Programmering og sproganalyse

Eksamensopgave 2

Skrevet af: Jonas Reventlow Petersen og David Jonas Nitze

Eksamensopgave 1:

Antal anslag: 11029

Antal sider: 4,6

Eksamensopgave 2:

Antal anslag: 20667

Antal sider: 8,6

Indledning

I denne opgave har vi fået til ansvar at analyserer og fortolke på forskellige anmeldelser af fitnesskæder Fitness World. Anmeldelserne er hentet fra hjemmesiden Trustpilot og splitter i henholdsvis positive og negative kommentarer. Vores løsning er udformet på baggrund af kvantitativ undersøgelser, hvor vi har til opgave at udpege tendenser der forekommer i anmeldelserne - både enkeltvis og i fællesskab. Denne rapport er dermed en detaljeret beskrivelse af vores undersøgerne og metoder, hvor vi vil forklare vores valg af analysemetoder igennem definitioner og bilag.

Den kvantitative undersøgelse er løst gennem en udformning af et kodningsprogram. Programmet er skrevet i kodningssproget Python (version 3.8) og er skrevet i platformen Jupyter Notebook. Programmet er udformet på baggrund af at får anskaffet information som opgaveformuleringerne tilkendegiver - dertil vil vi diskuterer hvordan man kan optimerer programmeret (Opgave 3), og hvilke andre kodninger der kan outputte relevante informationer.

Vores primære viden af besvarelserne af opgaverne er udformet af bogen:

Gries, Paul., Campbell, J. og Montojo, J. 2017. *Practical Programming. An introduction to Computer Science Using Python 3.6.* T. Coron (editor). 3. Edition.

Her er det essentielt også at nævne, at vi har brugt en del af NLTK pakken, som er en kodningspakke i programmeringssproget Python. Vores viden af NLTK er på baggrund af forelæsninger og hjemmesiden https://www.nltk.org/book/

Opgave 1

I opgave1 skal vi kreere en frekvensliste uden brug af nltk.

1a+b) Denne opgave har vi opdelt mere end vi har gjort med de andre, da dens formål og præmisser ændrer sig markant mere end de andre opgaver. Vi åbner først filerne med hhv. positive og negative anmeldelser. Herefter indlæser vi filerne og splitter dem til en liste med ord hvor vi stripper dem, så de ikke har fx. mellemrum/punktum i slutningen af sig. Herefter laver vi dem lowercase, så vi mindsker faldgruberne og får flest opfyldt vores kriterium bedst muligt. Slutteligt sorterer vi ordene efter frekvens fra flest til færrest med en minimumsgrænse på 25, da det er opgavens formål. Listerne udskrives til filen freqout1.txt.

Bilag 1 og Bilag 2 viser selve kodningen og resultatet.

In [1]:
```
import os
import string
from operator import itemgetter
with open('fit_pos.txt', encoding="utf-8") as fin:
    text=fin.read()
    text=text.split()
    words=[w.lower().strip(string.punctuation) for w in text]
    d = {w:words.count(w) for w in words}
    sortfreq1=sorted(d.items(),key=itemgetter(1),reverse=True)


with open('fit_neg.txt', encoding="utf-8") as fin:
    text=fin.read()
    text=text.split()
    words=[w.lower().strip(string.punctuation) for w in text]
    d = {w:words.count(w) for w in words}
    sortfreq2=sorted(d.items(),key=itemgetter(1),reverse=True)

with open('freqout1.txt','w', encoding="utf-8") as fout:
    fout.write('KVANTITATIV UNDERSØGELSE AF TEKSTERNE FIT_POS.TXT OG FIT_NEG.TXT'+'\n\n')
    fout.write('FREKVENSER for: FIT_POS.TXT MED FREKVENS MIN:25'+'\n')
    for element in sortfreq1:
        if element[1]>24:
            fout.write(str(element[1])+' '+element[0]+'\n')
    fout.write('\n\n'+'FREKVENSER for: FIT_NEG.TXT MED FREKVENS MIN: 25'+'\n')
    for element in sortfreq2:
        if element[1]>24:
            fout.write(str(element[1])+' '+element[0]+'\n')
```

Bilag 1

```
FREKVENSER for: FIT_POS.TXT MED FREKVENS MIN:25
228 og
167 er
157 i
134 at
131 jeg
92 det
80 har
75 til
64 der
62 med
61 fitness
57 gode
55 altid
55 af
55 for
50 world
49 godt
49 center
46 de
45 en
44 på
43 god
42 et
41 personale
38 hold
36 dejligt
35 super
34 træne
33 sted
33
29 alt
28 tilfreds
28 instruktører
28 man
27 fw
27 rent
27 meget
25 kan


FREKVENSER for: FIT_NEG.TXT MED FREKVENS MIN: 25
296 at
275 i
270 jeg
237 er
233 og
213 ikke
197 det
179 på
168 har
159 der
138 til
131 for
121 en
108 de
92 så
88 med
83 af
75 man
68 men
60 kan
58 mig
56 et
56
54 som
52 fitness
47 var
45 da
45 world
42 den
41 ved
38 hold
38 efter
37 bliver
37 dårlig
36 noget
36 om
35 kunne
34 fra
32 ud
31 min
29 få
29 når
28 ingen
28 skal
28 hvor
27 corona
27 kundeservice
27 service
26 bare
25 deres
25 under
25 mit
```

Bilag 2

1c) Her skal vi gøre det samme som i forrige opgave, men dog ekskludere visse ord fra listerne. Disse ord findes i filen stopord.txt. Ordene i denne fil bearbejder vi ligesom de andre filer - splitter til en liste for derefter at bruge den til at frasortere ordene i filen.
Bilag 3 og Bilag 4 viser kodningen og resultatet.

