

Building a Gigaword Corpus

Data Ingestion, Management, and Processing for NLP

Rebecca Bilbro

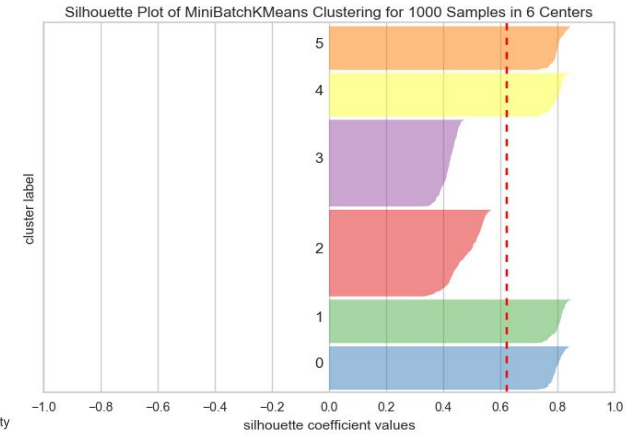
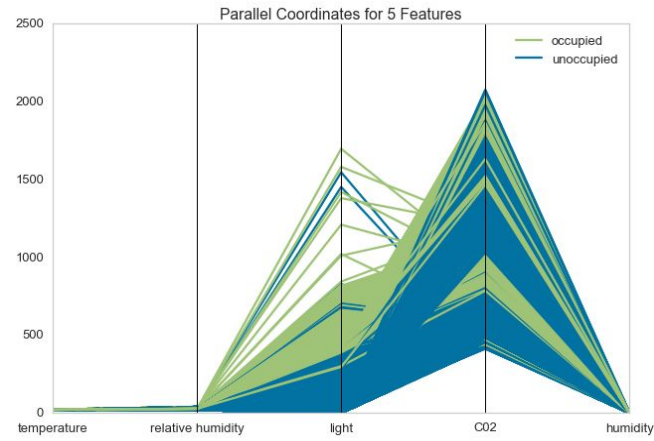
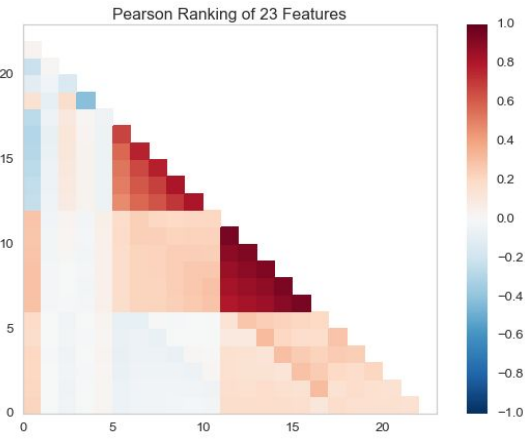
PyCon 2017

- Me and my motivation
- Why make a custom corpus?
- Things likely to go wrong
 - Ingestion
 - Management
 - Loading
 - Preprocessing
 - Analysis
- Lessons we learned
- Open source tools we made

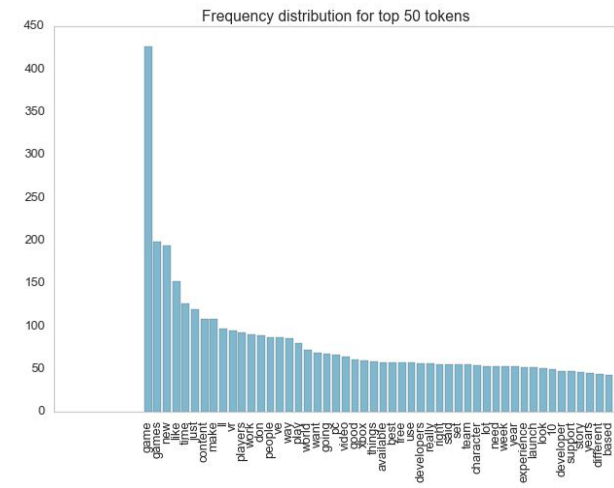
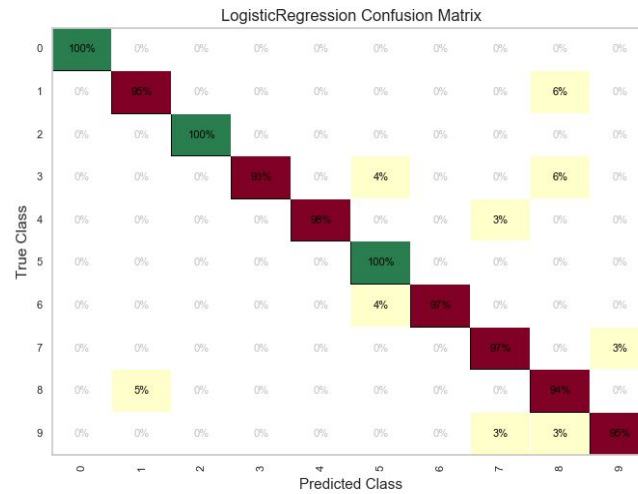
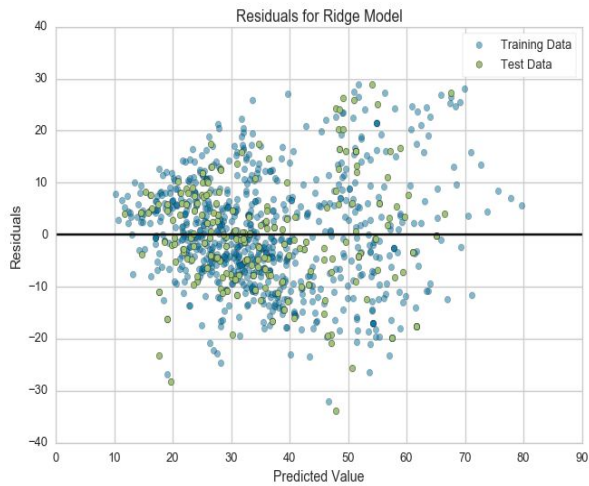
A photograph of the Jefferson Memorial at dusk, reflected in the water, with a shooting star in the sky and tree branches in the foreground. The scene is captured in a deep blue twilight, with the memorial's white dome and columns glowing against the dark sky. The water in the foreground is calm, creating a clear reflection of the building. A bright shooting star streaks across the sky above the memorial. The foreground is framed by dark, silhouetted tree branches.

