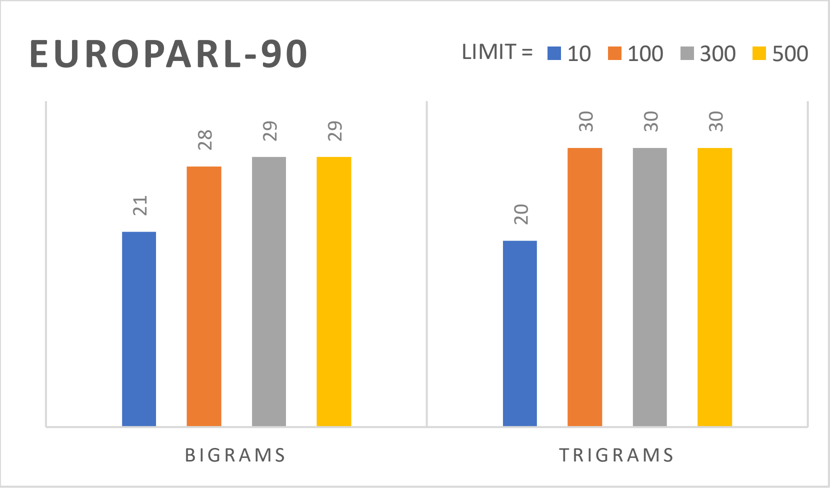
Language recognizer succes:

For the Europarl-90 testset, the bigram model succeeded in all 29 of the 30 languages. The trigram model did it better than the bigram model and had a perfect score: 30 out of 30 languages were classified correctly. For the Europarl-30 testset, the bigram model classified 27 languages correctly. The trigram model again beat the score, and classified 29 of the 30 languages correctly. The Europarl-10 was the hardest testset for the program. The bigram model was right in only 19 of the 30 cases, and the trigram model succeded in 22 of the 30 languages.

What we see is that as the length of the testset declines, the program makes more and more mistakes when classifying the languages. The trigram model does it better than the bigram model in all of the testsets.

Manipulating parameters:

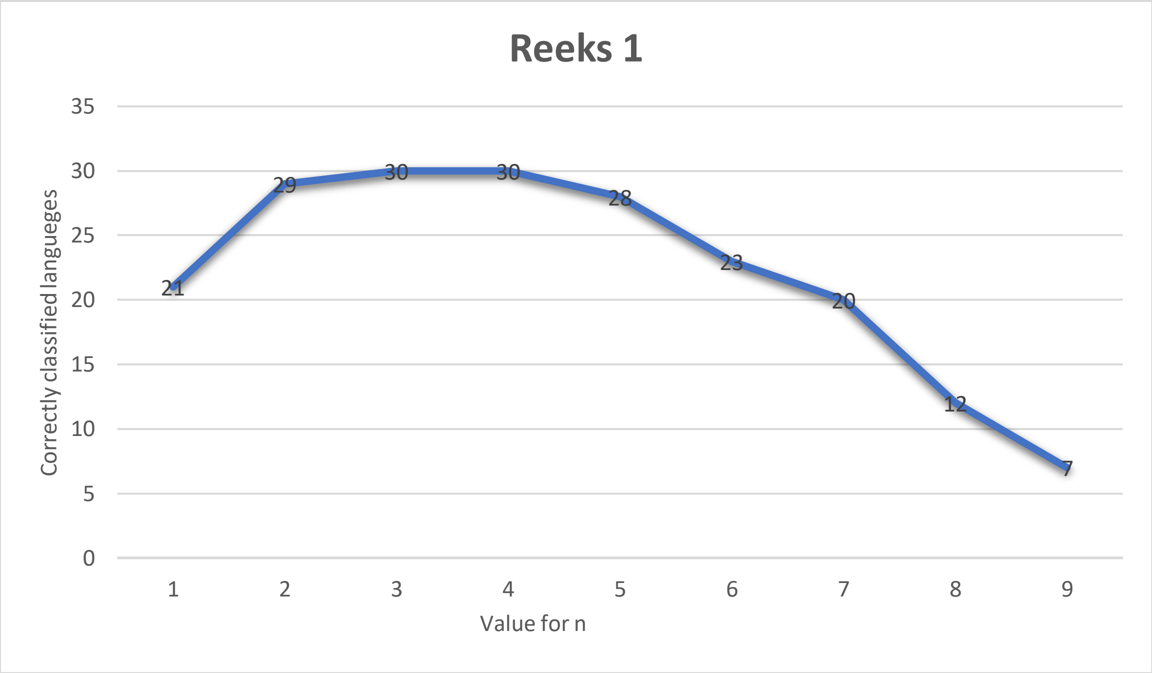
For the testset Europarl-90, we changed the limit of the maximum numbers of bi/trigrams. The number of correctly classified languages is the height of the bars. We also chose to add an really high and really low limit, to check how that would impact the score of the bi/trigrams. We saw that with a limit of 10, the bigrams (for the first time ever!) beat the trigrams. With trying a few different limits, we found that the cross is at a limit of 20. Both trigrams and bigrams have a score of 24 out of 30 languages.



Decreasing and increasing n:

When you decrease n to 1, so using unigrams, we managed to succesfully classify 21 out of the 30 langueges. We used the standard limit of 200. We also increased the n to higher values, and put

them all together in a line graph:



We see that the bi-, tri-, and quadgrams performe really good, and from there it's all downhill. Increeasing n to 5 or higher will only make the langueage detecter performe worse.

Reflecting on the project:

To start with, getting the group together took a week or two. We weren't all in the same seminar, but that wasn't a problem thanks to the group Whatsapp we have with (almost) the whole course. We didn't need to rush things anywhere during the project. We began by making sure everyone fully understood the assignment before we started to divide the tasks. The first part of the program; the 'langdetect module' was finished within a few days. Two of our group members already have quite some experience with python, which helped a lot. The other two group members were definitely able to also contribute a fair share on writing the functions. Some help here and there from our group's 'python experts' made us able to get all the functions working in no time.

After that we split our group in two. One duo was going to make the 'create_profiles' script, and the other group was put on the 'language recognizer'. We divided the groups to make two duo's consisting of a 'python-expert' and a 'python-dummy', to make sure the script and class would work correctly, and we could also learn a lot from our team members. The 'create_profiles' script went pretty smooth. The only obstacle we encountered, was a problem with retrieving the test- and training data. We couldn't get the language files loaded into our program. After some help from Fabio, we eventually managed to fix the problem.

With the 'LangMatcher' there was only one small problem. When reading the values of the frequencies of the bi- and trigrams, the program read it as a string, instead of an integer. It took us not more than 15 minutes to find the problem and to fix it. After that everything went without any significant obstacles.

Summarizing: the overall coöperation was excellent. Everybody was present at the seminars, so we could work on the project everytime. We made a good division of tasks, so everybody was able to contribute a good part to the project. When we read the whole assignment at the start, we thought it was going to take us a lot longer than it eventually did. This gave us an advantage for the last days; we had enough time to finish the last parts of the program and didn't need to rush anything.