In [2]:
```
import os
import string
from operator import itemgetter
with open('fit_pos.txt', encoding="utf-8") as fin:
    text=fin.read().split()
    words=[w.lower().strip(string.punctuation) for w in text]
    d = {w:words.count(w) for w in words}
    sortfreq1=sorted(d.items(),key=itemgetter(1),reverse=True)

with open('fit_neg.txt', encoding="utf-8") as fin:
    text=fin.read().split()
    words=[w.lower().strip(string.punctuation) for w in text]
    d = {w:words.count(w) for w in words}
    sortfreq2=sorted(d.items(),key=itemgetter(1),reverse=True)

with open('stopord.txt', encoding="utf-8") as fin:
    stopord=fin.read().split()

with open('freqout1.txt','w', encoding="utf-8") as fout:
    fout.write('KVANTITATIV UNDERSØGELSE AF TEKSTERNE FIT_POS.TXT OG FIT_NEG.TXT'+'\n\n')
    fout.write('FREKVENSER for: FIT_POS.TXT MED FREKVENS MIN:25'+'\n')
    for element in sortfreq1:
        if element[1]>=25 and element[0] not in stopord:
            fout.write(str(element[1])+' '+element[0]+'\n')
    fout.write('\n\n'+'FREKVENSER for: FIT_NEG.TXT MED FREKVENS MIN: 25'+'\n')
    for element in sortfreq2:
        if element[1]>=25 and element[0] not in stopord:
            fout.write(str(element[1])+' '+element[0]+'\n')
```

Bilag 3

```
KVANTITATIV UNDERSØGELSE AF TEKSTERNE FIT_POS.TXT OG FIT_NEG.TXT

FREKVENSER for: FIT_POS.TXT MED FREKVENS MIN:25
131 jeg
61 fitness
57 gode
50 world
49 center
41 personale
38 hold
36 dejligt
35 super
34 træne
33 sted
33
28 tilfreds
28 instruktører
27 fw
27 rent


FREKVENSER for: FIT_NEG.TXT MED FREKVENS MIN: 25
270 jeg
213 ikke
58 mig
56
52 fitness
45 world
38 hold
37 dårlig
31 min
28 ingen
27 corona
27 kundeservice
27 service
```

Bilag 4

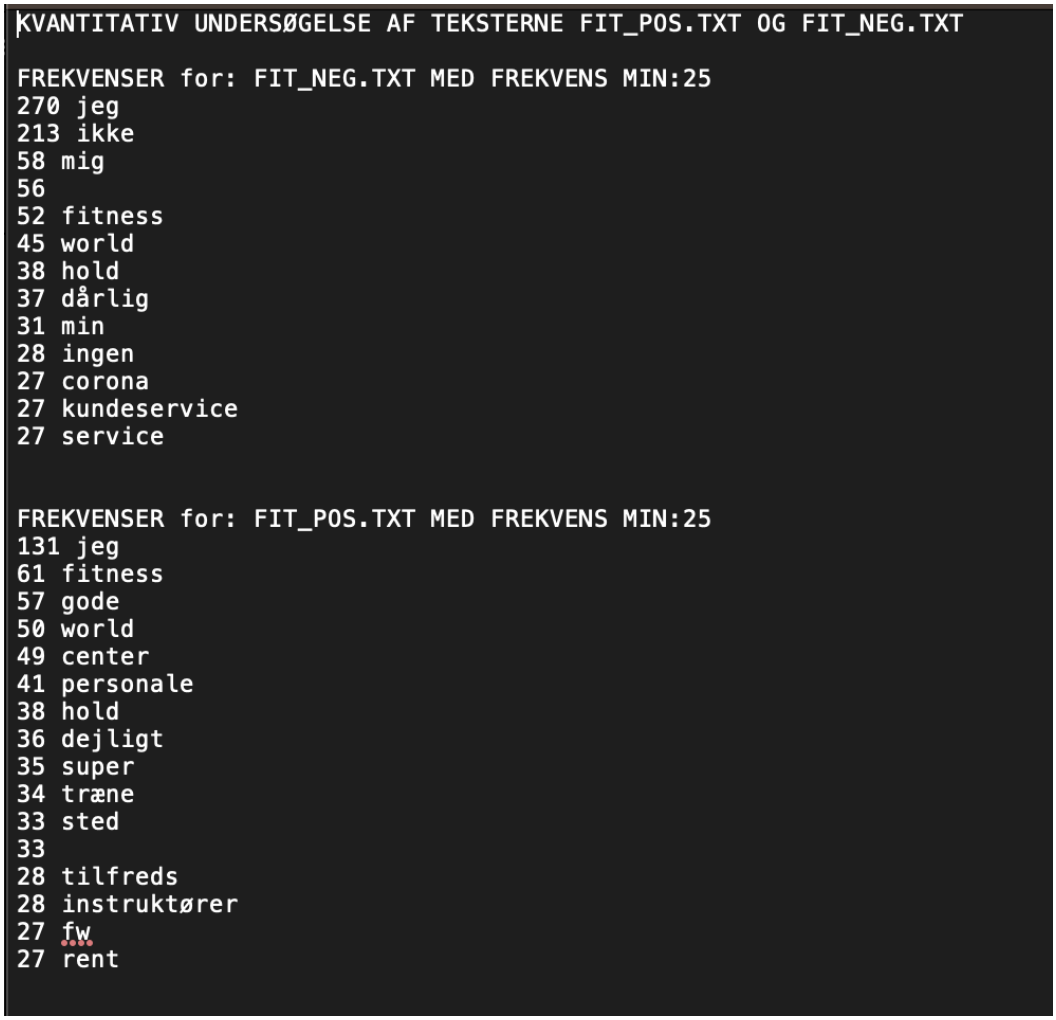
1d) Vi bygger yderligere ovenpå de forudgående opgaver, og nu skal vi have en bruger til at indtaste de ønskede filer (fit_pos.txt og fit_neg.txt) samt frekvenskriteriet for ordene i dem. Filerne skal ligge i samme sti som programmet, og det vil derfor kun være muligt at indtaste fit_pos.txt og fit_neg.txt. Lå der andre .txt-filer i stien ville de også kunne bruges.
Opgavens resultat ender ud i en fil, freqout1.txt, der indeholder en frekvensliste for hvert af brugerens valgte filer samt frekvenskriterier. Begge lister har et ord, der består af 0 tegn: hvad det præcis er, ved vi ikke. Vi har åbnet anmeldelserne og overveje muligheden, at det kunne være emojis eller dobbelt mellemrum el.lign., men har ikke identificeret det. Ønsker man det væk, kunne en mulighed være at sætte præmissen for længden af ord i listerne (len) til et minimum på 1 tegn. Selve vi ikke ved, hvad det er, har vi dog valgt at beholde det idet det må være meningsfuldt, da det findes i begge lister.
Bilag 5 og Bilag 6 viser kodningen og resultatet.

```python
def opgave1():
    fil1 = input("Indtast textfilnavn1: ")
    fil2 = input("Indtast textfilnavn2: ")
    kriterie = int(input("Hvilken frekvens skal ordene minimum have i frekvenslisten: "))
    with open(fil1, encoding="utf-8") as fin:
        text=fin.read().split()
        words=[w.lower().strip(string.punctuation) for w in text]
        d = {w:words.count(w) for w in words}
        sortfreq1=sorted(d.items(),key=itemgetter(1),reverse=True)

    with open(fil2, encoding="utf-8") as fin:
        text=fin.read().split()
        words=[w.lower().strip(string.punctuation) for w in text]
        d = {w:words.count(w) for w in words}
        sortfreq2=sorted(d.items(),key=itemgetter(1),reverse=True)

    with open('stopord.txt', encoding="utf-8") as fin:
        stopord=fin.read().split()

    with open('freqout1.txt','w', encoding="utf-8") as fout:
        fout.write('KVANTITATIV UNDERSØGELSE AF TEKSTERNE FIT_POS.TXT OG FIT_NEG.TXT'+'\n\n')
        fout.write('FREKVENSER for: ' + str(fil1).upper() + ' MED FREKVENS MIN:'+ str(kriterie) +'\n')
        for element in sortfreq1:
            if element[1]>=kriterie and element[0] not in stopord:
                fout.write(str(element[1])+' '+element[0]+'\n')
        fout.write('\n\n'+'FREKVENSER for: ' + str(fil2).upper() + ' MED FREKVENS MIN:'+ str(kriterie) +'\n')
        for element in sortfreq2:
            if element[1]>=kriterie and element[0] not in stopord:
                fout.write(str(element[1])+' '+element[0]+'\n')
```