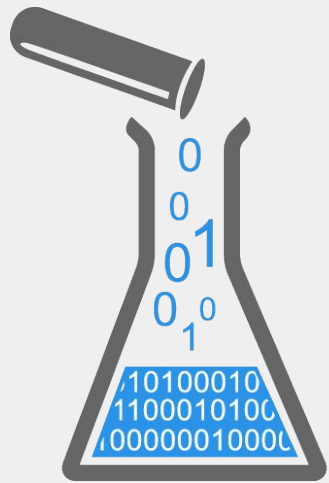
Rebecca Bilbro

Data Scientist



Yellowbrick



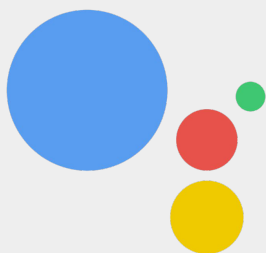


District**Data**Labs

Natural language processing



Everyone's doing it



theano

spaCy

TensorFlow



APACHE
Spark™

NLTK



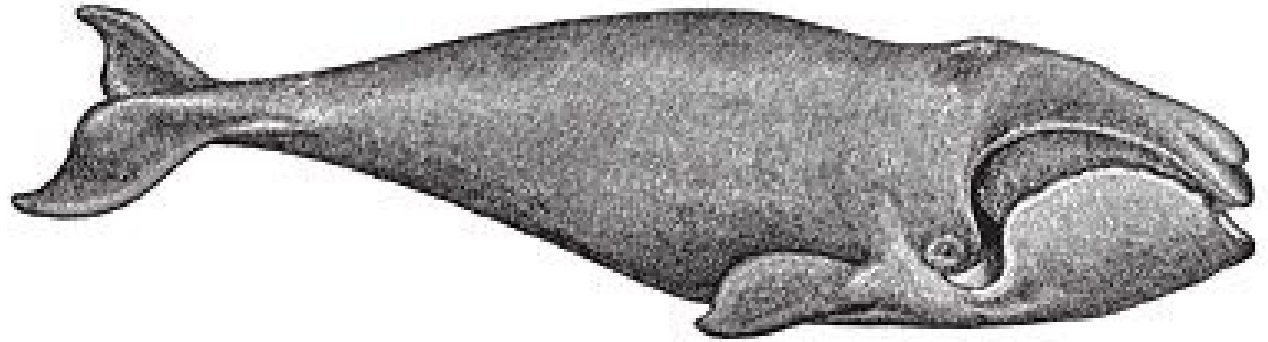
PYTORCH

gensim

Caffe



So many great tools



```
import nltk
```

```
moby = nltk.text.Text(nltk.corpus.gutenberg.words('melville-moby_dick.txt'))
```

```
print(moby.similar("ahab"))
```

```
print(moby.common_contexts(["ahab", "starbuck"]))
```

```
print(moby.concordance("monstrous", 55, lines=10))
```

The Natural Language Toolkit



```
import bz2
import gensim

# Load id to word dictionary
id2word = gensim.corpora.Dictionary.load_from_text('wikipedia_wordids.txt')

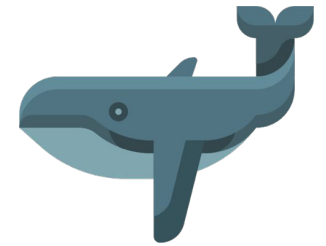
# Instantiate iterator for corpus (which is ~24.14 GB on disk after compression!)
mm = gensim.corpora.MmCorpus(bz2.BZ2File('wikipedia_tfidf.mm.bz2'))

# Do Latent Semantic Analysis and find 10 prominent topics
lsa = gensim.models.lsimodel.LsiModel(corpus=mm, id2word=id2word, num_topics=400)
lsa.print_topics(10)
```

Gensim + Wikipedia

A custom corpus





```
import os
import requests
import feedparser

feed = "http://feeds.washingtonpost.com/rss/national"

for entry in feedparser.parse(feed)['entries']:
    r = requests.get(entry['link'])
    path = entry['title'].lower().replace(" ", "-") + ".html"

    with open(path, 'wb') as f:
        f.write(r.content)
```

RSS

Ingestion

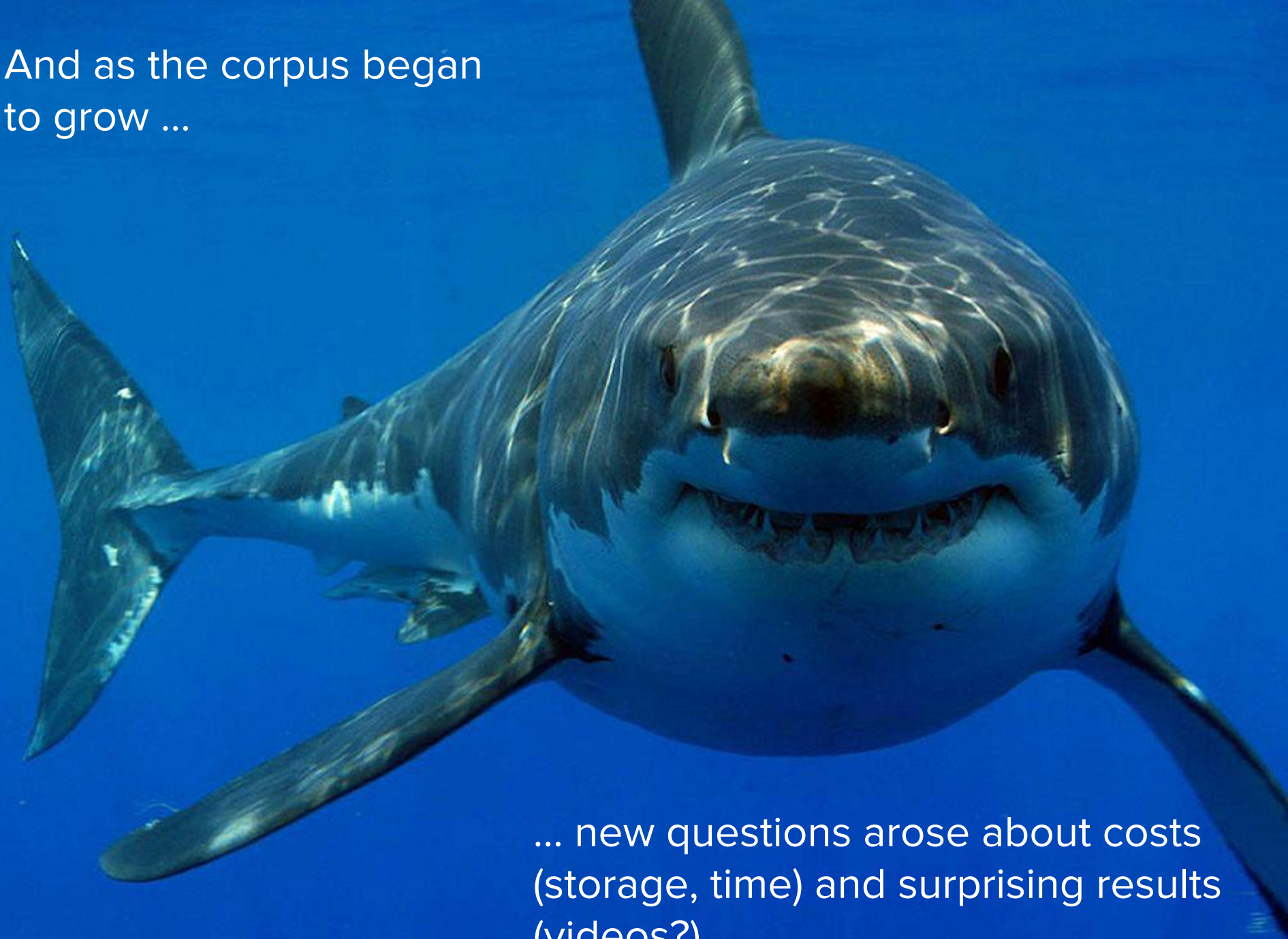
- Scheduling
- Adding new feeds
- Synchronizing feeds, finding duplicates
- Parsing different feeds/entries into a standard form
- Monitoring

Storage

- Database choice
- Data representation, indexing, fetching
- Connection and configuration
- Error tracking and handling
- Exporting



And as the corpus began
to grow ...



... new questions arose about costs
(storage, time) and surprising results
(videos?).


