**PART1:**

First of all, I convert the .conllu files to spacy files via terminal with the following commands.

python -m spacy convert .\tr\_imst-ud-train.conllu ..\Task1\ -c conllu -t spacy

python -m spacy convert .\tr\_imst-ud-dev.conllu ..\Task1\ -c conllu -t spacy

python -m spacy convert .\tr\_imst-ud-test.conllu ..\Task1\ -c conllu -t spacy

After loading these files, I add the token.dep\_ value of all the tokens in all the doc value in training data.

Then, I structured a training loop with 5 iteration because every iteration takes too long I kept it small. I shuffle the data in every iteration and divide it to batches. For every doc in batch I create an spacy Example object and call the spacy update function to update the model according the Example instance. I evaluate the development scores and provide its output in the code.

Then, I test the model in the test data and provide their UAS and LAS scores.

Labeled Attachment Score (LAS) on test set: 0.44

Unlabeled Attachment Score (UAS) on test set: 0.59

Unlabeled Attachment Score(UAS):

* Regardless of the dependency's label, UAS calculates the proportion of tokens in a text that are accurately matched to their heads in the dependency tree.
* It ignores the nature of the relationships and just evaluates if the syntactic structure, or who is connected to whom,is correct.

Labeled Attachment Score(LAS):

* It calculates the proportion of tokens with the proper dependence label correctly affixed to their heads.
* LAS takes into account the particular kinds of relationships (such as subject, object, modifier, etc.) that exist between words in addition to the dependency tree's structure.

Difference Between Them:

The way dependence labels are taken into account separates UAS and LAS from one another. UAS alone examines the structure, but LAS additionally takes label correctness into account.

Why?

LAS is typically lower than UAS because, in addition to recognizing the correct structure, gaining the correct label is an additional obstacle in LAS. It's more challenging to correctly identify the kind of relationship and affix a token to its head.

SENTENCE EXPERIMENTS:

"Güneş doğudan yükselir ve batıda batar."

This sentence is relatively straightforward with a clear subject (Güneş) and verbs (yükselir, batar). It's parsed correctly by the parser, because the training data included similar simple sentence structures.

A black and white diagram

Description automatically generated

Akıllı telefonlar, günümüzün vazgeçilmez teknolojik aletleridir.

This sentence, while not overly complex, contains a descriptive clause. The parser has been adequately trained on similar structures, so it correctly identifies the relationship between "akıllı telefonlar" and the rest of the sentence.

A diagram of a curved object with arrows

Description automatically generated with medium confidence

Yarınki toplantıda, müdürün asistanı tarafından sunulan, şirketin kazanç raporunu gözden geçireceğiz.

This sentence is more complex due to its length and the presence of multiple clauses. It poses a challenge in terms of correctly attaching the clauses to the main verb ("gözden geçireceğiz"). The parser struggles with this sentence more because of the lack of similar structures in training set.

A different types of arrows

Description automatically generated with medium confidence

Çocuğun parkta, köpeğiyle oynarken gördüğümüz adam, eski bir arkadaşım çıktı.

This sentence includes a relative clause and presents potential ambiguity in the parsing. The complexity arises in determining the relationships between "çocuğun," "adam," and "arkadaşım." This type of sentence proves to be challenging for the parser, depending on the training data and the parser's capabilities in handling relative clauses and nested structures.

A diagram of a diagram

Description automatically generated with medium confidence

Overall, training data representation, complex sentence structures and ambiguity seems to be potential causes of different parsing perfromances.