Topic Model Network Visualization

Lynn Cherny (@arnicas)
lynn@ghostweather.com

August 2014

WHAT IS TOPIC ANALYSIS?

Problems We're Attacking

- Document collections are hard work to explore/manage manually
- Sometimes the contents are completely or mostly "unknown" (e.g., an email archive, or a collection of research papers)
- We'd like at least semi-automated methods to group them, annotate them, explore relationships

The Topic Problem

Text 1

We present a statistical parsing framework for sentence-level sentiment classification in this article. Different from previous work employing linguistic parsing results for sentiment analysis, we develop a statistical parser to directly analyze the sentiment structure of a sentence. We show that the complicated phenomena in sentiment analysis (e.g., negation, intensification, and contrast) can be elegantly handled the same as simple and straightforward

Text 2

Sentiment analysis of Twitter data is performed. The researcher has made the following contributions via this paper: (1) an innovative method for deriving sentiment score dictionaries using an existing sentiment dictionary as seed words is explored, and (2) an analysis of clustered tweet sentiment scores based on tweet length is performed.

Text 3

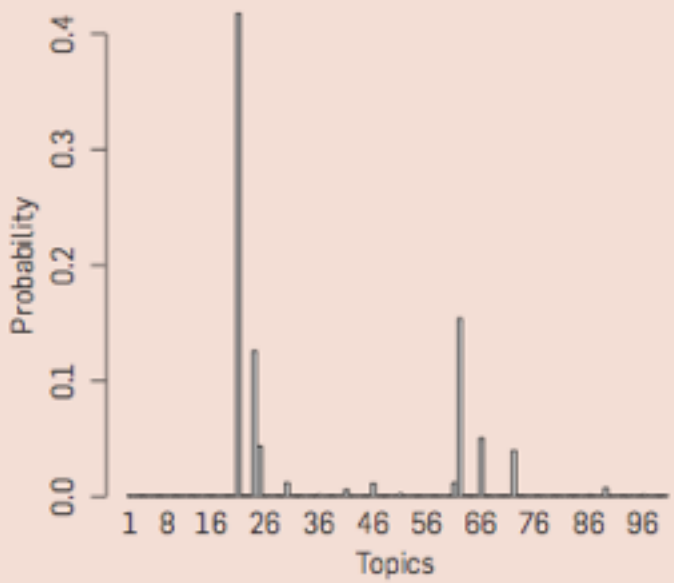
We perform a large-scale linguistic analysis of language diatopic variation using geotagged microblogging datasets. By collecting all Twitter messages written in Spanish over more than two years, we build a corpus from which a carefully selected list of concepts allows us to characterize Spanish varieties on a global scale. A cluster analysis proves the existence of well defined macroregions sharing common lexical properties

Intuitions

- Documents are composed of multiple words ("bag of words"). Documents may express multiple topics, using those words.
- Topics are considered unknown in advance — or "latent" — this is the problem we are trying to solve for.

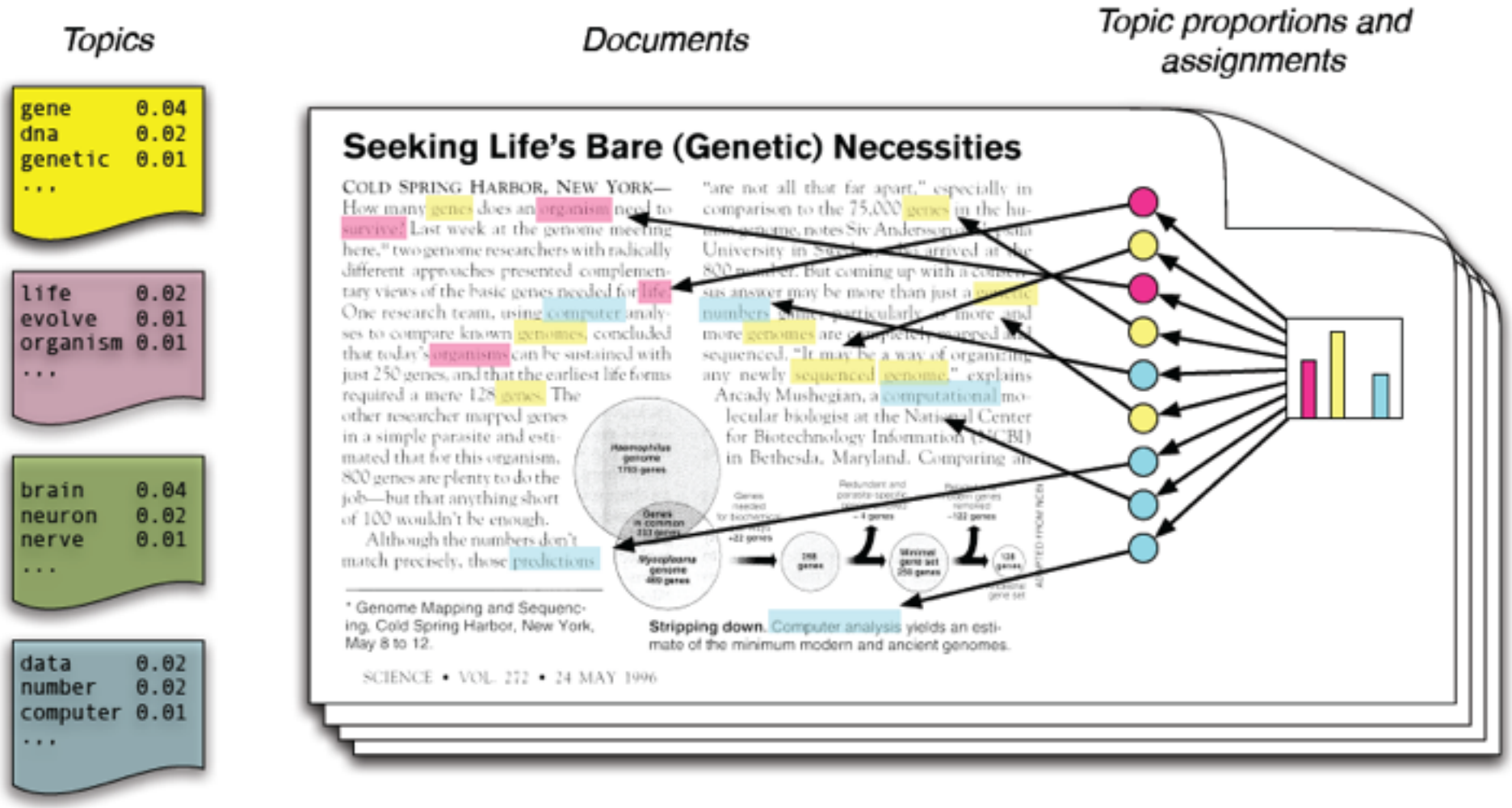
"LDA": Latent Dirichlet Allocation

Figure 2. Real inference with LDA. We fit a 100-topic LDA model to 17,000 articles from the journal *Science*. At left are the inferred topic proportions for the example article in Figure 1. At right are the top 15 most frequent words from the most frequent topics found in this article.

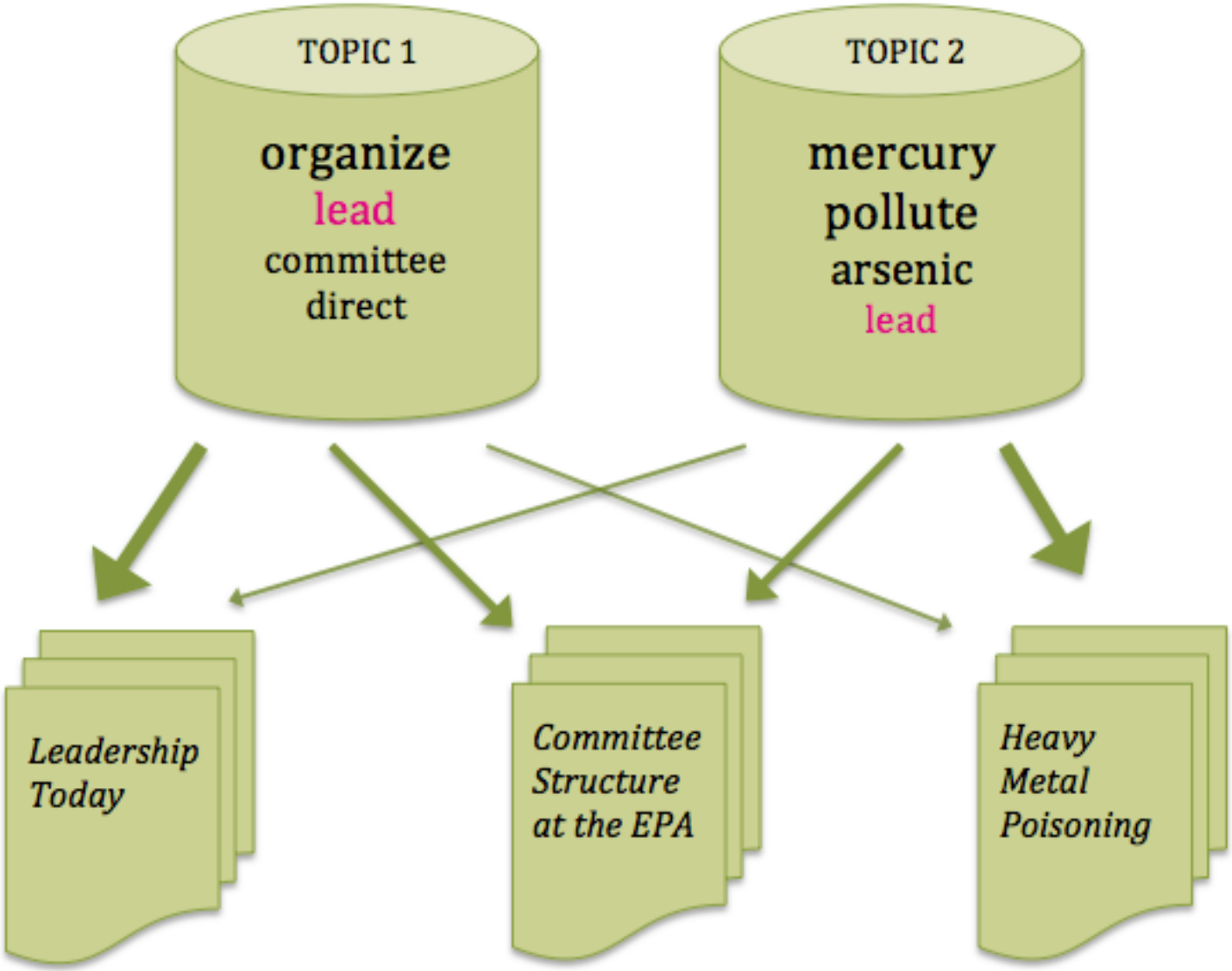


"Genetics"	"Evolution"	"Disease"	"Computers"
human	evolution	disease	computer
genome	evolutionary	host	models
dna	species	bacteria	information
genetic	organisms	diseases	data
genes	life	resistance	computers
sequence	origin	bacterial	system
gene	biology	new	network
molecular	groups	strains	systems
sequencing	phylogenetic	control	model
map	living	infectious	parallel
information	diversity	malaria	methods
genetics	group	parasite	networks
mapping	new	parasites	software
project	two	united	new
sequences	common	tuberculosis	simulations