Production-grade


ingestion: Baleen


Configuration
+ logging + database + flags


Utilities
+ timez

Admin
+ ingest_feeds() + ingest_opml() + summary() + run() + export()


 Logging
+ logger + mongolog

 Ingest
+ feeds() + started() + finished() + process() + ingest()

 Feed Sync
+ parse() + sync() + entries()

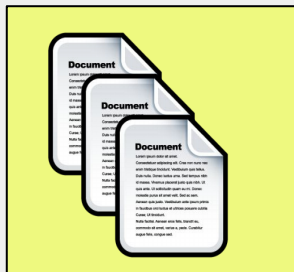
 Post Wrangler
+ wrangle() + fetch()





 OPML Reader
+ categories() + counts() + __iter__() + __len__()

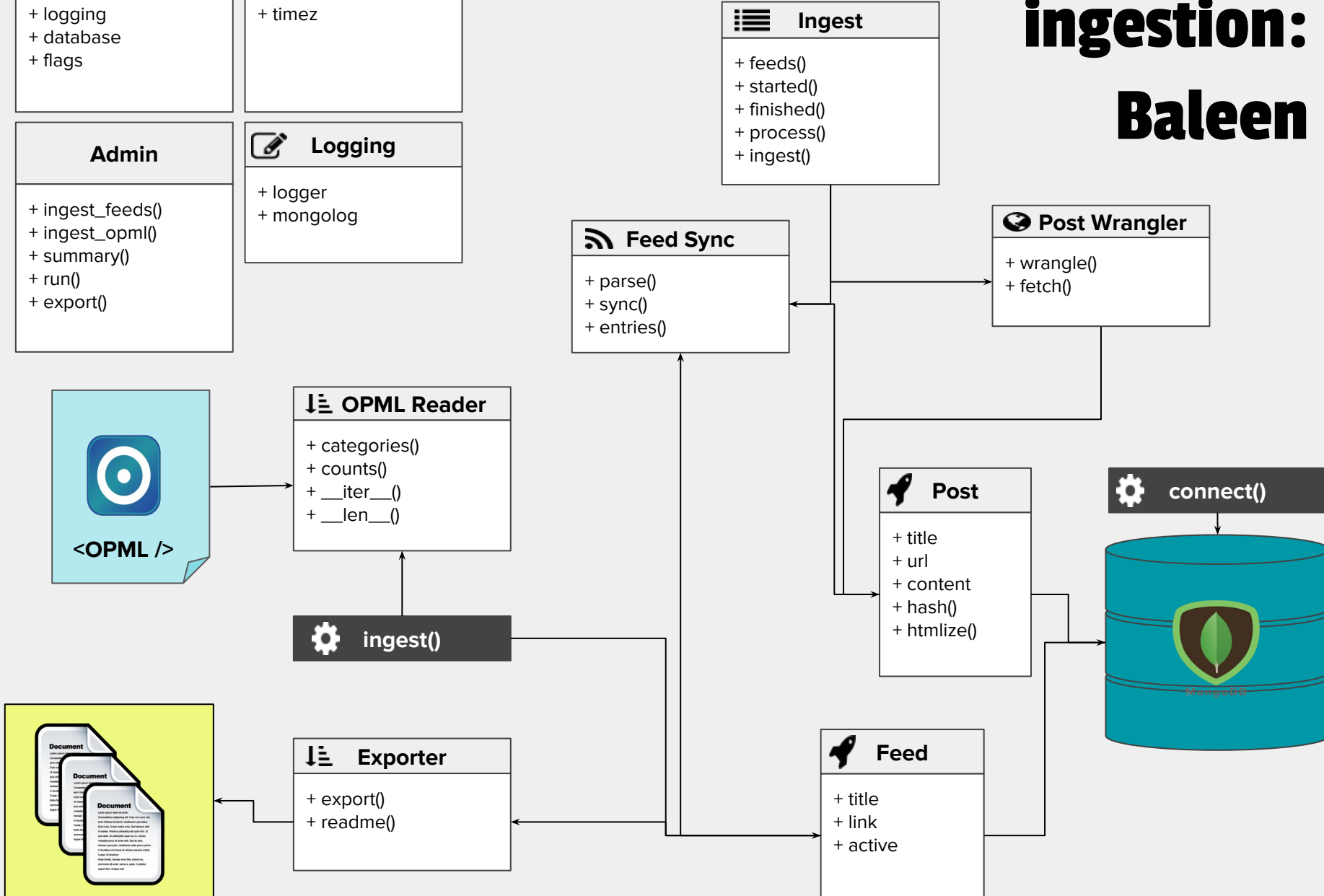
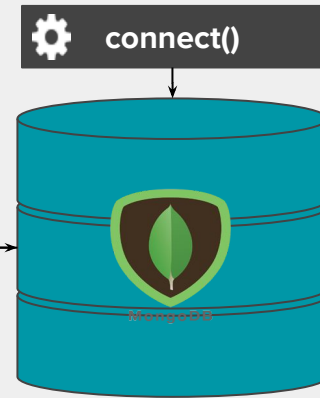
 ingest()

 Exporter
+ export() + readme()



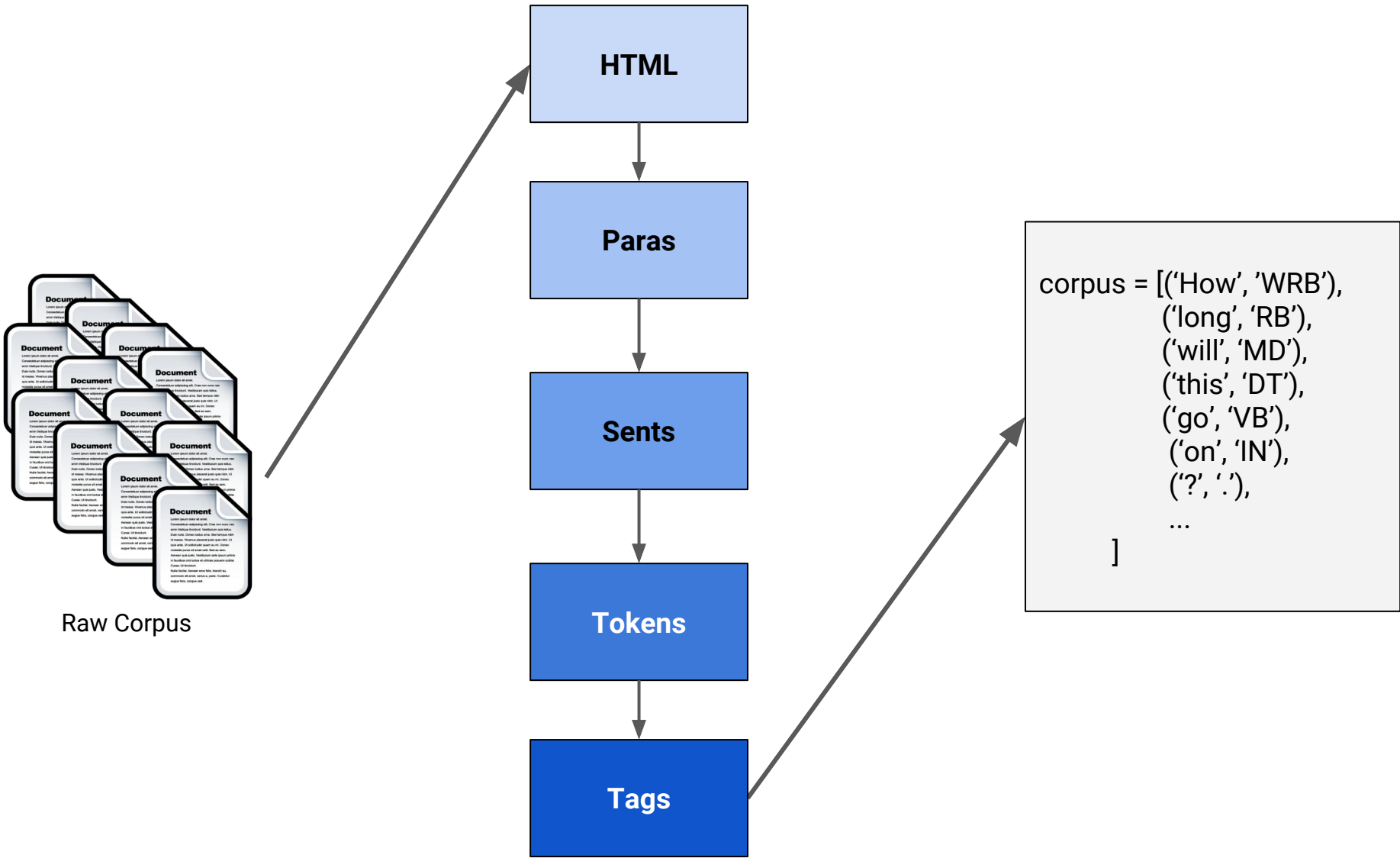
 Post
+ title + url + content + hash() + htmlize()

