Data Science for Bibliotekarer

January 28, 2018

Indhold

1 Indledning

Formålet med denne bog er, at give dig en smagsprøve på hvad data science er, og en fornemmelse af hvordan man leger med data. Dette gør vi, ved at starte med en introduktion til værktøj, og hvad data science er, - og derefter dykker vi ned i nogle real-world case studies.

Første trin er at komme *i gang med Jupyter*, som er er et værktøj til at lave data science i Python. Denne bog er skrevet som en række af Jupyter notesbøger, og dette kapitel sætter dig i stand til at køre, og eksperimentere med eksemplerne.

Derefter undersøger vi: *Hvad er data science*, hvilket giver det teoretiske fundament for at forstå tilgangen i eksemplerne.

En introduktion til *Python programmering*, giver det praktiske fundament for at kunne læse koden i eksemplerne.

Herefter kommer de egentlige case studies.

Og til slut afrunder vi med konklusion, og foreslag til videre studier.

1.1 Målgruppe

Denne bog er skrevet til en workshop om data science for bibliotekarer, så den forudsætter ingen forkendskab til programmering. Ligeledes er eksemplerne taget fra biblioteksverdenen.

1.2 Læringsmål

Du bliver i stand til at

- køre python *notebooks*, og lave små ændringer / eksperimenter i disse
- ...

2 I gang med Jupyter

Jupyter Notebook er det værktøj, som vi vil bruge til at eksperimentere med data science.

En Notebook er et dokument, som kan indeholde tekst, som kan indeholde kode, og som også kan indeholde resultatet af kørslen af koden. Eksempelvis er det dokument som du er i gang med at læse nu skrevet som Jupyter Notebooks.

Hvis man eksempelvis skriver et beregningsudtryk, vil den vise koden og resultatet således:

```
In [1]: 2 + 2
```

```
Out[1]: 4
```

Når man starter Jupyter, kører der en beregningskerne, *"kernel"*, som udfører den Python-kode som man skriver. Så udover simple beregninger kan man også skrive mere generel kode såsom:

```
In [2]: import bibdata;

        print(bibdata.title_creator(123))

        bibdata.meta[123]
```

```
Min broders vogter : roman - Leif Davidsen (book)
```

```
Out[2]: {'_id': '870970-basis:28188625',
         'creator': 'Leif Davidsen',
         'idx': 123,
         'language': 'Dansk',
         'subject-term': ['Sovjetunionen',
          'spænding',
          'læsekreds',
          '÷',
          '1930-1939',
          'fascisme',
          'sk',
          'Spanien',
          'Skønlitteratur',
          'den spanske borgerkrig',
          'som Da Vinci',
```

```
 'kommunisme',
 'ideologier',
 'spændende bøger'],
'title': 'Min broders vogter : roman',
'type': 'book'}
```

Herunder beskrives først: hvordan man installerer/kører Jupyter. Og dernæst: en introduktion til hvordan man bruger Jupyter.

2.1 Opsætning / Installation

Jupyter kan enten installeres lokalt på din egen computer, eller du kan køre det i skyen. Begge dele beskrives herunder.

2.1.1 Lokal installation

Den letteste måde at installere Jupyter Notebook lokalt på din maskine er via af værktøjet Anaconda:

- Åbn https://www.anaconda.com/download i din webbrowser.
- Download og kør installationsprogrammet for `Python 3.?` versionen af Anaconda, som passer til din computer (Linux/Windows/Mac, 32bit/64bit). Følg eventuelt vejledningen på websiden.
- Start *Anaconda Navigator*, som nu er installeret, og vælg derefter at starte / "Launch" Jupyter Notebook.

Dette starter Jupyter, og åbner en webbrowser, hvor du kan se filerne på din maskine, og redigere/køre Notebooks. Næste trin er at hente eksemplerne, så du selv kan lege med dem:

- Åbn https://github.com/solsort/data-science-for-bibliotekarer/releases i din webbrowser.
- Klik på / download *Source code (zip)* for den nyeste release, og udpak denne på din computer.

Når filerne er pakket ud, kan du igen vende tilbage til Jupyter Notebook i din webbrowser. Her kan du vælge den mappe hvor du har pakket `data-science-for-bibliotekarer` ud, og begynde at køre Notebooks.

2.1.2 Kørsel i skyen

Azure giver mulighed for at køre Jupyter Notebooks gratis i skyen. For at kunne køre eksemplerne, skal du gøre følgende:

- Åbn https://notebooks.azure.com i din webbrowser.
- Klik på `Sign In`, opret bruger og log ind.
- Klik på `Libraries`, og derefter `New Library`
  - Vælg `From GitHub`
  - Skriv `solsort/data-science-for-bibliotekarer` der hvor der står *Git repo (required)*.
  - Find på et navn/id til projektet (*Friendly Name* / *Unique id* ...), - dette kan være hvad som helst
  - Klik på import (først muligt når repository, name og id er udfyldt)

Herefter vil eksemplerne blive loadet. Du har nu et kørende Jypyter Notebook environment, og kan åbne de enkelte notebooks ved at klikke på dem.

Næste gang du logger ind, kan du bare klikke på dit igangværende projekt, og behøver ikke at vælge `New Library` etc.

2.2 Guide til Jupyter

Jupyter Notebook er et udviklingsmiljø. Det vil sige at man både kan bruge det til at skrive programmer, og bruge det til at køre dem.

Et dokument i en en Notebook, består af *celler*, hvor hver celle kan indeholde tekst eller kode. Koden kan udføres via den beregningskerne / *kernel* som Jupyter også kører.

En vigtig ting at være opmærksom på, når man arbejder med celler i Jupyter Notebook, er at man kan man være i to tilstande / *modes*:

- `edit mode` er når man har en celle åben, kan skrive kode / tekst.
- `command mode` er når man bevæger sig rundt mellem cellerne, og udfører kommandoer.

Hvis du vælger "User Interface Tour" fra "Help"-menu'en i Jupyter Notebook, får du et overblik over grænsefladen.

Prøv derefter at eksperimentere lidt med grænsefladen:

1. du kan lave en ny notebook ved at vælge "File", "New Notebook", "Python 3" i menuen.
2. I den nye notebook starter du i `edit mode`, skriv eksempelvis `1+2+3`, og tryk `Alt+Enter` - herved vil den udregne resultatet af koden i cellen, oprette en ny celle, hvor du kan skrive videre.
3. Hvis du trykke `Ctrl+Enter` udfører den cellen, og går derefter i command-mode. Brug pil-op/`K` og pil-ned/`J` til at vælge celle, og tryk herefter `Enter` for at vende tilbage tilbage til edit-mode.
4. Hvis du trykker `H` mens du er i command mode får du en liste over keybindings.
5. Hvis du trykker `P` mens du er i command mode, kan du søge efter kommandoer i Jupiter.
6. Du kan også åbne en eksisterende notebook ved at vælge "File", "Open" i menuen.
7. Vælg "Kernel", "Restart & Run All" fra menu'en, for at få udført alle celler i det åbne dokument. Hvis du arbejder videre med en celle i eksisterende kode, er det ofte vigtigt at du har kørt de foregående celler.

Celler kan være af forskellige typer, og udføres så forskelligt: *Kode-celler* indeholder Python-kode, som man kan udfører, og derved se resultatet af. *Markdown-celler* indeholder tekst skrevet i markdown-format, og hvis man "udfører" cellen betyder det blot at teksten bliver vist med formattering. Når man er i command-mode, kan man trykke `M` for at fortælle at en celle er markdown og `Y` for at fortælle at en celle indeholder kode (eller vælge det via toolbaren).

Jupyter indeholder også selv dokumentation. `P` og `H` i command-mode, er godt til at finde ud af hvordan man bruger Jupyter, - og under `"Help"` i menuen, er der også links til dokumentationen for Python etc.

3

Hvad er data science

Data science handler om, at finde indsigt og viden ud fra data, gennem en videnskabelig tilgang. Det er en blanding af forskellige faglige felter, her iblandt:

- Programmering og datalogi: Hvordan vi instruerer computeren i at finde viden fra data.
- Informationsvidenskab: Hvordan vi strukturerer og formidler den viden vi får ud af data.
- Statistik og matematik: Hvordan vi konceptuelt finder viden i data.