In [4]:
```python
opgave1()
```

```
Indtast textfilnavn1: fit_neg.txt
Indtast textfilnavn2: fit_pos.txt
Hvilken frekvens skal ordene minimum have i frekvenslisten: 25
```

Bilag 5



```
KVANTITATIV UNDERSØGELSE AF TEKSTERNE FIT_POS.TXT OG FIT_NEG.TXT

FREKVENSER for: FIT_NEG.TXT MED FREKVENS MIN:25
270 jeg
213 ikke
58 mig
56
52 fitness
45 world
38 hold
37 dårlig
31 min
28 ingen
27 corona
27 kundeservice
27 service


FREKVENSER for: FIT_POS.TXT MED FREKVENS MIN:25
131 jeg
61 fitness
57 gode
50 world
49 center
41 personale
38 hold
36 dejligt
35 super
34 træne
33 sted
33
28 tilfreds
28 instruktører
27 fw
27 rent
```

Bilag 6

2a) Her skal vi udføre samme opgave som i opgave 1a dog med brug af nltk. Til det er det oplagt at bruge nltk funktion, der hedder nltk.FreqDist (https://www.nltk.org/book/ch01.html kapitel 3.1), som betyder frequency distribution dvs. frekvensdistribution. Vi får igen sorteret frekvensbaseret udefra brugerinputtet, der bestemmer hvilke filer, der skal læses og hvad frekvensminimum skal være. freqdist1.most_common(kriterie) sætter begrænsningsansvaret hos brugeren og her er pythons brug af nltk klog nok til sortere frekvensmæssigt flest-->færrest.
Resultatet af denne opgave indsættes ikke i vores freqout1.txt dokument men forbliver blot i python.
Bilag 7 og Bilag 8 viser kodningen og outputtet.

In [5]:
```
import nltk
def opgave2():
    fil1 = input("textfil-1: ")
    fil2 = input("textfil-2: ")
    kriterie = int(input("antal: "))
    with open(fil1, encoding="utf-8") as fin:
        text = fin.read().split()
        words=[w.lower().strip(string.punctuation) for w in text]
        freqdist1 = nltk.FreqDist(words)
        sortfreq1 = freqdist1.most_common(kriterie)
    with open(fil2, encoding="utf-8") as fin:
        text = fin.read().split()
        words=[w.lower().strip(string.punctuation) for w in text]
        freqdist2 = nltk.FreqDist(words)
        sortfreq2 = freqdist2.most_common(kriterie)
    return sortfreq1, sortfreq2
```

In [6]:
```
sort1, sort2 = opgave2()
```

```
textfil-1: fit_neg.txt
textfil-2: fit_pos.txt
antal: 25
```

Bilag 7

```
In [8]: opgave2()
```

```
click to expand output; double click to hide output
antal: 25
```

```
Out[8]: ([('at', 296),
  ('i', 275),
  ('jeg', 270),
  ('er', 237),
  ('og', 233),
  ('ikke', 213),
  ('det', 197),
  ('på', 179),
  ('har', 168),
  ('der', 159),
  ('til', 138),
  ('for', 131),
  ('en', 121),
  ('de', 108),
  ('så', 92),
  ('med', 88),
  ('af', 83),
  ('man', 75),
  ('men', 68),
  ('kan', 60),
  ('mig', 58),
  ('et', 56),
  ('', 56),
  ('som', 54),
  ('fitness', 52)],
 [('og', 228),
  ('er', 167),
  ('i', 157),
  ('at', 134),
```

```
('jeg', 131),
('det', 92),
('har', 80),
('til', 75),
('der', 64),
('med', 62),
('fitness', 61),
('gode', 57),
('altid', 55),
('af', 55),
('for', 55),
('world', 50),
('godt', 49),
('center', 49),
('de', 46),
('en', 45),
('på', 44),
('god', 43),
('et', 42),
('personale', 41),
('hold', 38)])
```

Bilag 8

3) Denne opgave handler om bigrammer, dvs. to ord, der står i forlængelse af hinanden. Opgaven tager fortsat udgangspunkt i anmeldelserne om Fitness World.

3a+b+c+d) Her skal vi lave en frekvensliste med de hyppigste bigrammer først. Igen er det dikteret af brugerens input og kriterier om frekvenshyppighed samt stopordsfilens begrænsninger. Opgaven følger dermed tråd til de tidligere opgaver og bygger videre på analysen af anmeldelserne, som får mere og mere kontekst.