 Feed
+ title + link + active

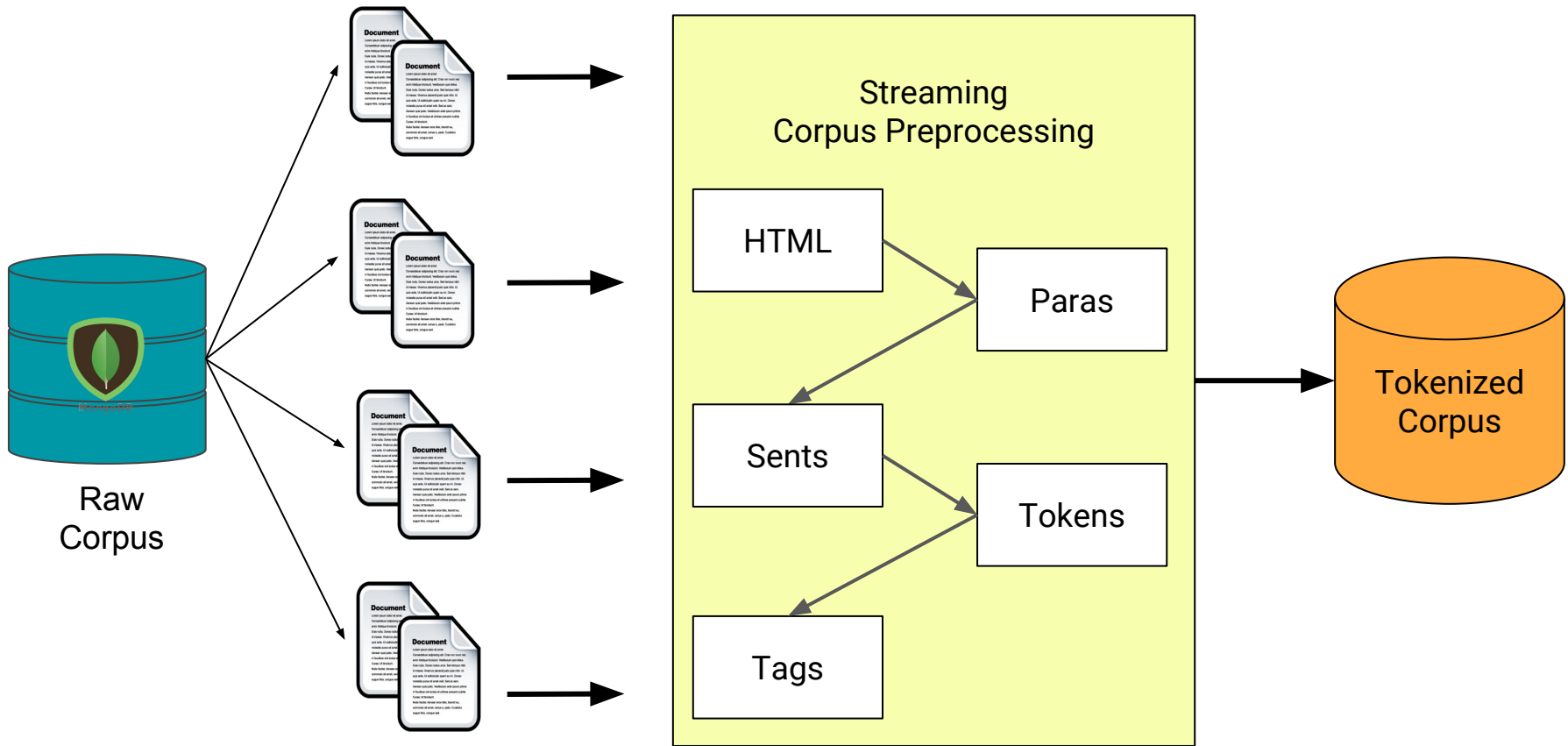


Raw corpus != Usable data





From each doc, extract html, identify paras/sents/words, tag with part-of-speech



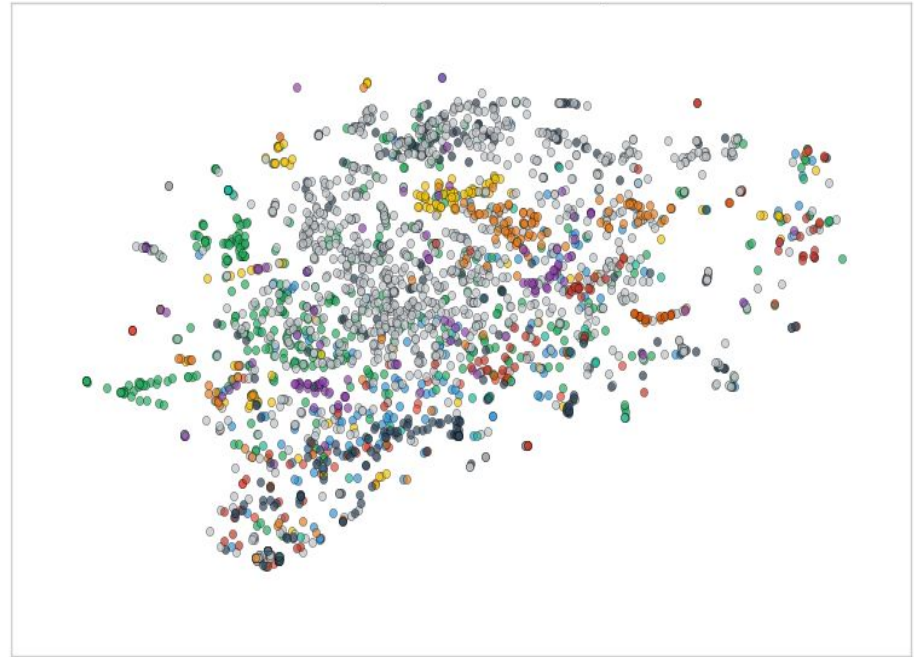
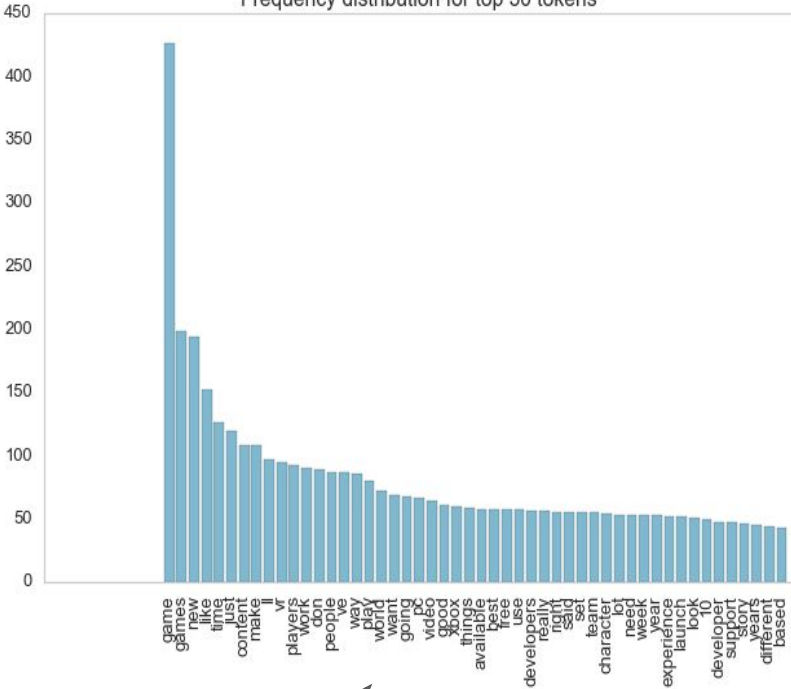
CorpusReader for streaming access, preprocessing, and saving the tokenized version

