Assignment 4 - Group 42

Joost Driessen - 2674286
Rick van Ravensberg - 2673204
Rohan Zonneveld - 2641655

May 23, 2023

1 Machine Learning & Neural Networks

(a) Stochastic Gradient Descent

Stochastic Gradient Descent is a computationally more efficient way of regular Gradient Descent, because instead of training on the entire dataset, the algorithm is iteratively trained on mini-batches, which are supposed to be a representative selection of the whole dataset. With each iteration, a new mini-batch is formed and the model parameters are updated accordingly. This method is useful in NLP to train the weights of neural language models, which can do all kinds of tasks such as word prediction, machine translation or text classification.

(b) Adam Optimizer

i

m is the rolling average of the gradients. Using this will mitigate sudden changes in the gradients, because the rolling average changes less than the most recent gradient changed. This results in a more steady training process without the model parameters changing in another direction with each subsequent iteration. It also reduces the number of iterations it takes to converge and is very robust in noisy landscapes with a lot of local minima.

ii

v is the rolling average of the squared gradients. Given that θ is divided by the square root of **v**, it's value is lower when **v** increases. **v** increases when the squared rolling average increases. This means that the learning rate for parameters that have been changed a lot recently receive a lower learning rate than parameters that have been 'stuck' for a while. This approach helps with exploration of variables that seem to be stuck in local minima.

(c) Dropout

i

Dropout is the process of randomly eliminating neurons from the neural network that is being trained in order to force it to make different connections, introduce noise and thus prevent overfitting.

ii

Dropout should not be used during evaluation because after dropout, the network has to be trained again. If one uses dropout during evaluation it will just impair the quality of the evaluation results, by deleting knowledge acquired during the training phase.

2 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
[ROOT]	[]	ROOT → attended	RIGHT-ARC

Table 1: Transition-based Dependency Parser

(b)

$2n + 1$. Every word has to be moved from the Buffer to the Stack (n). By creating new dependencies words are removed from the Stack one-by-one, this process continues until all words are removed (n). The $+1$ is the result of the initial configuration.

(c)

The data is in CoNNL-format. It consists of sentences, where each line represents a token. For each token specific information is specified in columns. The columns are POS-tag, index of the token that token is dependent on, dependency label. The model learns the dependencies by applying a neural net to

the features in each sentence. In this case, the features are the tokens and their POS-tags.

(d)

See `__init__` and `parse_step` functions in the `PartialParser` class in `parser_transitions.py`.

(e)

See `minibatch_parse` function in `parser_transitions.py`.

(f)

See the `__init__`, `embedding_lookup` and `forward` funcitons in `parser_model.py` and the `train_for_epoch` and `train` functions in `run.py`.

(g)

The model achieves an UAS of 88.61 on the dev set and 89.29 on the test set. UAS is a useful metric for evaluating dependency parsing because it measures how well a parser accurately assigns syntactic dependencies between words in a sentence. The evaluation metric directly reflects the achievement of the goal of the dependency parser by calculating the proportion of correctly predicted dependencies (ideally all dependencies would be predicted correctly, which coincides with a perfect score of 100). A downside of UAS is that it is blind to where a dependency parser is failing. It could be that the dependency parser never predicts a specific dependency label. Some task may depend on this specific label, the dependency parser will fail horribly although it has a high UAS score.

3 Error Analysis

(a)

i

- **Error type**: Verb Phrase Attachment Error
- **Incorrect dependency**: acquisition → citing
- **Correct dependency**: blocked → citing

ii

- **Error type**: Modifier Attachment Error
- **Incorrect dependency**: had → already
- **Correct dependency**: left → already

iii

- **Error type**: Prepositional Phrase Attachment Error
- **Incorrect dependency**: declined → decision
- **Correct dependency**: reasons → decision

iv

- **Error type**: Coordination Attachment Error
- **Incorrect dependency**: affects → one
- **Correct dependency**: plants → one

(b)