Vi bruger nltks bigrams pakke til at distingvere ordene i ordpar, hvor vi tidligere blot gjorde det med ordene separat. Samme måde som før bruger vi i denne opgave også nltks frekvensdistributionshyppighed (nltk.FreqDist(bigrams) og fdist.most_common()), hvor vi går fra flest hyppige til færrest hyppige. *If* statementet i slutningen af koden er et vigtigt parameter i præmisserne til den her opgave: vi har element[1] som er hyppigheden og element[0][0] samt element[0][1] som er bigrammerne. Ordene har to indextal fordi de er to unikke ord, som hver især

IKKE må være en del af stopordsfilen, da de så ikke vil indgå i bigramslisten. Denne opgave bliver igen appended til den eksisterende freqout1.txt-fil og bliver indskrevet med brugerens input som en del af stringen, så det er brugerens input, der står der. Dvs. om brugerens valg af den positive eller negative anmeldelse samtidig med frekvensbegrænsningen.
Bilag 9 og Bilag 10 viser kodningen og resultatet

```
import nltk
def opgave3():
    fil1 = input("Indtast textfilnavn1: ")
    fil2 = input("Indtast textfilnavn2: ")
    kriterie = int(input("Hvilken frekvens skal ordene minimum have i frekvenslisten: "))

    with open(fil1, encoding="utf-8") as fin:
        text=fin.read().split()
        words=[w.lower().strip(string.punctuation) for w in text]
        bigrams=nltk.bigrams(words)
        fdist1=nltk.FreqDist(bigrams)
        sortfreq1=fdist1.most_common()

    with open(fil2, encoding="utf-8") as fin:
        text=fin.read().split()
        words=[w.lower().strip(string.punctuation) for w in text]
        bigrams=nltk.bigrams(words)
        fdist2=nltk.FreqDist(bigrams)
        sortfreq2=fdist2.most_common()

    with open('stopord.txt', encoding="utf-8") as fin:
            stopord=fin.read().split()

    with open('freqout1.txt','a', encoding="utf-8") as fout:
        fout.write('\n\n'+'NEDENSTÅENDE ER BIGRAM FREKVENSER FOR TEKSTFILERNE'+'\n\n')
        fout.write('BIGRAM FREKVENSER FOR: ' + str(fil1).upper() + ' MED FREKVENS MIN:'+ str(kriterie) +'\n')
        for element in sortfreq1:
                if element[1]>=kriterie and element[0][0] not in stopord and element[0][1] not in stopord:
                    fout.write(str(element[1])+' '+element[0][0]+ ', ' + element[0][1] +'\n')
        fout.write('\n\n'+'BIGRAM FREKVENSER FOR: ' + str(fil2).upper() + ' MED FREKVENS MIN:'+ str(kriterie) +'\n')
        for element in sortfreq2:
                if element[1]>=kriterie and element[0][0] not in stopord and element[0][1] not in stopord:
                    fout.write(str(element[1])+' '+element[0][0]+ ', ' + element[0][1] +'\n')
```

Bilag 9

```
In [7]: opgave3()
        Indtast textfilnavn1: fit_neg.txt
        Indtast textfilnavn2: fit_pos.txt
        Hvilken frekvens skal ordene minimum have i frekvenslisten: 8
```

```
NEDENSTÅENDE ER BIGRAM FREKVENSER FOR TEKSTFILERNE

BIGRAM FREKVENSER FOR: FIT_NEG.TXT MED FREKVENS MIN:8
44 fitness, world
27 jeg, ikke
12 dårlig, rengøring
11 meldt, mig
8 dårlig, service


BIGRAM FREKVENSER FOR: FIT_POS.TXT MED FREKVENS MIN:8
49 fitness, world
14 dejligt, sted
11 dejligt, center
11 jeg, elsker
10 gode, instruktører
```

Bilag 10

4a+b+c) I denne opgave skal vi arbejde med fællesmængden af ord med samme udgangspunkt fra .txtfilerne som de tidligere opgaver. Vi bruger samme metode til at åbne filerne fra brugerinputtet, brugerens kriterier samt stripper .txtfilerne og gør dem lowercase, så vi mindsker faldgruber og indkapsler flest mulige resultater af samme ord. Noget der ikke står i opgavebeskrivelsen er brugen af stopord.txt. Den har vi valgt fortsat at gøre brug af, da der gennem opgaverne synes at være en rød tråd i at bruge den fortsat. Det skal pointeres, at det ikke er en del af selve opgaven men noget vi har valgt at tilføje.

Vi laver her to lister, der udmønter sig på hver sin fil: dvs. en liste, der bliver tilføjet ord til, hvis brugerkriterierne er opfyldt. Kriteriet er, at brugeren bestemmer en minimumsfrekvens af et ord samtidig med at ordet SKAL være i den anden fil samtidig med at den IKKE må være i stopord.txt. Kriteriet skal forstås sådan, at der skal være en minimumsfrekvens i den ene fil samtidig med at samme ord blot eksisterer i den anden fil. Dette er vigtigt for forståelsen af vores udførelse af

opgaven. Omvendt kunne man også have sat samme frekvenshyppighed for samme ord i begge lister, hvis det var ønsket.
Listerne lægger vi sammen og de appendes til freqout1.txtfilen.

Bilag 11 og 12 viser kodningen og resultatet

```
In [11]: import nltk
         import string
         def opgave4():
             fil1 = input("Indtast textfilnavn1: ")
             fil2 = input("Indtast textfilnavn2: ")
             kriterie = int(input("antal: "))

             with open(fil1, encoding="utf-8") as fin:
                 text = fin.read().split()
                 words1=[w.lower().strip(string.punctuation) for w in text]
                 freqdist1 = nltk.FreqDist(words1)
                 sortfreq1 = freqdist1.most_common()

             with open(fil2, encoding="utf-8") as fin:
                 text = fin.read().split()
                 words2=[w.lower().strip(string.punctuation) for w in text]
                 freqdist2 = nltk.FreqDist(words2)
                 sortfreq2 = freqdist2.most_common()

             with open('stopord.txt', encoding="utf-8") as fin:
                 stopord=fin.read().split()

             list1 = []
             list2 = []

             for element in sortfreq1:
                 if element[1]>=kriterie and element[0] in words2 and element[0] not in stopord:
                     list1.append(element[0])

             for element in sortfreq2:
                 if element[1]>=kriterie and element[0] in words1 and element[0] not in stopord:
                     list2.append(element[0])

             same_words = list(set(list1) & set(list2)) #Laver en liste, hvor orderne skal være i både "list1" OG "list2"
```

```
with open('freqout1.txt','a', encoding="utf-8") as fout:
    fout.write('\n\n' + 'ORD SOM OPTRÆDER I BEGGE FILER (FÆLLESMÆNGDEN) OG HVOR ORDETS FREKVENS I BEGGE FILER ER
    for word in same_words:
        fout.write(word +' ')
```

Bilag 11

```
In [9]: opgave4()
Indtast textfilnavn1: fit_neg.txt
Indtast textfilnavn2: fit_pos.txt
antal: 5
```

```
ORD SOM OPTRÆDER I BEGGE FILER (FÆLLESMÆNGDEN) OG HVOR ORDETS FREKVENS I BEGGE FILER ER MIN: 5
 rent fitnessworld holdtræning plads world covid-19 center min hold træner hele medarbejder mig fået fw jeg ikke jeres
medlemskab tid fitness personalet kundeservice tak træne maskiner siden centre service jer derfor medlem personale 5
rengøring centret abonnement
```

