Text Mining Exercise week 2

Guido Zuidhof, s4160703

My approach uses regular expressions to recognize and replace certain parts of the text. All regular expressions are Python regex. The text is processed as follows:

1. Subsequent newlines are flagged, matched using `'\n\n'`
2. Hyphens at the end of sentences are removed, the two parts of the words get merged. Achieved using `re.sub(r'\n([^A-Z])',r"\1", text)`
3. All newlines that do not have a capital letter as next character get removed: `re.sub(r'\n([^A-Z])',r"\1", unhyphened)`
4. Newlines are inserted where the following pattern is present: `dot, space, capital letter, (NOT (whitespace AND dot) OR dot)`. The operation for this is `re.sub(r'[.][ ]([A-Z])([^?\s.])', r'.\n\1\2', text)`.
5. Newlines are re-inserted where the flags were placed in step one.

This simple approach is quite succesful, I worked from simple rules to more advanced rules, back to simpler rules. It is not perfect, but it gets very close.

Encountered problems

I encountered many problems, the first was the encoding of the file. The windows commandline seemed to be unable to print certain characters, and Python wanted to convert everything to unicode. After a lot of googling I found out I could simply decode and encode the file.

The second was the use of hyphens mid-sentence. The solution was to only match hyphens at the end of sentences and remove these. The third problem had to do with capitals and dots used throughout the text, especially in names of people (with initials). By setting a minimal length of two for a sentence in the regex many of these problem areas seem to have been solved.

Finally, there was the pagenumbering/footnote problem. It turned out to be trivial to solve, as there were already newlines present in the original text, which I could simply restore.