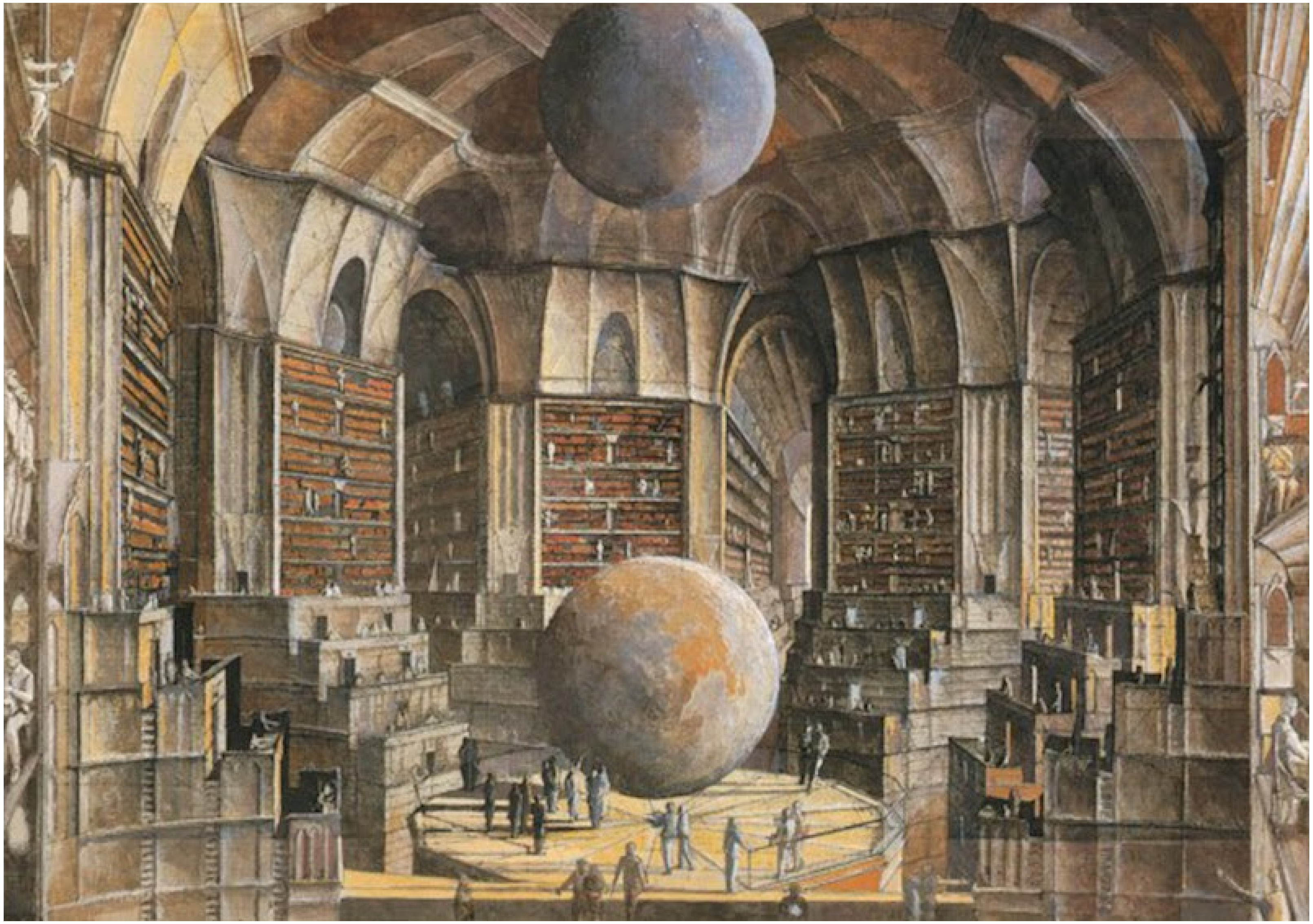
Session 15

Text as Data 1 - Supervised Learning and Text Classification

Snorre Ralund

Motivation



Information about what people do, think, and feel lies embedded in text.

Much of our social activity and communication is now done natively via text.

The internet is full of information stored in text, and textual traces range from social media, emails and instant messaging, to automatically transcribed videos from youtube or memos from doctors, goverment records and digitized libraries.