Blei(2012) review article: http://www.cs.princeton.edu/~blei/papers/Blei2012.pdf


Topics
Documents
Topic proportions and assignments
gene 0.04
dna 0.02
genetic 0.01
...
life 0.02
evolve 0.01
organism 0.01
...
brain 0.04
neuron 0.02
nerve 0.01
...
data 0.02
number 0.02
computer 0.01
...
Seeking Life's Bare (Genetic) Necessities
COLD SPRING HARBOR, NEW YORK—How many genes does an organism need to survive? Last week at the genome meeting here,* two genome researchers with radically different approaches presented complementary views of the basic genes needed for life. One research team, using computer analyses to compare known genomes, concluded that today's organisms can be sustained with just 250 genes, and that the earliest life forms required a mere 128 genes. The other researcher mapped genes in a simple parasite and estimated that for this organism, 800 genes are plenty to do the job—but that anything short of 100 wouldn't be enough.
Although the numbers don't match precisely, those predictions "are not all that far apart," especially in comparison to the 75,000 genes in the human genome, notes Siv Andersson of Uppsala University in Sweden, who arrived at the 800 number. But coming up with a consensus answer may be more than just a semantic numbers game, particularly as more and more genomes are completely mapped and sequenced. "It may be a way of organizing any newly sequenced genome," explains Arcady Mushegian, a computational molecular biologist at the National Center for Biotechnology Information (NCBI) in Bethesda, Maryland. Comparing an
* Genome Mapping and Sequencing, Cold Spring Harbor, New York, May 8 to 12.
Stripping down. Computer analysis yields an estimate of the minimum modern and ancient genomes.
SCIENCE • VOL. 272 • 24 MAY 1996

Blei (2011)



http://tedunderwood.com/2012/04/07/topic-modeling-made-just-simple-enough/

GETTING STARTED

Python Stuff you need

Python 2.7: ideally a virtual environment or install that includes libraries IPython notebook, nltk, pattern, pandas, numpy, networkx

1. Install miniconda: http://conda.pydata.org/miniconda.html.
2. Then:

```
>conda create -n topic_workshop ipython-notebook
        pandas numpy nltk pip
```

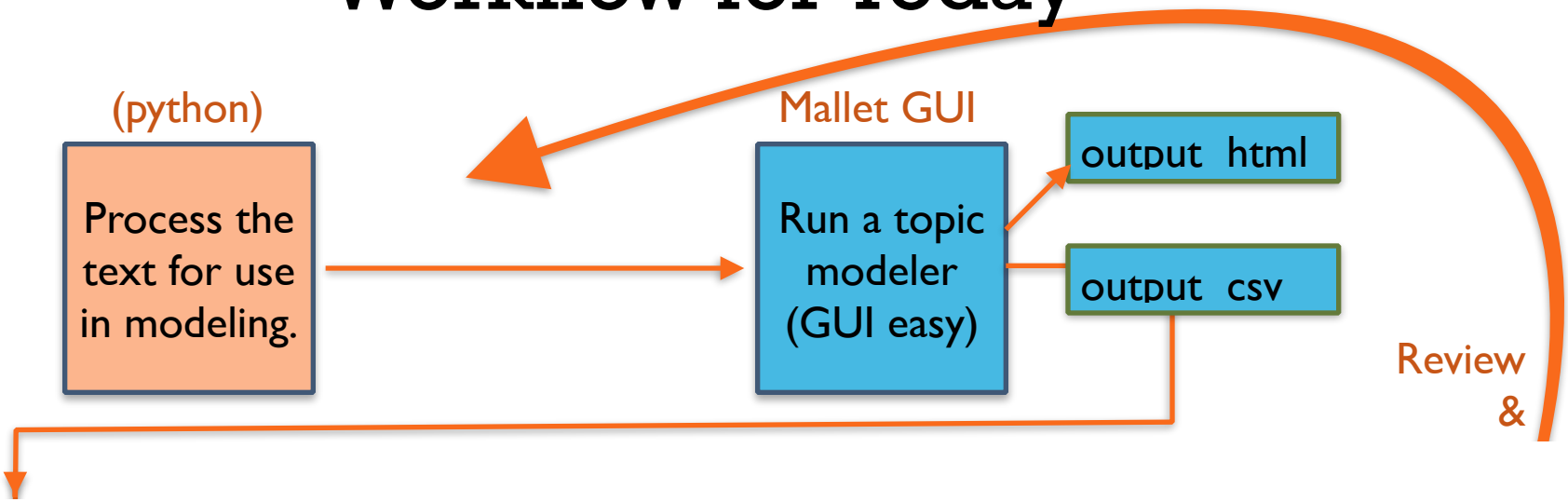
[accept the defaults]

```
>source activate topic_workshop
>pip install pattern
>pip install networkx
```

Other Software

- TopicModelingTool.jar — provided
- Gephi (gephi.org) — download and install.
- Also add plugins for Sigma Exporter and Circular Layout.
- Note: If you have issues running gephi, read this.

Workflow for Today



Cheat sheet: Scripts workflow

1. Optionally: Pre-process text files for part of speech…
   >python preprocess_files.py [original_dir] [new_dir] [part_of_speech]
2. Run Mallet on text files.
   Make an output directory. Double click on TopicModelingTool.jar.
3. Create gephi gdf files
   >python make_gephi_file.py [topics_dir_csv_path] [optl label]
4. View in Gephi, fix layout….
   Click on Gephi app and open your .gdf output file.
5. Optionally: Output gexf / json for d3
   >python d3_gexf.py [your output 'for excel' csvfile from 3] [label]
6. From Gephi, export as sigma.js site
   Find under export menu, if you installed that plugin
7. Fix up the sigma config and run server
   >python run_network.py [network_dir] [optl port]

Make an directory for your output data (“my_topics?”)

RUN THE TOPIC MODELING TOOL.JAR

The Topic Modeling black box: David Newman's Topic Modeling Tool

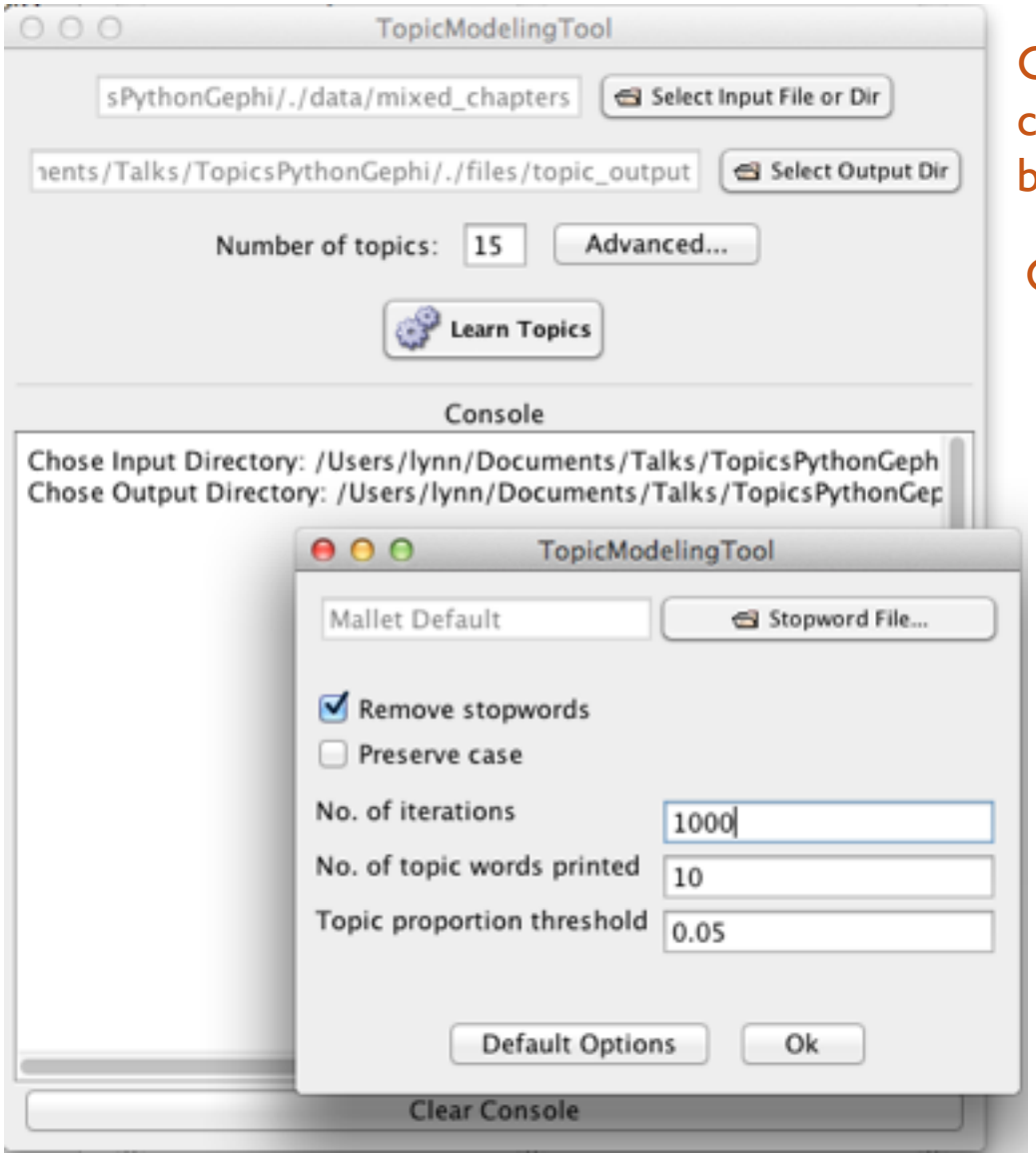
A tool available for non-technical audiences! A GUI wrapper on the state-of-the-art Mallet (a java-based app by David Mimno).

