Telling the Tale

Narratives and Emotional Intensity in Political Texts

Elliott Ash
ETH Zürich

SciPy 2021
Computational Social Science

Telling the (Scientific) Tale, with Python

- Time to preach to the choir:
  - Python is the best language for scientific programming.
  - Python is fun!

Telling the (Scientific) Tale, with Python

- Time to preach to the choir:
  - Python is the best language for scientific programming.
  - Python is fun!
- My team use Python for computational social science, mainly for empirical analysis using text data.

Overview

- Today I will introduce three projects showing what we can do:
  1. Measuring emotionality in text using embeddings.
  2. Unsupervised text classification using transformers.
  3. Mining narratives with semantic parsing.

Overview

- Today I will introduce three projects showing what we can do:
  1. Measuring emotionality in text using embeddings.
  2. Unsupervised text classification using transformers.
  3. Mining narratives with semantic parsing.
- We have (beta) open source code for our tools!

`https://bit.ly/Ash-SciPy`

Outline

Gennaro and Ash (2021)

"In politics, when reason and emotion collide, emotion invariably wins."

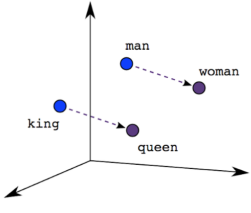
–Drew Westen

Corpus

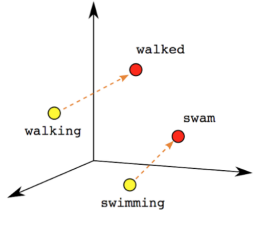
- Universe of floor speeches in U.S. Congress 1858-2014
  - Exclude non-speech content such as roll calls, bill sponsorships, and legislation (N=9 799 375)
- Pre-Processing:
  - Keep only nouns, adjectives, and verbs **[nltk.corpus.wordnet]**
  - Drop punctuation, capitalization, numbers, stopwords (including names of states, cities, politicians), and word endings (**[nltk.stem.snowball]**)
  - Drop tokens occurring in less than 10 speeches (63 334 words)

Word Embeddings

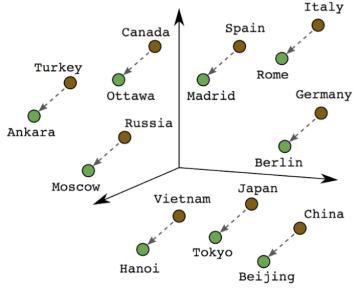
- Train a Word2Vec model with *gensim* on the corpus (300 dimensions, 8-word window) **[gensim.models.word2vec]**



Male-Female



Verb Tense



Country-Capital

Word Embedding is a technology from computational linguistics that represents words and phrases as vectors in a geometric space, where locations and directions encode meaning (Mikolov et al 2013)

Emotion and Reason Vectors

- Construct lexicons of emotion and reason words.
- Each lexicon gets a vector – its centroid in the word embedding space.
  - $\vec{A}$ for affect/emotion, $\vec{C}$ for cognition/reason **[numpy]**

[wordcloud]

Emotion and Reason in Speeches

- A document vector $\vec{d}_i$ is the weighted average of the vectors for each word in the speech.
  - $\vec{A}$ for affect/emotion, $\vec{C}$ for cognition/reason
- Relative emotionality of d is

$$Y_i = \frac{\text{sim}(\vec{d}_i, \vec{A}) + 1}{\text{sim}(\vec{d}_i, \vec{C}) + 1}$$

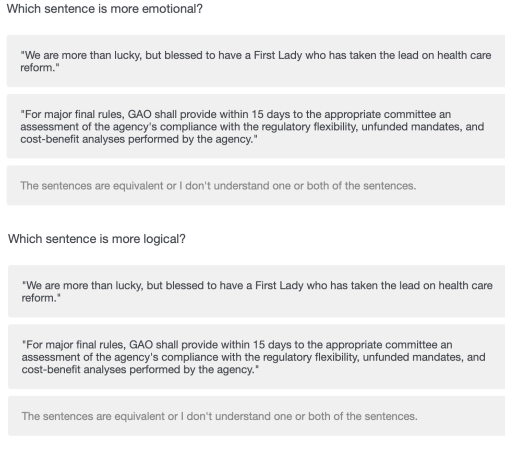
  - sim $= 1$ - **scipy.spatial.distance.cosine**

Which column is emotive and which is cognitive?