Bilag 12

5) I denne opgave skal vi finde gennemsnitssætningslængden for de to filer, fit_pos.txt og fit_neg.txt. Som i de tidligere opgaver kræver det et brugerinput. Dette står ikke i opgavebeskrivelsen til opgave 1)5), men står i introduktionen af hele opgave 1) - derfor vælger vi at have dette inkorporeret i vores løsning. Vi importerer først regularexpressions (https://docs.python.org/3/library/re.html) som vi gør brug af i opsplittelsen af tekstens helhed, når vi vil inddele i sætninger. Sætninger har i denne opgave valgt at definere ved .? eller !. Dvs. vi får faldgruber ved fx: 1) newline sætninger, 2) sætninger der ender med smileys (både emojis og kolon-smileys), 3) sætninger der ikke ender med nogen tegnsætning, eller 4) sætninger der ender med flere end én tegnsætning. Vi laver herefter et forloop, der tæller hver ordfrekvens inden for den sætningsdefinition, vi har opsat. Når det er gjort, bruger vi simpel matematik til at udregne gennemsnittet med antal ord/antal sætninger.
En kommentar til resultatet er, at de positive anmeldelse er mærkbart kortere end de negative, der i gennemsnit er tre ord længere.

Bilag 13 og Bilag 14 viser kodningen og resultatet

In [13]:
```
import re
def opgave5():
    fil1 = input("textfil-1: ")
    fil2 = input("textfil-2: ")
    with open(fil1, encoding="utf-8") as fin:
            text = fin.read() #indlæser fil
            sentences = re.split('[.?!]', text) #split tekst ind i en liste af sætninger. Splitter for hvert . ? og
            count = 0 #Tæller antal sætninger
            count_sentence = 0 #Tæller antal ord i hver sætning
            for sentence in sentences: #Lopper over hver sætning i listen af sætninger
                count_sentence = count_sentence + len(sentence.strip().split()) # Splitter hver sætning på mellemrum
                count = count + 1 #Tæller for hvert loop dvs tæller hvor mange sætninger, der er i listen af sætning
            gennemsnit1 = count_sentence/count #Udregner gennemsnit

    with open(fil2, encoding="utf-8") as fin:
            text = fin.read()
            sentences = re.split('[.?!]', text)
            count = 0
            count_sentence = 0
            for sentence in sentences:
                count_sentence = count_sentence + len(sentence.strip().split())
                count = count + 1
            gennemsnit2 = count_sentence/count
    with open('freqout1.txt','a', encoding="utf-8") as fout:
        fout.write('\n\n' + 'GENNEMSNITLIG SÆTNINGSLÆNGDE I ' + fil1 + ': '+ str(gennemsnit1) +'\n')
        fout.write('\n\n' + 'GENNEMSNITLIG SÆTNINGSLÆNGDE I ' + fil2 + ': '+ str(gennemsnit2) +'\n')
```

In [14]:
```
opgave5()
```

```
textfil-1: fit_neg.txt
textfil-2: fit_pos.txt
```

Bilag 13

```
GENNEMSNITLIG SÆTNINGSLÆNGDE I fit_neg.txt: 12.081471747700395

GENNEMSNITLIG SÆTNINGSLÆNGDE I fit_pos.txt: 9.176684881602915
```

Bilag 14

6) Denne del af opgaven er den sidste del i vores frekvensanalyser af de negative og positive anmeldelser. Her skal vi identificere ord i filerne som ikke fremkommer i den modsatte fil. Igen, skal denne programmering tilføres til freqout1 filen, og brugeren skal selv have mulighed for at bestemme hvad frekvensen af ordene er i begge filer.
Processen med indtastningen og åbningen af filerne er den samme, som vi har udarbejdet igennem hele opgave 1. Hvad der er nyt, er vores programmering angående skabelsen af to lister med ordene i hvert fil. For at skabe dette starter vi med at lave to tomme lister, hvor ordene som i sidste ende skal være outputtet i vores freqout. Vi skaber et for loop med et if statement for begge lister, som skal være kriterierne for hvilke ord der kommer i listerne. Her er det vigtigt for vores program, at indeks [1] for vores elementer er det samme eller højere end brugerens kriterie. Indeks [0] for elementet må ikke være et ord i den anden tekst, og ordet må heller ikke være i stopord filen. Hvis et ord lever op til disse kriterier vil den bliver tilført til listen igennem metoden .append.
Da det ikke er selve listerne vi vil have som output men selve de ord der er tilføjet til dem, laver vi et for loop i vores 'with open' metode'. Det er vigtigt at påpege, at vi igen bruger 'a' for append til vores fil, så vi dermed tilføjer noget til filen uden at slette dens indhold.
Bilag 15 og Bilag 16 viser kodningen og resultatet

In [15]:
```
import nltk
import string
def opgave6():
    fil1 = input("Indtast textfilnavn1: ")
    fil2 = input("Indtast textfilnavn2: ")
    kriterie = int(input("antal: "))

    with open(fil1, encoding="utf-8") as fin:
        text = fin.read().split()
        words1=[w.lower().strip(string.punctuation) for w in text]
        freqdist1 = nltk.FreqDist(words1)
        sortfreq1 = freqdist1.most_common()

    with open(fil2, encoding="utf-8") as fin:
        text = fin.read().split()
        words2=[w.lower().strip(string.punctuation) for w in text]
        freqdist2 = nltk.FreqDist(words2)
        sortfreq2 = freqdist2.most_common()

    with open('stopord.txt', encoding="utf-8") as fin:
        stopord=fin.read().split()

    list1 = []
    list2 = []

    for element in sortfreq1:
        if element[1]>=kriterie and element[0] not in words2 and element[0] not in stopord:
            list1.append(element[0])

    for element in sortfreq2:
        if element[1]>=kriterie and element[0] not in words1 and element[0] not in stopord:
            list2.append(element[0])
```

```
    with open('freqout1.txt','a', encoding="utf-8") as fout:
        fout.write('\n\n' + 'ORD SOM OPTRÆDER I '+ str(fil1).upper() +' HVOR ORDETS FREKVENS ER MIN: '+ str(kriterie
        for word in list1:
            fout.write(word +' ')
        fout.write('\n\n' + 'ORD SOM OPTRÆDER I '+ str(fil2).upper() +' HVOR ORDETS FREKVENS ER MIN: '+ str(kriterie
        for word in list2:
            fout.write(word +' ')
```