https://code.google.com/p/topic-modeling-tool/

(Also provided in the workshop files)

More of his work: http://www.ics.uci.edu/~newman/

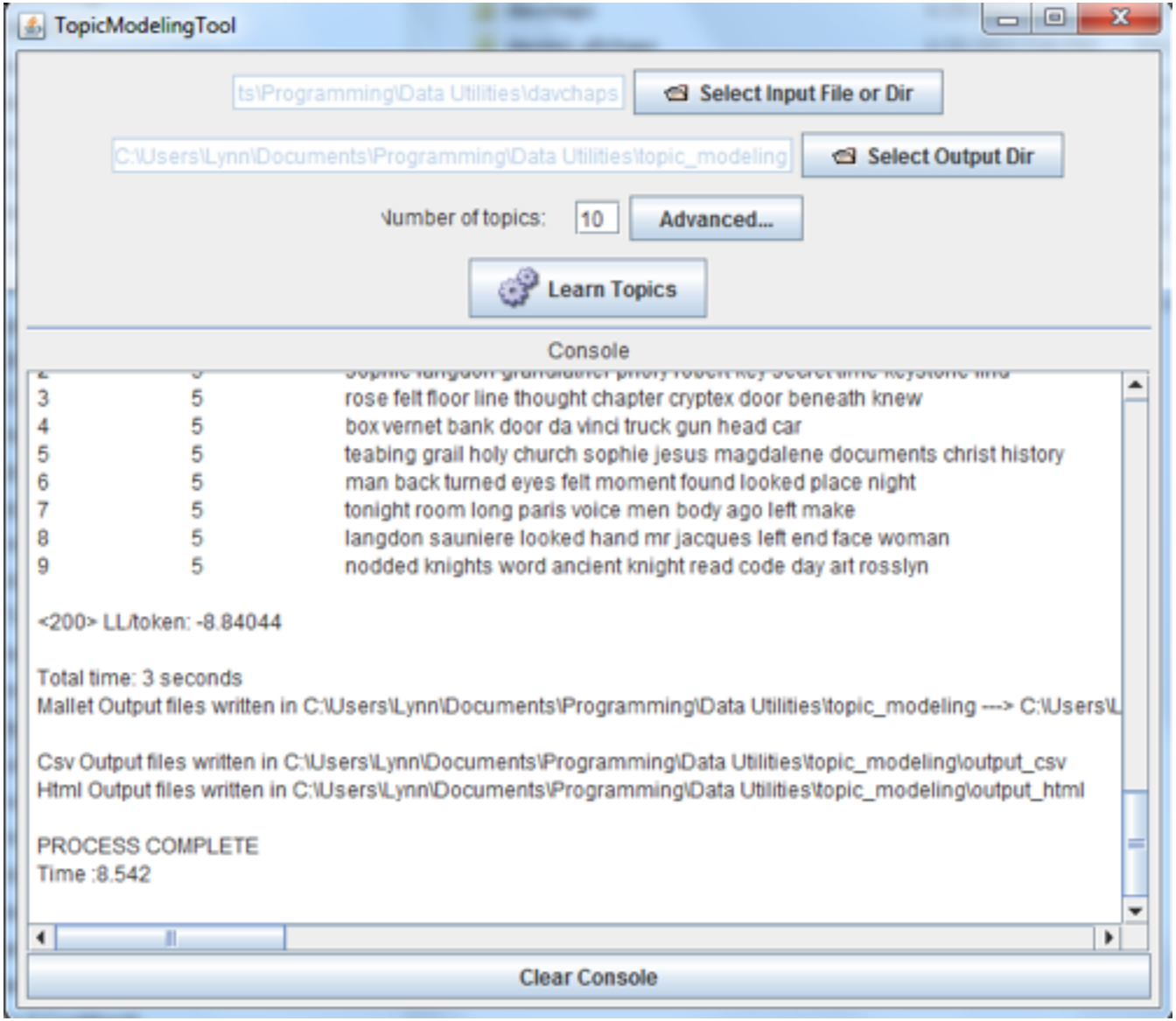
Topic Modeling Tool (GUI)



Create an output dir called "topic_output" before you try to select.

Click "Advanced…"

Post run…


TopicModelingTool
ts\Programming\Data Utilities\davchaps
Select Input File or Dir
C:\Users\Lynn\Documents\Programming\Data Utilities\topic_modeling
Select Output Dir
Number of topics:
10
Advanced...
Learn Topics
Console
3 5 rose felt floor line thought chapter cryptex door beneath knew
4 5 box vernet bank door da vinci truck gun head car
5 5 teabing grail holy church sophie jesus magdalene documents christ history
6 5 man back turned eyes felt moment found looked place night
7 5 tonight room long paris voice men body ago left make
8 5 langdon sauniere looked hand mr jacques left end face woman
9 5 nodded knights word ancient knight read code day art rosslyn
<200> LL/token: -8.84044
Total time: 3 seconds
Mallet Output files written in C:\Users\Lynn\Documents\Programming\Data Utilities\topic_modeling ---> C:\Users\L
Csv Output files written in C:\Users\Lynn\Documents\Programming\Data Utilities\topic_modeling\output_csv
Html Output files written in C:\Users\Lynn\Documents\Programming\Data Utilities\topic_modeling\output_html
PROCESS COMPLETE
Time :8.542
Clear Console

Understanding the Output

StackOverflow post: http://stackoverflow.com/questions/8447393/how-to-understand-the-output-of-topic-model-class-in-mallet

```
<1450> LL/token: -9.11846
<1460> LL/token: -9.11803
<1470> LL/token: -9.10896
<1480> LL/token: -9.11237
<1490> LL/token: -9.10845
```

Iteration number

Log Likelihood per word (we want this to increase as the algorithm runs)

Output files: Data (csv), text web site

output_csv

List of T topics:

Topics_Words.csv

List of topics in each of D documents:

TopicsInDocs.csv

List of top-ranked documents in each of T topics

DocsInTopics.csv

output_html

all_topics.html

Topic Modeling Mallet command line

You could also run mallet from the command line:

http://programminghistorian.org/lessons/topic-modeling-and-mallet

Or use a Python (or R) wrapper:

http://radimrehurek.com/2014/03/tutorial-on-mallet-in-python/

To do the rest of this workshop, you'd need to process the output files yourself similarly to our

py code (assume \t seps, not csv)

Pros/Cons vs CMD-Line Mallet

Pros of GUI

- Allows stopword file input
- Takes folder or file of text
- Produces csv and html output in a neat dir structure
- Has a GUI! (simpler to just get going without code and help)
- A nice intro to using mallet on the command line

Cons of GUI

- Runs with defaults, so no optimize-interval or other cmd line options
- No diagnostic output (a command-line option)
- Can get slightly fewer stats for your vis, as a result

2 of the 3 CSV Output files

topicId	words..
1	silas aringarosa remy teacher church dei opus bishop tomb vatican
2	fache collet police message neveu agent phone captain plane sir
3	sophie langdon grandfather priory robert key secret keystone time find
4	rose floor felt line thought chapter cryptex door knew began
5	box vernet bank vinci door head da louvre truck gun
6	teabing grail holy church sophie jesus magdalene documents history ch
7	man back turned felt eyes moment found looked place night

DocsInTopics.csv
Topics_Words.csv
TopicsInDocs.csv

docId	filename	top topics	and contribution to doc ...																	
1	C:\Users\I	9	0.21	4	0.188	8	0.147	7	0.13	5	0.13	6	0.063	1	0.055					
2	C:\Users\I	8	0.232	9	0.21	7	0.143	4	0.095	2	0.095	3	0.072	10	0.068	5	0.057			
3	C:\Users\I	4	0.274	1	0.212	7	0.137	8	0.127	5	0.062	3	0.053							
4	C:\Users\I	1	0.442	7	0.106	8	0.097	4	0.091	6	0.085									
5	C:\Users\I	6	0.175	3	0.165	4	0.146	9	0.124	7	0.098	5	0.097	1	0.076	8	0.062			
6	C:\Users\I	1	0.333	8	0.235	4	0.216	6	0.069											
7	C:\Users\I	1	0.222	2	0.197	7	0.191	8	0.098	6	0.095	9	0.065							
8	C:\Users\I	4	0.242	10	0.18	3	0.178	7	0.118	9	0.104	6	0.065	5	0.057					
9	C:\Users\I	9	0.252	3	0.187	6	0.134	4	0.097	7	0.087	10	0.08	5	0.055	8	0.054			
10	C:\Users\I	4	0.353	8	0.155	7	0.124	9	0.114	10	0.075	5	0.072							
11	C:\Users\I	2	0.314	9	0.24	3	0.097	7	0.09	8	0.073	10	0.072							
12	C:\Users\I	2	0.261	9	0.229	3	0.15	8	0.132	7	0.083									
13	C:\Users\I	9	0.309	2	0.16	3	0.157	8	0.091	7	0.091	6	0.058							
14	C:\Users\I	2	0.459	8	0.176	3	0.102	7	0.083	9	0.059									
15	C:\Users\I	1	0.25	4	0.182	7	0.12	8	0.104	5	0.089	3	0.089	6	0.057					
16	C:\Users\I	3	0.347	8	0.18	2	0.117	7	0.107	9	0.1	10	0.05	4	0.05					
17	C:\Users\I	2	0.354	8	0.173	3	0.116	9	0.104	10	0.065	7	0.057	5	0.051					
18	C:\Users\I	8	0.292	2	0.223	5	0.185	3	0.085	7	0.071	9	0.064	4	0.06					
19	C:\Users\I	8	0.244	1	0.199	4	0.124	3	0.11	9	0.09	5	0.076	7	0.07	6	0.059			
20	C:\Users\I	1	0.312	8	0.157	3	0.114	6	0.107	4	0.1	7	0.084	5	0.066					
21	C:\Users\I	10	0.368	9	0.284	3	0.066	7	0.06	5	0.06	2	0.053							
22	C:\Users\I	3	0.256	9	0.216	5	0.165	8	0.09	7	0.079	2	0.069	4	0.062					
23	C:\Users\I	4	0.461	1	0.145	8	0.07	7	0.068	9	0.066	10	0.062							
24	C:\Users\I	3	0.368	5	0.131	8	0.111	4	0.107	9	0.099	7	0.087							
25	C:\Users\I	4	0.336	8	0.185	1	0.185	3	0.126	6	0.059									
26	C:\Users\I	2	0.457	8	0.104	9	0.098	7	0.098	10	0.069	3	0.069	4	0.064					
27	C:\Users\I	5	0.34	9	0.261	7	0.084	4	0.07	3	0.07	10	0.059							