- Unlike the abortionist, Mr Speaker, who grows filthy rich by grinding, suctioning, poisoning, and dismembering the fragile bodies of unborn babies, the pro-lifers outside these baby slaughterhouses give of themselves to help both mother and child.
- Denying Federal funds to poor women for abortions while making available more money to wealthy women to do the same, If they so choose, simply does not make sense.
- We need a new approach because at the start of this decade we had the most murders, the worst schools, the most abortions, the highest infant mortality rate, the most illegitimacy, the most one-parent families, the most children in Jail, and the most children on Government aid in the world.

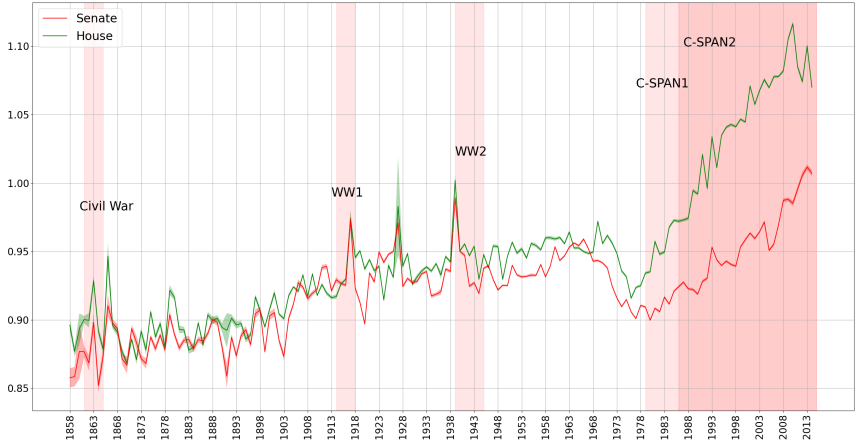
- The cases cited, reaffirmed in Casey, recognize that a State cannot subject women's health to significant health risks both in that context, and also where State regulations force women to use riskier methods of abortion.
- This Panel was convened to review the ethical considerations surrounding the use of tissue from induced abortions in federally funded human transplantion research.
- And I should note here that during the period in which DOD allowed private paid abortions in military facilities overseas, the regulations then clearly permitted third-trimester abortion only to preserve the life or physical health of the woman.

Human Validation Survey on M-Turk



- Accuracy is over 90% and stable across decades since the 1850s.

Results: Emotionality by Chamber over Time



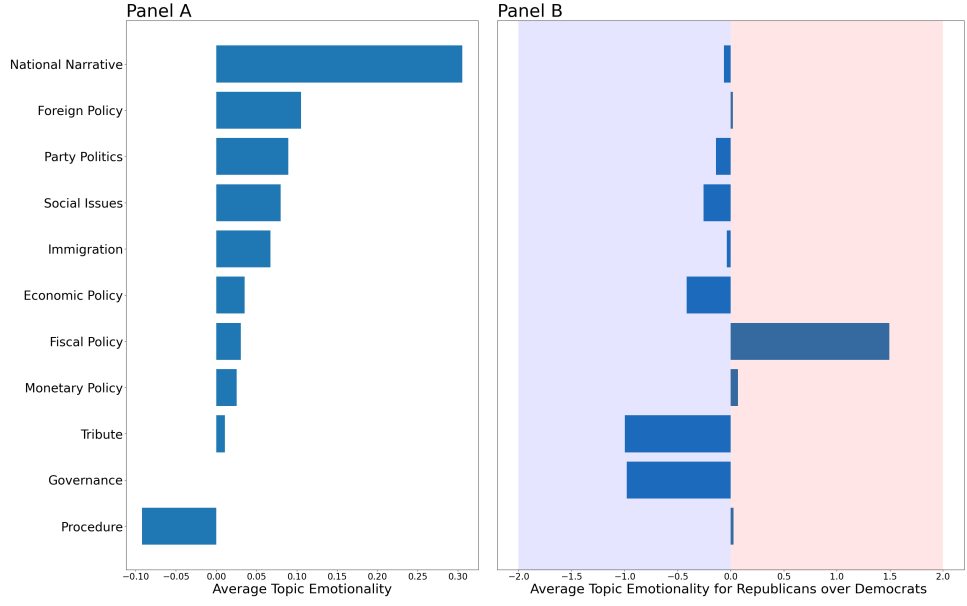
[matplotlib]

Emotionality by Topic

[gensim.models.wrappers.ldamallet]

