Lab 7 - Topic modeling

October 18, 2017

1 Lab Extension 2 : Topic Models on Harry Potter Books

In this lab, we'll apply Latent Dirichlet Allocation to the text data of the seven Harry Potter Books. The text has been downloaded from https://archive.org/stream/Book5TheOrderOfThePhoenix and will be preprocessed before applying topic modelling and other methods that will explore data. The downloaded text, present in the folder has been modified : titles of chapter have been corrected as there were some errors on them. The files containing the chapter names have also been scraped using http://www.harrypotterfacts.com/_chapters.htm.

2 Preprocessing the Data

We want to divide our data into chapter files so that we can extract interesting information such as the evolution of topics through the chapters. The pages appear in the file, we will then first need to remove then. Second we will need to create a new file each time we go to a new chapter. Finally we will remove useless trailing spaces.

```
In [1]: import os

        chapterToTitles = {}
        #Compute the dictionary of chapters
        chapter = 1
        for i in range(1,8):
            with open("Data/ChaptersBook"+str(i)+".txt") as infile:
                for line in infile.readlines():
                    chapterToTitles[chapter]=line.strip()
                    chapter = chapter + 1
            infile.close()

        titlesToChapter = {value:key for key,value in chapterToTitles.iteritems()}

        if not os.path.exists("HarryPotterBookComputedData"):
            os.makedirs("HarryPotterBookComputedData")

        print "We have now computed the Dataset to use for our Topic Modelling with the follow

        newBookChapter = []
```

```
    chapterToBook={}

    outfile = None
    chapter = 1
    for i in range(1,8):
        newBookChapter.append(chapter)
        print "BOOK "+str(i)
        totalChapters = chapter - 1
        with open("Data/HarryPotterBook"+str(i)+".txt","rb") as infile:
            for line in infile.readlines():
                textLine = line.strip().replace("","'")
                if textLine!="":
                    if textLine in titlesToChapter and textLine.isupper() and titlesToChap
                        if i==1:
                            print str(i)+"."+str(chapter-totalChapters)+" "+textLine
                        try:
                            outfile.close()
                        except:
                            pass
                        outfile = open("HarryPotterBookComputedData/Chapter"+str(titlesToCh
                        chapterToBook[chapter]=i
                        chapter = chapter + 1
                    else:
                        if not "Page | " in textLine and not "Rowling" in textLine:
                            outfile.write(textLine+" ")
        infile.close()
    outfile.close()
```

```
We have now computed the Dataset to use for our Topic Modelling with the following chapters :

BOOK 1
1.1 THE BOY WHO LIVED
1.2 THE VANISHING GLASS
1.3 THE LETTERS FROM NO ONE
1.4 THE KEEPER OF THE KEYS
1.5 DIAGON ALLEY
1.6 THE JOURNEY FROM PLATFORM NINE AND THREE-QUARTERS
1.7 THE SORTING HAT
1.8 THE POTIONS MASTER
1.9 THE MIDNIGHT DUEL
1.10 HALLOWEEN
1.11 QUIDDITCH
1.12 THE MIRROR OF ERISED
1.13 NICOLAS FLAMEL
1.14 NORBERT THE NORWEGIAN RIDGEBACK
1.15 THE FORBIDDEN FOREST
1.16 THROUGH THE TRAPDOOR
1.17 THE MAN WITH TWO FACES
```

```
BOOK 2
BOOK 3
BOOK 4
BOOK 5
BOOK 6
BOOK 7
```

```
In [2]: from glob import glob

        #We extract sorted files
        files = sorted(glob('HarryPotterBookComputedData\Chapter*.txt'),key=lambda chapter: int
        #We remove the last chapter as it happens years after the events of the previous chapte
        #as the topics addressed in it are not very related to the previous ones
        #files = files[:-1]
```

We then read the text into a numpy array.

```
In [3]: from sklearn.feature_extraction.text import CountVectorizer

        vect = CountVectorizer(lowercase=True,max_df=0.352,min_df=0.1,input='filename',stop_wor
```