A dense field of colorful seashells in various colors like blue, yellow, orange, and white. The shells are scattered across the entire frame, creating a complex and varied visual texture. The colors range from deep blues and purples to bright yellows and oranges, with many shells showing intricate patterns and textures. The shells are of various sizes and shapes, some appearing as smooth, rounded pebbles and others as more angular, broken pieces. The overall effect is a rich, multi-colored mosaic of natural objects.

Vectorization
...so many features

Yellowbrick

Frequency distribution for top 50 tokens

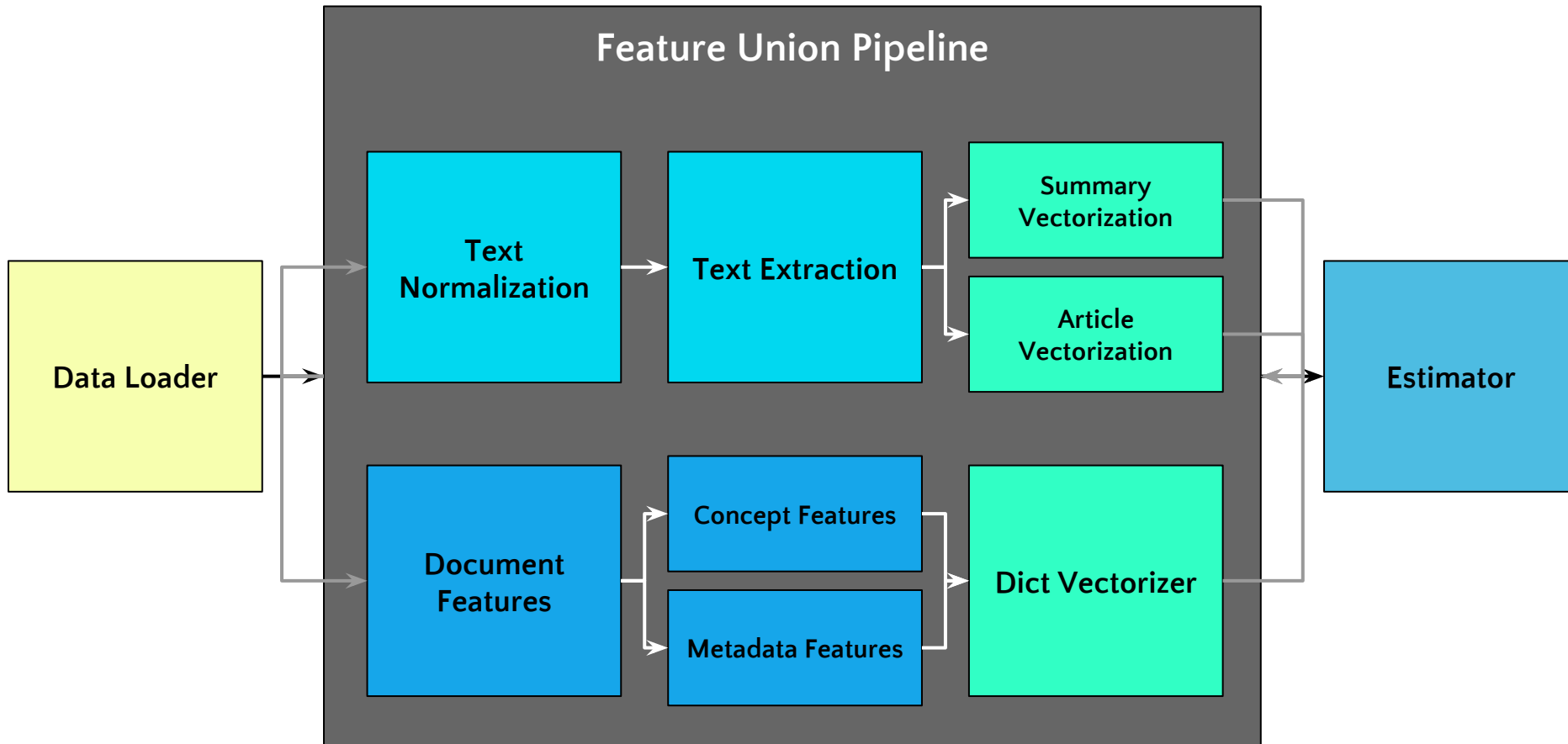
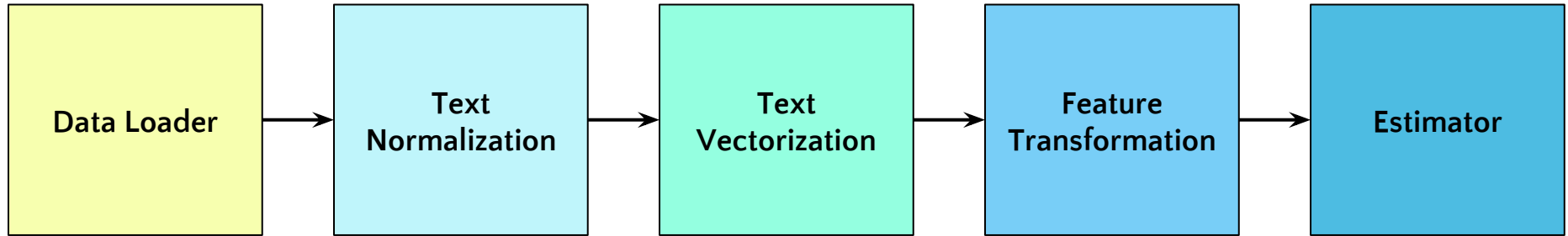


