**Reflection on our process towards building BUParsum**

We had initially started our project to build a program that would help linguistics process their dataset, providing a general overview of their dataset. BUParsum uses Universal Dependencies data, parses it, and returns a multiple bar chart demonstrating the frequencies of the features taken as input and how these frequencies vary from language to language, from data to data. So far we have been able to create a foundation for taking input from the user and returning the graphics. For the parsing part of the project, we intended to use two different parser projects available on GitHub. The initial parser function is adapted from Karahan Şahin’s project named Char-Level-Morphological-Parsing-with-Transformers. The second parser named Direct Attentive Dependency Parser, which we were going to use to parse plain text input from the user, would be a great addition to our project. However, we were not able to implement it since we could not retrieve the necessary data. The main reason for this was that the parser was actually a package and it took us a really long time even trying to understand how it worked in the background. The most challenging part for us has been figuring out the codes we did not write ourselves since we did not know the decision-making processes behind the functions and the rest of the code. Once we figured out how these codes function, we tested them using different datasets that have been annotated following the Universal Dependencies guidelines. The code failed with all the datasets except for English, so we tested the code constantly by printing the outcome after each change we made until we got the result we wanted.

After having handled the parsing part of our code, we focused on the user interface. Initially, we wanted to provide the user with a dropdown menu through which they could select the language(s) and the feature(s) they want BUParsum to analyze, but we could not find any resource that would help us with implementing the selected item in the code. After a brief discussion on this, we have decided to go with a simpler version which would also help us take multiple inputs. It was easier to build upon the code while working with the default language datasets we had. However, it was another challenge for us to figure out a way to take datasets from the user. For this part, we have utilized the Diaparser as mentioned above.

Another problem we faced along the way was the visualization part. At first, we only utilized Matplotlib to create the charts, however, when we tried to give multiple languages -or features- as input, the results were not how we wanted. So we also implemented pandas in our code. Then another problem came up since our chart did not demonstrate the exact numbers on the bars. Since this was a significant problem, we browsed over almost every source we found on the web. It was apparent to us that it was best we used .bar\_label for this, but we found out the version embedded in Google Colab was not up to date. We then figured out how to upgrade an already existing package, and solved the problem once and for all.

In our first meeting with our TA Karahan, he suggested we provided a correlation matrix to the user as well. However, so far, we have not succeeded in our trials, so our final version of the project does not include this property.