(fortsættes)

```
+' HVOR ORDETS FREKVENS ER MIN: '+ str(kriterie)+ ' - MENS ORDET IKKE ER I '+ str(fil2).upper() +'\n')

+' HVOR ORDETS FREKVENS ER MIN: '+ str(kriterie)+ ' - MENS ORDET IKKE ER I '+ str(fil1).upper() +'\n')
```

In [13]:
```
opgave6()
```

```
Indtast textfilnavn1: fit_neg.txt
Indtast textfilnavn2: fit_neg.txt
antal: 5
```

Bilag 15

```
ORD SOM OPTRÆDER I FIT_NEG.TXT HVOR ORDETS FREKVENS ER MIN: 5 - MENS ORDET IKKE ER I FIT_POS.TXT
dårlig kr venteliste dårligt meldt 1 betalt bruge beskidt maskine personer regning stå vide stykke åbenbart 2020 intet
time melde hjemmeside okt 99kr stykker toiletterne gebyr snydt sgu bøde resten året koster meldte kampagne svigter
beskidte elendig sko virker tilmeldt vælger lukke konto kontakt brugt 20 99 kr./md faktisk sker bemanding fikset
skriver små lange personligt

ORD SOM OPTRÆDER I FIT_POS.TXT HVOR ORDETS FREKVENS ER MIN: 5 - MENS ORDET IKKE ER I FIT_NEG.TXT
elsker bedste ros smilende fantastisk masser søde rart stemning udvalg dygtige professionel glæder faciliteter herfra
sødt atmosfære flinke og… kompetent forskellige træningscenter fremragende træningsfaciliteter behov hjælpsomme
fantastiske vende
```

Bilag 16

Opgave 2

Resultatet af opgave 1 har givet os mulighed for at udarbejde en masse data. Vi har især et fokus på brugen af de enkelte ord - både selvstændigt i anmeldelserne og på tværs af de negative- og positive anmeldelser. I denne opgave vil vi belyse hvad vi mener denne data viser os. Vil gå diskuterer opgaverne slavisk hvor der udarbejdes en sammenfatning til sidst.

Det første vi har udarbejdet, er frekvenser af ord i dels de negative anmeldelser og de positive anmeldelser. Vi valgte at sætte minimumsgrænsen på hyppigheden af ord på 25, da vi antager at der er et approberet hyppighed for at et ord bliver brugt tit. Det er tydeligt at både de negative- og positive anmelderne bærer præg af brugen af 'jeg' ordet. Det forekommer 270 gange i de negative anmeldelser og 131 gange i de positive anmeldelser, som gør det til det mest brugte ord begge steder. Det giver en god indikation på at alle anmelderne er baseret på de personlige oplevelser. Der er sket noget som har en eller anden form af påvirkning på anmelderen. Man kan anskue, at anmeldelserne bliver udformet på baggrund af det subjektive element, hvor enten den negative- eller positive diskurs er et resultat af ens personlige oplevelse. Man kan også argumentere for, at de 58 forekomster af ordet 'mig' i de negative anmeldelser er et belæg for denne antagelse.

Flere af ordene i de negative anmeldelser giver en indikation på hvilke kategorier der er i fokus. Vi anser at især service er et fokuspunkt, hvor både ordene service, kundeservice og hold indgår med frekvenserne 27, 27 og 38. Det giver et godt billede af hvad anmelderne har haft i tankerne når du har haft deres personlige negative oplevelser.
Derudover forekommer der det negative ladet ord 'Dårlig' 37 gange, som medvirker til at erkende af disse anmeldelser, er de negative. Man kan argumentere for, at ordet 'dårlig' kun er negativ ladet i dens sammenhæng den er skrevet i, men i denne frekvensanalyse antager vi at det er negative bemærkninger når ordet 'dårlig' er inkorporeret.

Ordet 'corona' skal også forklares i denne sammenhæng. Der forekommer 27 gange i de negative anmelderne, som giver anledning til at Fitness World har fået negativ respons på deres håndtering af coronarestriktionerne. Ordet forekommer ikke over 25 gange i den positive anmelderne, så enten

tyder på at det ikke er et fokuspunkt på den positive side, eller at der ikke er noget positivt omkring kategorien corona, og ville dermed være et belæg for dens fremkomst i de negative anmeldelser.

I de positive anmelderne forekommer der flere positive ladet adjektiver såsom ’dejlig’ og ’gode’ som forekommer 36 og 57 gange. Man kan, igen, argumenter for om de er positivt ladet ord, da man reelt set kunne putte ordet ‘ikke’ ind foran, og sætningen ville dermed blive negativ. Der er dog ikke en hyppighed af ‘ikke’ ordet over 25 gange, og må derfor antages at det ikke er brugt hyppigt.
Hvad der beskrives som positivt kan også anskues i denne frekvensanalyse. Både ‘center’, ‘personale’, ‘hold’, ‘sted’ og ‘instruktør’ bliver brugt over 25 gange. Det giver et meget godt billede af hvad anmelderne ser som positive elementer i fitness world. Det tyder på at særligt medarbejder får en vis form for positiv respons da ‘personale’ og instruktør’ tilsammen forekommer 69 gange.

For at få et overblik over ordene i deres sammenhæng, har vi fået dataresultater af bigrammer for både de negative og positive anmelderne. Disse bigrammer viser hvilke to ord der forekommer sammen mest hyppigt. Vores minimumskriterie på frekvensen har vi sat som otte.
Disse bigrammer kan hjælpe os med at forstå hvilken sammenhæng der er mellem de hyppige ord vi har forklaret tidligere i opgave 2.

Hos de negative anmelderne fremkommer det, at ordet ‘dårlig’ tit er efterfulgt af ‘rengøring’ og ‘service’. Anmelderne mener dermed, at de har haft dårlige oplevelser med rengøring, og har i et eller andet udstræk haft problemer med den service som Fitness World tilbyder. I den anledning, kan der både være personalet der er snakke om, men også service i for eksempel kvaliteten af maskinerne, eller hvis salgsautomaterne ikke fungere optimalt.
Dertil forekommer ordet ‘meldt mig’ 11 gange. Det gør at man bliver tilbøjelig til at mene, at der nogle problemer i forhold til sin tilmelding. Det kan både være sit medlemskab men også de holdtilmeldinger som Fitness World tilbyder. Der kunne også opbygges en hypotese omkring det økonomiske aspekt i disse to ord, hvor den negative anmeldelse omkring at personen har meldt sig til for eksempel et abonnement som ikke fungerer. Vi mener dog at der skal understøttende programmering til denne hypotese.