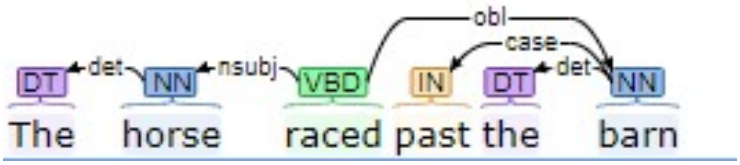

Figure 1: CoreNLP's interpretation of the garden path sentence before it's finished

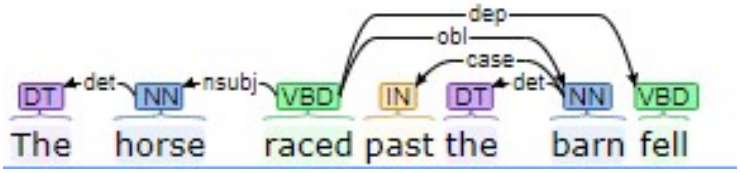

Figure 2: CoreNLP's interpretation of the garden path sentence after it's finished

A transition-based dependency parser would probably classify the relations in the sentence 'The horse raced past the barn fell' correctly, because the root is the last relationship that is determined. This means that after all words except 'raced' and 'fell' have exited the stack, the only possible correct way to correctly classify this sentence is by forming the relation 'fell' → 'raced' and assigning 'fell' as the root of the sentence.

(c)

Part-of-speech tags help this parser predict the correct relationships between words, because certain parts of speech are usually connected to other parts of speech. For instance, a determiner is often followed by a noun. So if the dependency parser receives the information that a determiner and a noun entered the stack subsequently, it would probably predict a relationship between the two, while it might not do that when these part-of-speech tags are not available.

4 Bonus

We first translated a set of English sentences into Russian manually to retain the grammatical integrity and the context. To ensure the highest fidelity in translation between English and Russian, our sentences were translated manually by a native Russian speaker who also possesses a high level of proficiency in English. This translation process aims to mirror the grammatical structures as closely as possible between the languages while maintaining the natural linguistic characteristics of each.

Following the translation, we utilized the UD models for UDPipe in Python, an environment well-suited for NLP tasks. This model enabled us to run a dependency parser on the translated sentences to examine how these sentences are broken down and analyzed in Russian. The UDPipe models used for this project were the latest 2.5-191206 files. In order to visualize the annotated output from the UDPipe parser in a more accessible format, We made use of the online CONLL-U formatter available at http://spyysalo.github.io/conllu.js.

(a) Prepositional Phrase Attachment Error

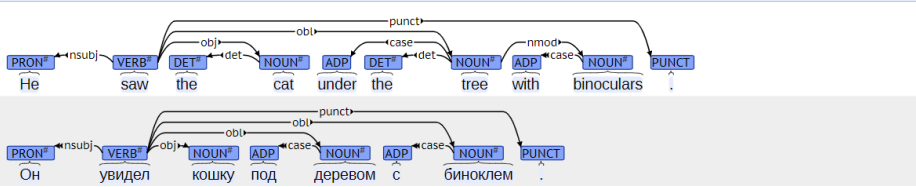

Figure 3: Prepositional Phrase Attachment

This error involves the incorrect attachment of a prepositional phrase to a word. For instance, in the English sentence 'He saw the cat under the tree with binoculars', the phrase 'with binoculars' could be associated with 'he saw' or 'the cat.' In this case, the parser incorrectly associates it with 'tree'. Conversely, in the Russian equivalent, the parser correctly attaches 'с биноклем' (with binoculars) to 'увидел' (saw).