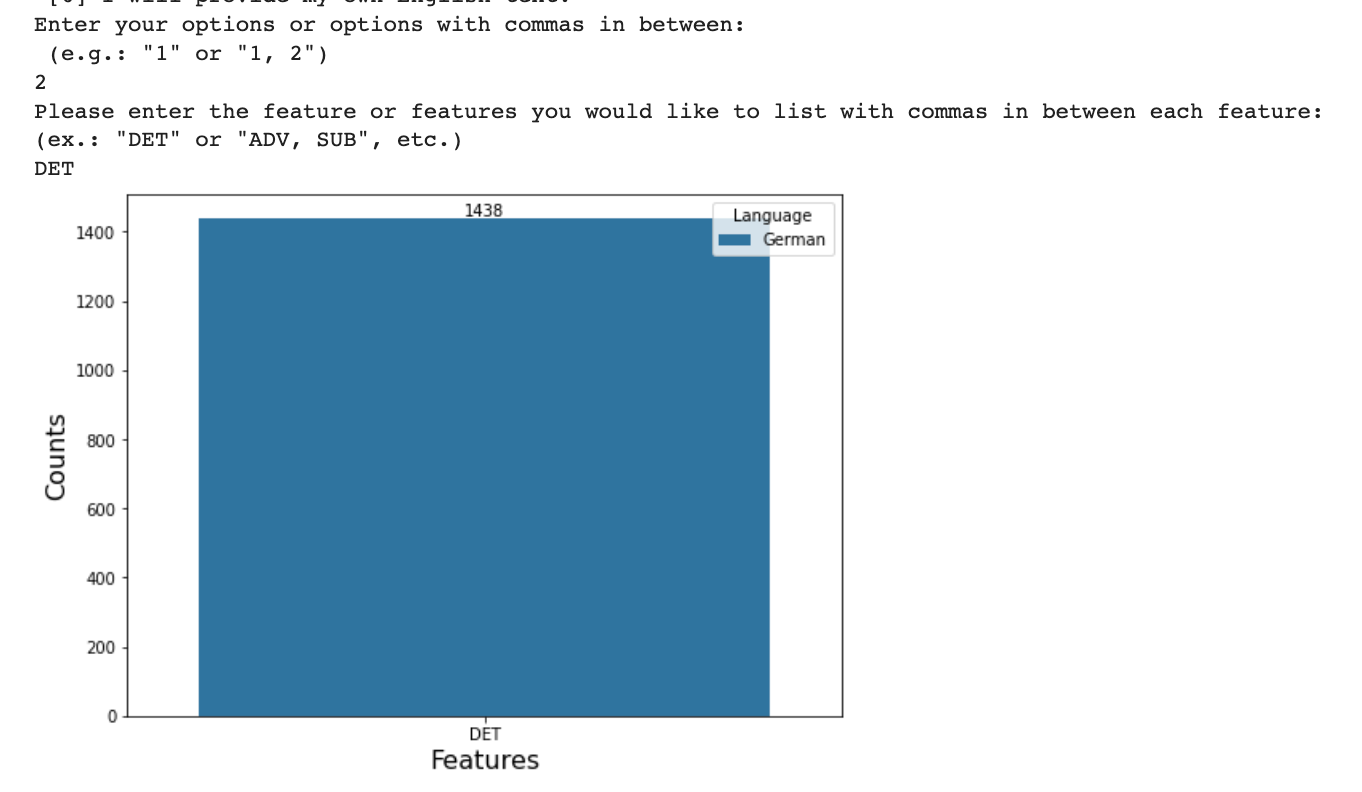
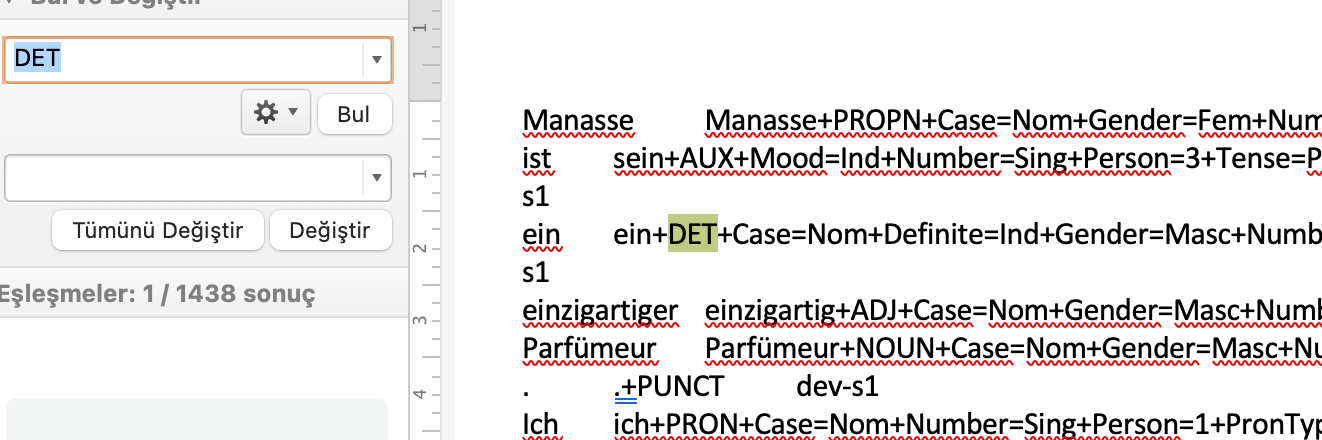
Finally, for the evaluation metric, we have decided to go with precision in that what we actually want to be sure of is whether our code (both the parsing and the visualizing part) gives correct numbers. In order to do this, we had to check whether the feature count the code gives us matches that of the actual document manually. We compared both results and they all matched. If we were to calculate the precision, we would have used the following formula:



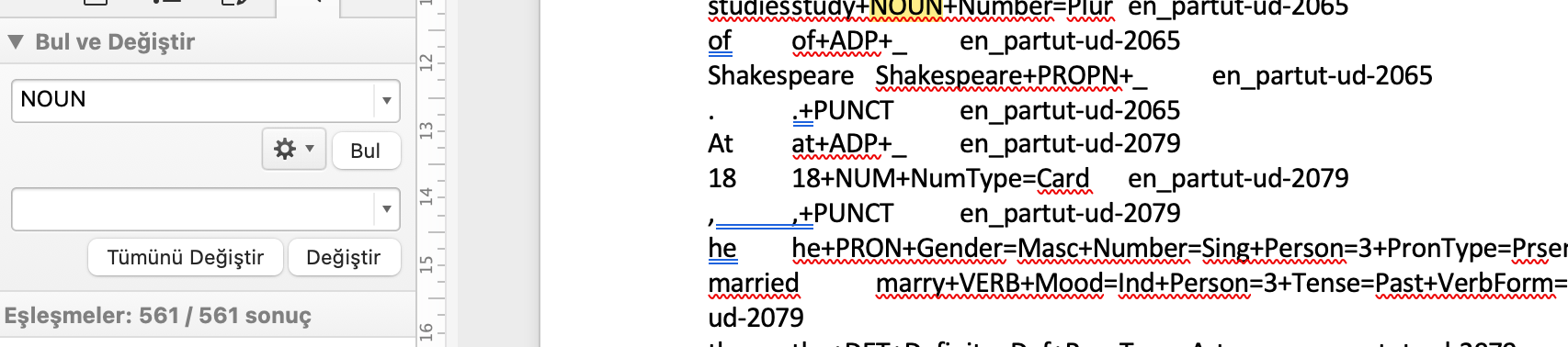
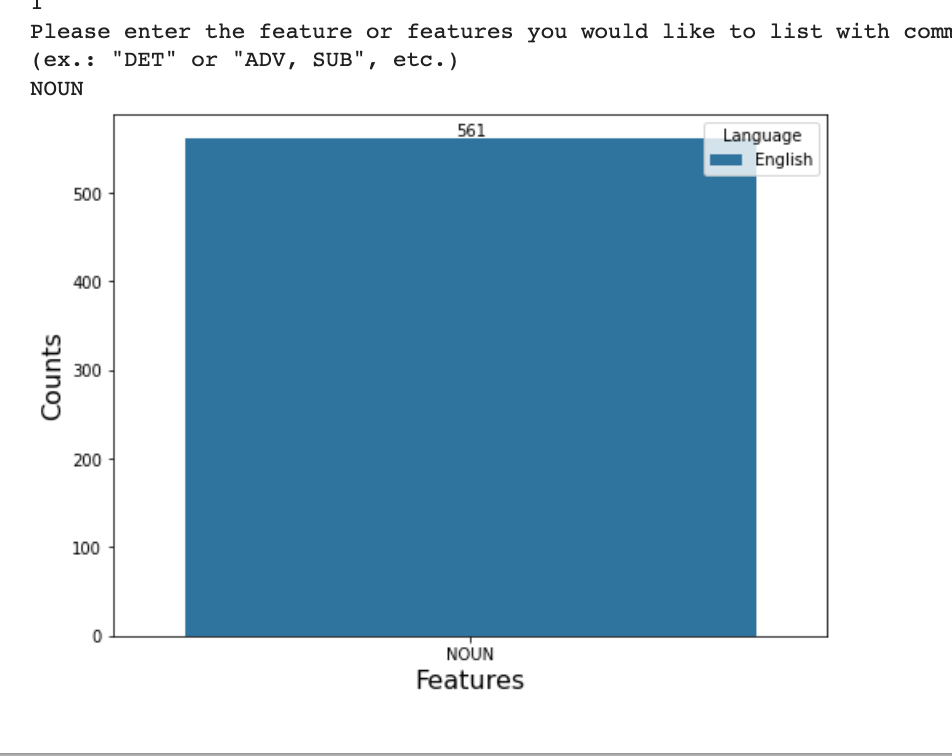
(e.g. precision = |1438 ∩ 1438| / |1438|

