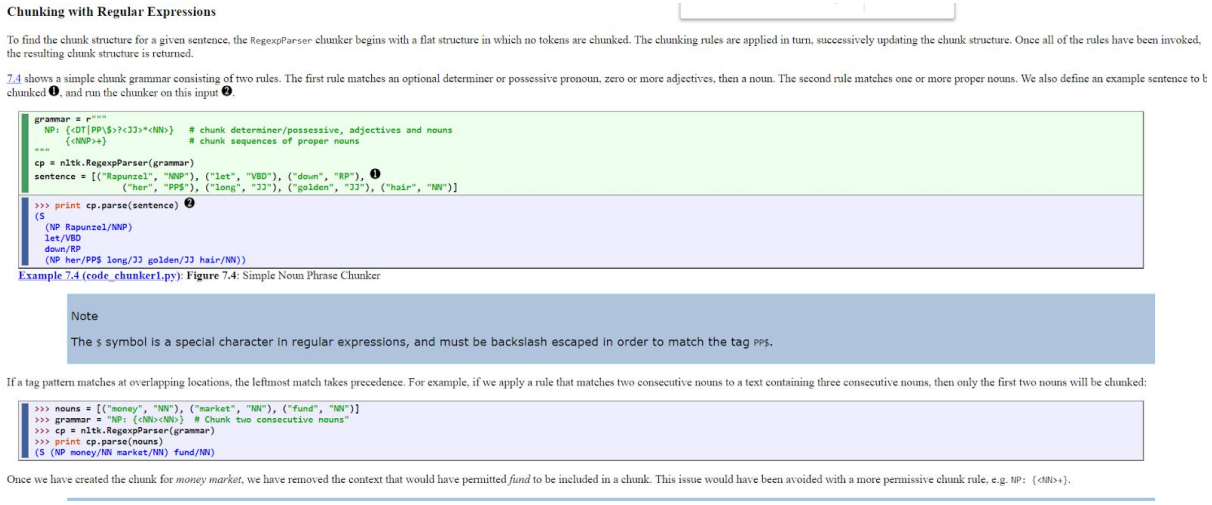
Assignment 4

Work done by:

- Helle van den Broek,
- Truls Berglund
- We did all tasks together

Exercise 1

Used the grammar from the https://www.nltk.org/book/ch07.html



Chunking with Regular Expressions

To find the chunk structure for a given sentence, the RegexpParser chunker begins with a flat structure in which no tokens are chunked. The chunking rules are applied in turn, successively updating the chunk structure. Once all of the rules have been invoked, the resulting chunk structure is returned.

7.4 shows a simple chunk grammar consisting of two rules. The first rule matches an optional determiner or possessive pronoun, zero or more adjectives, then a noun. The second rule matches one or more proper nouns. We also define an example sentence to b chunked ❶, and run the chunker on this input ❷.

```
grammar = r"""
  NP: {<DT|PP\$>?<JJ>*<NN>}   # chunk determiner/possessive, adjectives and nouns
      {<NNP>+}                # chunk sequences of proper nouns
"""
cp = nltk.RegexpParser(grammar)
sentence = [("Rapunzel", "NNP"), ("let", "VBD"), ("down", "RP"), ❶
            ("her", "PP$"), ("long", "JJ"), ("golden", "JJ"), ("hair", "NN")]
```

```
>>> print cp.parse(sentence) ❷
(S
  (NP Rapunzel/NNP)
  let/VBD
  down/RP
  (NP her/PP$ long/JJ golden/JJ hair/NN))
```

Example 7.4 (code_chunker1.py): **Figure 7.4:** Simple Noun Phrase Chunker

Note

The $ symbol is a special character in regular expressions, and must be backslash escaped in order to match the tag PP$.

If a tag pattern matches at overlapping locations, the leftmost match takes precedence. For example, if we apply a rule that matches two consecutive nouns to a text containing three consecutive nouns, then only the first two nouns will be chunked:

```
>>> nouns = [("money", "NN"), ("market", "NN"), ("fund", "NN")]
>>> grammar = "NP: {<NN><NN>}  # Chunk two consecutive nouns"
>>> cp = nltk.RegexpParser(grammar)
>>> print cp.parse(nouns)
(S (NP money/NN market/NN) fund/NN)
```

Once we have created the chunk for *money market*, we have removed the context that would have permitted *fund* to be included in a chunk. This issue would have been avoided with a more permissive chunk rule, e.g. NP: {<NN>+}.

