2023

A4: Dependency Parsing

Group 44

Course:	Natural Language Processing
Students:	Eduard Alexa 2709417 Ana Negru 12628859
Place:	Vrije Universiteit Amsterdam
Date:	23 May 2023

1. Contents

2. Table of figures

1. Part 1

1.1 a) (5 points) Stochastic Gradient Descent.

1.1.1 In 3-4 sentences, please explain stochastic gradient descent and how it could be useful for an NLP task.

Stochastic gradient descent (SGD) is often used as an optimization algorithm in NLP tasks like language modelling, machine translation and sentiment analysis where datasets are typically large and high dimensional, which can make it computationally expensive to train models using the entire dataset in each iteration. SGD works by iteratively and randomly selecting a small batch of training samples to update model parameters, which reduces the computational cost of training and can lead to faster convergence towards the optimal solution. NLP tasks also commonly involve non-convex optimization problems with local optima and SGD has been proven to be good at finding suitable solutions in such scenarios. Finally, SGD also allows for online learning, where models can learn from data arriving in a new stream which can be important for tasks such as text classification, where new data is constantly generated.

1.2 b) (5 points) Adam Optimizer

1.2.1 Briefly explain in 2–4 sentences (you don't need to prove mathematically, just give an intuition) how using m stops the updates from varying as much and why this low variance may be helpful to learning, overall.

The intuition behind momentum is that if gradients in consecutive iterations are similar, the updates will be more consistent and lead to quicker convergence. The optimizer can consider the direction of the previous gradients and adjust the current update accordingly by using a rolling average of the gradients. This leads to more stable updates that reduce oscillations in optimizations. The low variance is helpful to learning because it allows the model to avoid getting stuck in the local minima and leads to better generalization on unseen data

1.2.2 Since Adam divides the update by √v, which of the model parameters will get larger updates? Why might this help with learning?

In Adam optimizer, the model parameters with smaller values of √v will get larger updates. This is because the division by the square root of the rolling average of the magnitude of gradients (v) leads to a larger update for parameters with smaller √v values. Parameters that have large gradients and are changing rapidly will have smaller updates while parameters with smaller gradients that change slowly will have larger updates. This adaptive learning rate technique can be helpful in learning because it helps the optimizer to adjust the learning rate for each parameter based on the magnitude and history of the gradients. This is of use for NLP tasks where the

gradients of different parameters can have varying magnitudes. By using a different learning rate for each parameter, Adam can help the model converge faster and more accurately by giving more weight to the gradients of parameters that have a larger impact on the loss function. This learning rate technique can also help the optimizer escape from saddle points and plateaus (which are common in high-dimensional optimization problems) and is achieved by scaling the updates based on the curve of the loss function.

Parameters that have large gradients and are changing rapidly will have smaller updates, while parameters that have small gradients and are changing slowly will have larger updates. This allows the optimizer to converge more quickly and accurately to the optimal solution.

1.3 c) (5 points) Dropout regularization technique.

1.3.1 I. (2 points) Why should dropout be applied during training?

Dropout should be applied during training to prevent overfitting of the neural network to the training data. This occurs when the model learns noise of the training data and fits perfectly to it, making it difficult for it to generalize on new, unseen data. Dropout helps to prevent overfitting by randomly dropping units in the hidden layer during training, which forces the network to learn more robust/distributed representations of input features. Dropout also prevents the network from relying too much on any single feature, which also helps with overfitting

1.3.2 ii. (3 points) Why should dropout NOT be applied during evaluation? (Hint: it may help to look at the paper linked above in the write-up.)

Dropout is mainly intended for the training phase and shouldn't be applied for evaluation. During evaluation, the whole network with all its neurons is used to make predictions on new data and applying dropout during evaluation would lead to random noise being added to the output because different predictions would be made for the same input. Naturally this negatively impacts the predictive quality of the model. Dropout also affects the output scale of a network, making it difficult to compare outputs of different networks. This is because the output of a network with dropout is scaled by the dropout probability during training but not during evaluation. So, the output of a network where dropout is used during training can't be directly compared with the output of the network without training dropout during evaluation. As mentioned above, dropout is mainly used during training to prevent overfitting and during evaluation, the network should make predictions on new data and not require any regularization.

