Topic Models

Max Callaghan

2022-11-03

Introduction and Objectives

Summary of feedback

95.5% of respondents said they had learnt *something* about text as data!

Throughout the rest of the course I will try to provide

- More examples from real research
- Materials before Thursday morning!
- Exercises each non-homework week
- No more poems!

Homework Assignment 1

Marks have been communicated and solutions posted.

Each section was marked independently, with 100 points for any answer that got the job done, with 5 points removed for every minor issue, and 10 points removed for larger issues.

The median grade was slightly higher than our target, expect slightly harsher marks next time!

Objectives

By now we have figured out how to **represent** texts in simple and more complex ways

In this session we will start asking **questions** about our corpora, based on a simple, per-text question:

What is this text about?

Topics are the what.

If we can answer this about a corpus of text, we can analyse how different groups of texts use different topics, or how the use of different topics changes over time.

Topic models

The simplest version of topic models

Non-negative Matrix Factorization (NMF) (Lee and Seung, 1999) is a simple and effective method of learning topic models.

Essentially, it is a form of **dimensionality reduction**, in that it takes our multidimensional document-feature matrix V, and creates two matrices W and H, which help us to describe it.

$$V_{i\mu} \approx (WH)i\mu = \sum_{a=1}^{r} W_{ia} H_{a\mu}$$

In this equation, r is the number of topics, W is a matrix describing the score for each topic in each document, and H is a matrix describing the score for each word in each topic.

A simplified NMF problem

Let's take a simple corpus of documents, which we turn into a document feature matrix

```
from sklearn.feature_extraction.text import CountVectorizer
texts = [
    "Cats, elephants, camels",
    "Hawks doves pigeons",
    "Cats pigeons"
]
vec = CountVectorizer()
dfmat = vec.fit_transform(texts)
```

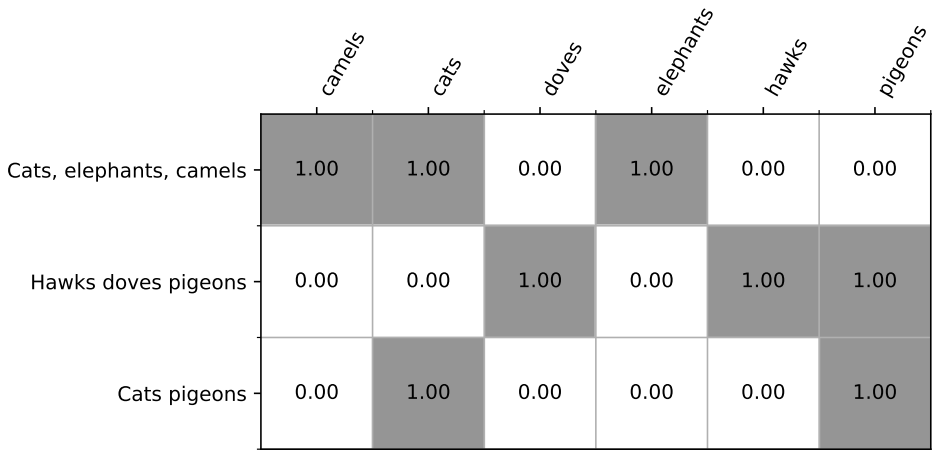
A simplified NMF problem

We'll also write a function to turn a matrix into a heatmap with x and y labels

```
import matplotlib.pyplot as plt
from matplotlib.colors import Normalize
import numpy as np

def plot_heatmap(X, xlabs, ylabs):
    fig, ax = plt.subplots()
    ax.imshow(
        X, cmap = "Greys",
        norm = Normalize(vmin=0, vmax=X.max()*2)
    )

    # Create a grid using minor ticks
    ax.set_xticks(np.arange(X.shape[1])+0.5, minor=True)
    ax.set_yticks(np.arange(X.shape[0])+0.5, minor=True)
    ax.grid(which="minor", zorder=5)

    # Set up x labels
    ax.xaxis.tick_top()
    ax.set_xticks(np.arange(X.shape[1]))
    ax.set_xticklabels(xlabs, rotation=60, ha="left", va="bottom")

    # Set up y labels
    ax.set_yticks(range(len(ylabs)))
    ax.set_yticklabels(ylabs)

    # Put the numbers in
    for m in range(X.shape[0]):
        for n in range(X.shape[1]):
            ax.text(n, m, f"{X[m, n]:.2f}", ha="center", va="center")
```