Hos de positive anmeldelser er det igen ordene såsom ‘dejligt’ og ‘gode’ som forekommer meget. I dette bigram forekommer de med ‘instruktører’, ‘center’, og ‘sted. Det er tydeligt at de positive anmelderne kommer med meget specifikke eksempler på hvad for nogle gode oplevelser selve anmelderen har haft. Det er som om at man har skrevet sin anmeldelse på baggrund af en selvstændig god oplevelse som har gjort indtryk på selve anmelderen. Hertil skal det nævnes at ‘jeg elsker’ forekommer 11 gange. Der er noget specifikt i Fitness World, som anmelderne ‘elsker’.

På frekvenserne på hvert enkelte ord og bigrammer har været data adskilt mellem de postitve og negative anmelderen. Der kan derfor også være godt at få en fællesmængde af ord i alle anmelderne. Den næste mængde data er derfor ord som optræder i begge filer ordet frekvens skal minimum være på otte. Denne data er supplerende argumenter for den data vi allerede har diskuteret. Det er mange af de samme emner, som anmelderne enten har haft en positiv- eller negativ oplevelse med. Igen, er der ordene ‘min’ og jeg’ som tyder på at det er anmelderens egen personlige oplevelser der ligger til baggrund for selve anmelderens udformning.

Den næste data viser os gennemsnitslængden på sætninger i både de negative og positive anmeldelser. Vores resultat er 12,1 for de negative og 9,2 for de positive. Disse tal kan godt afvige, da det er forskelligt hvilke kriterier man sætter for hvornår en sætning er færdig. I vores program har vi sat kriteriet til, at en sætning er slut efter punktum, udråbstegn eller spørgsmålstegn. Der er ikke den store forskel mellem de negative og positive anmeldelser. Hvad der kan udredes af det, er at når man skriver en anmeldelser, ligger der er en tendens i at de negative bliver længere. Det kan opstå ved, at der er skal bruges flere ord, eller flere argumenter, når man har haft en dårlig oplevelser hos Fitness World. Det kan også være at enklere at udtrykke sig positivt med færre ord.

Til sidst har vi de ord som optræder i anmeldelserne - nu adskilt hvor vi før havde de samlede ord. Vores kriterie for frekvens af hvert ord har vi her sat på 5. Da vi allerede har fået vist de mest brugte negative og positive ladet ord, får vi her et større perspektiv på hvilke ord der bruges. Det er samme resultat som vores frekvenser på de hyppige ord, men i denne data kan man få svar på nogle af hypoteser vi havde senere hen.
For eksempel i de negative anmelderne dukker der ord op som ‘regning’, ‘bemanding’, ‘toiletterne’ og der bliver nævnt priser såsom ‘99kr’. Det giver os mulighed for at kunne forklare mere detaljeret

disse negative anmeldelser. Nogle af disse ord er ikke så hyppige, men kan stadig være svar på nogle af de punkter som skal optimeres i Fitness World.

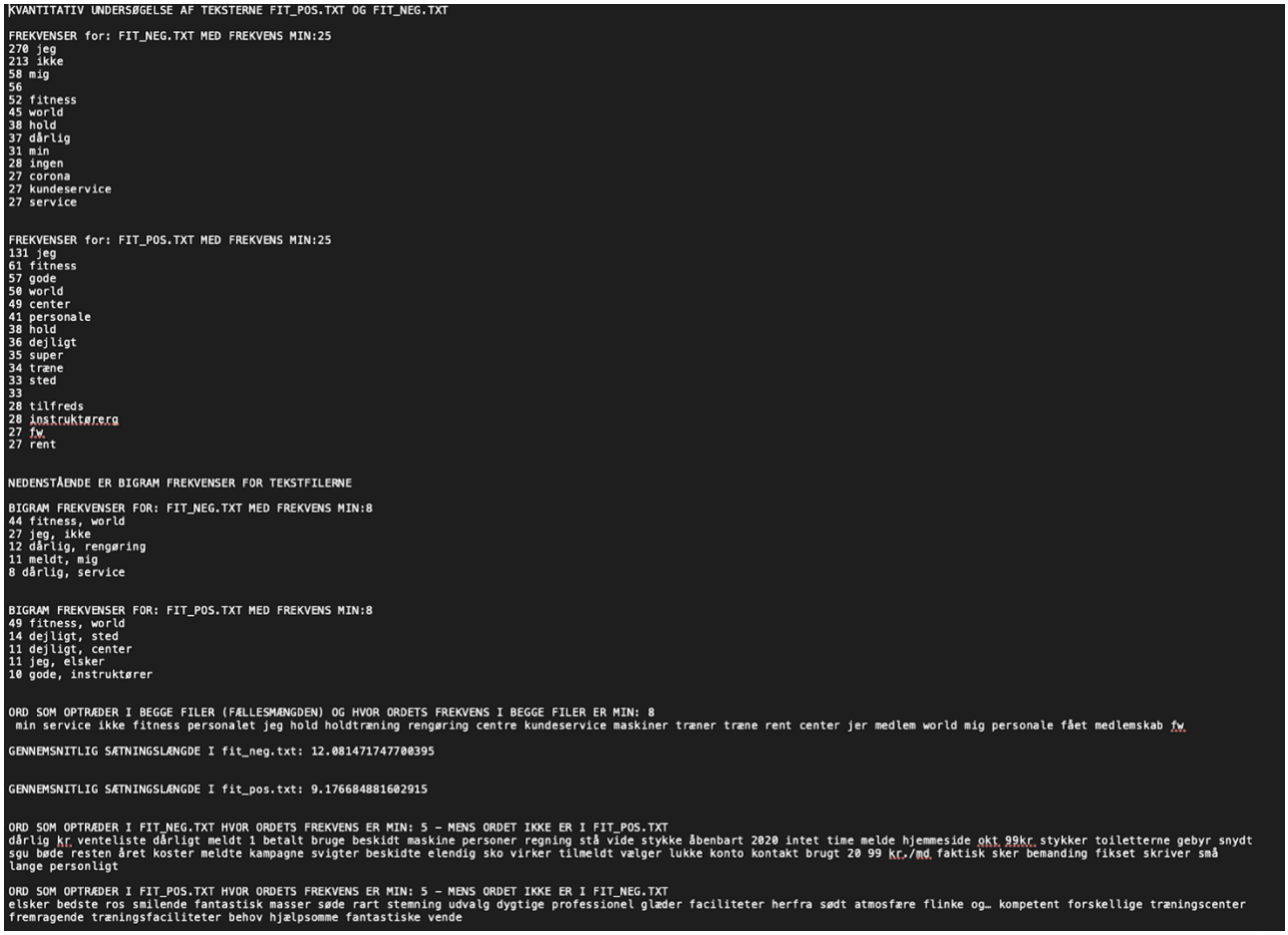
I de positive ord er der rigtig mange ord som refererer til de ansatte, som dermed supplere vores hypotese fra før. Igen, er der meget brug af positive adjektiver, som hører til menneskelige træk som eksempel 'smilende', 'søde' og 'flinke'.