Der er også en del overlap med de dele af datalogien, som hedder maskinlæring og data mining, som handler om hvordan man kan få computeren til (selv) at finde mønstre i data.

Data er alt hvad vi kan skrive ned. Eksempler på data er: 1) observationer, såsom en note hver gang en bog bliver udlånt, 2) tekst, såsom de fleste felter i metadata.

Kvalitative data er data der kræver menneskelig fortolkning, - det kan være interviews, beskrivelse af bøger, etc.

Kvantitative data er data som en computer direkte kan arbejde med, - det kan være statistiske data, lister af hændelser med tidspunkt, etc.

4 Python Programmering

TODO

What is programming

- Recipe, instructions for computer
  - computer = forvokset regnemaskine, kan kun gøre hvad den bliver bedt om
  - step by step instruction, computer is stupid, only does excatly what told
- Example of programming languages easily available
  - JavaScript
  - Python
  - Shell
- Getting started with Python/jupyter-notebooks
  - running within notebooks.azure.com
  - running locally with https://www.anaconda.com/download

What is data

- Concrete/paper examples: numbers, text, checkmark/questionmark, lists, indexes
- Structured data: JSON
  - example/exercise bibliographical data
- Numbers, Matrices/vectors
- Data types
  - Numbers
  - Strings
  - Lists
  - Dictionaries
  - Matrices/vectors
  - ... and many morre
- What is data science
  - "datalogi" - computer science vs. data science
  - statistics and machine learning
    - unsupervised
      - example: k-means

        * supervised
            · example: statistical

Smagsprøver på python

- forklaring af konstruktioner brugt i eksemplerne

4.1 Data, typer og strukturer

Computere er kun regnemaskiner, og for computeren er alle data tal. Så når vi arbejder med tekst, billeder, store datastrukturer, etc., er det alt sammen tal.

Eksempelvis er bogstaver i computeren oftest følgende tal: `A` er 65, `B` er 66, `C` er 67, `D` er 68, `E` er 69, `F` er 70, `G` er 71, `H` er 72, `I` er 73, `J` er 74, `K` er 75, `L` er 76, `M` er 77, og så videre, og mellemrum, `" "` er 32. Du kan nu læse hvad der står her: `72 69 74 32 77 69 68 32 68 73 71`.

Vi vil gerne arbejde med forskellige *typer* af data. Nogle typiske datatyper er:

- Tal, som både kan være heltal (... `-2`, `-1`, `0`, `1`, `2`, ...), eller kommatal (i.e. `-17.42`, `0.12345`, `1001.37`).
- Tekststrenge eller strenge, som er kortere eller længere stykke tekst. Ordet streng kommer af at tekst er en række/sekvens/tråd/streng af enkelte bogstaver. Vi vil ofte skrive eksempler på tekststreng i anførselstegne, i.e. `"dette er en streng"`.
- Lister af data, som kan indeholder strenge, tal, sandhedsværdier, og også selv indeholde lister og ordbøger. Vi vil ofte skrive lister med firkantede paranteser, eksempel: `["Høeg", "Murakami", "Blixen"]`.
- Opslagsværker eller ordbøger/dictionaries, som giver mulighed for at slå en tekststreng op og få data tilbage. Et eksempel på en ordbog kan være metadata for en bøger, hvor man eksempelvis kan slå "forfatter" og "titel" op, og måske får "A. A. Milne" og "Peter Plys" tilbage. Ordet/strengen, som man slår op i ordbogen kaldes ofte nøglen, og de data man får tilbage kaldes værdien. Ordbøger skrives ofte med krøllede paranteser, så ordbogen med metadata kunne skrives som: `{"titel": "Peter Plys", "forfatter": "A. A. Milne"}`.

Ud fra disse datatyper, kan vi nu strukturere data, eller lave *datastrukturer*. Her er et eksempel på en liste af metadata for et par bøger:

```
In [1]: books = [
          { "titel": "Peter Plys",
            "forfatter": "A. A. Milne",
            "nøgleord": ["børnebog"],
            "udlån": 4312
          },
          { "titel": "Turen går til Berlin",
            "forfatter": "Kirstine Therkelsen",
            "nøgleord": ["Berlin", "rejseører"],
            "udlån": 329
          },
          { "titel": "Mimbo Jimbo og den lange vinter",
            "forfatter": "Jakob Martin Strid",
            "udlån": 1234
          }
        ]
```

Her gemmer vi datastrukturen i en variabel der hedder `books`. Den måde som vi har skrevet data på kaldes *JSON*, og er en af de mest almindelige måde at skrive strukturerede data. Eksemplet indeholder en liste af ordbøger, som igen kan indeholde andre datatyper.

I modsætning til JSON, er Python ligeglad med om vi bruger `"` eller `'`, så hvis vi finder datastrukturen igen, skrives den ud således:

```
In [2]: books
```

```
Out[2]: [{'forfatter': 'A. A. Milne',
          'nøgleord': ['børnebog'],
          'titel': 'Peter Plys',
          'udlån': 4312},
         {'forfatter': 'Kirstine Therkelsen',
          'nøgleord': ['Berlin', 'rejsefører'],
          'titel': 'Turen går til Berlin',
          'udlån': 329},
         {'forfatter': 'Jakob Martin Strid',
          'titel': 'Mimbo Jimbo og den lange vinter',
          'udlån': 1234}]
```

Når man har en liste eller en dictionary, kan man bruge `[]` til at hente de enkelte elementer. Bemærk: pladserne i listen er navngivet `0, 1, 2, ...` og ikke `1, 2, 3, ....`

```
In [3]: books[0]
```

```
Out[3]: {'forfatter': 'A. A. Milne',
         'nøgleord': ['børnebog'],
         'titel': 'Peter Plys',
         'udlån': 4312}
```

```
In [4]: books[1]["nøgleord"]
```

```
Out[4]: ['Berlin', 'rejsefører']
```

Hvis man prøver at finde noget der mangler, får man en fejl:

```
In [5]: books[2]["nøgleord"]
```

```
        ---------------------------------------------------------------------------
        KeyError                                  Traceback (most recent call last)

        <ipython-input-5-a6602ef92512> in <module>()
    ----> 1 books[2]["nøgleord"]


        KeyError: 'nøgleord'
```

Her kan det være praktisk at bruge `.get`-metoden til at tilgå data, man kan fortælle hvad man ønsker per default, hvis data mangler:

```
In [ ]: books[1].get("nøgleord", [])
```

```
In [ ]: books[2].get("nøgleord", [])
```

5

Case: Genre-rum

Vi har en idé om, hvad *afstanden* er mellem to punkter. Hvis vi har to prikker på et stykke papir, så véd vi, hvordan vi måler afstanden mellem dem. Hvis vi har to steder i vores dagligstue, så har vi også en idé om deres afstand.

Hvordan kan vi tale om afstand mellem to bøger? Vi vil nok forvente, at bøger indenfor samme genre ligger tæt på hinanden, eksempler:

- Afstanden mellem "Peter Plys" og "Frøken Smillas fornemmelse for sne" er nok større end afstanden mellem "Cirkeline" og "Cykelmyggen Egon".
- En rejsebog om Berlin er nok tættere på en rejsebog om Paris, end på en håndarbejdsbog.
- Bøger af samme forfatter befinder sig nok i nærheden af hinanden.

Forestil dig, at vi har et genre-rum, hvor der er et punkt for hver eneste bog, og vi kan måle afstanden mellem dem. Dette vil give nye muligheder for at gå på opdagelse i litteraturen. Anbefalinger skabes ved at finde de nærmeste nabopunkter. En genre består af punkter i nærheden af hinanden. Dette gør, at vi kan bruge computeren til at udforske litteraturen.

På papir, i vores dagligstue såvel som i genrerummet, kan afstand defineres matematisk således: Hvis vi har to punkter, a og b, så er afstanden mellem dem $\sqrt{(a-b)^2}$. I én dimension er det afstanden mellem to tal. Eksempel: afstanden mellem 1 og 3 er 2, hvilket kan udregnes som $\sqrt{(1-3)^2} = \sqrt{(-2)^2} = \sqrt{4} = 2$. I to dimensioner er det afstanden mellem to prikker på et stykke papir, også kaldet den "Euklidiske afstand", $\sqrt{(x_a-x_b)^2+(y_a-y_b)^2}$. Eksempel: afstanden mellem koordinaterne $(1,2)$ og $(4,6)$ er 5 hvilket vi kan udregne som $\sqrt{(1-4)^2+(2-6)^2} = \sqrt{3^2+4^2} = \sqrt{9+16} = \sqrt{25} = 5$. I tre dimensioner er det afstanden mellem to punkter i rummet, i.e. $\sqrt{(x_a-x_b)^2+(y_a-y_b)^2+(z_a-z_b)^2}$, - og det fortsætter på samme måde i fire, fem, seks, ... dimensioner.