A simplified NMF problem

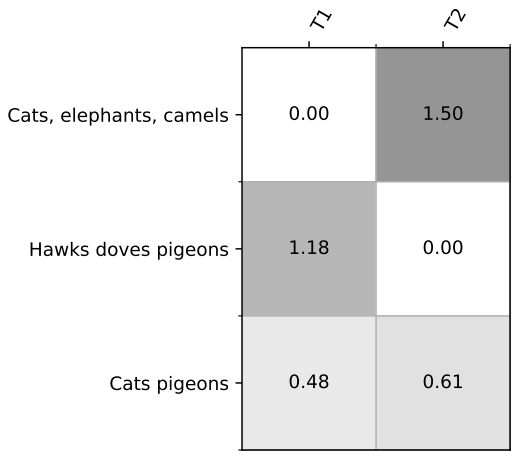
Let's take a simple corpus of documents, which we turn into a document feature matrix



Fitting an NMF model

To fit an NMF model, we just initialise an NMF instance from sklearn, and apply the `fit_transform()` method to our document feature matrix.

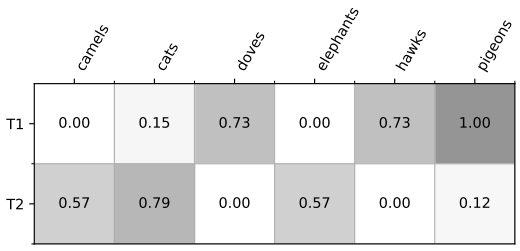
```
from sklearn.decomposition import NMF
nmf = NMF(2)
W = nmf.fit_transform(dfmat)
plot_heatmap(W, ["T1","T2"], texts)
plt.savefig("plots/W.pdf", bbox_inches="tight")
```



Inspecting the topics

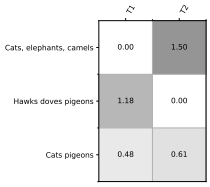
The Topic-word scores are contained in the matrix H, which we can access in the `components_` attribute of our nmf instance

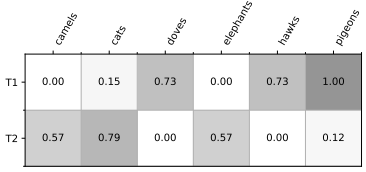
```
H = nmf.components_
plot_heatmap(
  H,
  vec.get_feature_names_out(),
  ["T1","T2"]
)
plt.savefig("plots/H.pdf", bbox_inches="tight")
```

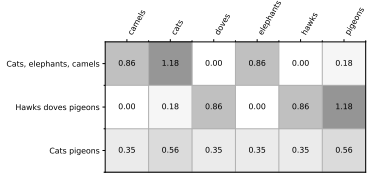


Combining the parts to approximate the whole

The dot product of W and H should give us a matrix that is similar to our document feature matrix



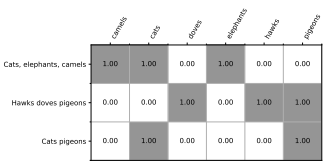


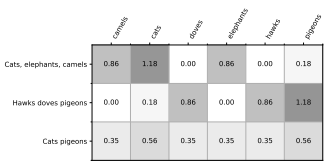


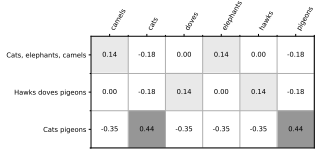

```
WH = W.dot(H)
plot_heatmap(WH, vec.get_feature_names(), texts)
plt.savefig("plots/WH.pdf", bbox_inches="tight")
```

Combining the parts to approximate the whole

The difference between the real document-feature matrix and the product of our document-topic and topic-word matrices is what our algorithm tries to minimize.








```
error = V - WH
plot_heatmap(error, vec.get_feature_names(), texts)
plt.savefig("plots/error.pdf", bbox_inches="tight")
```

Hyperparameters for NMF

What are hyperparameters?