It is to be noted that the parameter max_df is especially important has the most present words will often appear in the computed topic. max_df must be well chosen so that we don't get too common words but also make sure we keep the important ones.

```
In [4]: X = vect.fit_transform(files)

In [5]: X.shape

Out[5]: (199, 2078)

In [6]: ivoc = {j:i for i,j in vect.vocabulary_.items()}
```

Statistics

1. What are the highest-frequency words in the dataset?
2. How many words are there in the dataset ?

```
In [7]: import pandas as pd
        import numpy as np

        df = pd.DataFrame(np.transpose(X.sum(axis=0)),columns=["frequency"])
        df['word']=pd.DataFrame.from_dict(ivoc,orient='index')
        df = df.sort_values(["frequency","word"],ascending=False)
        print df.head(10),"\n"
        print "The data is composed of "+str(df["frequency"].sum())+" words."
```

```
      frequency      word
1940        629  umbridge
1942        543     uncle
1185        507     moody
1980        505    vernon
738         474     fudge
500         467     dobby
2068        441       yeh
538         434    dudley
1641        432       sir
1666        410  slughorn

The data is composed of 120740 words.
```

3 Running the topic model

We create a `LatentDirichletAllocation` model before fitting it.

```
In [8]: from sklearn.decomposition import LatentDirichletAllocation

        lda = LatentDirichletAllocation(n_components=10)
```

After tweaking the parameters for the LDA, the following LDA seems to give meaningfull topics :

```
In [9]: lda = LatentDirichletAllocation(max_iter=50,evaluate_every=5,verbose=1,learning_method=
        theta = lda.fit_transform(X);

iteration: 1 of max_iter: 50
iteration: 2 of max_iter: 50
iteration: 3 of max_iter: 50
iteration: 4 of max_iter: 50
iteration: 5 of max_iter: 50, perplexity: 1715.3605
iteration: 6 of max_iter: 50
iteration: 7 of max_iter: 50
iteration: 8 of max_iter: 50
iteration: 9 of max_iter: 50
iteration: 10 of max_iter: 50, perplexity: 1625.1401
iteration: 11 of max_iter: 50
iteration: 12 of max_iter: 50
iteration: 13 of max_iter: 50
iteration: 14 of max_iter: 50
iteration: 15 of max_iter: 50, perplexity: 1596.1581
iteration: 16 of max_iter: 50
iteration: 17 of max_iter: 50
iteration: 18 of max_iter: 50
iteration: 19 of max_iter: 50
```

```
iteration: 20 of max_iter: 50, perplexity: 1581.3037
iteration: 21 of max_iter: 50
iteration: 22 of max_iter: 50
iteration: 23 of max_iter: 50
iteration: 24 of max_iter: 50
iteration: 25 of max_iter: 50, perplexity: 1571.5552
iteration: 26 of max_iter: 50
iteration: 27 of max_iter: 50
iteration: 28 of max_iter: 50
iteration: 29 of max_iter: 50
iteration: 30 of max_iter: 50, perplexity: 1564.5550
iteration: 31 of max_iter: 50
iteration: 32 of max_iter: 50
iteration: 33 of max_iter: 50
iteration: 34 of max_iter: 50
iteration: 35 of max_iter: 50, perplexity: 1559.7330
iteration: 36 of max_iter: 50
iteration: 37 of max_iter: 50
iteration: 38 of max_iter: 50
iteration: 39 of max_iter: 50
iteration: 40 of max_iter: 50, perplexity: 1556.2214
iteration: 41 of max_iter: 50
iteration: 42 of max_iter: 50
iteration: 43 of max_iter: 50
iteration: 44 of max_iter: 50
iteration: 45 of max_iter: 50, perplexity: 1553.3430
iteration: 46 of max_iter: 50
iteration: 47 of max_iter: 50
iteration: 48 of max_iter: 50
iteration: 49 of max_iter: 50
iteration: 50 of max_iter: 50, perplexity: 1551.1613
```