(b) Verb Phrase Attachment Error

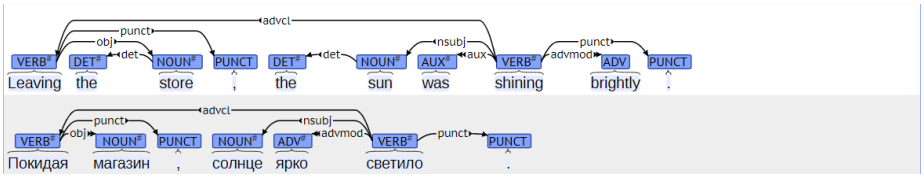

Figure 4: Verb Phrase Attachment

The parser can misinterpret which verb a phrase modifies. In 'Leaving the store, the sun was shining brightly', the parser correctly attaches 'Leaving the store' to 'was shining brightly', indicating the sun was shining as the subject was leaving the store. The parser also correctly handles the Russian equivalent, 'Покидая магазин, солнце ярко светило'.

(c) Modifier Attachment Error

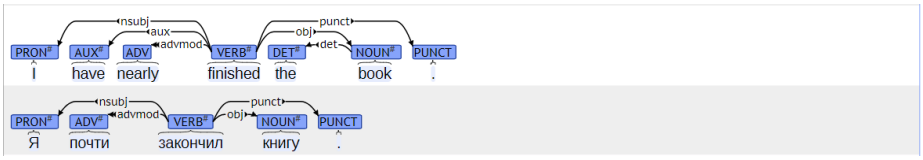

Figure 5: Modifier Attachment

This error occurs when the parser inaccurately associates a modifier with a word. In 'I have nearly finished the book', the parser correctly attaches 'nearly' to 'finished', indicating the subject is close to finishing the book. The parser also handles the Russian equivalent 'Я почти закончил книгу' accurately.

(d) Coordination Attachment Error

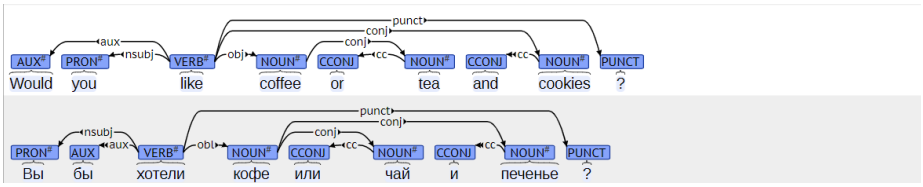

Figure 6: Coordination Attachment

This error involves misinterpretation of words connected by coordinating conjunctions. In 'Would you like coffee or tea and cookies?', the parser incorrectly

groups 'tea and cookies' together, suggesting the choice is between 'coffee' or 'tea and cookies'. The parser made the same error with the Russian equivalent, 'Вы бы хотели кофе или чай и печенье?'.

When we ran our sentences through the parser, we didn't consistently find the common parsing issues such as Prepositional Phrase Attachment, Verb Phrase Attachment, Modifier Attachment, or Coordination Attachment. However, it's crucial to note that the absence of these errors in our sample doesn't mean they don't occur. In fact, parsing errors are prevalent when dealing with complex and ambiguous sentences, even though UDPipe handled our specific sentences quite well.

The difficulty of parsing doesn't get easier when transitioning from English to Russian. However, differences in the languages can sometimes play a role. For instance, a well-known example in English: 'I saw the man on the hill with a telescope' can be misinterpreted due to the ambiguity of 'with a telescope'. It could either mean that the speaker used a telescope to see the man, or it could mean that the man on the hill had a telescope.

However, in Russian, this ambiguity is less likely to arise. This is because of the different word order in Russian and the nature of Russian grammar. In Russian, the prepositional phrase 'с телескопом' is closer to 'мужчину на холме' (the man on the hill), implying that the man had a telescope. If the speaker were using the telescope, a different preposition would typically be used (через телескоп - through a telescope), or the sentence structure would be different to make it clear.

In conclusion, despite the robustness of modern parsing tools like UDPipe, the unique features and complexity of different languages can present challenges that influence parsing accuracy. It's crucial to remain mindful of that when using these advanced tools.

Contribution

Joost did Part 1 and Part 3. Rohan did Part 2. Rick did the Bonus. After an assignment was completed the corresponding team member explained the steps and answered remaining questions by the other team members.