Hyperparameters are options that need to be set or left as default that are not strictly set by theory, but affect the result.

It is useful to check different values of these

- TFIDF (usually a good idea)
- numer of topics
- alpha_W - low values indicate documents should be composed of few topics
- alpha_H - low values indicate topics should be composed of few words

Latent Dirichlet Allocation

Latent Dirichlet Allocation (Blei, Ng, and Jordan, 2003) does a very similar job, the way it gets there and the assumptions it makes are just slightly different.

It also has a hyperparameter α that encodes your prior about how many topics are present in a document, and a β parameter that encodes your prior about how many words are present in a topic (not settable with topicmodels!).

Some nice features of these models

- Documents are mixtures of topics
- The model outcome is a matrix of documents x topics, which is in exactly the format of all the other representations we have been looking at
- Our features are small in number, and interpretable by humans

Some limitations

Topics are very fuzzily defined and may mean different things in different contexts or even within the same model.

Suppose we have a topic model of research where the largest topic is **climate change**, and there are 3 smaller topics on **cancer**, **asthma**, and **heart disease**. Can we say that most research is on climate change?

Topic models can be sensitive to the dataset, if you only have few samples from a certain subclass, expect the model to fit these poorly.

Naming topics

A quick heuristic for naming topics is to concatenate the top 3 terms for each topic.

If we want to use our model then we should give meaningful names to topics by inspecting the top terms and top documents associated with each topic.

Extensions

Structural Topic models

We often compare the prevalence of topics in documents of different types.

Structural Topic Models incoporate additional variables into the modelling process, and allow us to test hypotheses with statistical significane about how the prevalence of topics varies with document characteristics.

STM is only available in R!! It is used a lot by political scientists and was invented by those who wrote the textbook on Text as Data.

Dynamic Topic Models

Dynamic topic models incorporate time into the modelling process. Dynamic Topic Models by from the Blei lab (Github) does this by allowing the words associated with topics to update in each time period.

I haven't come across any satisfying implementations that allow **new** topics to emerge..

Topics from embeddings

The embedding space (into which we dipped our toes last session) is where a lot of things are happening in NLP these days.

Top2Vec, and BERTopic work by identifying clusters of documents in an embedding space (potentially generated by the latest fancy language models), and then finding the words that are common in that cluster of documents.

One drawback is that documents can only be assigned to a *single* topic.

Validation

Validating topic models

Creating a topic model is a bit like magic. You feed in a list of documents, and they come out the other side with topics that seem to make sense.

But can you just use these topics unreflectingly? How can you be sure if this topic model was the "best" topic model?

We are estimating **latent** (that is, hidden) variables, which means that unlike with supervised learning, there is no easy way to measure the "rightness" of the model.

Data-driven measures of topic rightness

We have two frequently used measures of topic quality that capture *some* aspects of topic quality

Loss/heldout-likelihood based measures work by comparing the topic **predictions** of words in documents to the actually observed numbers of words in documents.

Coherence based measures work by assessing whether the words in a topic are similar. If words in a topic only infrequently co-occur in documents, then this topic is considered less coherent.

Human measures

Can the "best" topic model be found by maximising a metric? Or is this more a case of interpretation?

Chang, Boyd-Graber, and Blei, 2009 find that data-driven measures do not necessarily correspond to human evaluations.

They propose 2 human annotation methods for topics.

In **word intrusion** humans are presented with a set of words, of which all but one are from a topic.

In **topic intrusion** humans are presented with a document and a set of topics of which all but one are related to the document.

Where humans consistently identify the odd one out, we can say that the topic model is performing well.

What is a good topic anyway?

The definition of a "good" topic is not universal but task dependent.

A good topic model is one that helps us to answer the research question we have in mind, or helps us to better perform the task we have.

Sometimes we want the big picture (few topics), sometimes we want fine-grained detail (many topics).

What we should **ensure** is that any results we present are not an **artefact** of an arbitrary model choice we make, but are robust to a variety of *reasonable* specifications.