I matematik kalder vi ofte koordinaterne for *vektorer*. Eksempelvis er en 5-dimensionel vektor, blot en liste af fem tal, og en 100-dimensionel vektor er en liste af hundrede tal.

Hvordan skaber vi et sådan genrerum? Vi har en masse statistik om lån på bibliotekerne. Hvis vi antager, at man ofte låner indenfor samme genre, så kan computeren ud fra disse data udregne et genrerum.

Jeg har udregnet et sådan genrerum for 10.000 biblioteksmaterialer, og i det følgende vil vi undersøge, om afstanden mellem bøger i genrerummet giver mening.

5.1 Eksperimenter med afstand mellem bøger

Når vi laver et program, må vi først fortælle computeren, hvilken funktionalitet vi har brug for: `bibdata` indeholder bibliografiske data, og genrerummet, som jeg har beregnet. `numpy` indeholder matematikfunktionalitet:

```
In [1]: import bibdata
        import numpy
```

I bibdata er der en funktion, som returnerer titel/forfatter, hvis vi kommer med nummeret på et biblioteksmateriale:

```
In [2]: bibdata.title_creator(8955)
```

```
Out[2]: 'Peter Plys : komplet samling fortællinger og digte - A. A. Milne (book)'
```

Herover ser vi at bog nummer 8955 er "Peter Plys". Ligeledes kan vi se de øvrige bøger, som vi vil eksperimentere med herunder:

```
In [3]: [(book_number, bibdata.title_creator(book_number))
           for book_number in [8955, 8214, 616, 580, 149, 278, 126, 29, 688]]
```

```
Out[3]: [(8955,
          'Peter Plys : komplet samling fortællinger og digte - A. A. Milne (book)'),
         (8214, 'Frøken Smillas fornemmelse for sne : roman - Peter Høeg (book)'),
         (616, 'Cirkeline bliver til - Hanne Hastrup (book)'),
         (580, 'Cykelmyggen Egon - Flemming Quist Møller (book)'),
         (149, 'Turen går til Berlin - Therkelsen Kirstine (book)'),
         (278, 'Turen går til Paris - Aske Munck (book)'),
         (126, 'Alt om håndarbejdes strikkemagasin -  (other)'),
         (29, '1Q84 - Haruki Murakami (audiobook)'),
         (688, 'Kafka på stranden - Haruki Murakami (book)')]
```

Disse kan derefter navngives, så de er lettere at arbejde med.

```
In [4]: peter_plys = 8955
        smilla = 8214
        cirkeline = 616
        cykelmyggen = 580
        berlin = 149
        paris = 278
        strikning = 126
        q84 = 29
        kafka = 688
```

Vi kan finde punktet i genrerummet for en bog via bibdata.genres. Selve genrerummet er 100-dimensionelt, så vektoren består af 100 tal. I programmering kaldes vektorer ofte for *arrays*.

```
In [5]: bibdata.genres[peter_plys]
```

```
Out[5]: array([-0.00555056,  0.02921515,  0.02892058, -0.00930832,  0.0100181 ,
                0.02586022,  0.07639047,  0.0186474 , -0.00086742,  0.06219033,
                0.00679732,  0.11657441,  0.10813324, -0.02801489,  0.09085855,
                0.17314938,  0.01396204,  0.00813998,  0.02280807, -0.070943  ,
                0.10048762,  0.19593475,  0.07772335, -0.02303249, -0.02877597,
                0.0482331 ,  0.2106633 ,  0.10839563, -0.06060356,  0.04430847,
                0.00402697,  0.08289498,  0.13924087, -0.27573585,  0.2362854 ,
               -0.06875856,  0.16701048, -0.12419402, -0.092553  , -0.2432362 ,
```

```
-0.04190033,  0.0515306 , -0.13153618,  0.02286854,  0.17851359,
 0.04017429, -0.11496432, -0.19422885, -0.29368186, -0.23722212,
-0.05779668,  0.11127474, -0.00292388,  0.11682518, -0.04726336,
 0.24068264,  0.11335513, -0.00626563, -0.06929475, -0.06640247,
-0.02567991, -0.0427829 ,  0.04892466, -0.00911773, -0.04064843,
 0.04555216,  0.02176529,  0.01931233,  0.05836495, -0.05277547,
 0.06144368, -0.05780331, -0.09876966, -0.01271217, -0.03567164,
-0.01191766,  0.07763795,  0.06327907,  0.05489877,  0.0089579 ,
 0.18206413,  0.07300671,  0.00060291, -0.0095353 , -0.00836633,
-0.04015189,  0.02045897, -0.05925296, -0.02297055, -0.03071   ,
 0.0417429 ,  0.00575653, -0.08316413, -0.01834758,  0.04315409,
-0.13101666, -0.01981491, -0.02141316, -0.18829995,  0.07193003])
```

Som det næste vil vi definere en funktion, `distance`, som udregner afstanden mellem to punkter/vektorer. Denne bruger funktionen `numpy.linalg.norm(v)`, der udregner $\sqrt{v^2}$. Navnet `linalg` står for "linær algebra", som er den del af matematikken, der blandt andet handler om at regne med vektorer.

```
In [6]: def distance(a, b):
            return numpy.linalg.norm(a - b)
```

Vi afprøver derefter funktionen ved at finde afstanden mellem $(1, 2)$ og $(4, 6)$. Dette skal være 5, ligesom vi udregnede tidligere. Her bruger vi `numpy.array`, som laver en liste af tal om til en vektor, så computeren kan regne på den.

```
In [7]: distance(numpy.array([1, 2]), numpy.array([4, 6]))
```

```
Out[7]: 5.0
```

Afstanden mellem "Peter Plys" og "Frøken Smilla" er:

```
In [8]: distance(bibdata.genres[peter_plys], bibdata.genres[smilla])
```

```
Out[8]: 1.3247944045025537
```

Afstanden mellem "Cirkeline" og "Cykelmyggen Egon" er:

```
In [9]: distance(bibdata.genres[cirkeline], bibdata.genres[cykelmyggen])
```

```
Out[9]: 1.0181284635824785
```

Afstanden mellem rejsebøger om "Berlin" og "Paris" er:

```
In [10]: distance(bibdata.genres[berlin], bibdata.genres[paris])
```

```
Out[10]: 0.25173324294787786
```

Afstanden mellem rejsebog om "Berlin" og en håndarbejdsbog er:

```
In [11]: distance(bibdata.genres[berlin], bibdata.genres[strikning])
```

```
Out[11]: 1.3969369158183382
```

Afstanden mellem to bøger af "Haruki Murakami" er:

```
In [12]: distance(bibdata.genres[q84], bibdata.genres[kafka])
```

```
Out[12]: 0.4369157294393759
```

Konklusionen er, at afstanden mellem bøger i genrerummet faktisk giver mening. De tre forventninger, som jeg formulerede i starten af kapitlet, holder.

Bemærk at forventningerne blev formuleret, før eksperimenterne blev programmeret og kørt. Når man laver data science / videnskabelige eksperimenter, gælder det om først at formulere hypotese, og hvorledes man kan teste den, - og derefter, at udføre testen for, at se om hypotesen faktisk holder.

6 Case: Anbefalinger

```
In [1]: import bibdata
        import numpy

In [2]: [peter_plys, smilla, cirkeline, cykelmyggen, berlin, paris, strikning,
           q84, kafka] = [8955, 8214, 616, 580, 149, 278, 126, 29, 688]

In [3]: def distance(a, b):
            return numpy.linalg.norm(a - b)
```

Genrerummet bør også kunne bruges til, at finde litterære anbefalinger.