This workshop has lots of code to process these files… (and a script: make_gephi_file.py)

```
In [36]: topics_per_doc = read_doctopics(topic_docs)  # keep in mind the input GUI said to cut off classification a
```

```
chap_0 {'1': '0.055', '5': '0.130', '4': '0.188', '7': '0.130', '6': '0.063', '9': '0.210', '8': '0.147'}
chap_1 {'10': '0.068', '3': '0.072', '2': '0.095', '5': '0.057', '4': '0.095', '7': '0.143', '9': '0.210'
chap_10 {'1': '0.212', '3': '0.053', '5': '0.062', '4': '0.274', '7': '0.137', '8': '0.127'}
chap_100 {'1': '0.442', '8': '0.097', '4': '0.091', '7': '0.106', '6': '0.085'}
chap_101 {'1': '0.076', '3': '0.165', '5': '0.097', '4': '0.146', '7': '0.098', '6': '0.175', '9': '0.124
chap_102 {'1': '0.333', '8': '0.235', '4': '0.216', '6': '0.069'}
chap_103 {'1': '0.222', '2': '0.197', '7': '0.191', '6': '0.095', '9': '0.065', '8': '0.098'}
chap_104 {'10': '0.180', '3': '0.178', '5': '0.057', '4': '0.242', '7': '0.118', '6': '0.065', '9': '0.10
chap_105 {'10': '0.080', '3': '0.187', '5': '0.055', '4': '0.097', '7': '0.087', '6': '0.134', '9': '0.25
chap_106 {'10': '0.075', '5': '0.072', '4': '0.353', '7': '0.124', '9': '0.114', '8': '0.155'}
chap_11 {'10': '0.072', '3': '0.097', '2': '0.314', '7': '0.090', '9': '0.240', '8': '0.073'}
chap_12 {'9': '0.229', '8': '0.132', '3': '0.150', '2': '0.261', '7': '0.083'}
chap_13 {'3': '0.157', '2': '0.160', '7': '0.091', '6': '0.058', '9': '0.309', '8': '0.091'}
chap_14 {'9': '0.059', '8': '0.176', '3': '0.102', '2': '0.459', '7': '0.083'}
```

The GUI's HTML output is a little lacking…

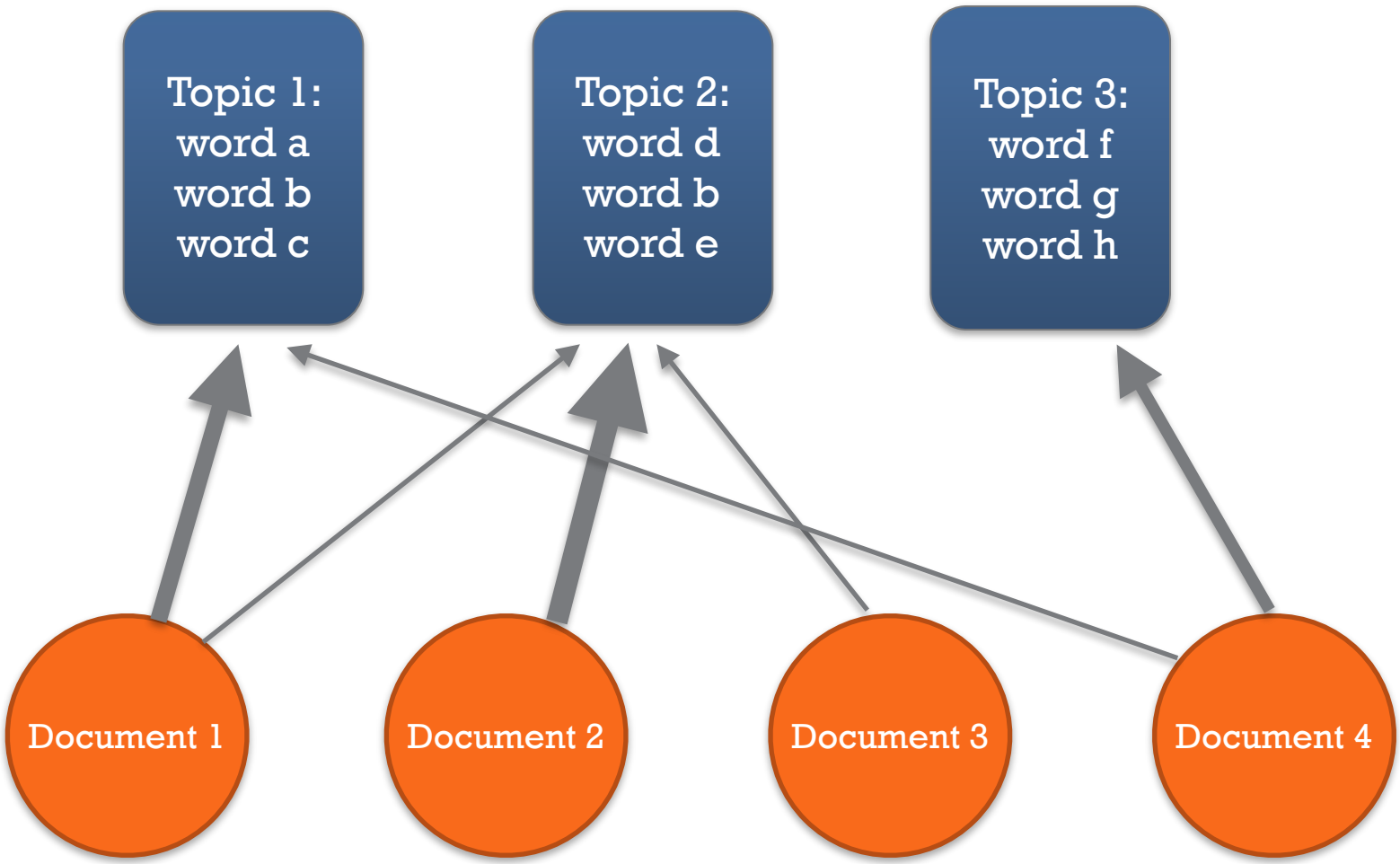
TOPIC : man back turned felt eyes moment found looked place night ...

top-ranked docs in this topic (#words in doc assigned to this topic)

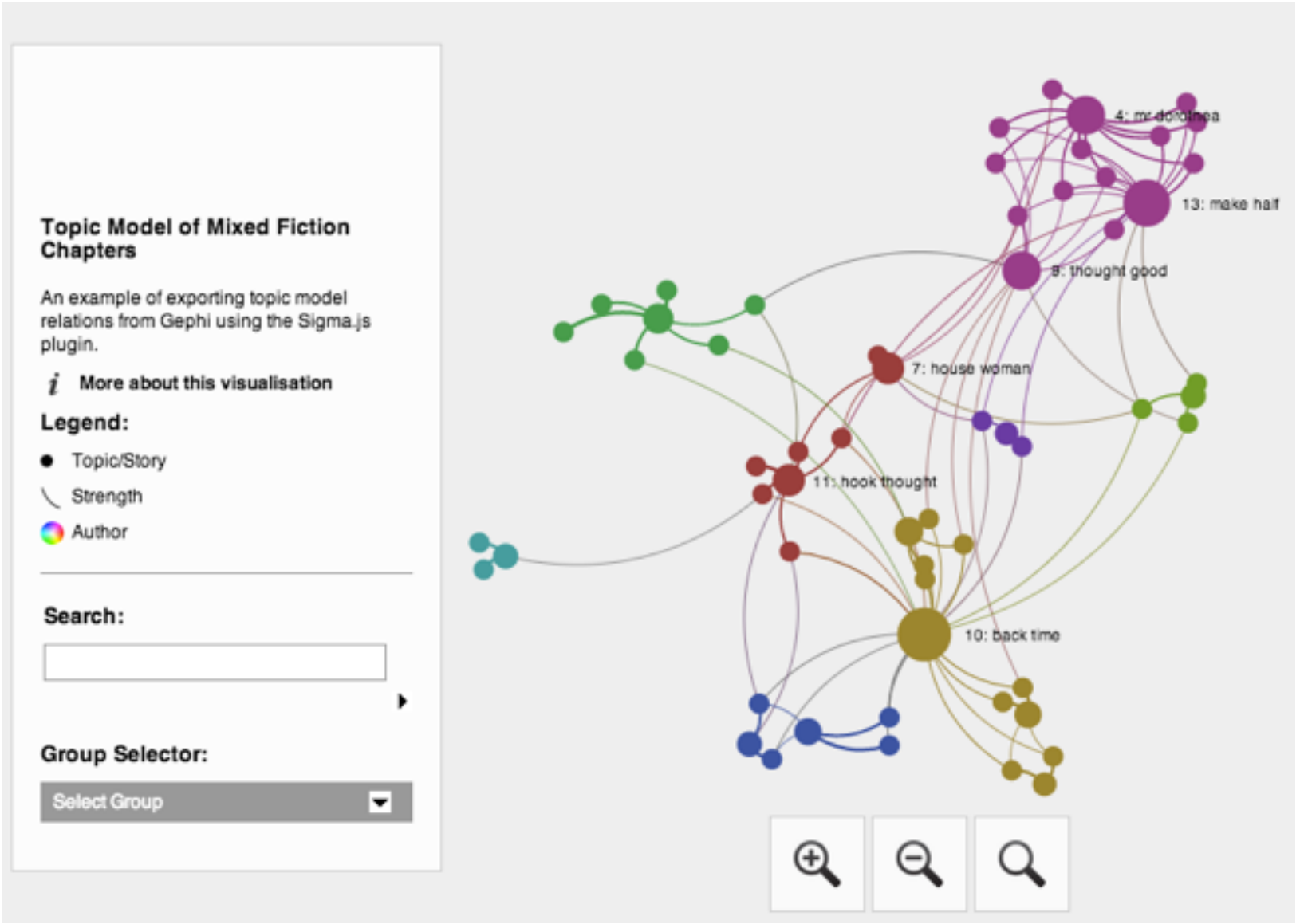
2. (219) chap_67.txt
3. (193) chap_104.txt
4. (180) chap_84.txt
5. (179) chap_99.txt
6. (160) chap_51.txt
7. (153) chap_32.txt
8. (145) chap_81.txt

A bipartite graph of chapters and topics is an obvious vis method….