- design
- tech
- business
- gaming
- politics
- news
- cooking
- data_science
- sports
- cinema
- books
- do_it_yourself

visualize top tokens,
document distribution
& part-of-speech
tagging

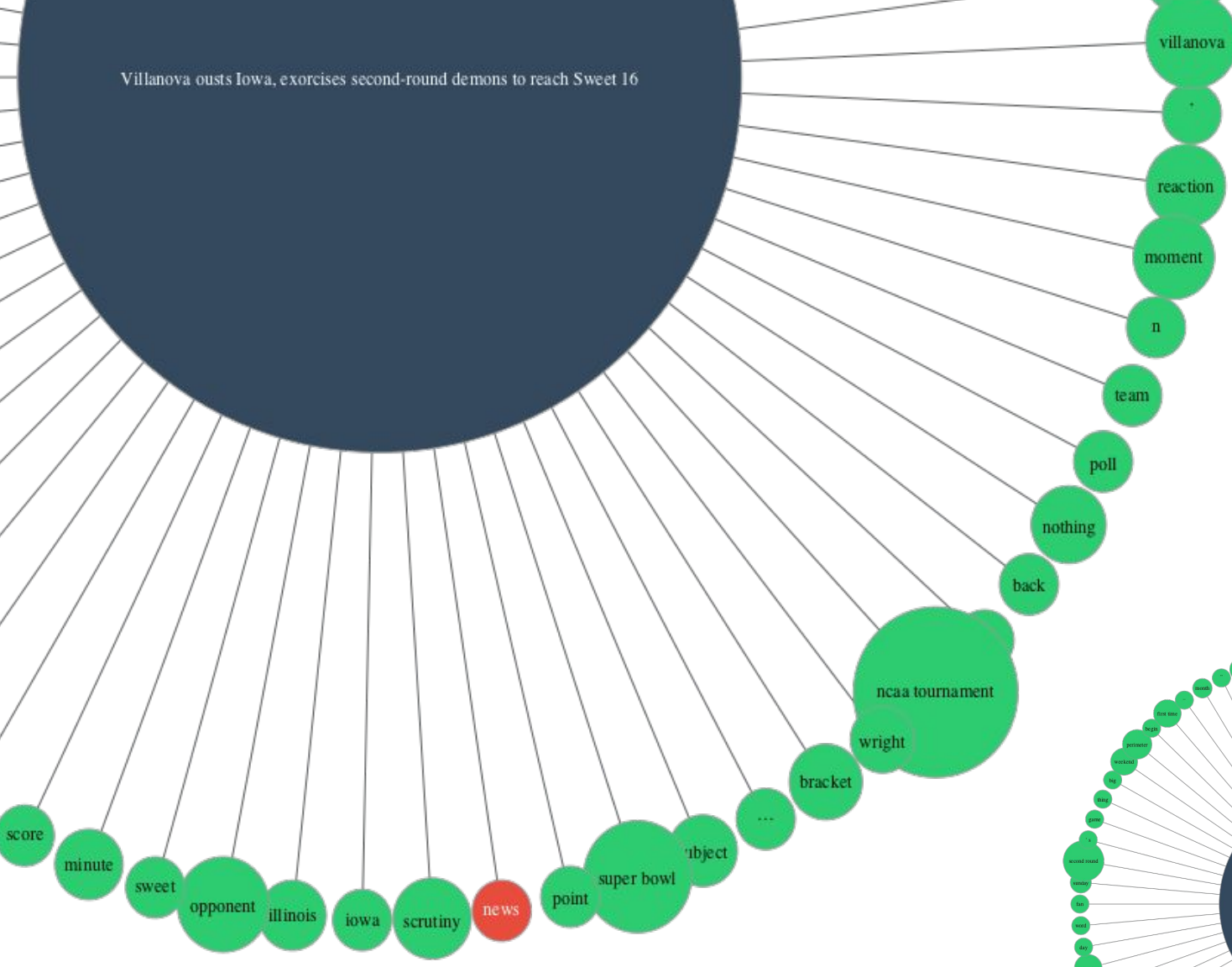
Algebra (from Arabic `al-jabr` meaning `reunion of broken parts`) is one of the broad parts of mathematics , together with number theory , geometry and analysis . In its most general form , algebra is the study of mathematical symbols and the rules for manipulating these symbols ; it is a unifying thread of almost all of mathematics .

Minke



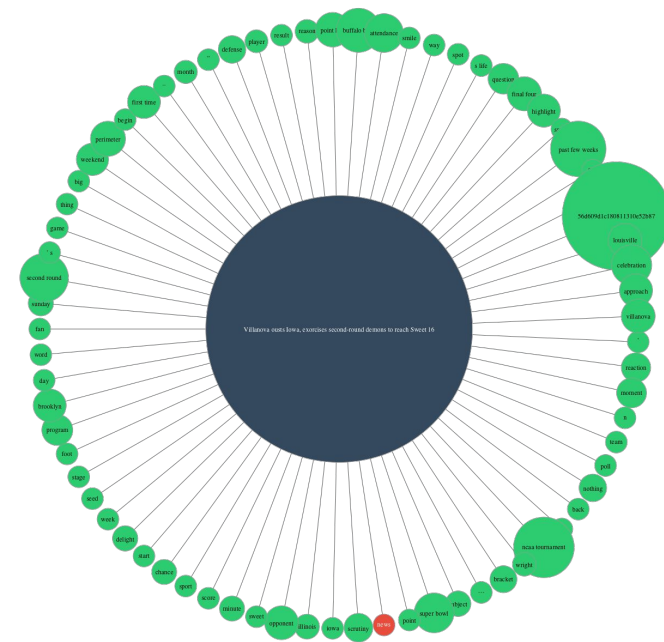
Dynamic graph analysis





Keyphrase Graph:

- 2.7 M nodes
- 47 M edges
- Average degree of 35



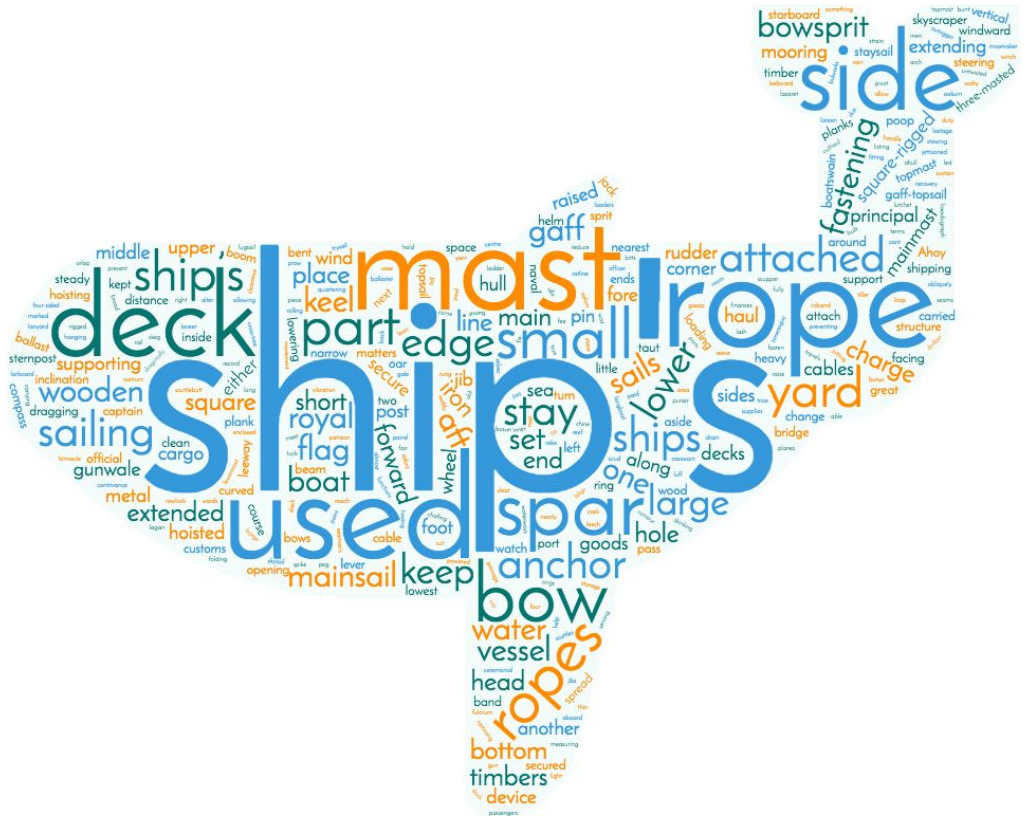
1.5 M documents, 7,500 jobs, 524 GB (uncompressed)