2. Part 2

2.1 a)

Go through the sequence of transitions needed for parsing the sentence "I attended lectures in the NLP class". The dependency tree for the sentence is shown below. At each step, give the configuration of the stack and buffer, as well as what transition was applied this step and what new dependency was added (if any).

Step	Stack	Buffer	New Dependency	Transition
0	[ROOT]	[I, attended, lectures, in, the, NLP, class]		Initial configuration
1	[ROOT, I]	[attended, lectures, in, the, NLP, class]		SHIFT
2	[ROOT, I, attended]	[lectures, in, the, NLP, class]		SHIFT
3	[ROOT, attended]	[lectures, in, the, NLP, class]	Attended, I	LEFT-ARC
4	[ROOT, attended, lectures]	[in, the, NLP, class]		SHIFT
5	[ROOT, attended, lectures, in]	[the, NLP, class]		SHIFT
6	[ROOT, attended, lectures, in, the]	[NLP, class]		SHIFT
7	[ROOT, attended, lectures, in]	[NLP, class]	The, in	LEFTARC
8	[ROOT, attended, lectures]	[NLP, class]	In, lectures	LEFTARC
9	[ROOT, attended]	[class]	Lectures, attended	RIGHTARC
10	[ROOT, attended, NLP]	[class]		SHIFT
11	[ROOT, attended, NLP, class]	[]	Class, NLP	LEFTARC
12	[ROOT, attended, NLP]	[]	NLP, attended	RIGHTARC
13	[ROOT]	[]	Root, attended	RIGHTARC

2.2 b)

A sentence containing n words will be parsed in how many steps (in terms of n?). Briefly explain in 1-2 sentences why.

A sentence with n words is going to be parsed in n-1 steps. This is because every word except for the root needs to have one parsing step to determine its dependency relation. Because there are n-1 non-root words in a sentence of length n, the parsing process will take n-1 steps.

2.3 c)

Inspect the data you will be using manually. What format is it in? How is it separated? Explain how the data is labelled already for your use.

Upon manual inspection of the data, we see that it is in tab-separated format. Each individual line represents a tokenized word in a sentence, and the columns represent different properties of the word like its word index, word form, lemma, POS-tag, dependency label and syntactic head index. Columns are separated by tabs.

The data is labelled with linguistic annotations. For instance, the “pos” column gives the part-of-speech tags for each word. The “dep” column gives the dependency labels which indicate the syntactic relationships between words. The “head” column indicates the index of the word's syntactic head in the sentence, which governs/controls the entire phrase and determines its grammatical properties/syntactic behaviour. All these annotations mentioned help us to understand the grammatical structure of the sentence as well as relationships between words.

2.4 d)

(4 points) Implement the __Init__ and parse step functions in the PartialParse class in parser transitions.py. This implements the transition mechanics your parser will use. You can run basic (non-exhaustive) tests by running python parser transitions.py part c.

2.5 e)

(6 points) Our network will predict which transition should be applied next to a partial parse. We could use it to parse a single sentence by applying predicted transitions until the parse is complete. However, neural networks run much more efficiently when making predictions about batches of data at a time (i.e., predicting the next transition for any different partial parses simultaneously). We can parse sentences in minibatches with the following algorithm.

2.6 f)

(28 points) We are now going to train a neural network to predict, given the state of the stack, buffer, and dependencies, which transition should be applied next.

2.7 g)

(4 points) Report the best UAS your model achieves on the dev set and the UAS it achieves on the test set in your write-up. Why is UAS a useful metric for evaluating dependency parsing? Cite specific examples where the UAS metric does and does not provide meaningful insight into the performance of your parser.

The best UAS score is 89.06 after 10 epochs.

```
Epoch 10 out of 10
100%|
Average Train Loss: 0.06704714936022828
Evaluating on dev set
1445850it [00:00, 17630929.29it/s]
- dev UAS: 88.89
New best dev UAS! Saving model.

================================================================================
TESTING
================================================================================
Restoring the best model weights found on the dev set
Final evaluation on test set
2919736it [00:00, 24953206.87it/s]
- test UAS: 89.06
Done!
```

Figure 1: Best UAS score

The Unlabelled Attachment Score (UAS) is a standard metric to evaluate dependency parsing. It measures the percentage of correctly predicted head-dependent attachments without considering the specific dependency labels. [1]

This metric can be particularly useful in the following scenarios:

- Parsing Accuracy: UAS provides an overall assessment of how well the parser predicts the proper syntactic relationships between words. It demonstrates the parser's capacity to identify a sentence's fundamental structure [2].
- Comparative Analysis: Comparing various parsing models or systems is made facilitated with UAS. It can determine which model or system performs better in terms of fundamental attachment correctness by comparing the UAS scores [2].

