	[NLP, class]		SHIFT
[ROOT, attended, in, the, NLP]	[class]		SHIFT
[ROOT, attended, in, the, NLP, class]	[]		SHIFT
[ROOT, attended, in, the, class]	[]	class→NLP	LEFT-ARC
[ROOT, attended, in, class]	[]	class→the	LEFT-ARC
[ROOT, attended, class]	[]	class→in	LEFT-ARC
[ROOT, attended]	[]	attended→class	RIGHT-ARC

Table 1: Sequence of Transitions for Parsing "I attended lectures in the NLP class"

b

In every case, all words in the buffer have to be moved to the stack using a SHIFT transition, this takes n steps. From then, all words will be assigned a LEFT-ARC or RIGHT-ARC transition. This last part also takes n steps. The total number of steps will thus be $n + n = 2n$ steps.

c

The dataset is marked with part-of-speech (POS) tags and dependency relationships. Every token within a sentence receives an annotation regarding its part-of-speech and its connection to the head token in the dependency parse tree.

g

The best Unlabeled Attachment Score (UAS) achieved on the development set is 88.62%. On the test set, the UAS achieved is 88.95%. UAS is a metric for evaluating dependency parsing performance as it measures the percentage of words correctly attached to their heads, focusing on structural accuracy without considering dependency labels. This metric is particularly useful for assessing the parser's ability to identify head-dependent pairs, which is fundamental for understanding sentence structure. UAS is language-independent and provides valuable insights into the accuracy of head assignments, which is crucial for many NLP tasks. However, UAS does not account for the correctness of dependency labels, which is important for detailed syntactic analysis, and may be less informative for languages with complex non-projective structures. Therefore, UAS is only essential for evaluating structural accuracy,

4 Error Analysis

a

Solution:

- Sentence 1:
  - Error type: Verb Phrase Attachment Error
  - Incorrect dependency: acquisition → citing
  - Correct dependency: blocked → citing
  - Explanation: Citing and the rest of the sentence provides extra information about why it is blocked.
- Sentence 2:
  1.
     - Error type: Modifier Attachment Error
     - Incorrect dependency: had → already
     - Correct dependency: left → already
     - Explanation: The adverb "already" should link to the main verb "left" to indicate the timing of the action.
  2.
     - Error type: Modifier Attachment Error
     - Incorrect dependency: left → early
     - Correct dependency: Friday → early
     - Explanation: "Early" belongs as a compound together with "Friday" and "afternoon" and this compound already has a dependency through "left" and "afternoon"
- Sentence 3:
  - Error type: Prepositional Phrase Attachment Error

  - Incorrect dependency: declined → decision
  - Correct dependency: reasons → decision
  - Explanation: The government decision should clarify the reasons, not the declination to comment itself.

- Sentence 4:
  - Error type: Coordination Attachment Error
  - Incorrect dependency: affects → one
  - Correct dependency: plants → one
  - Explanation: One specifies the amount of plants in Quebec, and should thus link to plants instead of affects

b

Figure 1 shows the output from inputting the garden path sentence into the dependency parser available at: https://demos.explosion.ai/displacy. The parser incorrectly links "horse" to the verb "raced," changing the intended meaning of the sentence. It also incorrectly links "fell" as a compound to "raced." The correct dependency parsing is illustrated in Figure 2, where "raced past the barn" is correctly linked to the noun "horse," serving as a modifier. This structure is akin to rephrasing the sentence as "The horse, that was raced past the barn, fell."

Given this, and considering our experience with transition dependency parsers, it is expected that they handle garden path sentences similarly to the depiction in Figure 1. The dependency parser from demos.explosion.ai is based on the transition system dependency parsing framework discussed by Honnibal and Johnson (2015)[1]. These parsers often make errors akin to those made by humans reading the sentence for the first time, as they build context incrementally from the input buffer, leading to initial misattachments that are explained only after more information is processed.