Lad os vælge en bog og derefter finde afstanden fra denne til alle andre bøger. Vi inkluderer titel/forfatter for at gøre resultatet mere læsbart. Når vi skriver `[:10]` betyder det, at vi kun viser de første 10 resultater i stedet for hele listen.

```
In [4]: distances_from_peter_plys = [
            (distance(bibdata.genres[peter_plys], bibdata.genres[other_book]),
                bibdata.title_creator(other_book))
            for other_book in range(0, 10000)
        ]
        distances_from_peter_plys[:10]

Out[4]: [(1.4011684494447065, 'Fifty shades - E. L. James (book)'),
         (1.4332511957653991, 'Journal 64 : krimithriller - Jussi Adler-Olsen (book)'),
         (1.4319693083341756,
          'Marco effekten : krimithriller - Jussi Adler-Olsen (book)'),
         (1.4377953467714748, 'Taynikma - Jan Kjær (f. 1971) (book)'),
         (1.4561631856697466, 'De glemte piger : krimi - Sara Blædel (book)'),
         (1.4390826249649267,
          'Den grænseløse : krimithriller - Jussi Adler-Olsen (book)'),
         (1.2877310264792368, 'Vildheks - Lene Kaaberbøl (book)'),
         (1.4363966653595654, 'Dødesporet : krimi - Sara Blædel (audiobook)'),
         (1.4501109638315552, 'Dødsenglen : krimi - Sara Blædel (book)'),
         (1.4287816916279745,
          'Flaskepost fra P : krimithriller - Jussi Adler-Olsen (book)')]
```

Hvis vi nu sorterer listen af bøger efter afstanden til den valgte bog, så får vi en liste af anbefalinger.

```
In [5]: sorted(distances_from_peter_plys)[:10]
```

```
Out[5]: [(0.0,
          'Peter Plys : komplet samling fortællinger og digte - A. A. Milne (book)'),
         (0.53454719612995338,
          'Bogen om Emil fra Lønneberg : samlet udgave med alle historierne om Emil fra Lønne
         (0.53718166508329679,
          'Astrid Lindgrens allerbedste historier - Ingrid Vang Nyman (book)'),
         (0.54053996866025411,
          'Mumitrolden : de samlede striber - Tove Jansson (book)'),
         (0.5654914190345337,
          'Klatremus og de andre dyr i Hakkebakkeskoven - Thorbjørn Egner (book)'),
         (0.57263271765819757,
          'Pippi Langstrømpe går om bord - Astrid Lindgren (audiobook)'),
         (0.57288576301783933, 'Bogen om Pippi Langstrømpe - Astrid Lindgren (book)'),
         (0.58822503022521333,
          'Anne Marie Helger læser Vinden i piletræerne - Anne Marie Helger (audiobook)'),
         (0.60406738110296454,
          'Pippi Langstrømpe i Sydhavet - Astrid Lindgren (audiobook)'),
         (0.61582753873808027,
          'Han er her endnu - Emil fra Lønneberg - Astrid Lindgren (book)')]
```

Vi kan nu definere dette i en funktion, hvor vi kun returnerer titlerne. Hvis man i programmering har et par data: `rec = (afstand, titel)` kan man få fat i `titel` ved at skrive `rec[1]` (og få fat i `afstand` ved at skrive `rec[0]`).

```
In [6]: def recommendations(book):
            return [recommendation[1] for recommendation in
                sorted([
                    (distance(bibdata.genres[book], bibdata.genres[other_book]),
                        bibdata.title_creator(other_book))
                    for other_book in range(0, 10000)])]
```

Med denne funktion kan vi så udforske anbefalingerne til forskellige bøger:

```
In [7]: recommendations(smilla)[:10]
```

```
Out[7]: ['Frøken Smillas fornemmelse for sne : roman - Peter Høeg (book)',
         'Den kroniske uskyld - Klaus Rifbjerg (audiobook)',
         'Vinter-Eventyr - Karen Blixen (book)',
         'Kongens Fald - Johannes V. Jensen (f. 1873) (book)',
         'Ved Vejen - Herman Bang (book)',
         'Rend mig i traditionerne - Leif Panduro (audiobook)',
         'Kronprinsessen : roman - Hanne-Vibeke Holst (book)',
         'Det forsømte forår - Hans Scherfig (book)',
         'Drageløberen - Khaled Hosseini (book)',
         'Pelle Erobreren : barndom - Martin Andersen Nexø (book)']
```

```
In [8]: recommendations(cirkeline)[:10]
```

```
Out[8]: ['Cirkeline bliver til - Hanne Hastrup (book)',
         'Godmorgen Cirkeline - Hanne Hastrup (book)',
         'Cirkeline - tæl til 10 - Hanne Hastrup (book)',
         'Godnat Cirkeline - Hanne Hastrup (book)',
         'Cirkeline på opdagelse - Ulla Raben (book)',
```

```
          'Cirkeline godnat - Hanne Hastrup (book)',
          'Den store bog om Cirkeline - Hanne Hastrup (book)',
          'Cirkeline flytter til byen - Hanne Hastrup (book)',
          'Kender du Pippi Langstrømpe? : billedbog - Ingrid Vang Nyman (book)',
          'Cirkeline - Hanne Hastrup (book)']

In [9]: recommendations(cykelmyggen)[:10]

Out[9]: ['Cykelmyggen Egon - Flemming Quist Møller (book)',
          'Den store bog om den glade løve - Louise Fatio (book)',
          'Bennys badekar - Flemming Quist Møller (book)',
          'Pandekagekagen - Sven Nordqvist (book)',
          'Stakkels Peddersen - Sven Nordqvist (book)',
          'Gok-gok i køkkenhaven - Sven Nordqvist (book)',
          'Findus flytter hjemmefra - Sven Nordqvist (book)',
          'Og hanen gol - Sven Nordqvist (book)',
          'Paddington - Michael Bond (book)',
          'Rasmus får besøg - Jørgen Clevin (book)']

In [10]: recommendations(berlin)[:10]

Out[10]: ['Turen går til Berlin - Therkelsen Kirstine (book)',
          'Politikens visuelle guide - Berlin - Søndervang Allan edt (book)',
          'Top 10 Berlin - Jürgen Scheunemann (book)',
          'Turen går til Amsterdam - Anette Jorsal (book)',
          'Turen går til Prag - Hans Kragh-Jacobsen (book)',
          'Turen går til Hamburg og Nordtyskland - Jytte Flamsholt Christensen (book)',
          'Turen går til Californien & det vestlige USA - Preben Hansen (f. 1956) (book)',
          'Turen går til London - Gunhild Riske (book)',
          'Politikens Kort og godt om Berlin - Charmetant Jim (book)',
          'Politikens visuelle guide - Prag - Vladimír Soukup (book)']

In [11]: recommendations(strikning)[:10]

Out[11]: ['Alt om håndarbejdes strikkemagasin -  (other)',
          'Kreativ strik -  (other)',
          "Ingelise's strikkemagasin -  (other)",
          'DROPS : strikkdesign - Garnstudio (periodica)',
          'Alt om håndarbejdes symagasin -  (other)',
          'Maries ideer : håndarbejde, strik, sy, hækl, bolig, inspiration -  (other)',
          'Burda style -  (other)',
          'Burda -  (other)',
          'Burda modemagasin : verdensberømt mode -  (other)',
          "Ingelise's symagasin -  (other)"]

In [12]: recommendations(kafka)[:10]

Out[12]: ['Kafka på stranden - Haruki Murakami (book)',
          'Trækopfuglens krønike - Haruki Murakami (book)',
          'Norwegian wood - Haruki Murakami (book)',
          'Sønden for grænsen og vesten for solen - Haruki Murakami (book)',
          'Efter midnat - Haruki Murakami (book)',
          'Sputnik min elskede - Haruki Murakami (book)',
```

```
'Hvad jeg taler om når jeg taler om at løbe - Haruki Murakami (book)',
'Brooklyn dårskab : roman - Paul Auster (book)',
'1Q84 - Haruki Murakami (audiobook)',
'Efter skælvet - Haruki Murakami (book)']
```

Vi har hermed lavet koden for en lille anbefalingsservice.