The perplexity is quite low and converges quickly, let us then see what topics we are then given :

```
In [10]: def show_topics(lda,ivoc,number_words=10,topics=range(10)):
             for k,topic in enumerate(lda.components_):
                 if k in topics:
                     print(k+1,[str(ivoc[i]) for i in topic.argsort()[::-1][:number_words]])
```

```
In [11]: show_topics(lda,ivoc)
```

```
(1, ['lockhart', 'luna', 'crabbe', 'goyle', 'nick', 'seamus', 'filch', 'headless', 'peeves', 'i
(2, ['tonks', 'kitchen', 'mundungus', 'prophet', 'hedwig', 'albus', 'moody', 'car', 'rita', 'a
(3, ['moody', 'crouch', 'cedric', 'krum', 'bagman', 'diggory', 'fleur', 'tournament', 'madame'
(4, ['yeh', 'ter', 'wood', 'team', 'snitch', 'broom', 'field', 'yer', 'firebolt', 'buckbeak'])
(5, ['umbridge', 'trelawney', 'cho', 'lesson', 'parvati', 'james', 'angelina', 'lavender', 'ho
(6, ['fudge', 'minister', 'dementors', 'train', 'scabbers', 'shop', 'crookshanks', 'azkaban',
```

```
(7, ['kreacher', 'bellatrix', 'master', 'wormtail', 'sword', 'eater', 'snake', 'luna', 'horcru
(8, ['slughorn', 'riddle', 'sir', 'filch', 'bathroom', 'quirrell', 'chamber', 'map', 'lesson',
(9, ['dobby', 'sir', 'elf', 'elves', 'master', 'squeaked', 'bludger', 'tea', 'clothes', 'kitch
(10, ['uncle', 'vernon', 'dudley', 'aunt', 'petunia', 'dursleys', 'kitchen', 'car', 'drive', '
```

The topics we have are very interesting. Indeed some make sense for people who know well the universe of Harry Potter. The third topic refers to the Goblet of Fire Tournament. The seventh topic refers to the Horcruxes of Voldemort. The ninth topic refers to the elves and Dobby especially. The tenth topic refers to Harry's adoptive family. Let us see an example of the distribution of topics over a chapter. To do so, we pick a chapter that has the most entropy, that is, the one where the distribution of the topics is the most balanced, so that we can visualize well different topics within a chapter. The chapter with the most even distribution is then :

```
In [12]: def getChapter(chapter):
             return "Book "+str(chapterToBook[chapter])+" Chapter "+str(chapter-newBookChapter

         max_entropy_chapter = np.argmax(-np.multiply(np.log(theta),theta).sum(axis=1))+1
         print "Chapter with the highest entropy : ",max_entropy_chapter,"(",getChapter(max_en
         print "The corresponding topic distribution is then : ",[float('{0:.2f}'.format(a)) fo
```

```
Chapter with the highest entropy :  57 ( Book 3 Chapter 22 OWL POST AGAIN )

The corresponding topic distribution is then :  [0.06, 0.09, 0.0, 0.3, 0.05, 0.28, 0.15, 0.0, 0
```