However, UAS has limitations in certain cases:

- Fine-grained Analysis: The semantic relation (such as Subj) used to label the bond between the head and the kid is not taken into account by UAS. [2].
- Error Patterns: Information regarding individual parser error patterns is not provided by UAS. It does not distinguish between different kinds of errors; rather, it treats all faulty attachments equally. Consequently, UAS may not identify a particular parser improvement opportunities. [2].

There are other standard metrics that evaluate parsers [1]:

- Labelled attachment score (LAS) the percentage of words with the proper head and dependence labels, or the labelled attachment score
- Labelled accuracy score (LabAcc), which measures the percentage of words with the correct dependence label is known as the labelled accuracy score.

3. Part 3

3.1 3 a) We'd like to look at example dependency parses and understand where parsers like ours might be wrong.

3.1.1 i.

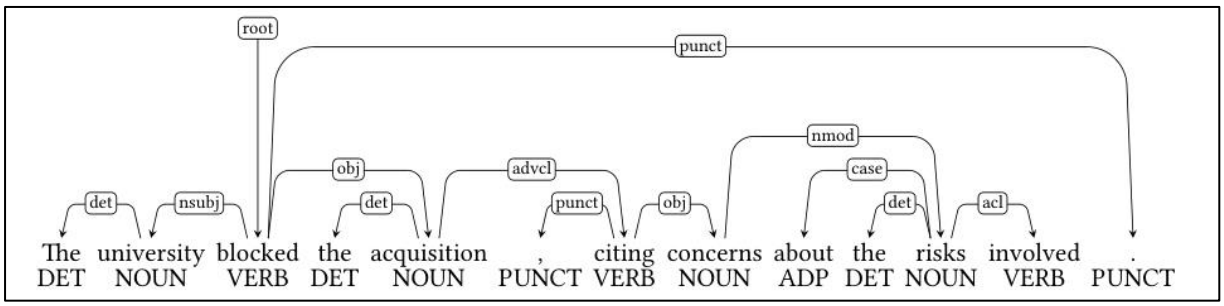

Figure 2: sentence 1

The university blocked the acquisition, citing concerns about the risks involved.

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

3.1.2 ii.

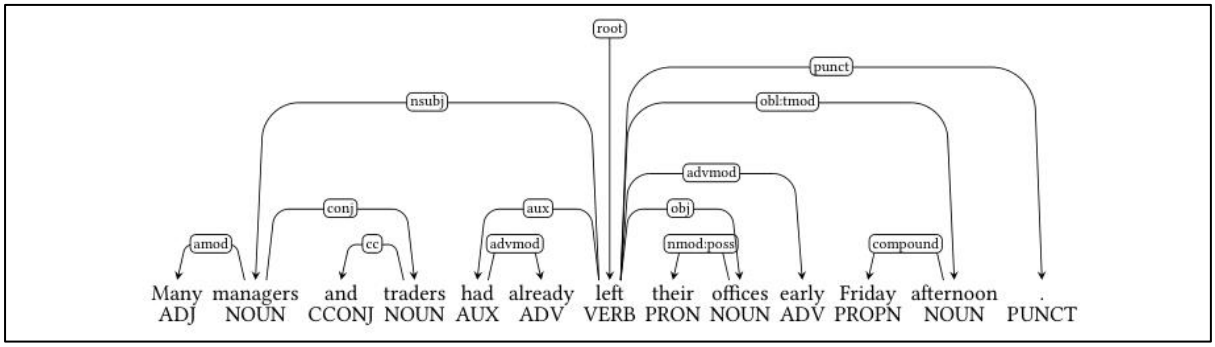

Figure 3: sentence 2

Many managers and traders had already left their offices early Friday afternoon.

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

3.1.3 iii.