For perspektivering, sammenlign eksempelvis resultaterne med anbefalingerne på bibliotekernes hjemmesider. Bemærk at vi her kun kigger på et lille udvalg af materialebestanden. På bibliotek.dk kan anbefalinger findes ved, at fremsøge en bog, og derefter klikke på "Inspiration", og "Andre der har lånt...". Genrerummet som vi benytter her, er faktisk udregnet fra (en lille delmængde af) de "Andre der har lånt"-data, som DBC havde med på Hack4DK.

Bemærk at bøgerne, som vi finder anbefalinger for, er valgt på forhånd (og derved uafhængigt af hvordan anbefalingerne bliver). Derfor må kvaliteten af anbefalinger forventes at være repræsentativ.

6.1 Øvelser

- Ændr antallet af resultater der vises for nogle af de ovenstående bøger.
- Udforsk anbefalingerne for andre bøger end de ovenstående.

7 Case: Emneord

Et andet spørgsmål er om vi kan bruge genrerummet til at forbedre metadata. En hypotese er emneord ligge i klumper i genrerummet. I så fald kan vi bruge dette til at have et mål for hvor typisk et materiale er for et emne.

Lad os først kigge på hvorledes metadata ser ud:

```
In [1]: import bibdata
        from collections import Counter
        import numpy
        bibdata.meta[1234]
```

```
Out[1]: {'_id': '870970-basis:51363035',
         'creator': 'Jakob Martin Strid',
         'idx': 1234,
         'language': 'Dansk',
         'subject-term': ['årstider',
          'for 5 år',
          'sk',
          'sjove bøger',
          'Skønlitteratur',
          'for 4 år',
          'for 6 år',
          'for 3 år',
          'venner',
          'vejret',
          'dyrefortællinger'],
         'title': 'Mimbo Jimbo og den lange vinter',
         'type': 'book'}
```

Som vi kan se er der, for hvert materiale et antal emneord `subject-term`.

Lad os fokusere på de mest populære emneord. Vi kan finde disse ved at lave en liste af alle anvendte emneord `subjects`, og derefter finde de mest hyppige af disse. Listen af alle emneord finder vi ved at gennemløbe alle bibliografiske poster `bib` og for hver af disse tilføje samtlige emneord `subject` til listen af alle emneord `subjects`. Herefter kan den indbyggede `Counter` i Python finde de hyppigst brugte emneord:

```
In [2]: subjects = []
        for bib in bibdata.meta:
            for subject in bib.get('subject-term', []):
```

```
            subjects.append(subject)
    Counter(subjects).most_common(10)
```

```
Out[2]: [('sk', 7036),
         ('Skønlitteratur', 6445),
         ('for 6 år', 1109),
         ('krimi', 1100),
         ('for 5 år', 1097),
         ('for 7 år', 1063),
         ('let at læse', 1047),
         ('for 4 år', 999),
         ('for 8 år', 932),
         ('Spillefilm', 904)]
```

Vi er i stand til at finde bøgerne med et givent emneord, ved at gennemløbe alle bøgerne (book_id), og kun beholde dem der har det pågældende emneord. Hvis vi eksempelvis vil finde børnefilmene kan vi gøre det således:

```
In [3]: films = [book_id for book_id in range(len(bibdata.meta))
                 if 'Børnefilm' in bibdata.meta[book_id].get('subject-term', [])]
```

I koden herover er `len(bibdata.meta)` antallet af materialer, og `range(n)` returnere en liste med `n` tal, i.e. `[0,1,2,...]`. Når vi bruger `.get('subject-term', [])` får vi enten en liste af emneord, eller en tom liste hvis metadata mangler, - ellers ville koden fejle hvis vi manglede metadata.

Nu da vi har listen af børnefilm kan vi finde punkterne i genrerummet for disse:

Vi kan herefter udskrive de først 10 film:

```
In [4]: print([bibdata.title_creator(film) for film in films[:10]])
```

```
['Bamses billedbog dvd - Katrine Hauch-Fausbøll (movie)', 'Kaj og Andrea - Fausbøll Katrine H
```

Hver af disse film svare til et punkt(vektor) i genrerummet, som vi kan finde således:

```
In [5]: vectors = [bibdata.genres[film] for film in films]
```

Vi forestiller os at materialerne med samme emneord, ligger i en klump i nærheden af hinanden i genrerummet. I så fald er midten af klumpen gennemsnits-punktet(`mean`), og størrelsen af klumpen den statistiske spredning(`std` = standard deviation), hvilket let findes i Python:

```
In [6]: mean = numpy.mean(vectors, axis=0)
        std = numpy.std(vectors, axis=0)
```

Vi kan herefter finde ud af hvor tæt et materiale ligger på klumpen, ved at tage afstanden i genrerummet, vægtet med klumpens størrelse (spredning). I.e. vi tager punktet for materialet(`bibdata.genres[book_id]`), trækker gennemsnitspunktet fra klumpen fra(`mean`), dividere med størrelsen(`std`) og tager længden af dette `numpy.linalg.norm(...)`. Så for alice i eventyrland, cirkeline(616) og turen går til berlin(149) er deres børnefilms-agtighed (jo lavere jo bedre):

```
In [7]: print('alice i eventyrland(film)',
              numpy.linalg.norm((bibdata.genres[810] - mean) / std))
        print('cirkeline',
              numpy.linalg.norm((bibdata.genres[616] - mean) / std))
        print('berlin',
              numpy.linalg.norm((bibdata.genres[149] - mean) / std))
```

```
alice i eventyrland(film) 11.79158192
cirkeline 19.9179050997
berlin 38.8215720144
```

Vi kan nu lave en liste af biblioteksmaterialer, kombineret med hvor meget de minder om børnefilm. Hvis vi derefter sorterer disse, finder vi de mest børnefilms-agtige materialer:

```
In [8]: weighted_books = [
            (numpy.linalg.norm((bibdata.genres[book_id]-mean)/std), book_id)
            for book_id in range(len(bibdata.meta))
        ]
        weighted_books.sort()
        print(weighted_books[:10])
```

```
[(6.4345398281629294, 3801), (6.6494780896369434, 738), (6.6877823144785085, 3216), (6.702091
```

- hvilket er følgende materialer:

```
In [9]: [bibdata.title_creator(book_id) for (weight, book_id) in weighted_books][0:10]
```

```
Out[9]: ['Shrek - Elliott Ted aus (movie)',
         'Ratatouille - Walt Disney firma (movie)',
         'Ice age 2 : på tynd is - Carlos Saldanha (movie)',
         'Lilo & Stitch - Walt Disney firma (movie)',
         'Find Nemo - Reynolds David (movie)',
         'Shrek 2 - Vernon Conrad drt (movie)',
         'Toy story 2 - Walt Disney firma (movie)',
         'Junglebogen - Walt Disney firma (movie)',
         'Happy feet - Miller George f. 1945 (movie)',
         'Madagascar - Darnell Eric (movie)']
```

Nu har vi sorteret biblioteksmaterialerne efter børnefilms-agtighed, - dette kunne være praktisk at kunne, for hvilket som helst emneord. Derfor definere vi en ny funktion, som samler denne funktionalitet:

```
In [10]: def books_by_subject(subject):
             books = [book_id
                 for book_id in range(len(bibdata.meta))
                 if subject in bibdata.meta[book_id].get('subject-term', [])]
             vectors = [bibdata.genres[book] for book in books]
             mean = numpy.mean(vectors, axis=0)
             std = numpy.std(vectors, axis=0)
             weighted_books = [
                 (numpy.linalg.norm((bibdata.genres[book_id]-mean)/std), book_id)
                 for book_id in range(10000)]
             sorted_books = [
                 book_id for (weight, book_id) in sorted(weighted_books)]
             return sorted_books
```

Hvis vi tager de første materialer fra listen, som ikke har emneordet `Børnefilm` blandt dets emneord(`subject-term`), kan det være at vi finder nogle bibliografiske poster, som kunne beriges med disse metadata:

```
In [11]: [bibdata.title_creator(book_id)
          for (weight, book_id) in weighted_books
          if not 'Børnefilm' in bibdata.meta[book_id].get('subject-term', [])][:10]

Out[11]: ['Shrek - Elliott Ted aus (movie)',
          'Shrek 2 - Vernon Conrad drt (movie)',
          'Bee movie - det store honningkomplot - Hickner Steve (movie)',
          'Shrek den tredje - Miller Chris instruktør (movie)',
          'Robin Hood - Walt Disney firma (movie)',
          'Koslænget - Oedekerk Steve (movie)',
          'Pinocchio - Walt Disney firma (movie)',
          'Oliver & Company - Walt Disney firma (movie)',
          'Dumbo - Walt Disney firma (movie)',
          'Boog & Elliot 2 - vilde venner mod husdyrene - Carls John (movie)']
```

Konklusionen fra dette ekperiment er vist, at genrerummet godt kan bruges til at finde kandidater til berigelse af metadata.