We will now highlight the text with the 3 topics with highest probability : topic 4, topic 6, topic 7

```
In [13]: def highlight(colour, text):
             if colour == "black":
                 return "\033[1;40m" + str(text) + "\033[1;m"
             if colour == "red":
                 return "\033[1;41m" + str(text) + "\033[1;m"
             if colour == "green":
                 return "\033[1;42m" + str(text) + "\033[1;m"
             if colour == "yellow":
                 return "\033[1;43m" + str(text) + "\033[1;m"
             if colour == "blue":
                 return "\033[1;44m" + str(text) + "\033[1;m"
             if colour == "magenta":
                 return "\033[1;45m" + str(text) + "\033[1;m"
             if colour == "cyan":
                 return "\033[1;46m" + str(text) + "\033[1;m"
             if colour == "gray":
                 return "\033[1;47m" + str(text) + "\033[1;m"
             return str(text)

         topicToColor={0:"black",1:"grey",7:"yellow",3:"blue",4:"green",5:"yellow",6:"red",8:"m
         print "COLORS :","\n",highlight("blue","Topic 4 : Quidditch"),"\n",highlight("yellow"
```

```
COLORS :
Topic 4 : Quidditch
Topic 6 : Minister
Topic 7 : Horcrux
```

```
In [14]: wordTopic={}
         for k,topic in enumerate(lda.components_):
             if k==3 or k==5 or k==6:
                 arr = [str(ivoc[i]) for i in topic.argsort()[::-1][:300]]
                 for word in arr:
                     wordTopic[word]=k
         with open (files[max_entropy_chapter-1]) as fin:
             for line in fin.readlines():
                 words = line.split()
                 for word in words:
                     if word.lower() in wordTopic:
                         print highlight(topicToColor[wordTopic[word.lower()]],word),
                     else:
                         print word,
```

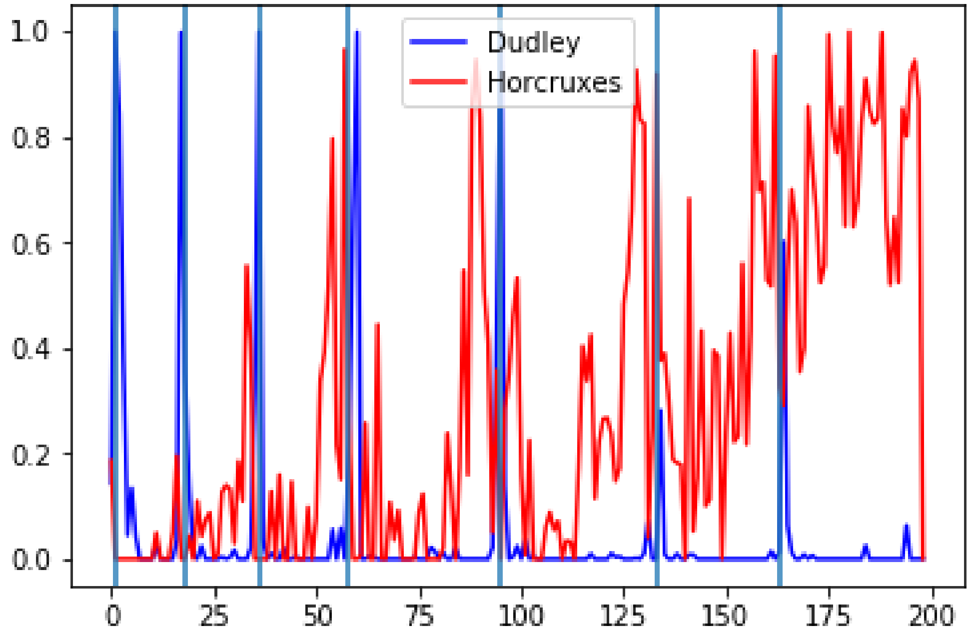
```
[illegible] fluffy [illegible]
```

3.1 How topics evolve through chapters ?

Let us select two topics : the Horcruxes and Harry's adoptive family.

```
In [15]: horcrux_topic = 7
         dudley_topic = 10
```

```
In [16]: import matplotlib.pyplot as plt

         plt.plot(theta[:,dudley_topic-1],color='blue',label="Dudley")
         plt.plot(theta[:,horcrux_topic-1],color='red',label="Horcruxes")
         for element in newBookChapter:
             plt.axvline(element)
         plt.legend()
         plt.show()
```