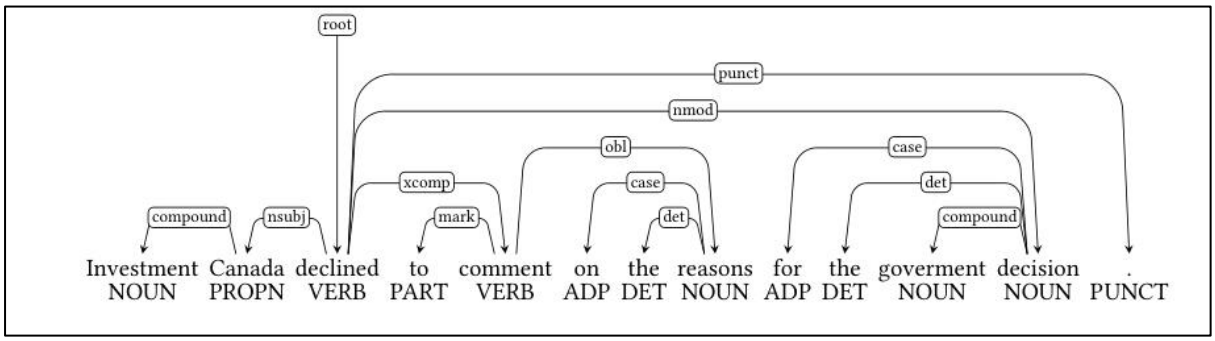

Figure 4: sentence 3

Investment Canada declined to comment on the reasons for the goverment decision.

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

3.1.4 iv.

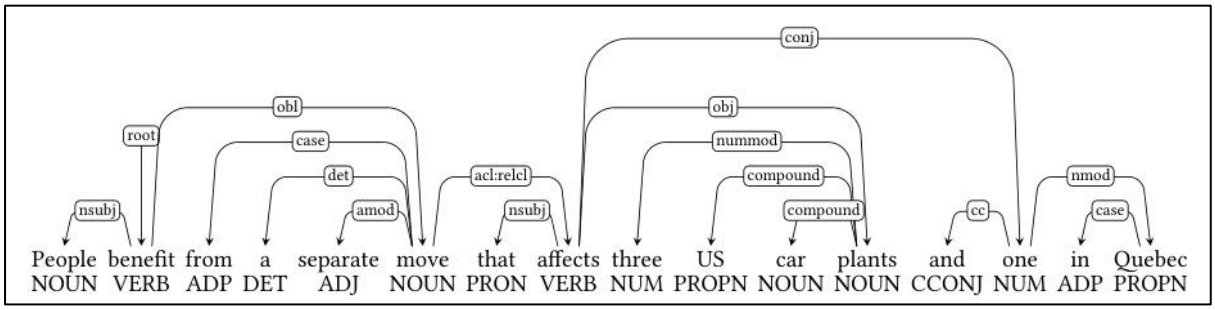

Figure 5: sentence 4

People benefit from a separate move that affects three US car plants and one in Quebec

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

3.2 3 b)

In class, we saw an example of a garden-path sentence, or a grammatically correct sentence that starts in such a way that a listener's or reader's likely parses the structure incorrectly. The sentence we discussed was "The horse raced past the barn fell". Using an online dependency parser (e.g. CoreNLP), show (i) the initial "garden path" parse structure computed before the parse is corrected, and (ii) the final, correct parser of the sentence. Having now worked with a transition-based dependency parser, how do you think it would handle such sentences?

The incorrect version:

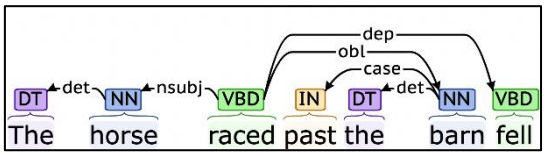

Figure 6: Sentence

A transition-based dependency parser would handle the sentence “The horse raced past the barn” as follows:

Step	Stack	Buffer	New Dependency	Transition
0	[ROOT]	[The, horse, raced, past, the, barn]		Initial configuration
1	[ROOT, The]	[horse, raced, past, the, barn]		SHIFT
2	[ROOT, The, horse]	[raced, past, the, barn]		SHIFT
3	[ROOT, The, horse, raced]	[past, the, barn]		SHIFT
4	[ROOT, The, horse, raced, past]	[the, barn]		SHIFT
5	[ROOT, The, horse, raced, past, the]	[barn]		SHIFT
6	[ROOT, The, horse, raced, past]	[barn]	The, horse	LEFTARC
7	[ROOT, The, horse, raced]	[barn]	Past, raced	LEFTARC
8	[ROOT, The, horse]	[barn]	Raced, horse	LEFTARC
9	[ROOT, The]	[barn]	Horse, The	LEFTARC
10	[ROOT, The, barn]	[]		SHIFT
11	[ROOT, The, barn]	[]	Barn, the	RIGHTARC
12	[ROOT, The]	[]	The, ROOT	RIGHTARC