Scaled up examples

Topic modelling with R

Let's load our old Hertie research dataset and see if we can dig deeper into what has been researched at this school. We'll start by preparing a document feature matrix

```
library(quanteda)
library(readr)
library(dplyr)
df <- read_csv("../datasets/hertie_papers.csv")
df <- df %>% filter(!is.na(abstract) & !is.na(publication_year)) %>%
  distinct(abstract, .keep_all = TRUE) # remove duplicates
dfmat <- df$abstract %>%
  tokens(remove_punct = T) %>%
  tokens_remove(pattern=stopwords("en")) %>%
  dfm()  %>%
  dfm_trim(min_termfreq = 5) # Remove infrequent terms

rownames(dfmat) <- df$id # use meaningful names

dfmat
```

```
## Document-feature matrix of: 1,098 documents, 3,461 features (98.34% sparse) and 0 docvars.
##                                  features
## docs                              > 50 limiting warming 2 ° c recent scenarios
##   https://openalex.org/W2195453830 1  1        1       1 1 1 1      1         1
##   https://openalex.org/W18536190   0  0        0       0 0 0 0      0         0
##   https://openalex.org/W2092902022 0  0        0       0 0 0 0      0         0
##   https://openalex.org/W2041842081 0  0        0       0 0 0 0      0         0
```

Running an LDA model

We can simply pass this to the LDA function of topic models to learn our first model, we just need to tell it how many topics we want

```
library(topicmodels)
lda <- LDA(dfmat, 15)
```

Using LDA output

Our document-topic matrix is stored in the "gamma" attribute, and our topic-term matrix is stored in the "beta" attribute.

We can turn these into "tidy" data with tidytext

```
library(tidytext)
print(dim(lda@gamma))
```

```
## [1] 1098   15
```

```
print(dim(lda@beta))
```

```
## [1]   15 3461
```

```
topic_words <- tidy(lda, matrix="beta") %>%
  group_by(topic) %>%
  slice_max(beta, n = 5) %>%
  ungroup() %>%
  arrange(topic, -beta)
topic_words
```

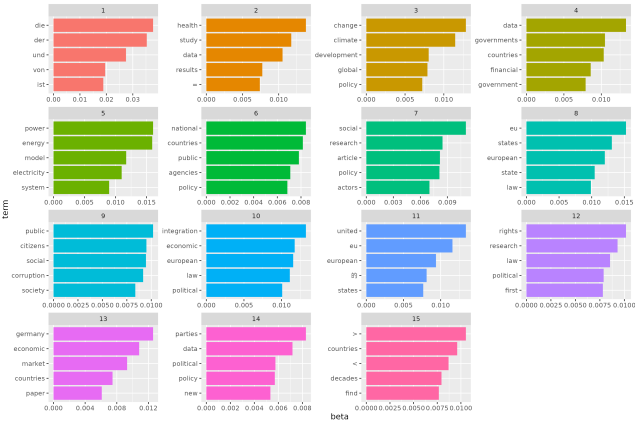
```
## # A tibble: 75 x 3
##    topic term       beta
##    <int> <chr>     <dbl>
##  1     1 die     0.0377
##  2     1 der     0.0353
##  3     1 und     0.0274
```

Plotting the top words of each topic

We can plot the top words of each topic using this tidy data and ggplot()

```
library(ggplot2)
topic_words %>%
  mutate(term = reorder_within(term, beta, top
  ggplot(aes(beta, term, fill = factor(topic))
  geom_col(show.legend = FALSE) +
  facet_wrap(~ topic, scales = "free") +
  scale_y_reordered()
```

```
ggsave("plots/top_terms_r.png", width=12, heig
```



Using the doc-topic data

We can also inspect the share of a topic in a particular group of documents by tidying our doc-topic data. In this case we get the share of documents on a topic in each year