Using the nltk RegexParser on the grammar, and tagged_sents on the brown corpus. Looping through all the tagged sentences we use the parser on every one of them, creating a tree.
For all the subtrees in the tree we check whether or not they are labeled with a clause, adding them to a list of tuples.

Output showing the 20 first words in alphabetical order:

----------EXERCISE1----------
('account', 'for', 'NP')
('achieve', 'in', 'NP')
('acquiesce', 'in', 'NP')
('acquire', 'in', 'NP')
('act', 'in', 'NP')
('act', 'on', 'NP')
('act', 'pro', 'NP')
('act', 'within', 'NP')
('add', 'to', 'NP')

('address', 'with', 'NP')
('adjust', 'to', 'NP')
('admire', 'in', 'NP')
('advance', 'in', 'NP')
('advise', 'on', 'NP')
('afford', 'with', 'NP')
('agree', 'on', 'NP')
('agree', 'with', 'NP')
('aid', 'in', 'NP')
('aid', 'with', 'NP')
('allow', 'for', 'NP')

Exercise 2

Using the features from https://github.com/foxbook/atap/blob/master/snippets/ch07/model.py, we made our twitter corpora compatible with both the ngram counter and the smoothing-implemented KneserNey model. This took some time, but after a while we managed to write input and get an estimated completed sentence back. Because of the relatively small amount of data available from the tweets, our input mostly had to correspond directly with an ngram in order to get a viable answer in return. The output below shows *input*, *completed sentence*, and its corresponding *probability*.