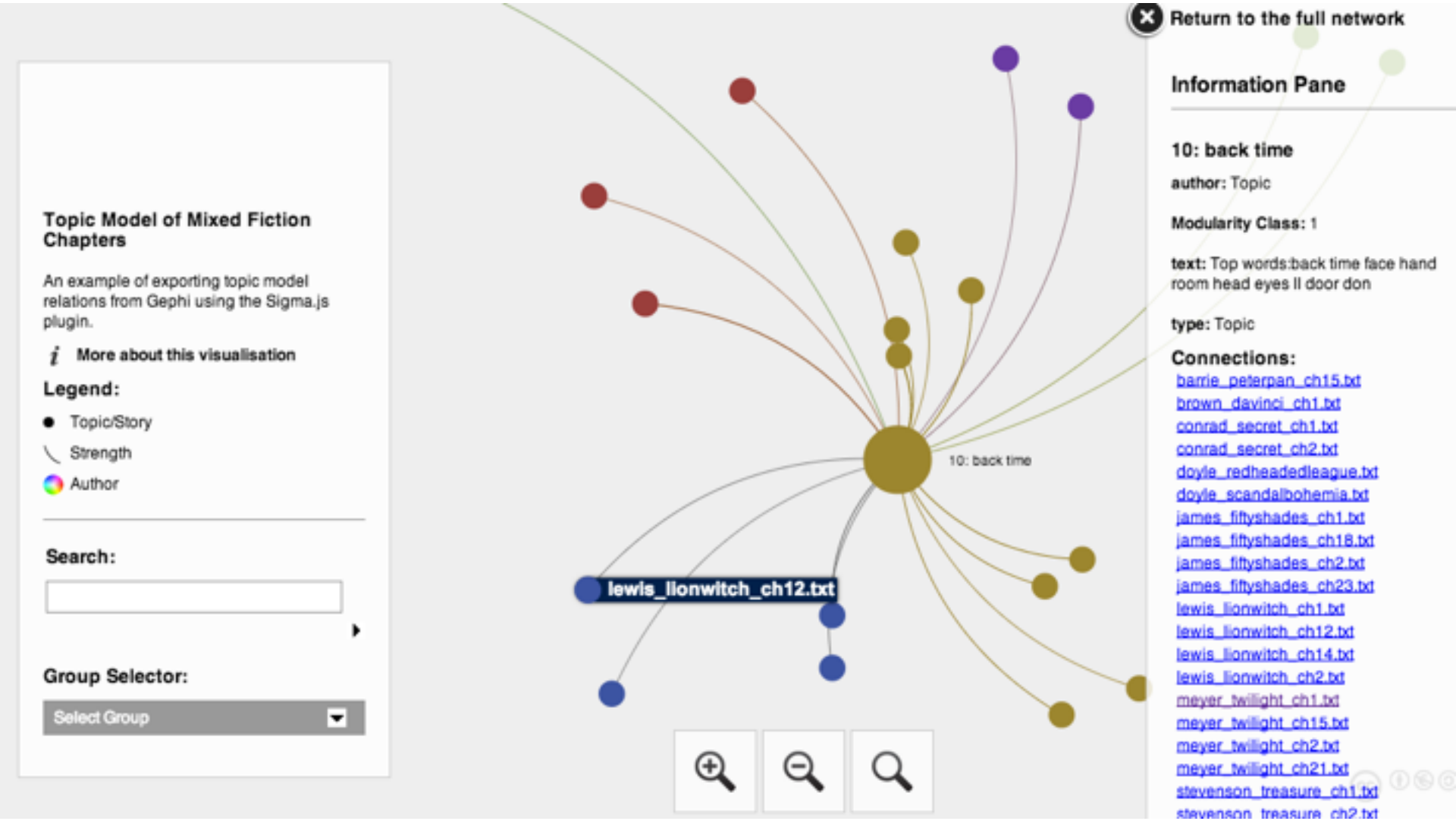
The results of topic modeling



Our Goal


Topic Model of Mixed Fiction Chapters
An example of exporting topic model relations from Gephi using the Sigma.js plugin.
More about this visualisation
Legend:
Topic/Story
Strength
Author
Search:
Group Selector:
Select Group
13: make half
8: thought good
7: house woman
11: hook thought
10: back time

Our Goal


Return to the full network
Information Pane
10: back time
author: Topic
Modularity Class: 1
text: Top words:back time face hand room head eyes ll door don
type: Topic
Connections:
barrie_peterpan_ch15.txt
brown_davinci_ch1.txt
conrad_secret_ch1.txt
conrad_secret_ch2.txt
doyle_redheadedleague.txt
doyle_scandalbohemia.txt
james_fiftyshades_ch1.txt
james_fiftyshades_ch18.txt
james_fiftyshades_ch2.txt
james_fiftyshades_ch23.txt
lewis_lionwitch_ch1.txt
lewis_lionwitch_ch12.txt
lewis_lionwitch_ch14.txt
lewis_lionwitch_ch2.txt
meyer_twilight_ch1.txt
meyer_twilight_ch15.txt
meyer_twilight_ch2.txt
meyer_twilight_ch21.txt
stevenson_treasure_ch1.txt
Topic Model of Mixed Fiction Chapters
An example of exporting topic model relations from Gephi using the Sigma.js plugin.
More about this visualisation
Legend:
Topic/Story
Strength
Author
Search:
Group Selector:
Select Group
10: back time
lewis_lionwitch_ch12.txt

PROCESS CSV OUTPUT FROM GUI TOOL

Cheat sheet: Scripts workflow

1. Optionally: Pre-process text files for part of speech…

   >python preprocess_files.py [original_dir] [new_dir] [part_of_speech]

2. Run Mallet on text files.

   Make an output directory. Double click on TopicModelingTool.jar.

3. Process output: Create gephi gdf files

   >python make_gephi_file.py [topics_dir_csv_path] [optl label]

4. View in Gephi, fix layout….

   Click on Gephi app and open your .gdf output file.

5. Optionally: Output gexf / json for d3

   >python d3_gexf.py [your output 'for excel' csvfile from 3] [label]

6. From Gephi, export as sigma.js site

   Find under export menu, if you installed that plugin

7. Fix up the sigma config and run server

   >python run_network.py [network_dir] [optl port]

Our next step: Process GUI CSV Output

After running the Topic Modeling Tool, we start with the IPython notebook "Topic Analysis of Mixed Fiction.ipynb."

If you want to run the notebooks, make sure you are in an active virtual environment, set up as I described.

```
> source activate topic_workshop
(topic_workshop)> ipython notebook
```

If you don't want to run it, you can achieve the same outputs with the path to the gui output csv file:

```
(topic_workshop)> cd files
(topic_workshop)> python make_gephi_file.py topic_output/output_csv all
```

Our next steps… visualize

1. Excel pivot table analysis with the for_excel.csv file
2. Gephi for the gdf file output!

Tips on Gephi layout and UI are in the PDF:

GephiToSigmaJS_Mixed.pdf

EXCEL: OPEN FOR_EXCEL_ALL.CSV

GEPHI: LAUNCH IT, LOAD THE GDF FILE
OPEN GEPHITOSIGMAJS_MIXED.PDF

GETTING YOUR WEBSITE UP

Export from Gephi

- Under Export, if you have the plugin installed, you should see an option for SigmaJs. Do that…
- It will create a directory called "network."
- The files need a little preprocessing…

  Run:

  ```
  >python run_network.py network 8000
  ```

REVIEW YOUR SITE!

How can we improve on the results?

Iterate on number of topics you output. Try 10, instead of 15!

Rerun the GUI with 10 topics, then use command line:

```
> cd files
> python make_gephi_file.py topic_output/output_csv 10
```

How else can we improve?

Pre-process the documents — change what's modeled! Maybe only verbs?

- Use stop words tuned for your data set (don't want proper nouns? or only proper nouns?)
- Read in a document, parse it, save out a new "document" of the POS you want, then use those in the topic modeler

Cheat sheet: Scripts workflow

1. Optionally: Pre-process text files for part of speech…
   >python preprocess_files.py [original_dir] [new_dir] [part_of_speech]
2. Run Mallet on text files.
   Make an output directory. Double click on TopicModelingTool.jar.
3. Create gephi gdf files
   >python make_gephi_file.py [topics_dir_csv_path] [optl label]
4. View in Gephi, fix layout….
   Click on Gephi app and open your .gdf output file.
5. Optionally: Output gexf / json for d3
   >python d3_gexf.py [your output 'for excel' csvfile from 3] [label]
6. From Gephi, export as sigma.js site
   Find under export menu, if you installed that plugin
7. Fix up the sigma config and run server
   >python run_network.py [network_dir] [optl port]

Optionally: Preprocess the text before modeling.

