Corpus Explorer

a shiny-driven app for exploratory text analysis

Goal

- A visual exploratory tool for comparing and contrasting documents in a small-to-medium sized corpus
- High-level document summaries - topic models, significant terms, similar documents
- Easily assess document similarity and clustering visually

Broad overview

- Preprocessing
- Modeling
- Data structures/conversions (Py↦R)
- Postprocessing
- Final product: exploratory app

Preprocessing

- Apply custom regex substitution sequence to each document
- Tokenize the results using Gensim.utils.simple_preprocess
- Removed tokens with > max.length characters (antidisestablishmentarianism: 28)
- Detect bigrams using Gensim.models.Phrases
  - based on Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality." *Advances* ... 3.

$$\text{score}(w_i, w_j) = \frac{\text{count}(w_i w_j) - \delta}{\text{count}(w_i) \times \text{count}(w_j)}.$$

- This process is iterable: detect n-grams by joining terms to (n-1)-grams, etc.

Preprocessing (cont.)

- Remove common stop words and terms appearing in > p.max% of documents
- For terms appearing in < p.min% of documents, remove only those occurring < min.count times in the corpus
- Save results to log file:

```
Generating corpus-wide dictionary:
38573 initial terms in the dictionary.

Computing df's for terms in dictionary.
70 terms occur in 118 or more documents.
17934 terms occur in 1 or fewer documents.

Identifying sparse terms:

14851 of the rare terms occur 5 or fewer times in any document; removing.

Removing stop words and sparse terms:
23569 terms remain in the dictionary after stopword and rare word removal.
```