The sentence “The horse raced past the barn fell” would be handled as follows by the transition-based dependency parser:

Step	Stack	Buffer	New Dependency	Transition
0	[ROOT]	[The, horse, raced, past, the, barn, fell]		Initial configuration
1	[ROOT, The]	[horse, raced, past, the, barn, fell]		SHIFT
2	[ROOT, The, horse]	[raced, past, the, barn, fell]		SHIFT
3	[ROOT, The, horse, raced, past]	[past, the, barn, fell]		SHIFT
4	[ROOT, The, horse, raced, past]	[the, barn, fell]		SHIFT

5	[ROOT, The, horse, raced, past, the]	[barn, fell]		SHIFT
6	[ROOT, The, horse, raced, past, the, barn]	[fell]		SHIFT
7	[ROOT, The, horse, raced, past, the]	[fell]	Barn, the	LEFTARC
8	[ROOT, The, horse, raced, past]	[fell]	The, horse	LEFTARC
9	[ROOT, The, horse, raced]	[fell]	Past, raced	LEFTARC
10	[ROOT, The, horse]	[fell]	Raced, horse	LEFTARC
11	[ROOT, The]	[fell]	Horse, the	LEFTARC
12	[ROOT, The, fell]]	[]		SHIFT
13	[ROOT, The, fell]	[]		RIGHTARC
14	[ROOT, The]	[]	Fell, the	RIGHTARC
15	[ROOT]	[]	The, root	RIGHTARC

Differences compared to the previous sentence ("The horse raced past the barn"):

- Step 7: The transition is a LEFTARC operation with "barn" as the dependent and "The" as the head, capturing the dependency relationship between "barn" and "The".

- Step 12: A SHIFT operation is performed to process the word "fell" and move it from the buffer to the stack.

- Step 13: The transition is a RIGHTARC operation, marking "fell" as a dependent of "The" and removing "fell" from the stack.

- Step 14: Another RIGHTARC operation is performed, creating the dependency relationship between "fell" and "The".

- Step 15: The final RIGHTARC operation establishes the dependency between "The" and the ROOT node, completing the parsing process

4. Part 4

4.1 Bonus: Cross-lingual dependency parsing (10 points)

Choose a language you are familiar with other than English, and explore how the four types of parsing errors introduced in (3a) may be challenges in cross-lingual dependency parsing. To do this, follow these steps:

• Choose an example English sentence that demonstrates each error type. You may use sentences from this assignment or come up with your own.

• Translate these sentences into the language you are working with. You may use an automatic translator or translate the sentence yourself. Explain your translation process.

• Run a dependency parser for your language on the translated sentences. For common NLP languages, online demos like CoreNLP may support your language. For other languages, you may need to install a new parser: see the Universal Dependencies Website for supported UD languages.

• Assess how the challenges of English dependency parsing using a transition-based parser translate into the language you have chosen. Are the challenges the same? Are there new challenges that English may not face? Feel free to add additional examples that may not be directly translatable from English.

i. The university blocked the acquisition, citing concerns about the risks involved.

a. Translation: L'università ha bloccato l'acquisizione, esprimendo preoccupazioni per i rischi coinvolti.

b. Comment: in this case the word that proved to be more difficult was “involved” where it could be translated into many different words with different meanings depending on the context (connessi, impliciti, legati)