It is very interesting to see that the Dudley topic appears at the beggining of every book as every book start with Harry being with his adoptive family. We can clearly see too that the Horcrux topic appear at the end of the second book, when Harry destroys an Horcrux (the snake and gets the sword), and that this topic has higher probability at the end, when he intends to destroy all Horcruxes. Let us now explore the chapters at which the Horcrux topic and the Dudley topic are maximum : **Horcrux topic**

```
In [17]: max_probability_horcrux_chapter = np.argmax(theta[:,horcrux_topic-1])+1

         print "The Horcrux topic has maximum probability density for the chapter :",max_probal

         def show_words(lda,ivoc,number_words=10,chosenTopic=0):
             for k,topic in enumerate(lda.components_):
                 if k==(chosenTopic-1):
                     words = [str(ivoc[i]) for i in topic.argsort()[::-1][:number_words]]
                     for word in words:
                         print word+",",

         print "The 50 main words in it are :","\n"
         show_words(lda,ivoc,50,horcrux_topic)
```

```
The Horcrux topic has maximum probability density for the chapter : 181 ( Book 7 Chapter 19 THI

The 50 main words in it are :

kreacher, bellatrix, master, wormtail, sword, eater, snake, luna, horcrux, fleur, goblin, luci
```

Let us now view the corresponding document :

```
In [18]: wordTopic={}
         for k,topic in enumerate(lda.components_):
             if k==(horcrux_topic-1):
                 arr = [str(ivoc[i]) for i in topic.argsort()[::-1][:500]]
                 for word in arr:
                     wordTopic[word]=k
         with open (files[max_probability_horcrux_chapter-1]) as fin:
             for line in fin.readlines():
                 words = line.split()
                 for word in words:
                     if word.lower() in wordTopic:
                         print highlight(topicToColor[wordTopic[word.lower()]],word),
                     else:
                         print word,
```

[illegible]

The chapter corresponds well to the topic as it is the time when Voldmort plans for creating the Horcruxes. Let us now look at the other topic : **Dudley topic**

```
In [19]: max_probability_dudley_chapter = np.argmax(theta[:,dudley_topic-1])+1

         print "The Dudley topic has maximum probability density for the chapter :",max_probab

         def show_words(lda,ivoc,number_words=10,chosenTopic=0):
             for k,topic in enumerate(lda.components_):
                 if k==(chosenTopic-1):
                     words = [str(ivoc[i]) for i in topic.argsort()[::-1][:number_words]]
                     for word in words:
                         print word+",",

         print "The 50 main words in it are :","\n"
         show_words(lda,ivoc,50,dudley_topic)
```

```
The Dudley topic has maximum probability density for the chapter : 96 ( Book 5 Chapter 2 A PEC

The 50 main words in it are :

uncle, vernon, dudley, aunt, petunia, dursleys, kitchen, car, drive, privet, birthday, letters
```

Let us now view the corresponding document :

```
In [20]: wordTopic={}
         for k,topic in enumerate(lda.components_):
             if k==(dudley_topic-1):
```

```
                arr = [str(ivoc[i]) for i in topic.argsort()[::-1][:500]]
                for word in arr:
                    wordTopic[word]=k
        with open (files[max_probability_dudley_chapter-1]) as fin:
            for line in fin.readlines():
                words = line.split()
                for word in words:
                    if word.lower() in wordTopic:
                        print highlight(topicToColor[wordTopic[word.lower()]],word),
                    else:
                        print word,
```

[illegible]

The results are very coherent with the chapter as the chapter mainly focus on Harry's relations with his adoptive family.