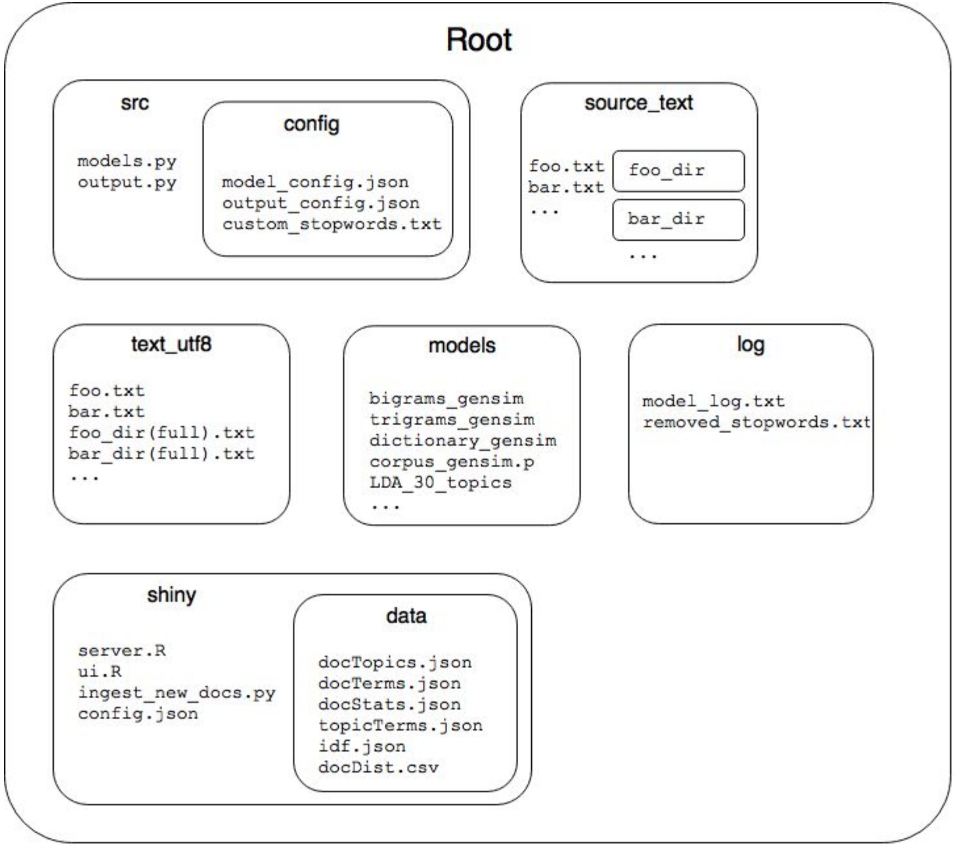
Modeling

- How to summarize documents for easy comparison?
- Significant terms is one way (TF-IDF)
- *Topic Models* are another way: summarize docs as mixtures of “topics” - intuitively, clusters of terms appearing often together
- Why topic models?
  - Low-dimensional representation: *10’s* of topics vs *1000’s* of terms
  - Semantic coherence: a good quality topic model can produce topics that *make sense* (interpretable)
- Most tried and true flexible topic model: LDA
  - Allows document comparison by fitting one set of topics to the entire corpus

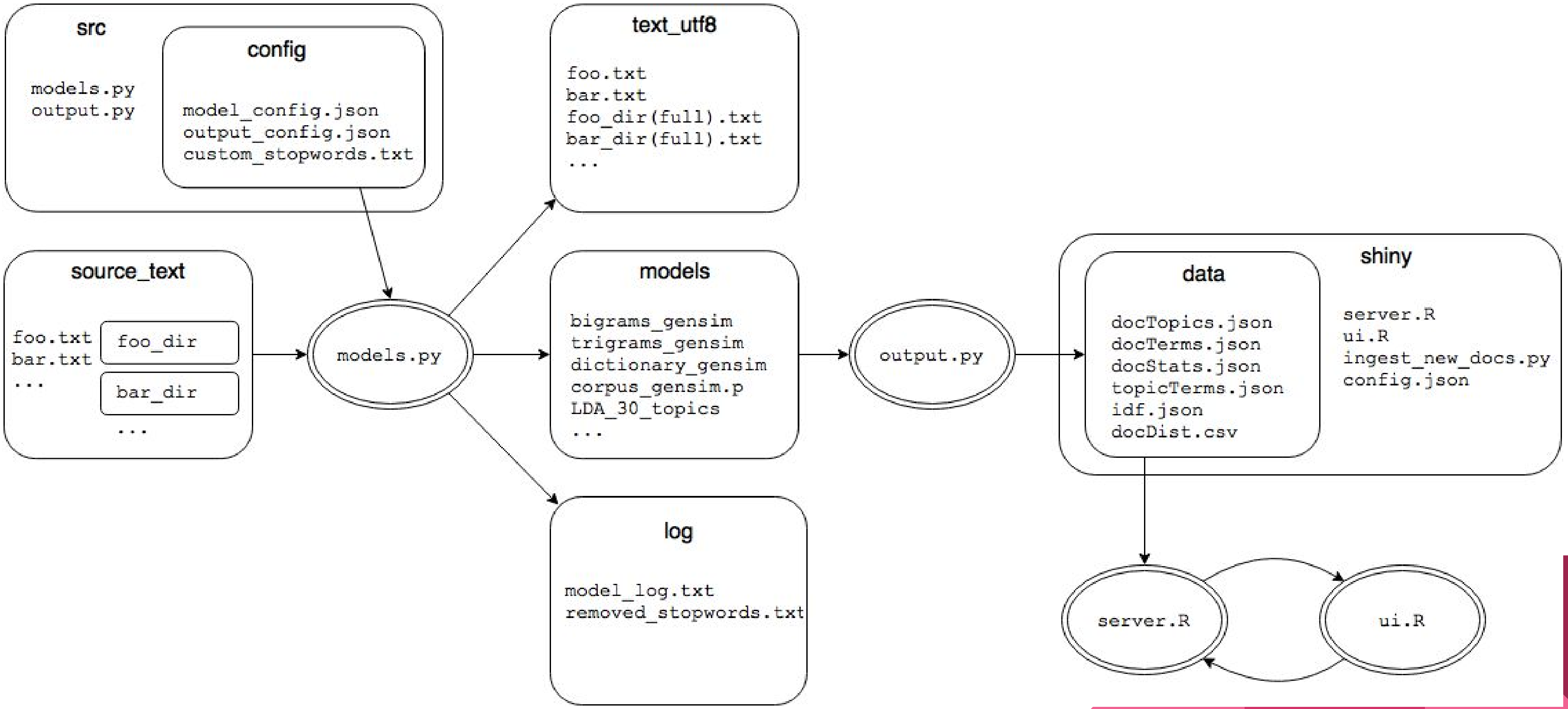
Modeling (cont.)