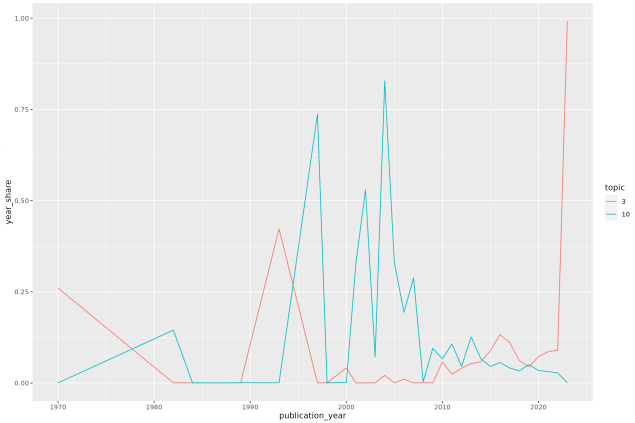
```
doc_topics <- tidy(lda, matrix="gamma") %>%
  left_join(df, by=c("document"="id"))

yearly_topics <- doc_topics %>%
  group_by(publication_year, topic) %>%
  summarise(gamma = sum(gamma)) %>%
  group_by(publication_year) %>%
  mutate(year_share = gamma/sum(gamma)) %>%
  ungroup() %>%
  mutate(topic = factor(topic))

yearly_topics
```

```
## # A tibble: 480 x 4
##    publication_year topic    gamma year_share
##               <dbl> <fct>    <dbl>      <dbl>
##  1             1970 1     0.000318   0.000318
##  2             1970 2     0.736      0.736
##  3             1970 3     0.260      0.260
##  4             1970 4     0.000318   0.000318
##  5             1970 5     0.000318   0.000318
##  6             1970 6     0.000318   0.000318
##  7             1970 7     0.000318   0.000318
```

Using the doc-topic data

```
ggplot(filter(yearly_topics, topic %in% c(3,10
    x=publication_year, y=year_share,
    group=topic, colour=topic, fill=topic
  )) +
  geom_line()
```

```
ggsave("plots/topic_groups.png", width=12, hei
```



An interactive site for exploring a topic model

LDAvis is a nice package for producing interactive visualisations of our topic models. These are really helpful for exploring the model and understanding how it has worked

To use this with the `topicmodels` package, we just need to slightly transform our default output data, we can reuse this function from the internet.

```
library(LDAvis)
topicmodels2LDAvis <- function(x, ...){
  post <- topicmodels::posterior(x)
  if (ncol(post[["topics"]]) < 3) stop("The model must contain > 2 topics")
  mat <- x@wordassignments
  json <- LDAvis::createJSON(
    phi = post[["terms"]],
    theta = post[["topics"]],
    vocab = colnames(post[["terms"]]),
    doc.length = slam::row_sums(mat, na.rm = TRUE),
    term.frequency = slam::col_sums(mat, na.rm = TRUE)
  )
  return(json)
}
json <- topicmodels2LDAvis(lda)
serVis(json)
```

Topic modelling in python + a topic "map" in an embedding space

In the notebook tms.ipynb, you will find similar functions, including an extension that displays the documents in their topic space, using UMAP to represent this in 2 dimensions.

Exercise

In pairs, independently come up with your own specification of the same topic model by altering the parameters (including in pre-processing).

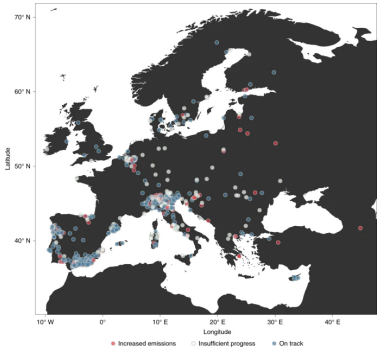
Discuss which model is better, or has more "truthiness"

Topic models in research

Performance determinants show European cities are delivering on climate mitigation

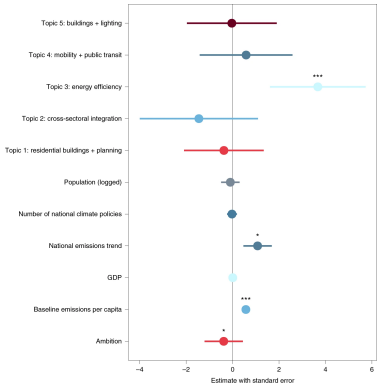
In Hsu et al., 2020, the authors use topic modelling to inspect a corpus of 1,066 European cities' Sustainable Energy and Climate Action Plans.