3.2 A WORD CLOUD VISUALIZATION

We will finish by creating a word cloud for the two previous chapters and the whole Harry Potter Book serie **Horcrux chapter**

```
In [21]: ! pip install WordCloud
```

```
Requirement already satisfied: WordCloud in c:\python27\lib\site-packages
Requirement already satisfied: numpy>=1.6.1 in c:\python27\lib\site-packages (from WordCloud)
Requirement already satisfied: matplotlib in c:\python27\lib\site-packages (from WordCloud)
Requirement already satisfied: pillow in c:\python27\lib\site-packages (from WordCloud)
Requirement already satisfied: six>=1.10 in c:\python27\lib\site-packages (from matplotlib->Wo
Requirement already satisfied: pytz in c:\python27\lib\site-packages (from matplotlib->WordClou
Requirement already satisfied: cycler>=0.10 in c:\python27\lib\site-packages (from matplotlib-
Requirement already satisfied: python-dateutil in c:\python27\lib\site-packages (from matplotl
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=1.5.6 in c:\python27\lib\sit
Requirement already satisfied: functools32 in c:\python27\lib\site-packages (from matplotlib->
Requirement already satisfied: olefile in c:\python27\lib\site-packages (from pillow->WordClou
```

```
In [22]: def showWordCloud(text_files):
             from os import path
             from wordcloud import WordCloud, STOPWORDS

             # Read the whole text.
             text = ""
             for chapter in text_files:
                 text = open(chapter).read() + text

             # Add stopwords
             stopwords=set(STOPWORDS)
```

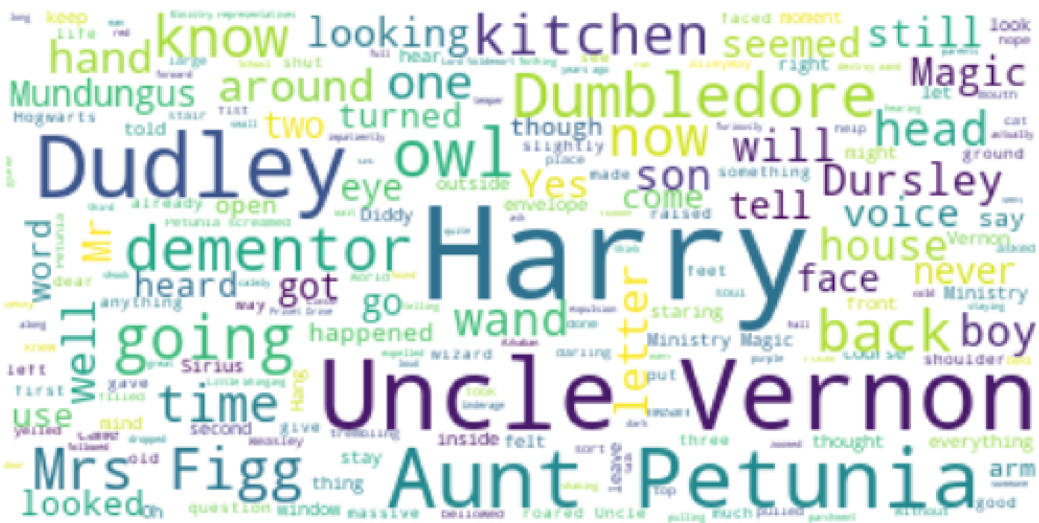
```
stopwords.add("said")

# Generate a word cloud image
wordcloud = WordCloud(stopwords=stopwords,max_font_size=50,background_color="white

# Display the generated image:
# the matplotlib way:
import matplotlib.pyplot as plt
plt.figure( figsize=(10,5) )
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

Horcrux Topic

```
In [23]: showWordCloud([files[max_probability_horcrux_chapter-1]])
```

Dudley Topic

```
In [24]: showWordCloud([files[max_probability_dudley_chapter-1]])
```



Harry Potter Serie

```
In [25]: showWordCloud(files)
```

3.3 Conclusion

I personally thought topic modelling worked really well on the Harry Potter serie. It was really interesting and stimulating to do it on this book as for people who know the universe of Harry Potter, the topics discovered make a lot of sense. I personally enjoyed this lab a lot. I would then have been interesting to test the influence in the number of documents for LDA. Indeed, I could qualitatively see that the topics were much more relevant with more documents (more books and chapters). I started only studying the Book 1 but ended adding them all as topics were becoming more and more relevant. The preprocessing part took a lot of time in this lab as the data had to be built from scratch. Tweaking the LDA to have relevant topics took also a lot of time as the hyperparameters must be well set. Considering is is a randomized algorithm, I tried changing the random seed for a long time. Thanks for reading this lab !