PRE-PROCESS FOR PARTS OF SPEECH, ETC

Our next step… Preprocess the doc text

Use the notebook POS_Text_Conversion.ipynb

In this notebook, we'll look at how to handle text: tokenize it, clean it, strip out words/ punctuation, find parts of speech…

For faster, command-line use (requires pattern and nltk installed!)

```
>cd files
>python preprocess_files.py ../data/mixed_chapters ../data/verbs_only VB
```

Now look at those results…

1. Make a directory for the new topic modeled files under files:

   ```
   >mkdir verb_output
   ```

2. Rerun the GUI with this as output directory, and the verb files as your input files!
3. Then use command line:

   ```
   > cd files
   > python make_gephi_file.py verb_output/output_csv verbs
   ```

4. Then find the output for_gephi_topics_verbs.gdf & visualize in Gephi.

Optionally…

OTHER WAYS TO VIEW TOPICS (AND JSON/D3 EXPORT)

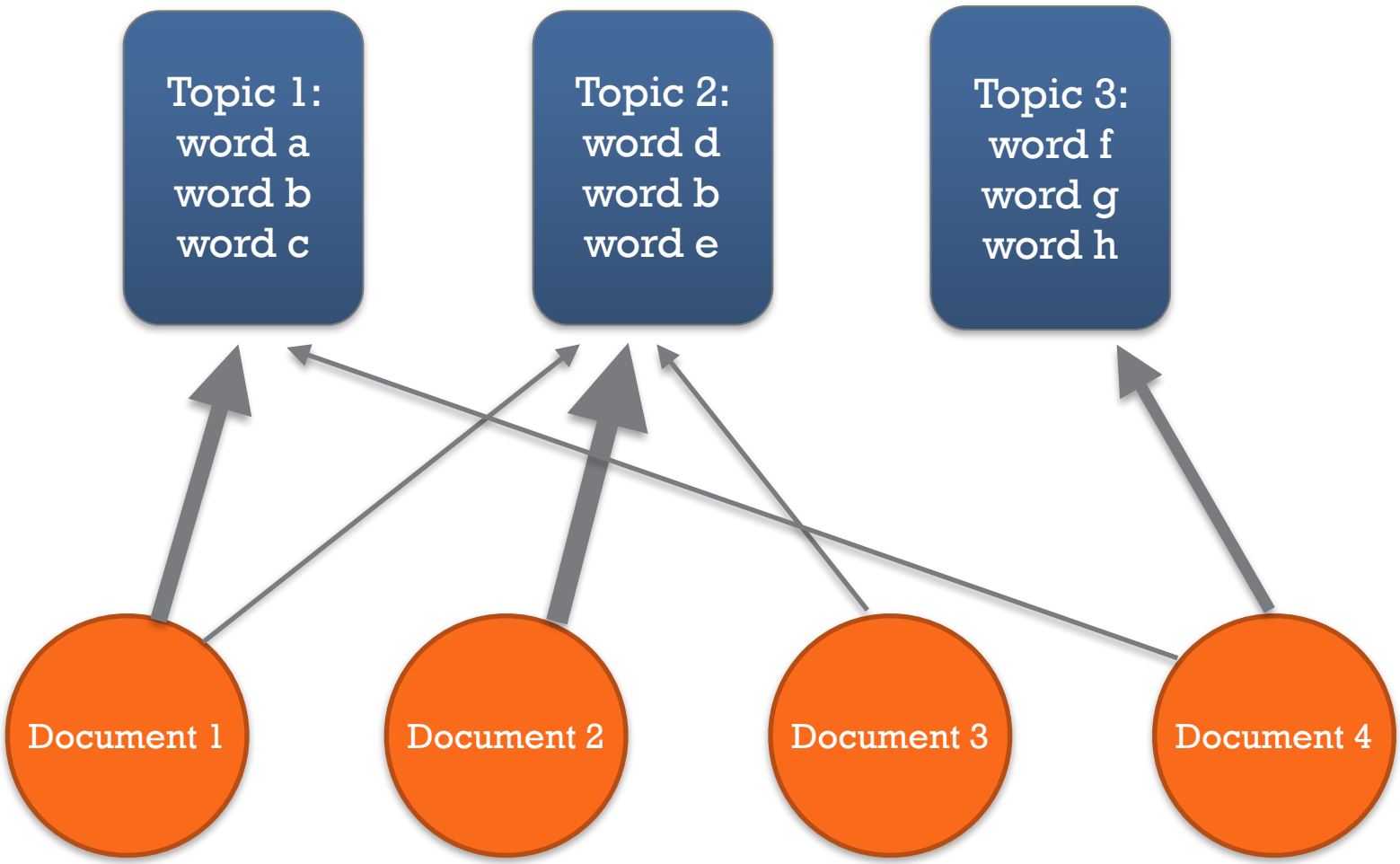
Cheat sheet: Scripts workflow

1. Optionally: Pre-process text files for part of speech…
   >python preprocess_files.py [original_dir] [new_dir] [part_of_speech]
2. Run Mallet on text files.
   Make an output directory. Double click on TopicModelingTool.jar.
3. Create gephi gdf files
   >python make_gephi_file.py [topics_dir_csv_path] [optl label]
4. View in Gephi, fix layout….
   Click on Gephi app and open your .gdf output file.
5. Optionally: Output gexf / json for d3
   >python d3_gexf.py [your output 'for excel' csvfile from 3] [label]
6. From Gephi, export as sigma.js site
   Find under export menu, if you installed that plugin
7. Fix up the sigma config and run server
   >python run_network.py [network_dir] [optl port]

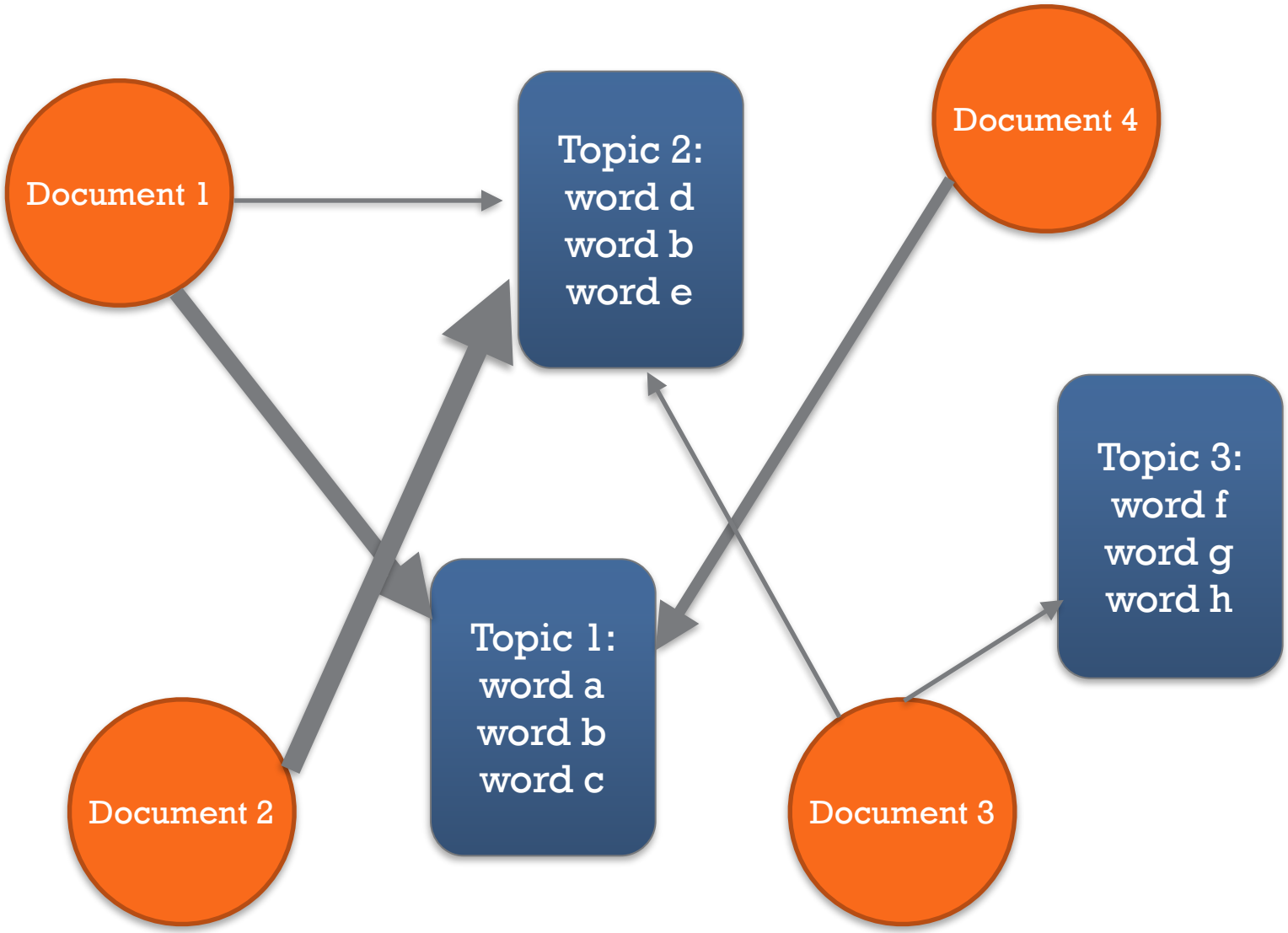
Going to D3

- Raw nodes-edges json:
  - export json from gephi (using JSON plugin)
  - or post-process and create the json: see next step
  - Example of d3 network use: http://bl.ocks.org/mbostock/4062045
- Export gexf and use Elijah Meeks' code to process and display it from gexf format:
  - http://bl.ocks.org/emeeks/9357371