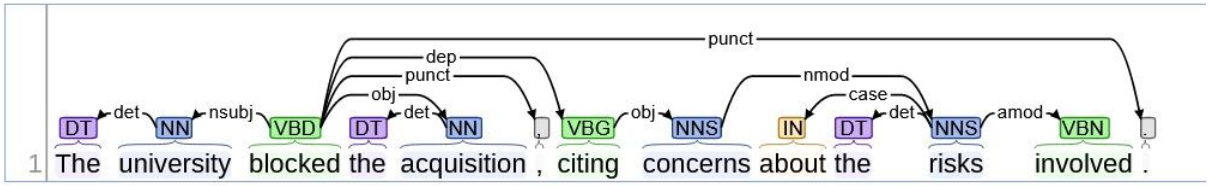

Figure 7: English 1

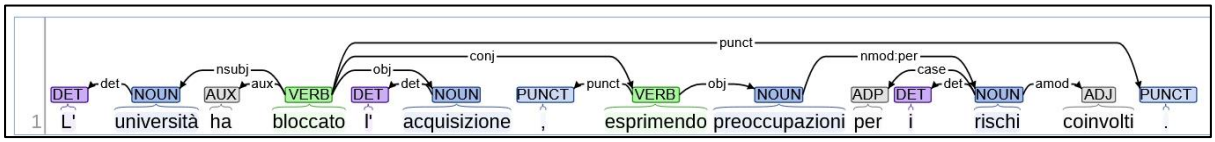

Figure 8: translation 1

ii. Many managers and traders had already left their offices early Friday afternoon.

a. Translation: Molti manager e trader avevano già lasciato i loro uffici nel primo pomeriggio di venerdì.

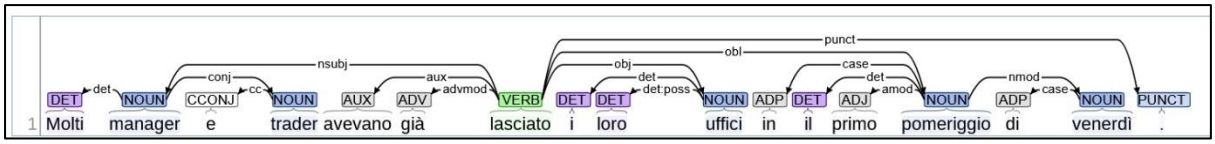

Figure 9: translation 2

iii. Investment Canada declined to comment on the reasons for the goverment decision.

a. Translation: Investment Canada ha rifiutato di commentare le ragioni della decisione del governo.

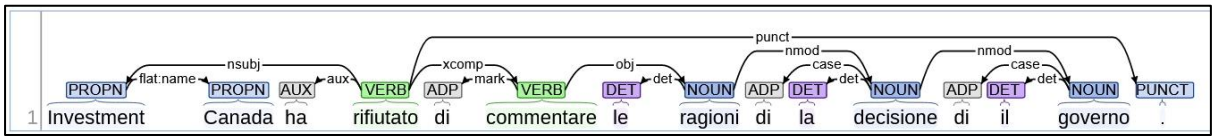

Figure 10: translation 3

iv. People benefit from a separate move that affects three US car plants and one in Quebec

a. Translation: I cittadini beneficiano di un movimento separato che riguarda tre stabilimenti automobilistici statunitensi e uno in Quebec

b. Comment: in this case the difficulty was to understand what does move mean in Italian as it can be translated into several different words (spostamento, mossa, azione, manovra).

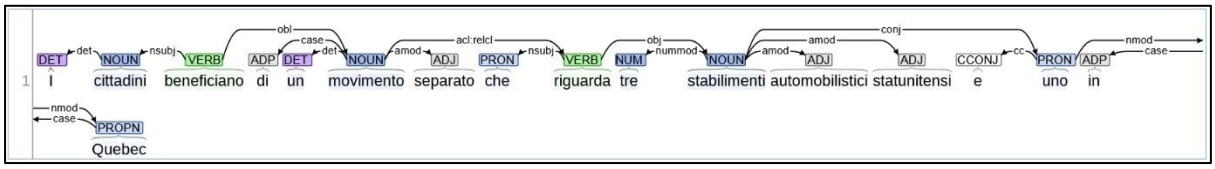

Figure 11: translation 4

The translation process into Italian proved to more difficult. The objective was to keep the meaning of the structure of the sentences, while using a natural language. As to not sound like a robot. Also, there was an increased difficulty into choosing some specific words that could transmit the same meaning.

The challenges increase as in Italian there is the presence of gender which can be used to change the structure of many words. This will affect the conjugation of the verbs, adjectives, articles etc.

5. References

[1] J. Foster and J. van Genabith, 'Parser Evaluation and the BNC: Evaluating 4 constituency parsers with 3 metrics'.

[2] 'Dependency parsing', *NLP-progress*. http://nlpprogress.com/english/dependency_parsing.html (last accessed May 23, 2023).

6. Contribution

Ana wrote part 1, worked together with Eduard on the code.

Eduard contributed to the code and bonus part.

Both students worked on part 3, and generally contributed to the project in equal measure.