Tilsvarende kan vi også finde de Børnefilm, der er mindst børnefilms-agtige:

```
In [12]: [bibdata.title_creator(book_id)
          for (weight, book_id) in reversed(weighted_books)
          if 'Børnefilm' in bibdata.meta[book_id].get('subject-term', [])][0:5]

Out[12]: ['Star wars - the clone wars - Murphy Scott aus (movie)',
          'Kidnappet - Muasya Vibeke (movie)',
          'Star wars - the clone wars. Sæson 1 - Lucas George (movie)',
          'Inside out - Walt Disney firma (movie)',
          'Eragon - Hugh Johnson (movie)']
```

Hvilket stadigt er børnefilm, så med dette konkrete eksperiment fandt vi ikke nogle underlige materialer i forhold til emnet.

Det kunne dog være interessant, at lave lignende eksperimenter med andre emneord end "børnefilm", så lad os omskrive koden til en funktion så vi lettere kan ekspementere med det. Her er en funktion, som givet et emneord finder 1) de materialer, som ikke har emneordet, men minder mest om det i genrerummet, og 2) de materialer, som har emneordet, men minder mindst om det:

```
In [13]: def subject_outliers(subject):
             print('Without subject "' + subject + '":')
             print('\n'.join(
                 ['  ' + bibdata.title_creator(book_id)
                  for book_id in books_by_subject(subject)
                  if not subject in bibdata.meta[book_id].get('subject-term', [])
                 ][:5]))
             print('With subject "' + subject + '":')
             print('\n'.join(
                 ['  ' + bibdata.title_creator(book_id)
                  for book_id in reversed(books_by_subject(subject))
                  if subject in bibdata.meta[book_id].get('subject-term', [])
                 ][:5]))

         subject_outliers('eventyr')
```

```
Without subject "eventyr":
  Sjov med feer og alfer - Daniela Dogliani (book)
  Alfer - Moffett Patricia (book)
  Tommelise - Josef Palecek (book)
  Snehvide - Pikka (book)
  Alice i Eventyrland - Walt Disney firma (book)
With subject "eventyr":
  Et sted i nærheden - Cecelia Ahern (book)
  Dragens dom - Bodil Bang Heinemeier (book)
  Aktive fortællinger, rim og remser : dialogisk læsning med børn - Lotte Salling (book)
  Eventyret om Gammelpot og den forheksede tepotte - Thom Roep (book)
  Eventyret om Gammelpot og den sorte kimono - Thom Roep (book)
```

Hvis vi skal gå videre på opdagelse i materialer, med givne emneord, kan det være praktisk med en liste med de mest almindelige emneord, hvliket findes således:

```
In [14]: "; ".join([subject for (subject, count) in Counter(subjects).most_common(200)])
```

```
Out[14]: 'sk; Skønlitteratur; for 6 år; krimi; for 5 år; for 7 år; let at læse; for 4 år; for
```

7.1 Forslag til metadata for materiale

Nu har vi set hvorledes, vi kan finde materialer, der ligger tæt ved, eller langt fra, et givent emneord. Det kunne også være sjovt at vende det om, og se hvilket emneord, der ligger tæt på et materiale ud fra hvor det ligger i genre rummet.

Første trin er at lave en statistisk model for populære emneord, eksempelvis emneord, der er tildelt til mere end 10 materialer. Lad os allerførst lave en liste over alle emneord, og hvilke materialer der har dem:

```
In [15]: subject_dict = {}

         for book in bibdata.meta:
             for subject in book.get('subject-term', []):
                 subject_dict[subject] = subject_dict.get(subject, []) + [book.get('idx')]

         subject_dict = {
             subject: book_ids
             for subject, book_ids in subject_dict.items()
             if len(book_ids) >= 10
         }
```

Vi kan herefter lave finde gennemsnittet og spredningen, for hvert emne.

```
In [16]: subjects = []
         for subject, book_ids in subject_dict.items():
             vectors = [bibdata.genres[book_id] for book_id in book_ids]
             mean = numpy.mean(vectors, axis=0)
             std = numpy.std(vectors, axis=0)
             std = std / numpy.linalg.norm(std)
             subjects.append({"name": subject, "mean": mean, "std": std})
```

Givet et `book_id` kan vi herefter sortere emnerne efter hvor godt de passer til en given bog. Herunder tager vi 10 tilfældige bøger, og udskriver hvilke emner der passer bedst til dem i genrerummet, samt viser hvad det er for en bog, og hvilke metadata den har fået tildelt.

Vi finder tilfældige bøger via `random`, og `random.seed(1)` sikre at vi får de samme bøger hver gang vi kører koden. Afstanden mellem bøger og emner bliver fundet på samme vis som tidligere herover. Denne gang er det dog emnerne, og ikke bøgerne, vi sorterer.

```
In [17]: import random; random.seed(1)

         for book_id in [random.randint(0,10000) for i in range(10)]:
             sorted_subjects = sorted([(
                 numpy.linalg.norm(
                     (bibdata.genres[book_id] - subject.get('mean')) /
                         subject.get('std')),
                 subject.get("name")) for subject in subjects])

             print(book_id, bibdata.title_creator(book_id))
             print('metadata:', bibdata.meta[book_id].get('subject-term'))
             print('genrerum:', [name for weight, name in sorted_subjects[:10]])
             print()
```

```
2201 Den uendelige historie - Michael Ende (audiobook)
metadata: ['sk', 'Eventyrromaner', 'Skønlitteratur']
genrerum: ['Oplæsning 6.- 7. kl.', 'fantastiske fortællinger', 'Fantastisk fortælling', 'Fant

9325 Magia og den forsvundne baby - Rose Impey (book)
metadata: ['for 0-2. klassetrin', 'Lettal 13', 'for folkeskolenfor 0-2. klassetrin', 'for 9 å
genrerum: ['Læse let blå 2', 'Fril. 0.-2.kl. 95 a', '1. klasse', 'Hekse', 'for 7-8 år', 'heks

1033 Min store Peter Pedal findebog - Margret (book)
metadata: ['aber', 'gå på opdagelse', 'for 5 år', 'sk', 'sjove bøger', 'Skønlitteratur', 'for
genrerum: ['gå på opdagelse', 'tal', 'for 3-6 år', 'hulbøger', 'for 5 år', 'bogstaver', 'alfa

4179 Mellem sommerens længsel og vinterens kulde : en roman om en forbrydelse - Leif G. W. Pe
metadata: ['Sverige', 'politiromaner', 'sk', 'Skønlitteratur', 'Stockholm', '÷', '1980-1989'
genrerum: ['Stockholm', 'politiromaner', 'økonomisk kriminalitet', 'Sverige', 'narkokriminali

1931 Iris og Ruby - Rosie Thomas (book)
metadata: ['kærlighed', 'mor-datter forholdet', 'Skønlitteratur', 'Cairo', 'den 2. verdenskri
genrerum: ['underholdningsromaner', 'dameromaner', 'Australien', 'bk', 'Irland', 'England', '

8117 Esben - Benni Bødker (book)
metadata: ['mystik', 'for 12 år', 'for 11 år', 'Kreta', 'spændende bøger', 'spænding', 'Serie
genrerum: ['SerieM1', 'SerieM', 'mystik', 'for 10 år', 'Anbefales: 3. klasse', 'helte', 'for

7364 Morgans run - Colleen McCullough (book)
metadata: ['mænd', 'Norfolk Island', 'Australien', 'sk', 'Skønlitteratur', 'fængsler', '1700-
genrerum: ['Australien', 'Egypten', 'historie', 'indianere', 'England', 'Sydamerika', 'Amerik

7737 Guds fjende - Bernard Cornwell (book)
metadata: ['England', 'legender', 'sagn', 'sk', 'Skønlitteratur', '400-499', 'riddere']
genrerum: ['vikinger', '1200-1299', 'vikingetiden', 'sagn', 'middelalderen', 'krig', '1100-11
```

```
6219 Lille Nørd - Dyr - Falck Anders (movie)
metadata: ['77.7', '58', 'Spillefilm', 'Zoologi', 'for 5 år', 'dyr', 'for 4 år', 'for 6 år',
genrerum: ['77.74', 'Børnefilm', 'venskab', 'tv-musik', 'superstærk', 'drenge', 'eventyr', 'f

3439 Bjørnen Bjørns ordbog - Tove Krebs Lange (book)
metadata: ['begrebsindlæring', 'for 2 år', 'sk', 'Legetek', 'for 4 år', 'Skønlitteratur', 'fo
genrerum: ['sange', 'Legetek', 'for 2-5 år', 'begrebsindlæring', '78.69', 'Vokalmusik for bør
```