They use emissions data to figure out which cities are on track to meeting emissions reductions targets, and investigate what is mentioned more in successful plans.



Performance determinants show European cities are delivering on climate mitigation II

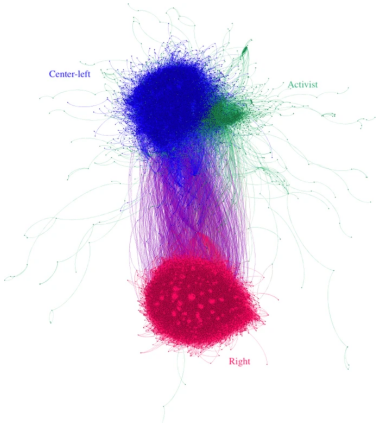
Using a Structural Topic Model, they find that increased prevalence of the topic on energy efficiency is associated with a significant reduction in emissions.



Attention and counter-framing in the Black Lives Matter movement on Twitter

In Klein et al., 2022, the authors use topic modelling to explore a corpus of 118 million tweets about Black Lives Matter

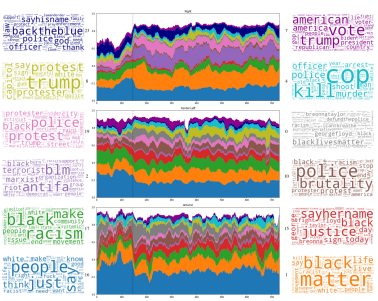
- They cluster a retweet network in order to find distinct communities



Attention and counter-framing in the Black Lives Matter movement on Twitter

In Klein et al., 2022, the authors use topic modelling to explore a corpus of 118 million tweets about Black Lives Matter

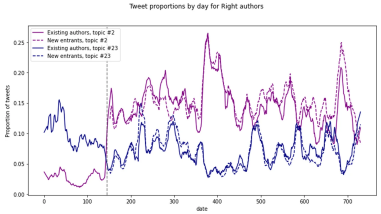
- They cluster a retweet network in order to find distinct communities
- They use LDA to identify topics, and show how number of tweets referring to these topics has changed over time



Attention and counter-framing in the Black Lives Matter movement on Twitter

In Klein et al., 2022, the authors use topic modelling to explore a corpus of 118 million tweets about Black Lives Matter

- They cluster a retweet network in order to find distinct communities
- They use LDA to identify topics, and show how number of tweets referring to these topics has changed over time
- In particular, they identify a sharp increase in the "antifa terrorist" topic in the immediate aftermath of George Floyd's death



Wrapup and Outlook

Wrapup

We have done machine learning!

We have now explored techniques used in real contemporary computational science research to investigate *what* texts are about.

Like many of our applications so far, our model forms a matrix with each document in a row, and features in a column. In our case, these features are of a manageable size defined by us, and they are **interpretable** with the help of another matrix which maps our features to the terms associated with them

Assignment 2

Assignment 2 is now live, and due on 17. November 23:59. Please submit on Moodle.

You are asked to use a topic model to answer a simple research question about the content of manifestos. For maximum *possible* marks, this should be **interesting** and **answerable**. For maximum likelihood of **good** marks, focus on the **answerable** part.

This time, there are more skills being assessed, not just coding but also **understanding**, **communication**, and **research** skills.

Comment on everything you are asked to comment on, and answer every question you are asked!!.

A mostly wrong answer is better than no answer at all.

Remember to post issues and to ask me when something is not clear or if you are struggling!

Good luck!

Assignment 3

Assignment 3 is due for the last week of term. Time to start thinking about it.

Assignment 2 is structured along the lines of answering a research question. This should prepare you for your group project where you will do just that.

In around 10 minutes, you will need to present

- Your research question and why it is interesting
- Your data and methods used to answer it
- The results of your analysis, your interpretation of these and your limitations and ideas for further work

Get into groups of 3-4. Consider joining forces across programmes, you will need complementary skills!

Please email me your group and your proposed topic by the end of **week beginning 14 November**.

Next week

Next week we will look at sentiment analysis, where we score texts according to whether they express positive or negative sentiment.