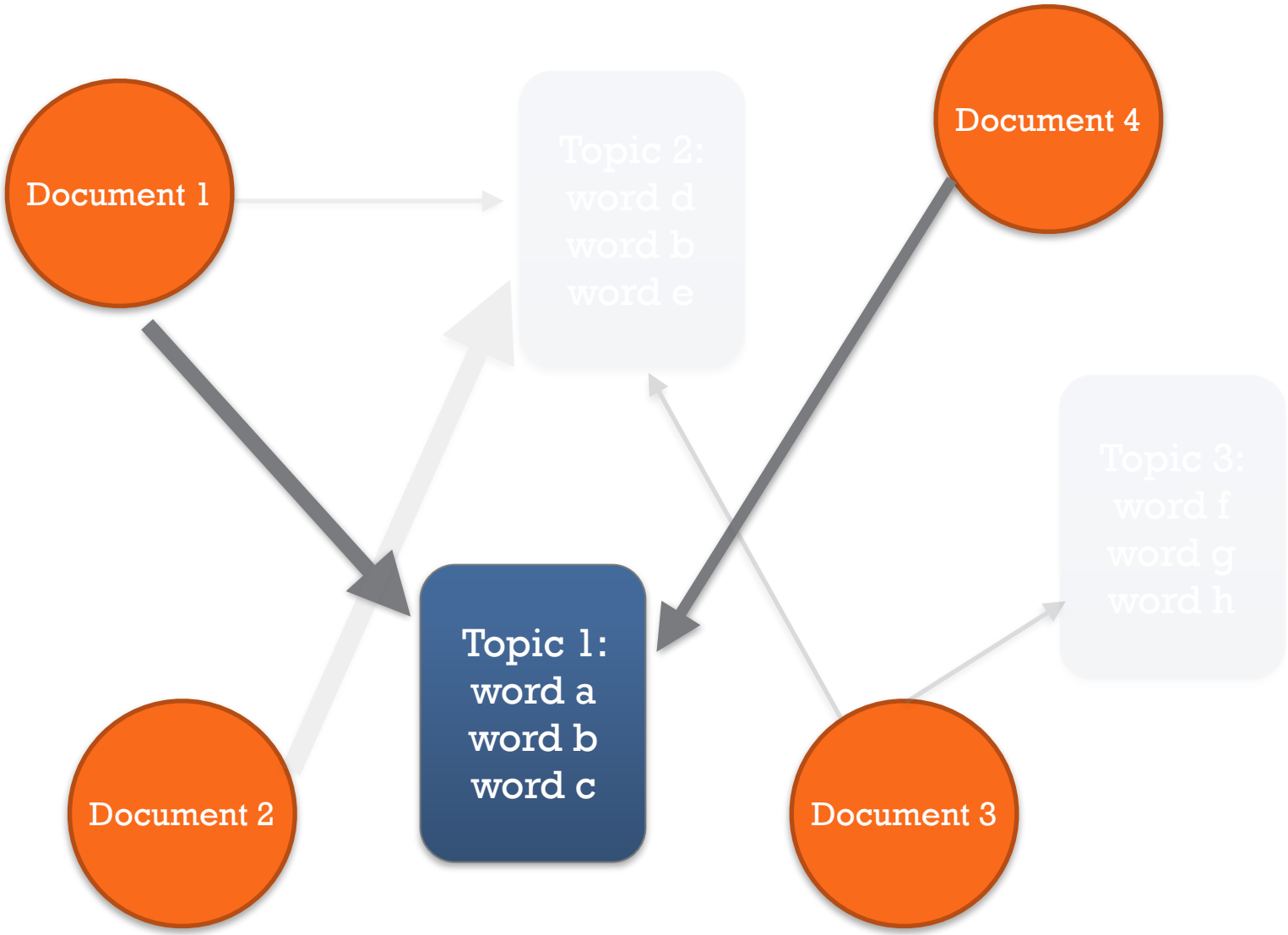
How we started…



Another view: More Like Ours



Another view: Related Document to Document



But we need to account for link weights

- Doc A — Topic 1, 40% words
- Doc B — Topic 1, 20% words

- How do you compute the weight of the relation between the Doc A and Doc B?
  - Options: 1 / difference (normalized)
  - Average, Median, Count of edges

Our next step… Advanced: Doc to Doc Network in D3.

We want to combine some of the output we created as JSON files.

Use code in Advanced-D3 and GEXF Network of Docs Only.ipynb or files/d3_gexf.py.

From the command line - to make a simple file:

```
>cd files
>python d3_gexf.py for_excel.csv all for_excel_verbs.csv verbs
```

To make a file with 2 sets of edges, to compare:

```
python d3_gexf.py for_excel.csv all for_excel_verbs.csv verbs
```

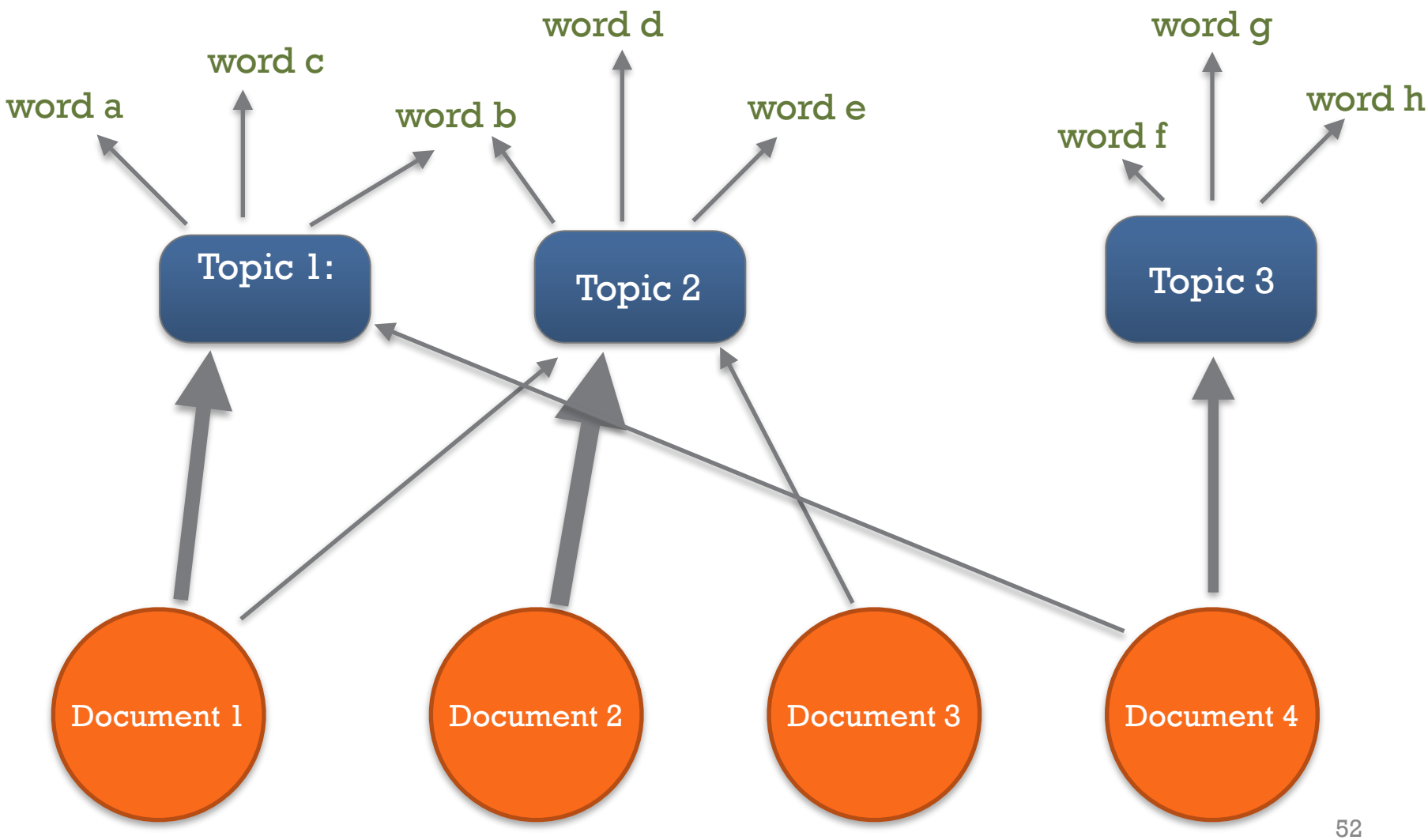
Output

- gexf file — for use in gephi if you want
- json files for use in d3
- Note: combined json, if you input 2 csv files to compare. Don't forget you need to include labels after each csv filename!

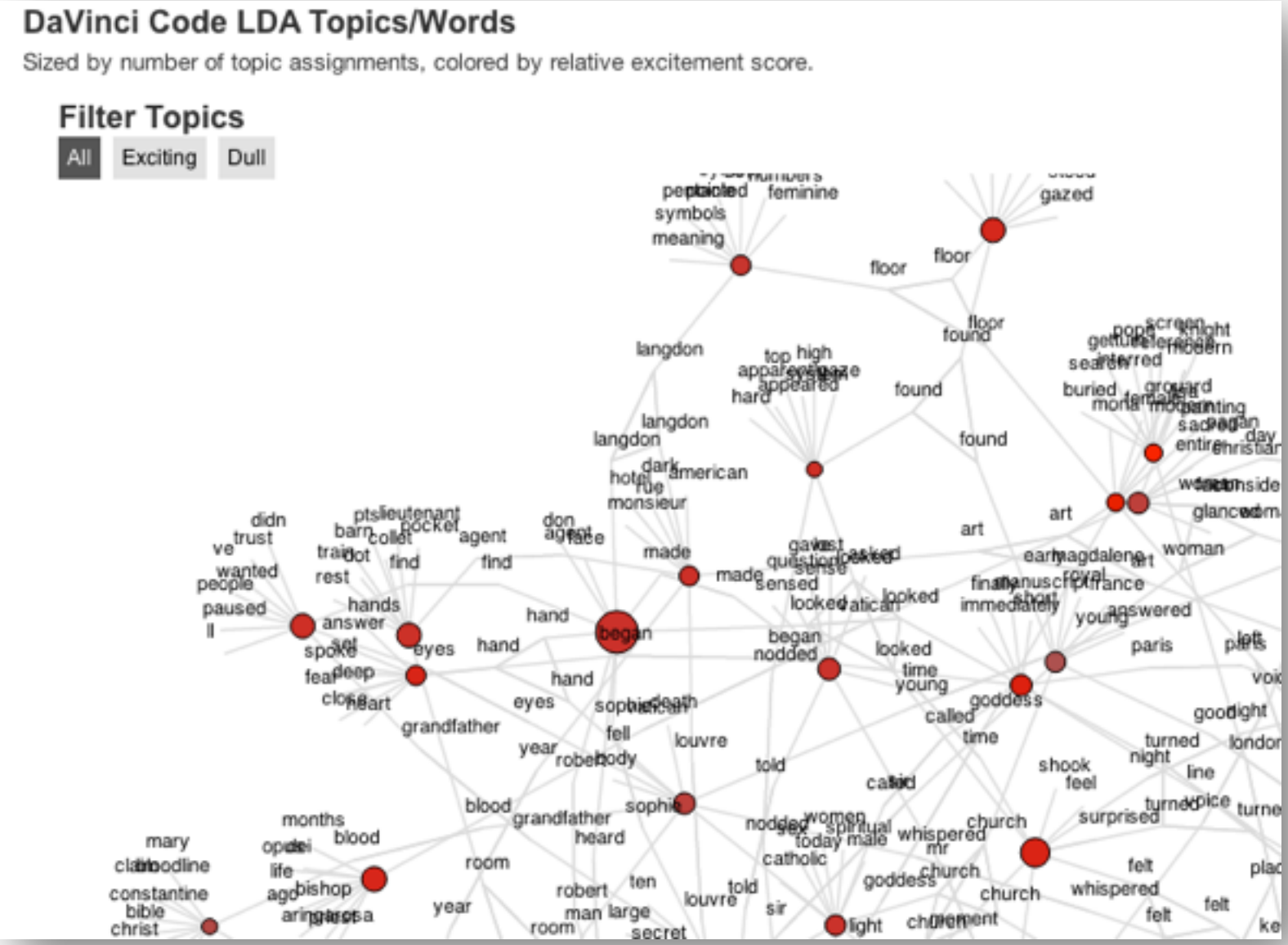
```
>cd verb_output
>python -m SimpleHTTPServer 8010
```

Load the d3_network.html file.