- Fit LDA to the corpus using Gensim's implementation
- Since interpretability is important to the end goal, kept topic count small
  - Tried 20, 30, 50, 70, 100, 150 topics
  - Although held-out log-perplexity decreased, topics more redundant, semantically incoherent
  - See: Chang, Jonathan, Sean Gerrish, Chong Wang, Jordan Boyd-Graber, and David M. Blei. "Reading tea leaves: How humans interpret topic models." *Advances in neural information processing systems*. 2009.
- Allowed Gensim to fit an asymmetric topic Dirichlet prior (alpha), generally a better fit than fixed, symmetric prior
  - see: Wallach, Hanna M., David M. Mimno, and Andrew McCallum. "Rethinking LDA: Why priors matter." *Advances in neural information processing systems*. 2009.

Directory Structure



Data Flow

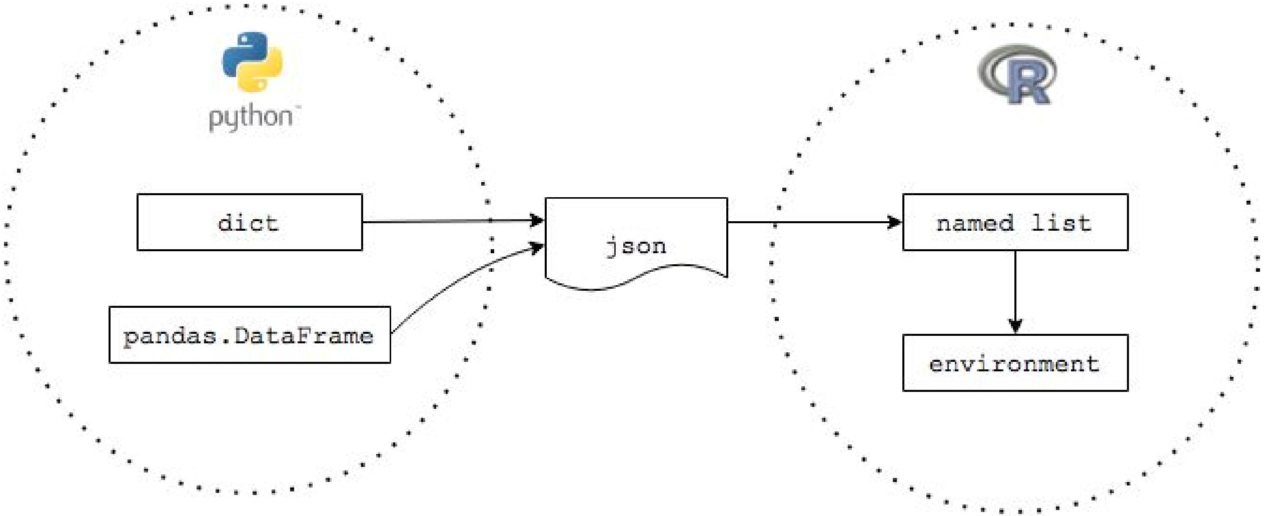

src
models.py
output.py
config
model_config.json
output_config.json
custom_stopwords.txt
source_text
foo.txt
bar.txt
...
foo_dir
bar_dir
...
models.py
text_utf8
foo.txt
bar.txt
foo_dir(full).txt
bar_dir(full).txt
...
models
bigrams_gensim
trigrams_gensim
dictionary_gensim
corpus_gensim.p
LDA_30_topics
...
log
model_log.txt
removed_stopwords.txt
output.py
shiny
data
docTopics.json
docTerms.json
docStats.json
topicTerms.json
idf.json
docDist.csv
server.R
ui.R
ingest_new_docs.py
config.json
server.R
ui.R

Data Structures/Formats

- Python needs to talk to R
- Tension between scalability and robustness: smaller ⇒ more fragile
- Ideally for a large corpus, doc-level stats (top terms, topic weights, idf dictionary) would be hashable by document/term name for fast user queries
- Python *dicts* do this
- R *environments* do as well