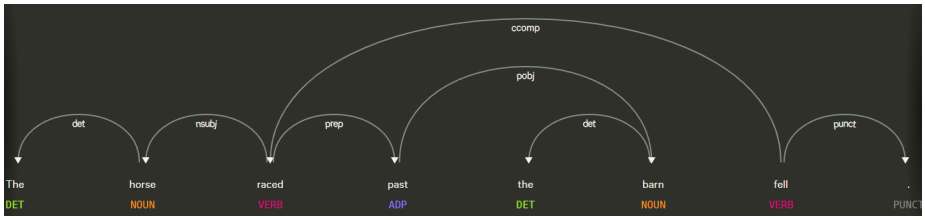

Figure 1: (i) the initial "garden path"

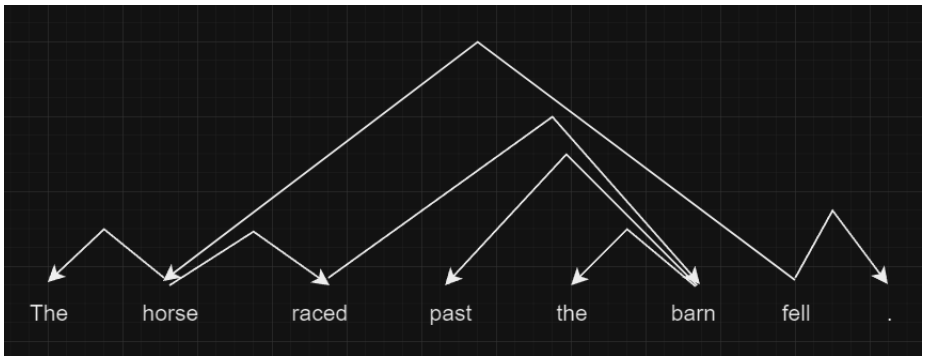

Figure 2: (ii) the final, correct parser of the sentence

c

First of all, it helps distinguishing words that have the same spelling but different meaning. For example, "can" can be used as "Can you help?" but also as "Give me that can". Using POS tags helps the parser distinguishing

POS tags can be used to determine the syntactic roles of words in sentences. For example, in the sentence "The old man the boats", the word "man" could be mistaken as a person rather than a verb, manning the boat.

In garden path sentences the POS tags can also help. For example, in the sentence "The horse raced past the barn fell", the correct parsing depends on recognising "raced" as something modifying "horse" rather than a simple past verb.

5 Bonus: Cross-lingual dependency parsing

Example sentences for each type of error:

- Prepositional Phrase Attachment Error: "He found the keys on the table"
- Verb Phrase Attachment Error: "Running in the park, she enjoyed the sunny weather."
- Modifier Attachment Error: "She saw the man riding a bike quickly."
- Coordination Attachment Error: "He likes playing football and watching movies."

The translation according to google translate:

- Prepositional Phrase Attachment Error: "Hij vond de sleutels op tafel"
- Verb Phrase Attachment Error: "Rennend in het park genoot ze van het zonnige weer"

- Modifier Attachment Error: "Ze zag de man snel fietsen."
- Coordination Attachment Error: "Hij houdt van voetballen en films kijken."

The dependency parser found the following dependencies for the sentences:

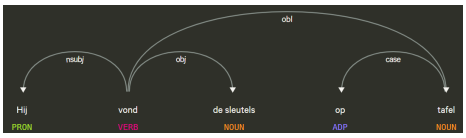

Figure 3: Hij vond de sleutels op de tafel

In both English and Dutch, prepositional phrases (PP) can lead to attachment errors. The ambiguity regarding whether the PP describes where the keys are or the action of finding is still present in both languages.

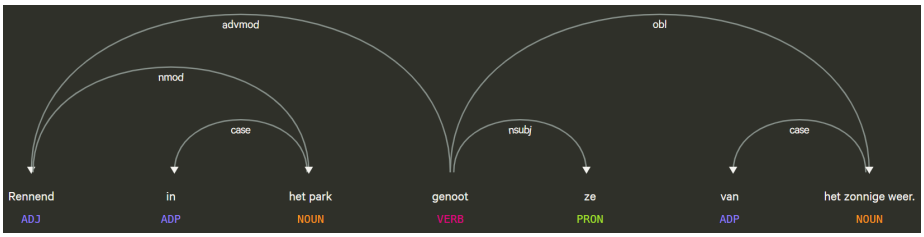

Figure 4: Rennend in het park genoot ze van het zonnige weer.