Dette eksperiment, giver os både mere indsigt i metadata såvel som genrerum. Når vi ser på det første par bøger, rammer genrerummet ret godt plet:

- "Den Uendelige Historie" - her rammer genrerummets emneord plet, og mere specifikt end de eksisterende metadata. Jeg vil give maskinen ret i at dette er en fantastisk fortælling om parallelle verdener for 6-7 klasse. Vi finder også her at der er forskellige emneord for "fantastiske fortællinger", hvilket kunne give mening at normalisere.
- "Magia" - her er det lidt interessant at metadata er "søskende" og "trylleri", hvor det er "hekse" i genrerummet. Dette minder om at genrerummet afspejler hvad der ellers lånes, hvilket indikere at bogens lånere går efter andre hekse-bøger. En søgning efter bogen, viser en heks på forsiden af bogen.

Det er at bemærke at den også skyder forkert. Eksempelvis: når der er geografi/tidsangivelser/etc. har betydning i metadata, kan dette også dukke op i genrerummet, men ofte skudt forkert, - i.e.: forkerte lande, vikinger omkring år 1200 i stedet for legender om riddere omkring år 400, etc.

Konklusionen er at genrerummet godt kan bruges til inspiration om metadata, men ofte også tager fejl, så man kan ikke automatisk udtrække emneord for materialet.

7.2 Anbefalinger af emneord

Dette leder også til overvejelser at vi kan få anbefalinger til andre emneord ud fra et givent emneord. Afstanden mellem emnerord findes på samme vis som afstandend mellem emneord og bøger, dog med den tilføjelse at der normaliseres med begge emneords spredning.

```
In [18]: subject_name = 'fantastiske fortællinger'
         subject1 = [subject
             for subject in subjects
             if subject.get('name') == subject_name
         ][0]

         sorted_subjects = sorted([(
                 numpy.linalg.norm(
                     (subject1.get('mean') - subject2.get('mean')) /
                      (subject1.get('std') * subject2.get('std'))),
                 subject2.get("name")) for subject2 in subjects])
         sorted_subjects[:20]

Out[18]: [(0.0, 'fantastiske fortællinger'),
          (32.589011241465435, 'Fantastisk fortæl.'),
          (37.335710868923528, 'Oplæsning 3.- 5. kl.'),
```

```
(39.926106704591696, 'Oplæsning 6.- 7. kl.'),
(40.539648464301365, 'Fantastiske fortællinger'),
(45.438862253162597, 'Fantastisk fortælling'),
(47.40923136227692, 'Fantastiske fortæll.'),
(49.158284916890935, 'Eventyr'),
(51.20541490301153, 'parallelle verdener'),
(51.598369067958373, 'Mellemskolingen'),
(51.703630185244045, 'Mellemskoling'),
(51.875821482898822, '6.-7. klasse'),
(52.337659955147281, 'Seriebøger'),
(54.247970145997982, 'Oplæsning 3.-5. kl.'),
(56.363399583178662, 'Spænding'),
(56.655407858672753, 'eventyrlige fortællinger'),
(58.706610226001743, 'Ungdomsbog'),
(58.729694544002122, 'tro'),
(58.85151876502978, 'Læs højt'),
(60.029052682330089, 'Oplæsning 3.- 5. kl')]
```

Konklusionen af stikprøven herover viser, at genrerummet også kan bruges til at finde relaterede emneord.

7.3 Øvelser

- Find listen af de 100 eller 200 mest populære emneord, i stedet for blot de 10 mest populære.
- Find ud af hvilke andre outliers der er, eksempelvis for emneord som "sex"ede bøger, "tv-serier", "Fodbold", "Biografier af enkelte personer", etc. som er outliers, enten fordi de ligner dette emne og ikke har emneorder, - eller har emneordet og ikke ligner de andre indenfor emnet. Kunne det give mening at fjerne/tilføje emneordet for disse?

8 Case: Genre-klynger

Jeg er nysgerrig om hvilke typer / kategorier af biblioteksmaterialer, der er. Særligt i litteratur, som ligger ud over klassifikationssystemer som DK5. Derfor vil jeg lave en mere udforskende / eksplorativ analyse. En tilgang til detter er, at bede computeren, at sortere biblioteksmaterialerne i et antal bunker(klynger), hvor ting, der ligner hinanden, ligger i samme bunke(klynge).

Dette kaldes klynge-analyse, og en af de enkleste måder at gøre dette kaldes k-means. Her starter man med at fortælle hvor mange bunker man ønsker. Dette antal kaldes ofter k. Algoritmen virker således: vi har et punkt i genrerummet for hver bunke. Hver bog lægges i den bunke, hvor afstanden mellem bunkens punkt og bogen er kortest. Derefter flyttes punktet for hver bunke, til gennemsnits-punktet for alle bøgerne i bunken. Dette gentager man indtil, at punkterne for bunkerne ikke længere flytter sig.

Bemærk at vi her beder computeren om, selv at finde en struktur i data. Dette kaldes unsupervised learning indenfor machine learning / data mining.

Når vi skal programmere det, findes k-means algoritmen allerede i Python's SciKit-Learn funktionsbibliotek. Så vi behøver blot importere det, fortælle at vi ønsker k-means med k klynger, samt at den skal køre algoritmen på genrerummet:

```
In [6]: import bibdata
        import sklearn.cluster
        k = 15
        kmeans = sklearn.cluster.KMeans(n_clusters = k)
        kmeans.fit(bibdata.genres)
```

```
Out[6]: KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
            n_clusters=15, n_init=10, n_jobs=1, precompute_distances='auto',
            random_state=None, tol=0.0001, verbose=0)
```

Nu har computeren fundet nogle klynger, som vi kan kigge på.