Sammenfatning

De negative anmeldelser af Fitness World har vist sig at have et særligt fokus på medlemmernes økonomi. Der er en utilfredshed angående medlemmernes betaling, både i form af deres abonnement og selve priserne. Derudover er der lagt vægt på selve servicen der bliver givet på centrene, hvor rengøring og beskidte maskiner bliver nævnt i flere anmeldelse.

Hos de positive anmelderne er det tydeligt at tendensen er hos bemandingen. De ansatte bliver rost for deres udstråling, og det tyder på at medlemmer får hjælp til deres problemer. Dertil bliver selve centrene rost, hvor medlemmerne er tilfredse med atmosfæren og stemningen når man bruger faciliteterne.

Se bilag for fulde kontekst af programmering for dette kapitels nævne analysering



```
KVANTITATIV UNDERSØGELSE AF TEKSTERNE FIT_POS.TXT OG FIT_NEG.TXT

FREKVENSER for: FIT_NEG.TXT MED FREKVENS MIN:25
270 jeg
213 ikke
58 mig
56
52 fitness
45 world
38 hold
37 dårlig
31 min
28 ingen
27 corona
27 kundeservice
27 service


FREKVENSER for: FIT_POS.TXT MED FREKVENS MIN:25
131 jeg
61 fitness
57 gode
50 world
49 center
41 personale
38 hold
36 dejligt
35 super
34 træne
33 sted
33
28 tilfreds
28 instruktørerg
27 fw
27 rent


NEDENSTÅENDE ER BIGRAM FREKVENSER FOR TEKSTFILERNE

BIGRAM FREKVENSER FOR: FIT_NEG.TXT MED FREKVENS MIN:8
44 fitness, world
27 jeg, ikke
12 dårlig, rengøring
11 meldt, mig
8 dårlig, service


BIGRAM FREKVENSER FOR: FIT_POS.TXT MED FREKVENS MIN:8
49 fitness, world
14 dejligt, sted
11 dejligt, center
11 jeg, elsker
10 gode, instruktører


ORD SOM OPTRÆDER I BEGGE FILER (FÆLLESMÆNGDEN) OG HVOR ORDETS FREKVENS I BEGGE FILER ER MIN: 8
 min service ikke fitness personalet jeg hold holdtræning rengøring centre kundeservice maskiner træner træne rent center jer medlem world mig personale fået medlemskab fw

GENNEMSNITLIG SÆTNINGSLÆNGDE I fit_neg.txt: 12.081471747700395


GENNEMSNITLIG SÆTNINGSLÆNGDE I fit_pos.txt: 9.176684881602915


ORD SOM OPTRÆDER I FIT_NEG.TXT HVOR ORDETS FREKVENS ER MIN: 5 - MENS ORDET IKKE ER I FIT_POS.TXT
dårlig kr venteliste dårligt meldt 1 betalt bruge beskidt maskine personer regning stå vide stykke åbenbart 2020 intet time melde hjemmeside okt 99kr stykker toiletterne gebyr snydt
sgu bøde resten året koster meldte kampagne svigter beskidte elendig sko virker tilmeldt vælger lukke konto kontakt brugt 20 99 kr./md faktisk sker bemanding fikset skriver små
lange personligt

ORD SOM OPTRÆDER I FIT_POS.TXT HVOR ORDETS FREKVENS ER MIN: 5 - MENS ORDET IKKE ER I FIT_NEG.TXT
elsker bedste ros smilende fantastisk masser søde rart stemning udvalg dygtige professionel glæder faciliteter herfra sødt atmosfære flinke og… kompetent forskellige træningscenter
fremragende træningsfaciliteter behov hjælpsomme fantastiske vende
```

Bilag

Opgave 3

Der er mange andre frekvensanalyser i programmering som vil kunne belyse disse anmeldelse. Vi mener at en udvidet sprogmodellering er den bedste mulighed. Det betyder at bygge videre på den analyse vi allerede har kreeret, og dermed få nogle med præcise resultater.
Især trigrammer vil være en mulighed i denne sammenhæng. Hvis man har importeret NLTK er det også en mulighed at lave trigrammer. Trigrammer er reelt set det samme som bigrammer - som vi allerede har udnyttet os af. Outputtet for et trigram er dog i stedet tre sammenhængende ord i stedet for to som forekommer i bigrammer. Denne analyse ville hjælpe os, da resultatet af bigrammerne viste mange sammenhængende ord uden at man fik konteksten med. Et eksempel kunne være at 'jeg' og elsker' forekommer hyppigt. Det kunne være en fordel at udarbejde trigrammer i denne

sammenhæng, og dermed få den mulighed af at få at vide, hvad anmelderen egentlig elsker omkring Fitness World.

For at blive i NLTK mulighederne, kunne man analysere videre på de ord der forekommer hyppigt i vores bi- eller trigrammer. For at fremhæve enkelte ord kan man bruge metoden .concordance. Denne metode giver en oversigt over en systematisk opstilling over stedet hvor et ord er fremtræden. Dertil kan man skabe kriterier omkring hvor mange sammenhænge man vil finde ordet i, og hvor lang de enkelte kontekster skal fremføres. Dette gøres for eksempel ved: fill2.concordance (‘elsker’,10,15).

Slutteligt kan man udnytte NLTK’s fulde potentiale og få skabt den sidste forståelse for de enkelte frekvenser. Metoder .similar og .commen_contexts, finder ord i en tekst som optræder i lignende kontekster. Et hyppigt brugt ord eller begreb vil dermed sættes i kontekst til andre relevante sætninger eller formuleringer. Det giver os mulighed for at forstå et ord, både i dens sammenhæng med dets brug i de enkelte sætninger, men også et generelt overblik over dens diversitet igennem flere af anmeldelserne. Det er dog vigtigt at påpege, at det er programmet NLTK der bestemmer outputtet af ordene og hvilken kontekst de er er i. Et ords kontekst kan være baseret på forskellige kriterier, som giver disse to metoder en vis usikkerhed.