"This vast supply of text has broad demand for natural language processing and machine learning tools to filter, search, and translate text into valuable data. And it has broad exciting opportunities for social scientists who knows how to get and handle text as data." **- James Evans, University of Chicago, Director of Masters Programme in Computational Social Science**

Projects(1)

Using supervised machine learning or dictionary methods to measure Phenomena in text

Projects:

- Gender Bias in Newspapers?
  - What about Gender and Emotions in newspapers?
  - Length of Reviews and gender?

Own Projects:

- Studying ethnic and political polarization.
  - Measuring relational insult types and the saliency of these accross ethnicities in Denmark accross time using supervised learning on large scale social media data.
- Analyzing developments in the Jobmarket.
  - Datadriven, and using a dictionary approach, developed using a semiautomated search algorithm to generate it.
- Conflict discourse and politization in Social Movements. Supervised learning.
- Using it as a way of validating a proxy of friendship. Textual features to qualify what a tag on facebook means.
- Analyzing growing Populism in Danish Politics. Measuring topics using a validated dictionary.

Projects(2)

Using techniques from information extraction, data mining and Natural Langauge Processing to discover, characterize and extract textual features used in a

Projects:

```
* Predicting bitcoin prices... Extracting textual features and using those as in
put to a predictor.
   - pretty hard´task.
   - however take a look at this paper: https://hobbs.human.cornell.edu/SensorsI
CWSM17.pdf

* Growing Sensationalism, and dumbing down the news? Measuring complexity of the
 text (LIX-number) and length of title as click bate indicator.
    * What about Emotions in news. How has it changed on DR?
    * Do they cite more sources within the text (or do they just advertise their
 own content)?
```

Research Examples

- Chen 2013: *"The effects of language on Economic Behaviour - Evidence from Savings Rates, Health Behaviors, and Retirement Assets"*
  - Quantifying the use of future tense in different languages and its connection to future oriented behavior: saving for retirement, practicing safe sex etc.
- Gentzkow and Shapiro 2007: *"What drives media slant? Evidence from U.S daily newspapers"*
  - Measuring slant as document distances to the words of different political actors.
- Danescu-Niculescu-M et. al 2012 *"Echoes of Power: Language Effects and Power Differences in Social Interaction"* (https://www.cs.cornell.edu/%7Ecristian/Echoes_of_power_files/echoes_of_power.pdf)
  - Measuring power dynamics as imitation of linguistic style. Where style is the use of word classes: *"articles, auxiliary verbs, conjunctions, high-frequency adverbs, impersonal pronouns, personal pronouns, prepositions, and quantifiers"*, that can be automatically extracted.

Agenda

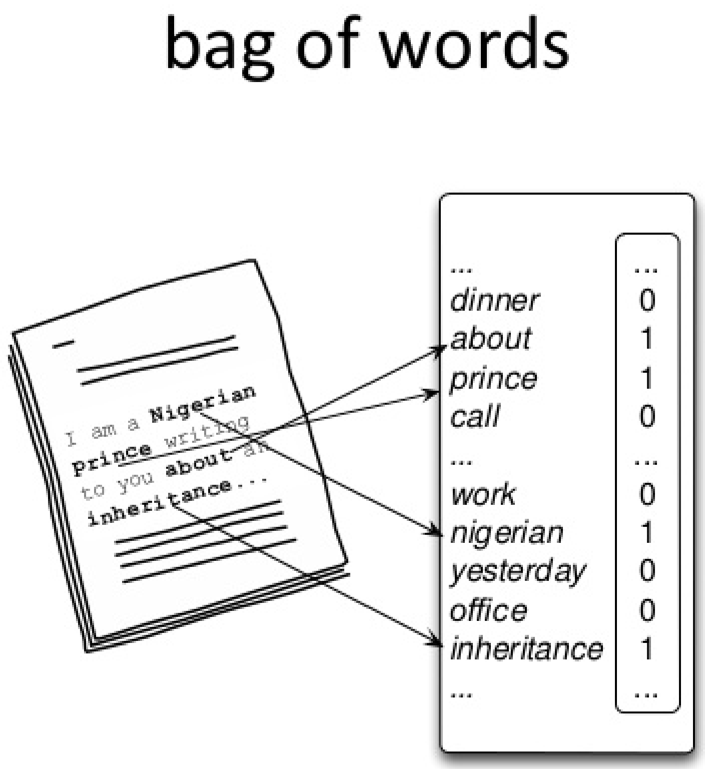
- Applying Supervised learning to text classification.
- From Text to Vector.
  - Bag of Words.
  - Term frequency Inverse Document Frequency
  - Ngrams
  - Tokenization
  - Text normalization

```
In [14]: # Packages
         import nltk # general framework
         #nltk.download('all') # Uncomment this and run
         import re # Regular expressions
         import gensim # For unsupervised learning and text
         # import spacy # A lot of Natural language processing capabilities (NER, POS, Sentiment
          analysis)
         # stanford core nlp written in java but you can also use it in python.
         ## MORE import polyglot for multilanguage
```

```
C:\Users\jbv933\AppData\Local\Continuum\anaconda3\lib\site-packages\gensim\utils.py:1
209: UserWarning: detected Windows; aliasing chunkize to chunkize_serial
  warnings.warn("detected Windows; aliasing chunkize to chunkize_serial")
```