Vi vil nu se hvad klyngerne indeholder. For hver klynge udskriver vi derfor titel/forfatter for de første bøger i hver klynge. Koden for dette læses lettest bagfra: `cluster_id` er nummeret på klyngen. For hvert `cluster_id` gennemløber alle 10000 bøger, og udvælger dem som faktisk ligger i klyngen. Til dette bruger vi `kmeans.labels_`, som er en liste over hvilken klynge hver bog hører til. Herefter udvælger vi de første `book_count` bøger, som vi returnere, så vi faktisk kan se hvad der er i klyngen:

```
In [7]: book_count = 7
        [[bibdata.title_creator(book_id)
          for book_id in range(10000)
          if kmeans.labels_[book_id] == cluster_id][0:book_count]
         for cluster_id in range(k)]
```

```
Out[7]: [['Os fra Blomsterkvarteret - Dy Plambeck (book)',
          'Troldmandens flyvefedt - Bent T. Lund (f. 1946) (book)',
          'Klods-Hans - H. C. Andersen (f. 1805) (audiobook)',
          'Hvad kan det være? - Keld Petersen (f. 1955) (book)',
          'Kejserens nye klæder - Birte Dietz (book)',
          'Den grimme ælling - Svend Otto (book)',
          'Læs og gæt gåder - Keld Petersen (f. 1955) (book)'],
         ['Naboens søn : roman - Jane AAmund (book)',
          'Knud, den store : roman - Hanne-Vibeke Holst (book)',
          'Zornig - vrede er mit mellemnavn - Lisbeth Zornig Andersen (f. 1968) (book)',
          'Vindue uden udsigt - Jane AAmund (book)',
          'Vi kan sove i flyvemaskinen - Ulla Terkelsen (audiobook)',
          'Hvor solen græder : en fortælling fra Syrien - Puk Damsgård Andersen (book)',
          'Onklerne og deres fruer : erindringsroman (mp3) - Jane Aamund (audiobook)'],
         ['Fifty shades - E. L. James (book)',
          'Journal 64 : krimithriller - Jussi Adler-Olsen (book)',
          'Marco effekten : krimithriller - Jussi Adler-Olsen (book)',
          'Taynikma - Jan Kjær (f. 1971) (book)',
          'De glemte piger : krimi - Sara Blædel (book)',
          'Den grænseløse : krimithriller - Jussi Adler-Olsen (book)',
          'Dødesporet : krimi - Sara Blædel (audiobook)'],
         ['Vildheks - Lene Kaaberbøl (book)',
          'Harry Potter og De Vises Sten - Joanne K. Rowling (book)',
          'Skammerens datter - Lene Kaaberbøl (book)',
          'Harry Potter og Hemmelighedernes Kammer - Joanne K. Rowling (book)',
          'Mustafas kiosk - Jakob Martin Strid (book)',
          'Wimpy Kid - Jeff Kinney (book)',
          'Gummi-Tarzan - Ole Lund Kirkegaard (audiobook)'],
         ['Vi unge : kun for piger -  (other)',
          'Livs liv - Rikke Dyrhave Nissen (book)',
          'Pretty little liars - Sara Shepard (book)',
          'Knus-klubben - Gitte Løkkegaard (book)',
          'Lego Indiana Jones - the original adventures - Glyn Scragg (game)',
          'Olivia -  (other)',
          'Toy story 3 - Walt Disney (firma) (game)'],
         ['Stå fast : et opgør med tidens udviklingstvang - Svend Brinkmann (book)',
          'Spis maven flad : sådan gør du! : 111 skønne opskrifter - Engell Majbritt L. aut (
          'Tarme med charme : alt om et undervurderet organ - Giulia Enders (book)',
          'Hvad kan jeg blive? : Politikens erhvervsvejledning - Dueled Knud (book)',
          'Verdens bedste kur : vægttab der holder - Bitz Christian aut (book)',
          'Dr. Zukaroffs testamente : en bog om menneskehjernen - Peter Lund Madsen (book)',
          'Stenalderkost : palæo-opskrifter til det moderne menneske - Thomas Rode Andersen (
         ['Min mormors gebis - Jakob Martin Strid (book)',
          'Kender du Pippi Langstrømpe? : billedbog - Ingrid Vang Nyman (book)',
          'Pippi Langstrømpe på Kurrekurredut-øen - Ingrid Vang Nyman (book)',
          'Lille frø - Jakob Martin Strid (book)',
          'Da Findus var lille og forsvandt - Sven Nordqvist (book)',
          'Fyrtøjet - H. C. Andersen (f. 1805) (book)',
          'Da Lille Madsens hus blæste væk - Jakob Martin Strid (book)'],
         ['Turen går til Hamburg og Nordtyskland - Jytte Flamsholt Christensen (book)',
          'Turen går til London - Gunhild Riske (book)',
```

```
 'Turen går til Rom - Alfredo Tesio (book)',
 'Turen går til Mallorca, Menorca & Ibiza - Jytte Flamsholt Christensen (book)',
 'Turen går til Tyrkiet - Fenger-Grøndahl Carsten (book)',
 'Turen går til Berlin - Therkelsen Kirstine (book)',
 'Turen går til Sydspanien - Jørgen Laurvig (book)'],
['Kriminalkommissær Barnaby - Caroline Graham (movie)',
 'Avatar - Fiore Mauro cng (movie)',
 'Hævnen - Bier Susanne (movie)',
 'Dirch - Zandvliet Martin Pieter (movie)',
 'The godfather - Willis Gordon (movie)',
 'Flammen & Citronen - Andersen Lars K. f. 1960-11-27 (movie)',
 'Black swan (blu-ray) - maclaughlin McLaughlin John J. (movie)'],
['Bamses billedbog dvd - Katrine Hauch-Fausbøll (movie)',
 'Kaj og Andrea - Fausbøll Katrine Hauch- (movie)',
 'Ninjago - masters of spinjitzu - Sørensen Thomas inv (movie)',
 'Pippi - Hellbom Olle (movie)',
 'Filmhits for børn -  (movie)',
 'Legends of Chima - Andreasen Tommy inv (movie)',
 'Op. Med bonus features disc - Docter Pete (movie)'],
['Niceville - Kathryn Stockett (book)',
 'Jeg skal gøre dig så lykkelig - Anne B. Ragde (audiobook)',
 'Den stjålne vej - Anne-Cathrine Riebnitzsky (book)',
 'Øen - Victoria Hislop (book)',
 'Jordemoderen fra Hope River - Patricia Harman (audiobook)',
 'Fra hus til hus : roman - Kristín Marja Baldursdóttir (book)',
 'Orkideens hemmelighed - Lucinda Riley (audiobook)'],
['Den gode opgave : opgaveskrivning på videregående uddannelser - Rienecker Lotte (b
 'Interview : en introduktion til det kvalitative forskningsinterview - Steinar Kval
 'Læring : aktuel læringsteori i spændingsfeltet mellem Piaget, Freud og Marx - Knud
 'Kvalitative metoder : en grundbog - Tanggaard Lene edt (book)',
 'Klassisk og moderne samfundsteori - Kaspersen Lars Bo edt (book)',
 'Helbredets mysterium : at tåle stress og forblive rask - Aaron Antonovsky (book)',
 'Modernitet og selvidentitet : selvet og samfundet under sen-moderniteten - Anthony
['Alt om håndarbejdes symagasin -  (other)',
 'Alt om håndarbejdes strikkemagasin -  (other)',
 'Alt om haven -  (other)',
 'Haven - Det Jyske Haveselskab (other)',
 'Danmarks store gør det selv leksikon - Nielsen Jørn (book)',
 'Burda -  (other)',
 'Politikens strikke- og hæklebog - Vivian Høxbro (book)'],
['Giganternes fald - Ken Follett (audiobook)',
 'Og bjergene gav genlyd - Khaled Hosseini (book)',
 'Det syvende barn - Erik Valeur (audiobook)',
 'Den hundredårige der kravlede ud ad vinduet og forsvandt - Jonas Jonasson (book)',
 '1Q84 - Haruki Murakami (audiobook)',
 'Min kamp : roman - Karl Ove Knausgård (book)',
 'Brobyggerne - Jan Guillou (book)'],
['Det blinde punkt : krimi - Julie Hastrup (book)',
 'Ensomme hjerters klub : kriminalroman - Lotte Hammer (book)',
 'Skrig under vand - Ole Tornbjerg (book)',
 'Alting har sin pris : kriminalroman - Lotte Hammer (book)',
```

```
'Skindød - Thomas Enger (book)',
'Møgkællinger - Gretelise Holm (f. 1946) (book)',
'Manden der ikke var morder - Hans Rosenfeldt (book)']]
```

Hvis vi kører klyngeanalysen igen, vil den komme med nogle andre klynger, men strukturen i resultatet vil være det samme, i.e. rejsebøger, børnefilm, forskellige typer krimier, forskellige typer børnebøger, andre forskellige typer romaner, studiebøger, etc.

Resultatet er i mine øjne interessant. Blandt andet ser jeg forskellige separate typer af skønlitteratur, hvis opdeling jeg ikke kendte til i forvejen. Det kunne være interessant, at køre dette mere finkornet, på større datasæt, og derefter danne nogle foreslag til emnehierakier for skønlitteratur, for at få mere indblik i denne del af litteraturen.

8.1 Øvelser

- Få computeren til at udføre denne notebook flere gange, og se hvordan de fundne klynger ændre sig, men stadigt har noget af den samme struktur. (Rækkefølgen ændres også, så læg i stedet mærke til karakteristiske klynge, og se at/om de går igen).
- Ændr antallet, `k`, af fundne klynger. Prøv eksempelvis med 25 eller flere, hvorved resultatet bliver mere finkornet.
- Ændr det maksimale antal af bøger `book_count`, som bliver vist fra hver klynge. Hvis man viser mere end de nuværende 7 materialer, giver det mere indblik i klyngen (men ville også bruge mere papir/skærmplads).

9 Efterskrift

TODO

- Konklusion
- Links til videre studier