A further level of network you could draw….



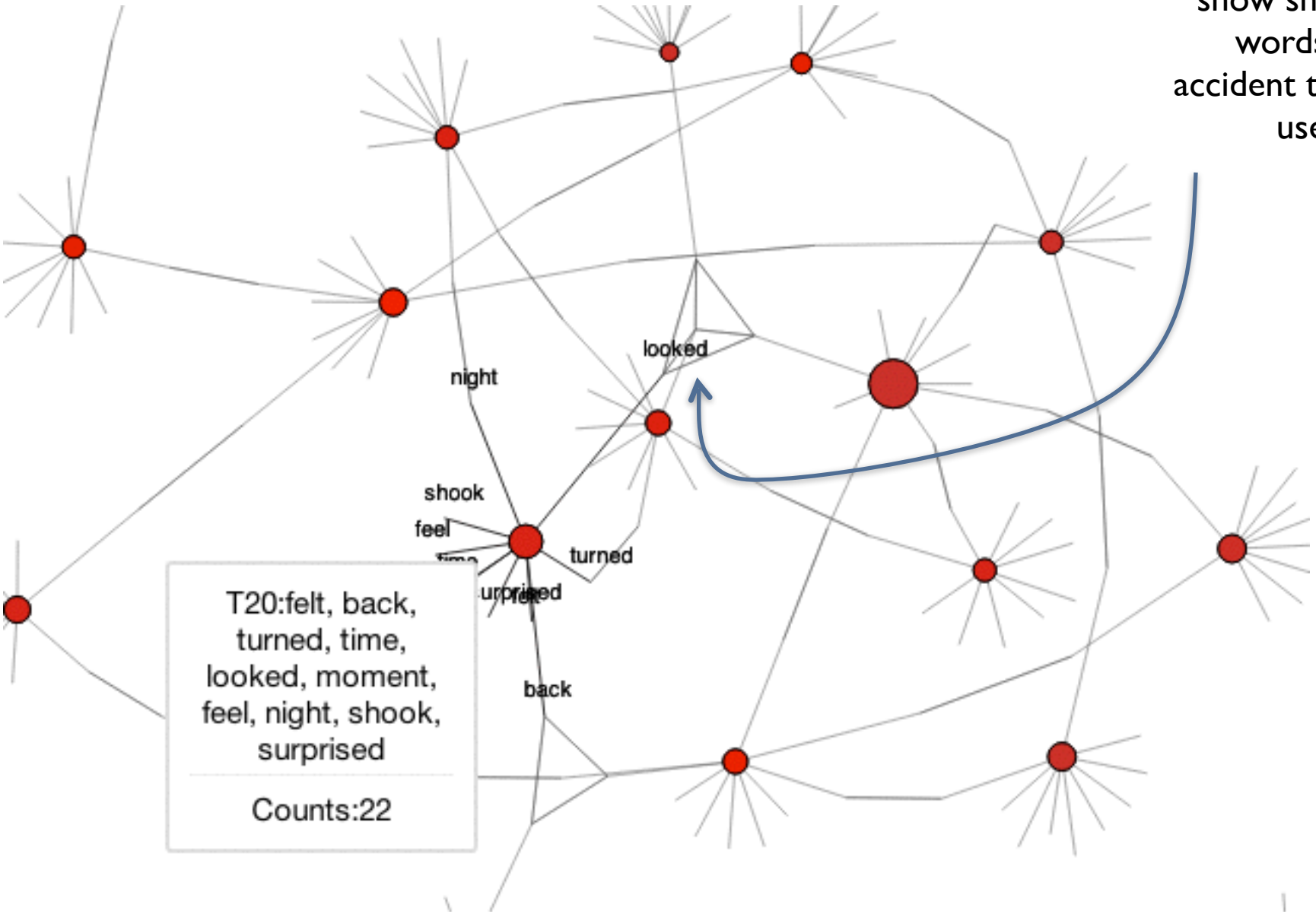
Maybe I need One More Tool. Any word relations of interest?
Let's try another hairball…



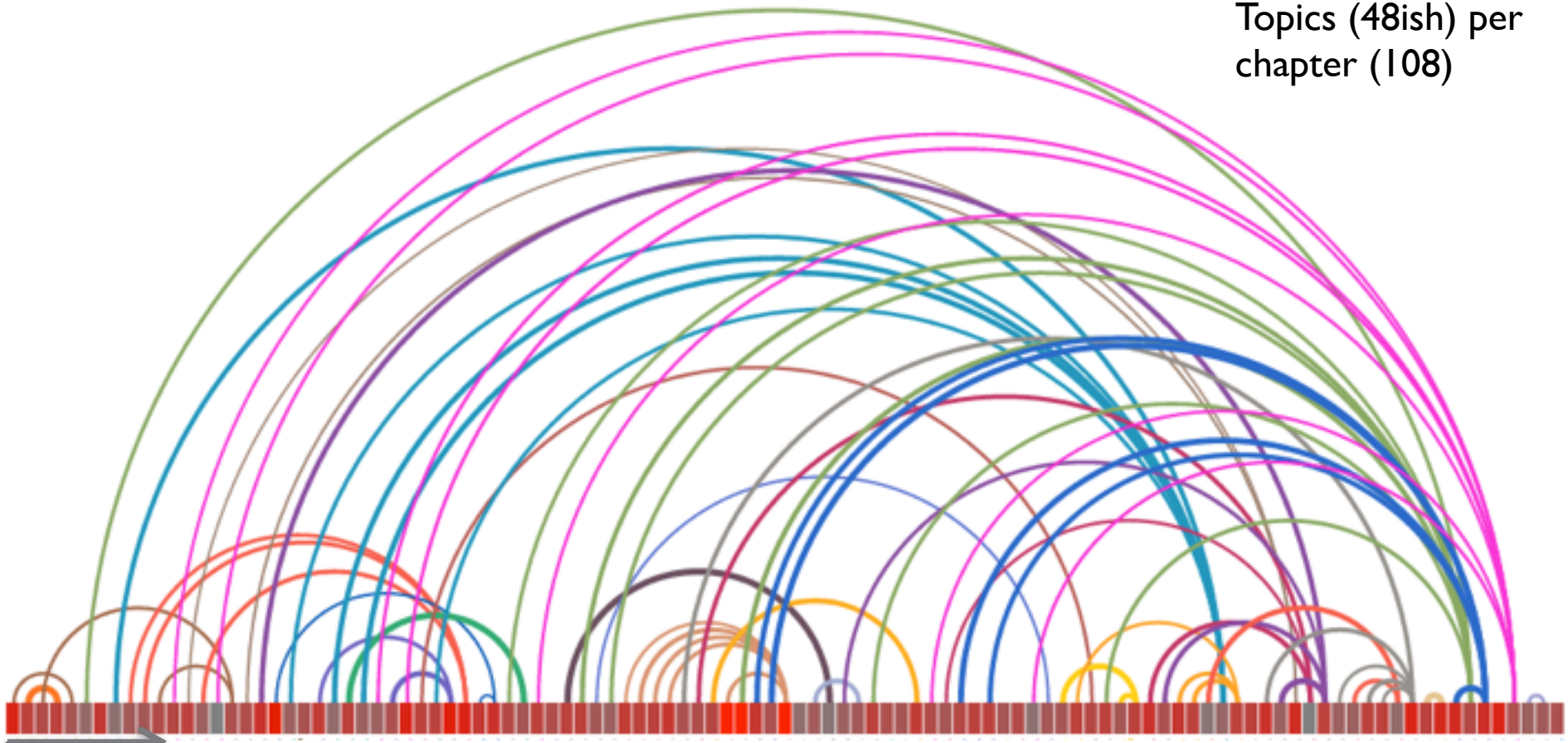
Demo: http://www.ghostweather.com/essays/talks/openvisconf/topic_words_network/index.html

Filtered to only the "exciting" nodes…

Small "constellations" show shared words (an accident that's useful!)



Another tool: Sequential documents, with topic arcs.
DaVinci Code topics to chapters mapping



Add some topic-tooltips and fade-outs….

Demo: http://www.ghostweather.com/essays/talks/openvisconf/topic_arc_diagram/TopicArc.html

But what did this show?

Some topics are just neither exciting nor dull – topic clustering (as I did it) had little to do with action scenes. It's slightly helpful for topics, though ☺

These nodes are shaded from gray (dull) to red (exciting)

WRAP UP!

Other Ways to Improve the Results Display

- Visualize differently or more (chords, matrix…)
- Look for the topic words "in context" - find sentences with them and use those as part of your topic description
- Construct phrases from your topic words to make them "better" for descriptors
- Use only the interesting output words for a topic
- Don't use the result immediately — use as input to other methods (it's a data reduction technique like principal components analysis)

A Few More References

- Matthew Jockers' post: The LDA Buffet is Now Open: http://www.matthewjockers.net/2011/09/29/the-lda-buffet-is-now-open-or-latent-dirichlet-allocation-for-english-majors/
- Scott Weingart's nice overview of LDA Topic Modeling in Digital Humanities: http://www.scottbot.net/HIAL/?p=221
- Elijah Meeks' lovely set of articles on LDA & Digital Humanties vis: https://dhs.stanford.edu/comprehending-the-digital-humanities/
- Topic Modeling Made Just Simple Enough - Ted Underwood post: http://tedunderwood.com/2012/04/07/topic-modeling-made-just-simple-enough/
- Some pure python (and C) implementations (toy code, primarily) are listed on Blei's website: http://www.cs.princeton.edu/~blei/topicmodeling.html