Getting from Text to vector

Our supervised models do not know how to use the .split function or what to search for using regular expressions.



Credit Byron C Wallace

Vector representation

Bag of Words (BoW) or Document Term Matrix

- Throw out the word order.
- And let each word be a feature. --> map word to an index in a matrix.

doc1: "i really love bacon"

doc2: "i really don't like bacon"

As a Document Term Matrix

document	**really**	**i**	**love**	**bacon**	**don't**	
doc1	0	0	1	1	0	
doc2	1	1	0	1	1	

Bag of Words(2) - Problem with polesemy

Consider the following to documents:

doc1: *"River A/S declared default by the bank."*

doc2: *"When camping my default is by the river bank."*

As a Document Term Matrix

document	**declared**	**by**	**default**	**bank**	**river**	**a/s**	**when**	**camping**	**my**
doc1	1	1	1	1	1	1	0	0	0
doc2	0	1	1	1	1	0	1	1	1

Bag of Words (3) - Lack of word order

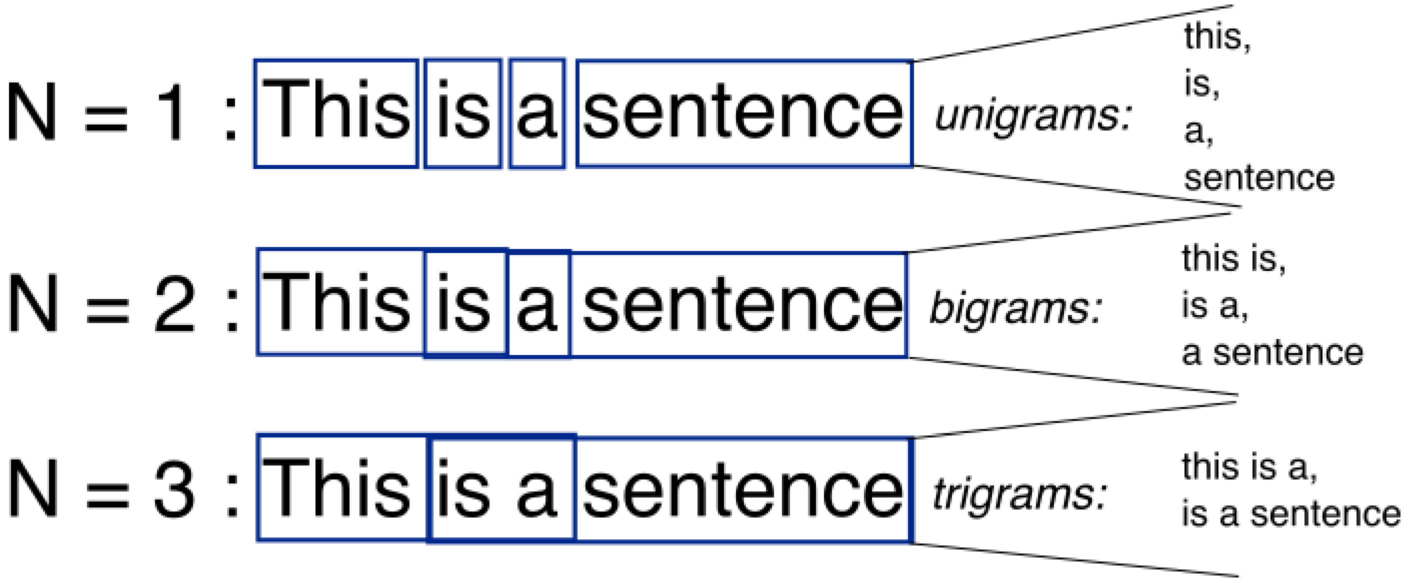
doc1 = 'this was not the best movie'

doc2 = 'was this not the best movie ever?'