Here it becomes more clear that the Dutch syntax allows for more flexibility without introducing additional punctuation. For example, we can easily change the sentence without sacrificing clarity to "Ze genoot van het zonnige weer rennend in het park". Whereas if we would change the sentence in english to "She enjoyed the sunny weather running in the park", we would typically like to add "while" before running to clarify that the enjoyment and running occurred simultaneously.

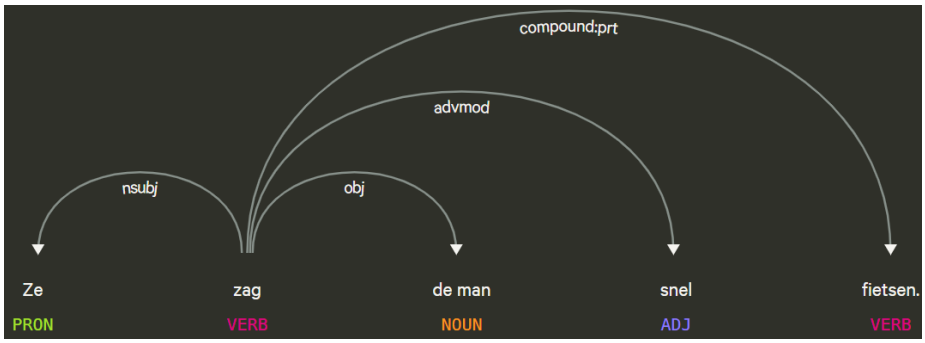

Figure 5: Ze zag de man snel fietsen.

In the English example, there's a potential risk of incorrectly parsing the connection between 'riding' and 'bike,' since these are separate words that require grammatical linking. However, in Dutch, this issue is mitigated because the action and the object are combined into a single verb 'fietsen,' which inherently includes the meaning of both riding and biking. This mitigates the risk of making such a form of a "Verb Phrase Attachment Error", where it is not a verb with a phrase but a verb with a head.

Furthermore, in Dutch, there's a potential hazard that 'snel' (quickly) might be incorrectly attached as modifying 'zag' (saw), meaning that the observation was done quickly rather than biking. Thus there is a bigger chance of a modifier attachment error, which is less likely in English in this context because adverb placement more clearly indicates which verb it modifies due to syntactic constraints. In English, saying 'She saw quickly the man biking' would mean that she saw the man quickly, instead of the fact that the man is biking quickly. However, as we can see in the dependency diagram in Dutch it can mean both, and the algorithm incorrectly drew a dependency between quickly and saw instead of biking and quickly.

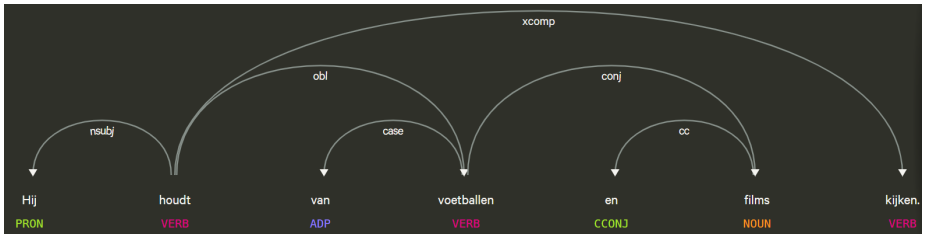

Figure 6: Hij houdt van voetballen en films kijken.

The "xcomp" relationship indicates that "voetballen" and "kijken" share a dependency on "van" without their own subjects, relying on the subject of the main verb "houdt." This structure might lead to complexity in parsing as the parser must recognize that "Hij" is the implicit subject of both "voetballen"

and "kijken."

Furthermore, "van" typically introduces a dependency that can be across multiple verbs or actions, as it must clearly link them back to the verb "houdt" (likes). This is a critical linkage that, if missed, could alter the understood meaning or relationships in the sentence.

References

[1] Honnibal, Matthew and Johnson, Mark, An Improved Non-monotonic Transition System for Dependency Parsing, in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Màrquez, Lluís and Callison-Burch, Chris and Su, Jian (eds.), Association for Computational Linguistics, Lisbon, Portugal, pp. 1373–1378, Sep. 2015. `https://aclanthology.org/D15-1162`, 10.18653/v1/D15-1162.