Data Structures/Formats

- JSON is an ideal 'glue' format for quickly translating between these types
  - not memory-efficient (redundant), but sparsity helps some:
    27.6MB of text ↦ 1.6MB of summary .json files (might want to do better for larger corpora)
- With future iterations in mind, JSON is native format for a custom JS app



Post-processing: document similarity

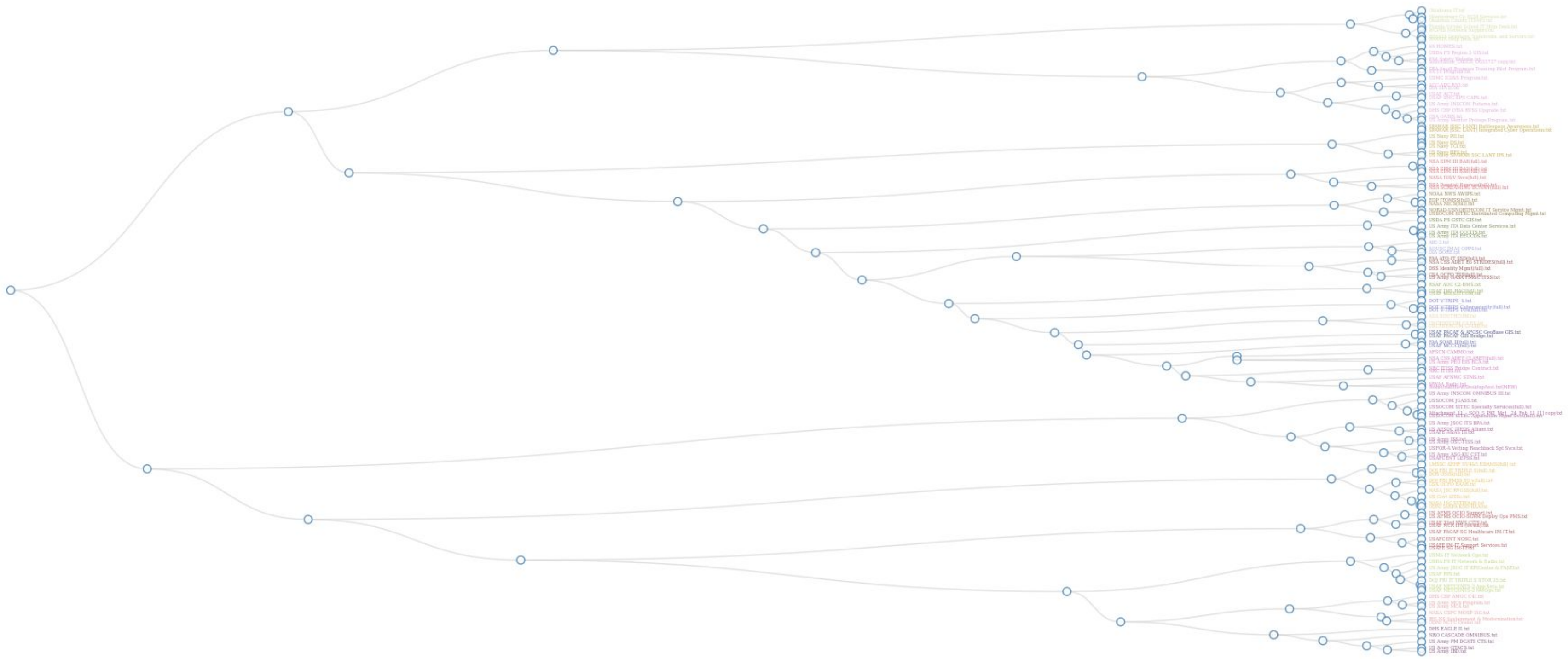
- Need a good similarity/distance measure
- What's the best notion of distance, given topic distributions? Candidates:
    - Cosine distance (*often outperformed by other metrics; better for tf-idf vectors*)
    - Kullback-Liebler Divergence (*asymmetric and often not defined for sparse distributions*)
    - Hellinger distance - basically a first-order approximation to:
    - Jensen-Shannon Divergence
        i. rooted in information theory
        ii. always defined
        iii. sqrt(J-S) is a proper metric in the mathematical sense - convenient
        iv. often among the best-performing (see, e.g.
        Huang, Anna. "Similarity Measures for Text Document Clustering." 2008.)