Output:
Tweets: ["“hiv is cured in 2nd patient, doctors report.” @nytimes such great news for so many. tremendous progress being made!', 'now that **[...]**
The calculated completed word for the input: 'hiv is' was:
hiv is cured
Probability: 0.25
~~~~~~
The calculated completed word for the input: 'crooked hillary' was:
crooked hillary clinton
Probability: 0.25
~~~~~~
The calculated completed word for the input: 'make america great' was:
make america great again
Probability: 0.125
~~~~~~

Exercise 3

Reading and saving the first five lines of the document as a string.

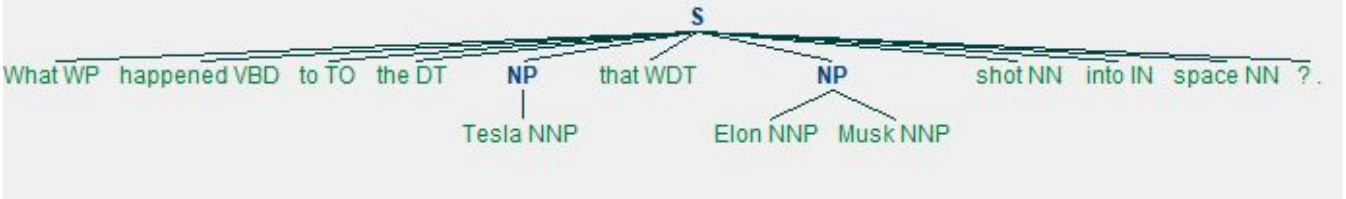
Processing the document by tokenizing sentences and words, and using the pos_tag from nltk on all sentences. Using the RegexParser on the given grammar and parsing the processed document.
For exercise B we basically do the same with finding subtrees as in exercise 1, appending all matches to a list.

Output:
----------THE DOCUMENT----------
What happened to the Tesla that Elon Musk shot into space?
There were plenty of spectacular moments during the maiden launch of SpaceX's Falcon Heavy rocket on Tuesday.
But perhaps the most dramatic scene occurred about four minutes after liftoff: The second stage of the rocket, headed deeper into space, discarded the white nose cone at its tip.
It revealed SpaceX CEO Elon Musk's cherry red sports car. Behind the wheel was a spacesuit-clad mannequin, named Starman. The car glided victoriously away from Earth as David Bowie's "Life on Mars?" blared on SpaceX's launch webcast.
The car is not on some scientific voyage. This was a test launch, so SpaceX needed a dummy payload -- and Musk previously said he wanted it to be the "[s]illiest thing we can imagine." So he picked his own luxurious Tesla roadster.

----------THE PROCESSED DOCUMENT----------
[[('What', 'WP'), ('happened', 'VBD'), ('to', 'TO'), ('the', 'DT'), ('Tesla', 'NNP'), ('that', 'WDT'), ('Elon', 'NNP'), ('Musk', 'NNP'), ('shot', 'NN'), ('into', 'IN'), ('space', 'NN'), ('?', '.')], [('There', 'EX'), ('were', 'VBD'), ('plenty', 'NN'), ('of', 'IN'), ('spectacular', 'JJ'), ('moments', 'NNS'), ('during', 'IN'), ('the', 'DT'), ('maiden', 'JJ'), ('launch', 'NN'), ('of', 'IN'), ('SpaceX', 'NNP'), ("'s", 'POS'), ('Falcon', 'NNP'), ('Heavy', 'NNP'), ('rocket', 'NN'), ('on', 'IN'), ('Tuesday', 'NNP'), ('.', '.')], [('But', 'CC'), ('perhaps', 'RB'), ('the', 'DT'), ('most', 'RBS'), ('dramatic', 'JJ'), ('scene', 'NN'), ('occurred', 'VBD'), ('about', 'IN'), ('four', 'CD'), ('minutes', 'NNS'), ('after', 'IN'), ('liftoff', 'NN'), (':', ':'), ('The', 'DT'), ('second', 'JJ'), ('stage', 'NN'), ('of', 'IN'), ('the', 'DT'), ('rocket', 'NN'), (',', ','), ('headed', 'VBN'), ('deeper', 'NN'), ('into', 'IN'), ('space', 'NN'), (',', ','), ('discarded', 'VBD'), ('the', 'DT'), ('white', 'JJ'), ('nose', 'JJ'), ('cone', 'NN'), ('at', 'IN'), ('its', 'PRP$'), ('tip', 'NN'), ('.', '.')], [('It', 'PRP'), ('revealed', 'VBD'), ('SpaceX', 'NNP'), ('CEO', 'NNP'), ('Elon', 'NNP'), ('Musk', 'NNP'), ("'s", 'POS'), ('cherry', 'NN'), ('red', 'JJ'), ('sports', 'NNS'), ('car', 'NN'), ('.', '.')], [('Behind', 'IN'), ('the', 'DT'), ('wheel', 'NN'), ('was', 'VBD'), ('a', 'DT'), ('spacesuit-clad', 'JJ'), ('mannequin', 'NN'), (',', ','), ('named', 'VBN'), ('Starman', 'NNP'), ('.', '.')], [('The', 'DT'), ('car', 'NN'), ('glided', 'VBD'), ('victoriously', 'RB'), ('away', 'RB'), ('from', 'IN'), ('Earth', 'NNP'), ('as', 'IN'), ('David', 'NNP'), ('Bowie', 'NNP'), ("'s", 'POS'), ('``', '``'), ('Life', 'NNP'), ('on', 'IN'), ('Mars', 'NNP'), ('?', '.'), ("''", "''")], [('blared', 'VBN'), ('on', 'IN'), ('SpaceX', 'NNP'), ("'s", 'POS'), ('launch', 'NN'), ('webcast', 'NN'), ('.', '.')], [('The', 'DT'), ('car', 'NN'), ('is', 'VBZ'), ('not', 'RB'), ('on', 'IN'), ('some', 'DT'), ('scientific', 'JJ'), ('voyage', 'NN'), ('.', '.')], [('This', 'DT'), ('was', 'VBD'), ('a', 'DT'), ('test', 'NN'), ('launch', 'NN'), (',', ','), ('so', 'RB'), ('SpaceX', 'NNP'), ('needed', 'VBD'), ('a', 'DT'), ('dummy', 'JJ'), ('payload', 'NN'), ('--', ':'), ('and', 'CC'), ('Musk', 'NNP'), ('previously', 'RB'), ('said', 'VBD'), ('he', 'PRP'), ('wanted', 'VBD'), ('it', 'PRP'), ('to', 'TO'), ('be', 'VB'), ('the', 'DT'), ('``', '``'), ('[', 'JJ'), ('s', 'NN'), (']', 'NNP'), ('illiest', 'JJ'), ('thing', 'NN'), ('we', 'PRP'), ('can', 'MD'),

('imagine', 'VB'), ('.', '.'), ("""", """")], [('So', 'RB'), ('he', 'PRP'), ('picked', 'VBD'), ('his', 'PRP$'), ('own', 'JJ'), ('luxurious', 'JJ'), ('Tesla', 'NNP'), ('roadster', 'NN'), ('.', '.')]]

----------PARSED----------
(S
What/WP
happened/VBD
to/TO
the/DT
(NP Tesla/NNP)
that/WDT
(NP Elon/NNP Musk/NNP)
shot/NN
into/IN
space/NN
?/.)



b)

----------MATCHES----------
['Tesla', 'Elon', 'spectacular', 'SpaceX', 'Falcon', 'Tuesday', 'minutes']