Lessons learned



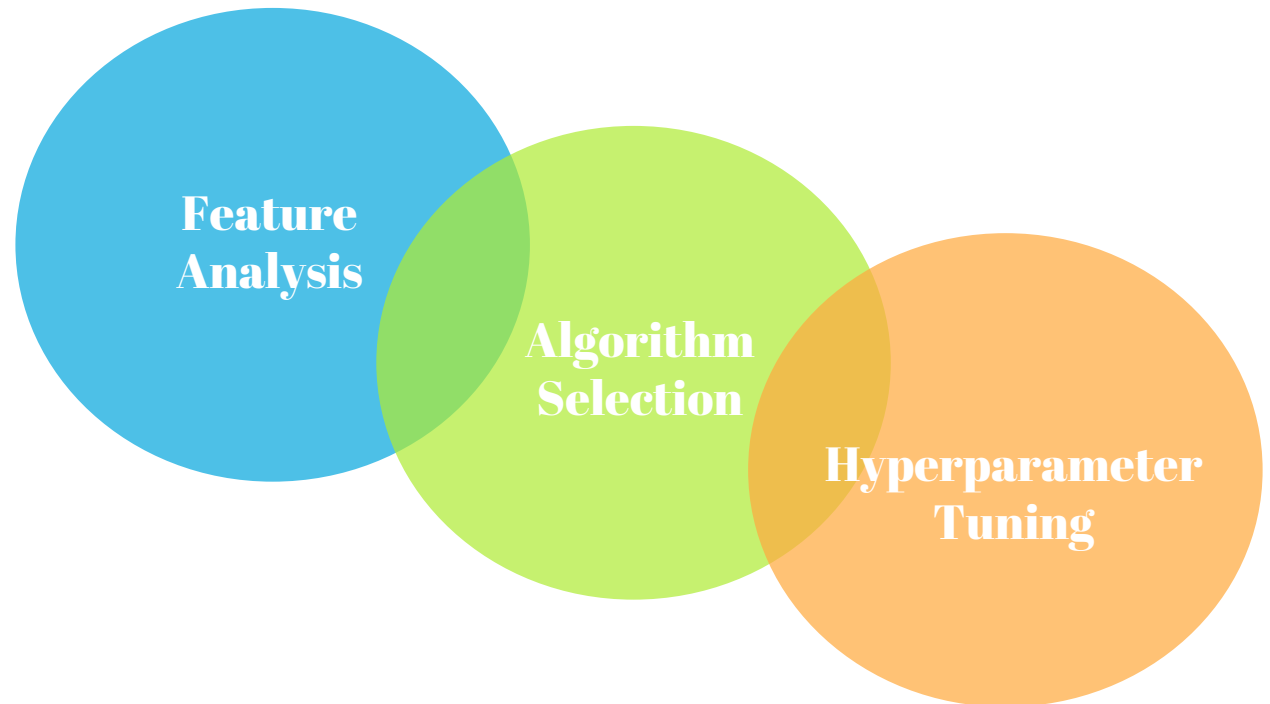
Meaningfully literate data products rely on...

...a custom, domain-specific corpus.



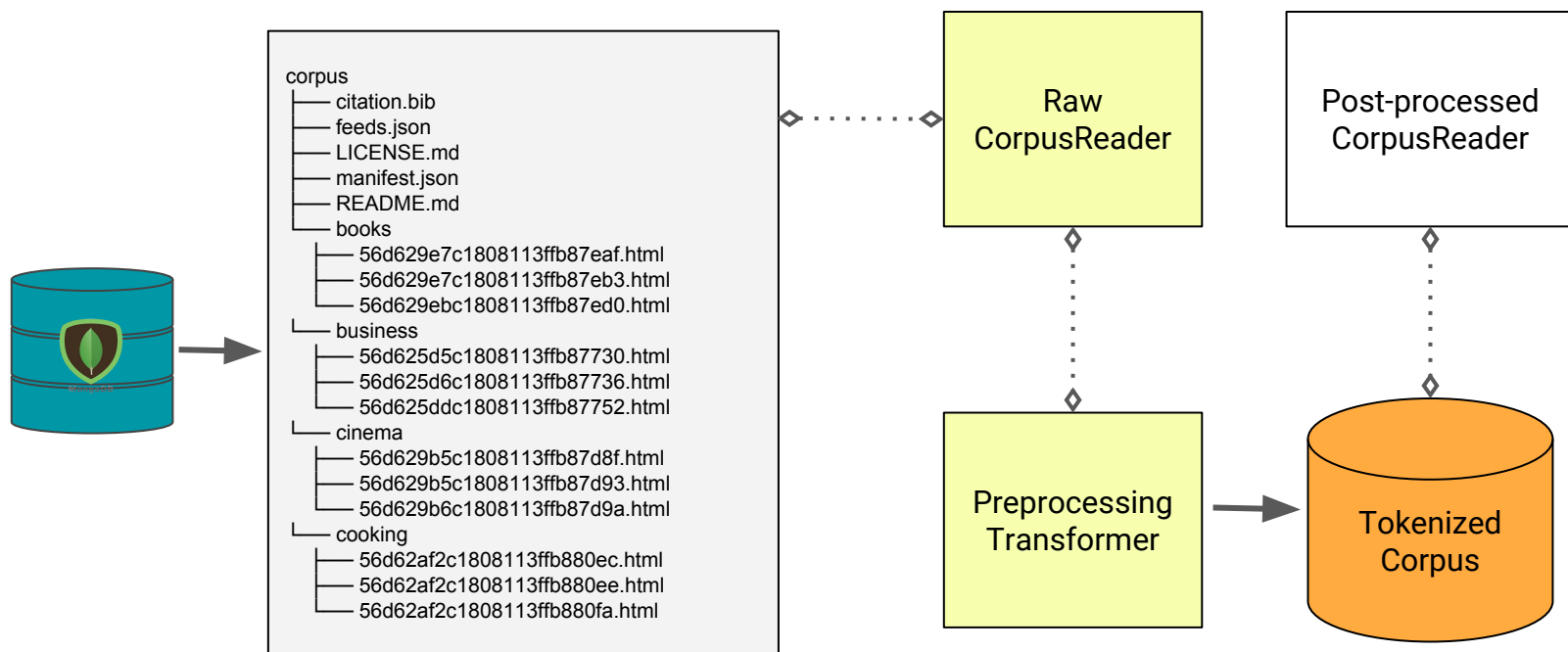
Meaningfully literate data products rely on...

...a data management layer for flexibility and iteration during modeling.