= 1

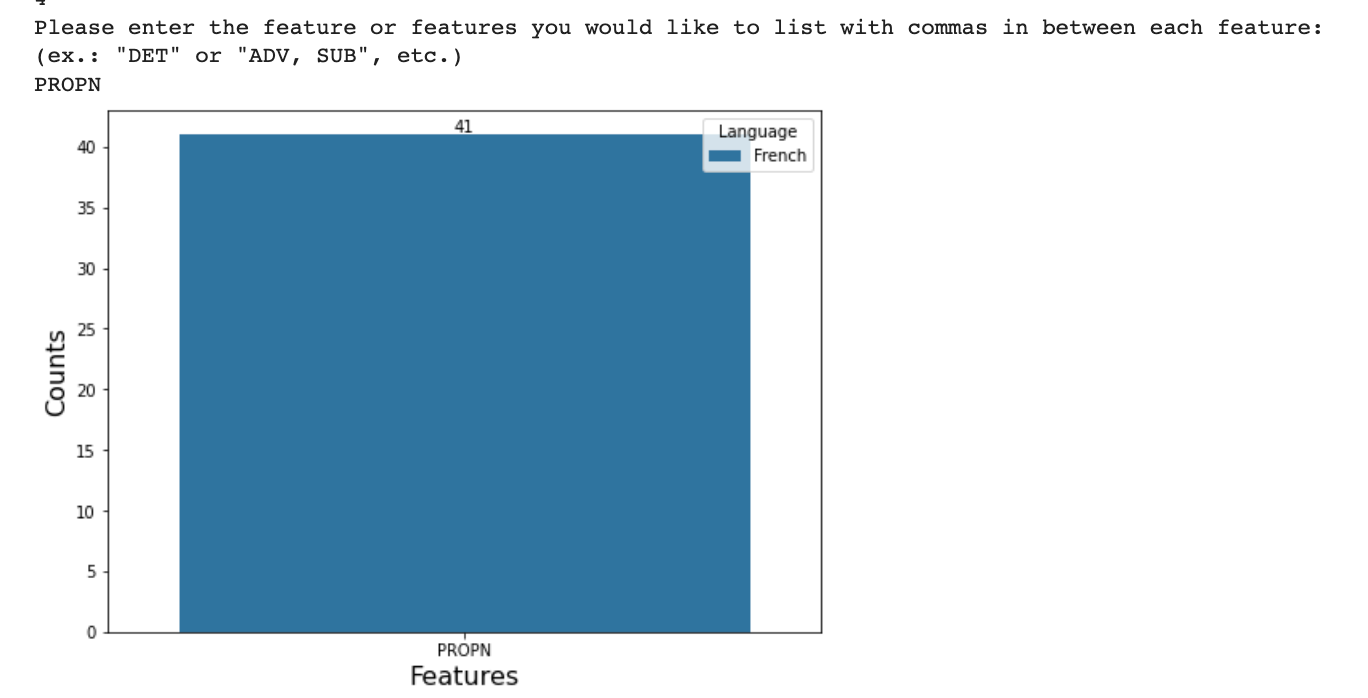
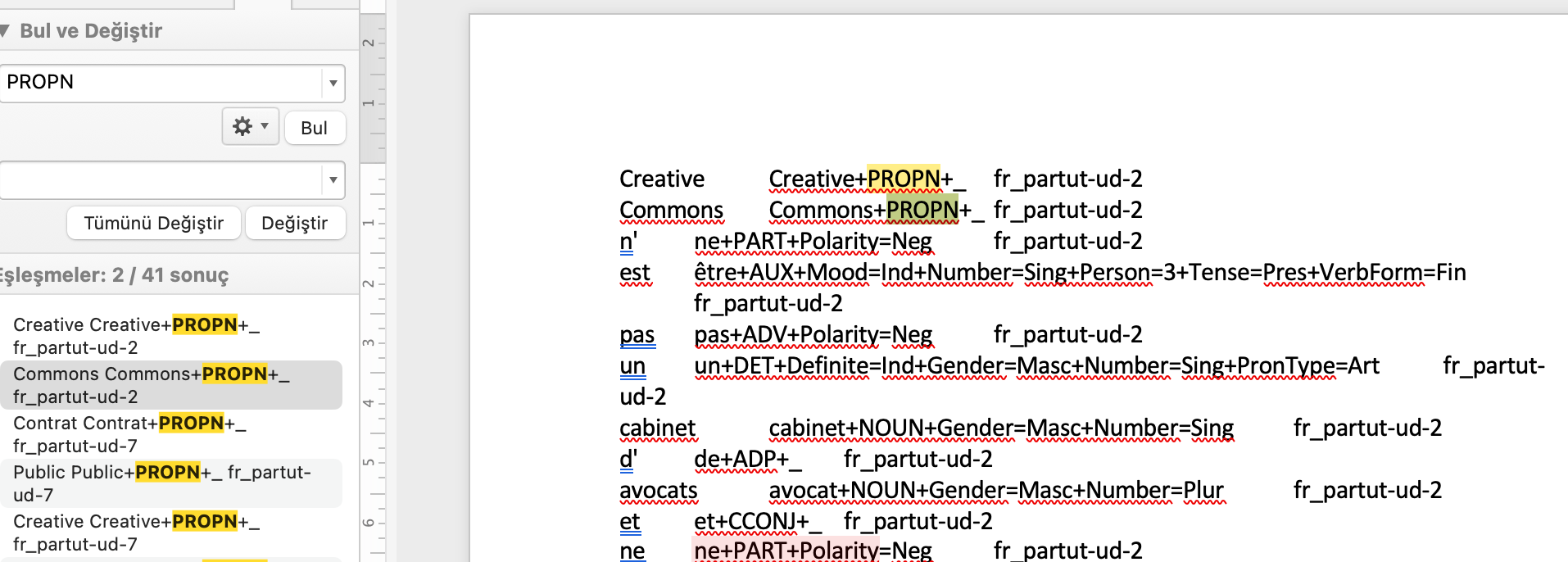
Unfortunately, we were not able to implement the metric into our code, but, below, you can find the screenshots demonstrating that BUParsum gives out the exact same results as the actual document.

(German: DET)



(English: NOUN)

(French: PROPN)