Post-processing: document clustering

- R reads docDist.csv, a doc-doc distance (J-S) matrix
- Performs agglomerative hierarchical clustering with hclust
- Labels a user-adjustable # of clusters (truncate the hclust merge tree
- Visualized with a zoomable dendrogram

Corpus Explorer

Document-Topic Network | Document Dendrogram | Document Stats | Upload New Documents



Font Size

3 — 5 — 12

3 4 5 6 7 8 9 10 11 12

Number of Clusters

1 — 20 — 50

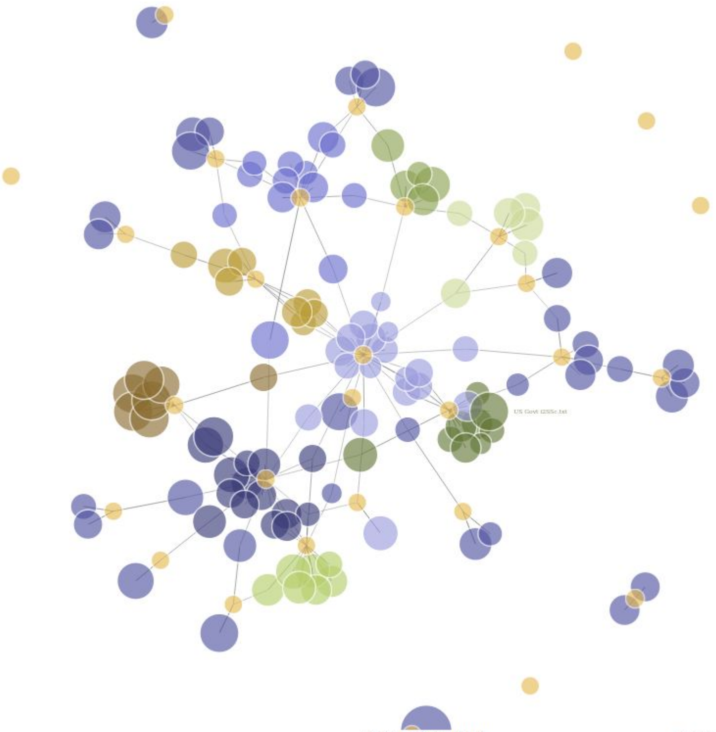
1 6 11 16 21 26 31 36 41 46 50

Post-processing: Doc-Topic network

- R reads docTopics.json, a sparse document-topic matrix (specifically, named list of small dataframes, one per doc)
- Builds a bipartite network with documents and topics as nodes
  - edge between doc D and topic T when D is more than p% topic T
  - threshold p is user-adjustable
  - edges are weighted by topic weight; more representative topics end up closer to their docs
  - nodes colored by cluster label from the hierarchical clustering

Corpus Explorer

Document-Topic Network | Document Dendrogram | Document Stats | Upload New Documents



Opacity

0.1 — 0.8 — 1

Number of Clusters

1 — 10 — 20

Topic Threshold

0 — 0.2 — 1

Documents

ACC-APG RS3.txt
AFSCN CAMMO.txt
AIE-3.txt
AOUSC JMAS OPPS.txt
ASA SOUTHCOM.txt

Topics

0 1 10 11 12 13 14 15 16 17
18 19 2 20 21 22 23 24 25 26
27 28 29 3 4 5 6 7 8 9

Future Work

- Better handling of OCR mistakes - splitting run-ons
- More comprehensive drill-down UI views - tweaking Shiny at the Javascript level
- Searchability of documents by topic, term, etc.
- Allow user to choose between multiple pre-computed topic models
- Allow user to upload new docs for inference and comparison
- Optimization
  - hierarchical clustering and force networks won't scale to larger corpora - other options?
    1. spherical or some other variant of k-means
    2. local doc-topic network centered on a subset of docs or topics
  - sparse doc-doc distance matrix

Questions?