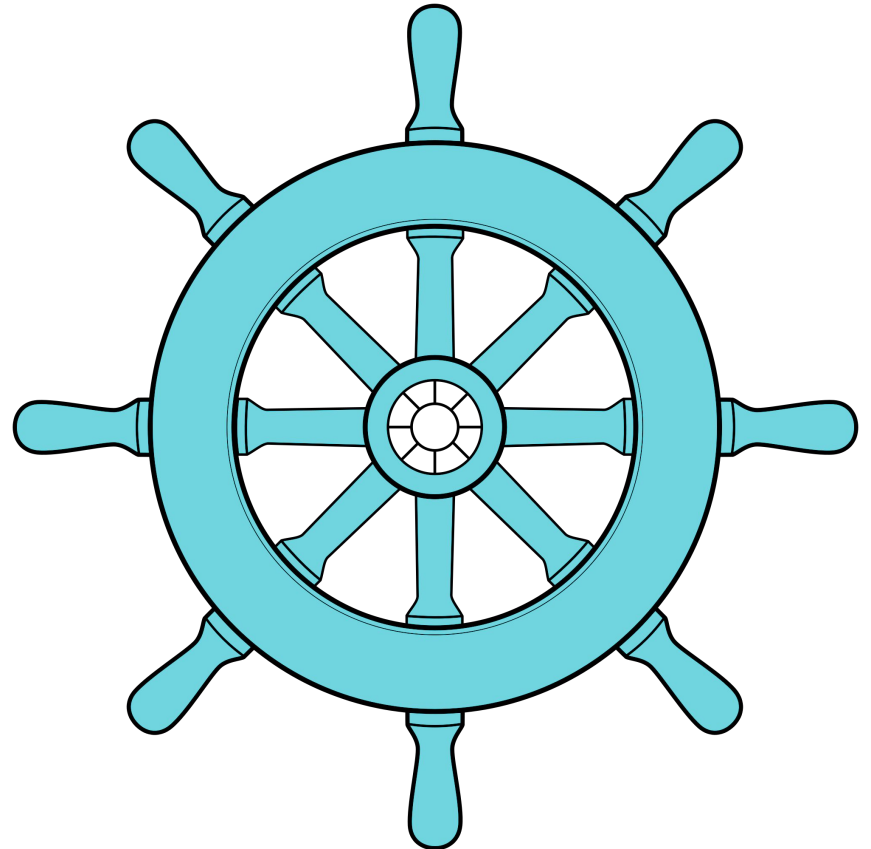
Meaningfully literate data products rely on...

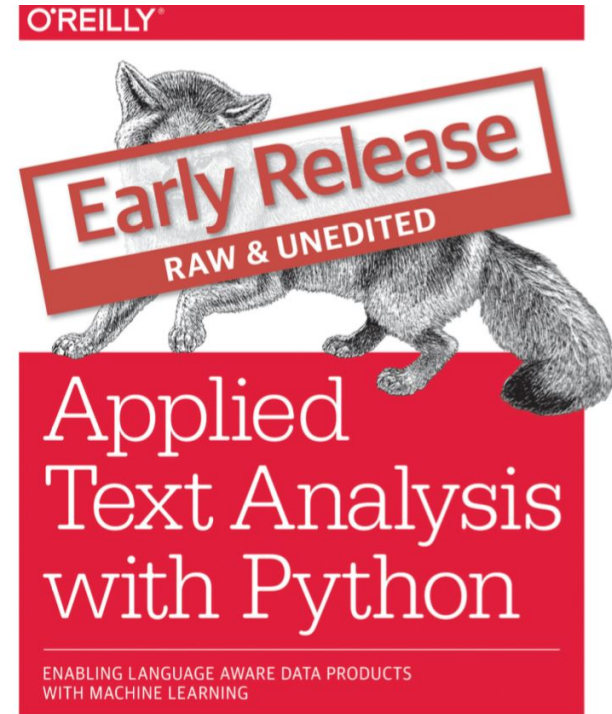
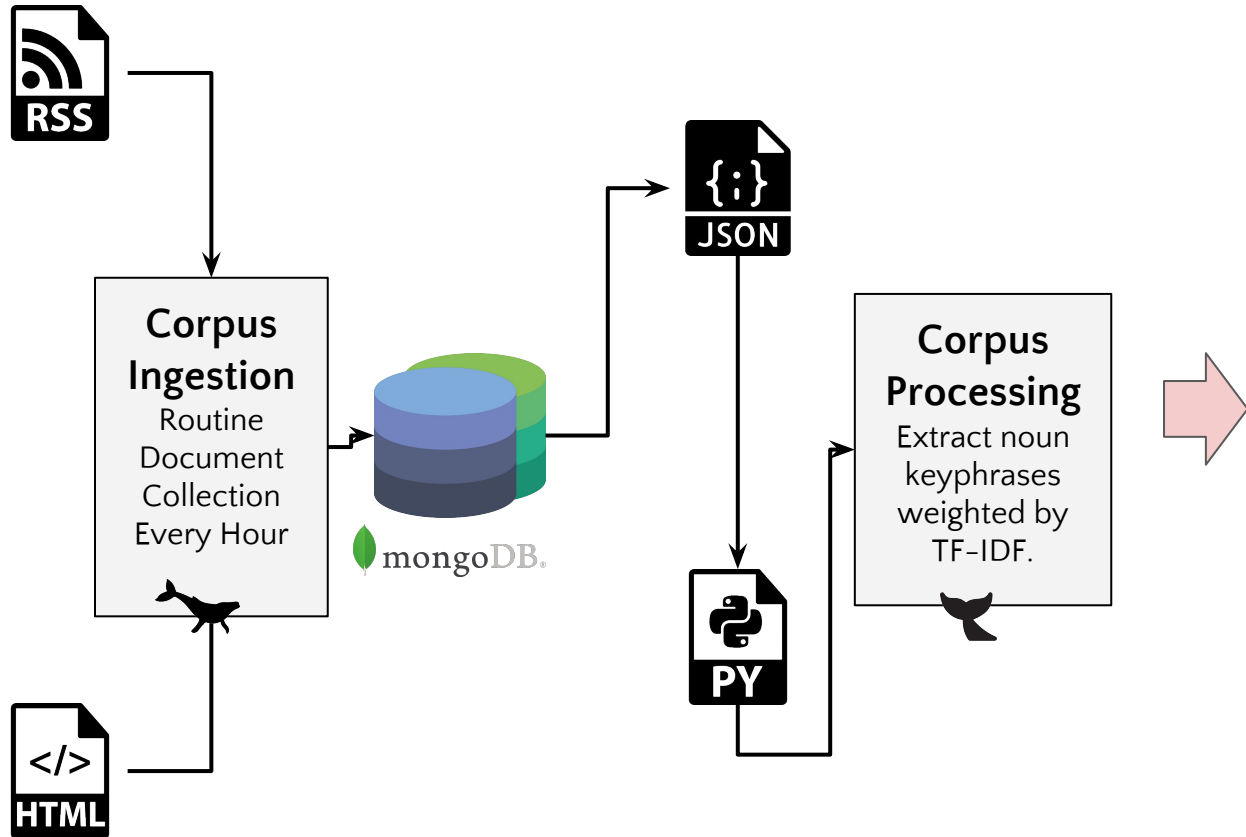
...a custom CorpusReader for streaming, and also intermediate storage.



Meaningfully literate data products rely on...

...visual steering and graph analysis for interpretation.





Benjamin Bengfort,
Rebecca Bilbro & Tony Ojeda

Baleen & Minke

Fork me on GitHub

Baleen

An automated ingestion service for blogs to construct a corpus for NLP research.

build passing coverage 61% docs latest Ready 3



Minke

Graph extraction and NLP analysis for Baleen Corpora

build passing coverage 15% health 80% Ready 3



Yellowbrick

Fork me on GitHub

Yellowbrick

build passing coverage 80% health 91% docs latest Ready 8

Visual analysis and diagnostic tools to facilitate machine learning model selection.



Getting data

- **(Tutorial)** “Fantastic Data and Where to Find Them” by Nicole Donnelly
- **(Poster)** “On the Hour Data Ingestion” by Benjamin Bengfort and Will Voorhees

Speed to insight

- **(Talk)** “Human-Machine Collaboration” by Tony Ojeda
- **(Poster)** “A Framework for Exploratory Data Analysis” by Tony Ojeda and Sasan Bahadaran

Machine learning

- **(Poster)** “Model Management Systems” by Benjamin Bengfort and Laura Lorenz
- **(Poster)** “Yellowbrick” by Benjamin Bengfort and Rebecca Bilbro

Also, **sprints!**

pycon.districtdatalabs.com



Thank you!

Rebecca Bilbro

Twitter: twitter.com/rebeccabilbro

Github: github.com/rebeccabilbro

Email: rebecca.bilbro@bytecubed.com