Will have very similar representations.

Ngrams to the Rescue

Word Ngram



Character Ngram: For people who do not like to tokenize.

Ngrams to the Rescue(2)

Problem with dimensionality Quad-grams Qvint Grams etc. Generates exponential number of features.

Solution Pick only the ngrams using statistical analysis of the word co-occurences. Check out methods for doing so in the Natural Language Processing Toolkit package nltk: `nltk.collocations`

Bag of Words features as input to Baseline models

- Baseline models: Naive Bayes, Logistic Regresion, K-nearest Neighbor and Support Vector Machines

The seem to work pretty well for many tasks: Wang and Manning (2015): *"Baselines and Bigrams: Simple, Good Sentiment and Topic Classification"*

Baseline Models(2) - Understanding the task at hand.

- Always run simple BoW model first. Check if you need a more complex model, and evaluate models based on gains in relation to a bag-of-word model.

- How hard is the learning task?
    - How well "linear" and BoW models do.
        - Learning what can be learned from atomized words alone tells you alot about the task at hand.
    - Look at examples of where they fail.
    - How much data you need. Learning curve.

From text to vector(2) - Tokenization

In [29]:
```
# simple example
string = 'I really love bacon :)'
string.split()
#import re
#token_re = re.compile('\w+')
#token_re.findall(string)

#token_re = re.compile('\w+|:\)')
#token_re.findall(string)
import nltk
# nltk.word_tokenize(string) # problem with emojies
import nltk.sentiment
# nltk.sentiment.vader.SentiText(string).words_and_emoticons # removal of stopwords
```

Lets see how it does on a real dataset.

```
In [ ]: import pandas as pd
        df = pd.read_csv('https://raw.githubusercontent.com/snorreralund/scraping_seminar/maste
        r/english_review_sample.csv') # Load data
```

In [31]:
```
tokenizers = {'nltk':nltk.word_tokenize,'regex':lambda x: token_re.findall(x)
              ,'nltk_sentiment':lambda x :nltk.sentiment.vader.SentiText(x).words_and_em
oticons,'.split':
             lambda x: x.split()}
from collections import Counter
string = ' '.join(df.reviewBody.values[0:1000])
for name,tokenize in tokenizers.items():
    print(name,len(Counter(tokenize(string))))
```

```
nltk 5282
regex 5075
nltk_sentiment 5489
.split 6965
```

From Tokens to Vectors

Bag of Words - Count Vectorizer

In [34]:

```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CounterVectorizer() # handles preprocessing and tokenization by default
```

Quick digression - Problem of Large feature spaces.

- Stopword removal ['i','you','and','or'].'nltk.stopwords
- Stemming and Lemmatizing:
    - Reducing a word to its basic form. Walking --> walk. Are --> be. nltk.stem,nltk.lemmatize

TFIDF - Term Frequency Inverse Document Frequency

A clever weighing scheme (especially used in unsupervised learning)

This is essentially removes the problem of irrelevant words, that are shared by all documents. And puts weight on the defining word of a document.

It does this usign the following formula:

$$IDF = \log \frac{N}{n_t}$$

$$TF = \frac{c_{t_i,d}}{d_c}$$

where

N is the number of documents.

n_t is the number of documents with the token present

$c_{t,d}$ is the is the number of times a token t is present in d

c_d is the number of tokens in document

```
In [36]: from sklearn.feature_extraction.text import TfidfVectorizer
         vectorizer = TfidfVectorizer() # handles preprocessing and tokenization by default
         X = vectorizer.fit_transform(df.reviewBody.values[0:50])
         X
```

```
Out[36]: <50x677 sparse matrix of type '<class 'numpy.float64'>'
                 with 1315 stored elements in Compressed Sparse Row format>
```

And now it is your turn. Go to Exercise Set 15 and use this as input to a classifier.