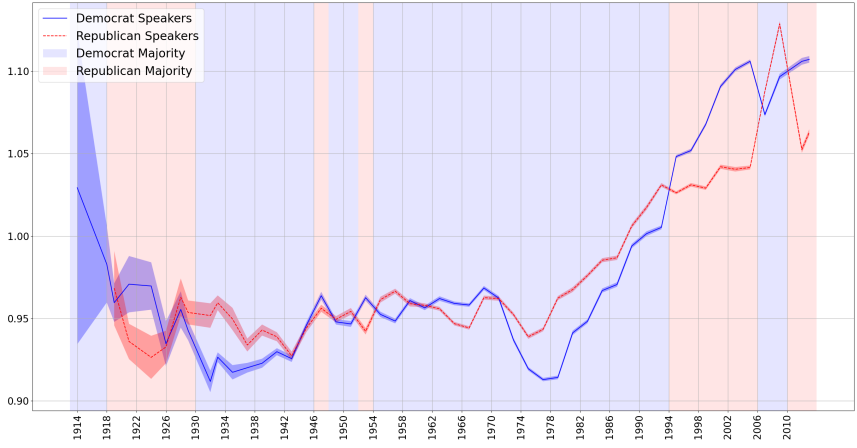
Emotionality by Topic

[gensim.models.wrappers.ldamallet]

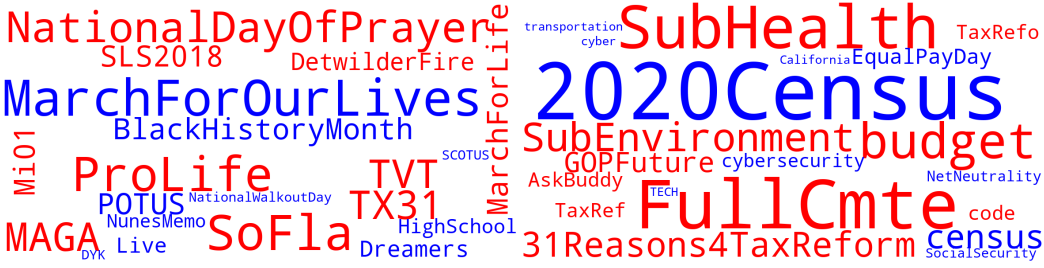


Emotionality by Minority/Majority Status

Emotionality by Minority/Majority Status



Application to Tweets: Most emotional/rational #HashTags



Top 10 Emotional (Left) and Cognitive (Right) Hashtags for Democrats and Republicans

Size is the odds ratio of belonging to the Rep-Emotional or Dem-Emotional class

EmotionMeter Python Package

- pull from `https://github.com/elliottash/emotionmeter.git`
- notebook: `https://bit.ly/Ash-SciPy`

```
!git clone https://github.com/elliottash/emotionmeter.git
from emotionmeter import EmotionMeter
meter = EmotionMeter(text_column='doc')
# compute emotion scores (takes a couple of minutes)
df = meter.calculate_score(docs)

# resulting data frame
df.head()
```

	id	doc	affection	cognition	ratio
0	9.845497e+16	Republicans and Democrats have both created ou...	0.561333	0.637114	0.953711
1	1.234653e+18	I was thrilled to be back in the Great city of...	0.690971	0.714896	0.986049
2	1.304875e+18	The Unsolicited Mail In Ballot Scam is a major...	0.624300	0.790885	0.906982
3	1.223641e+18	Getting a little exercise this morning!	0.624831	0.649300	0.985164
4	1.215248e+18	Thank you Elise!	0.310523	0.185268	1.105676

Outline

What is this about?

- The starting question in human and machine reading of text documents
- Previous methods:
  - word lists / rules
  - hand coding / supervised classification
  - topic models (eg LDA / STM)
  - clustering (e.g. k-means) applied to document vectors

What is this about?

- The starting question in human and machine reading of text documents
- Previous methods:
  - word lists / rules
  - hand coding / supervised classification
  - topic models (eg LDA / STM)
  - clustering (e.g. k-means) applied to document vectors
- DocSCAN (Stammbach and Ash 2021):
  - Apply tools from transformer-based NLP (sentence encoding) and computer vision – **Semantic Clustering by Adopting Nearest-neighbors (SCAN)** – to learn document topics without any labels.

SCAN: Semantic Clustering by Adopting Nearest-neighbors

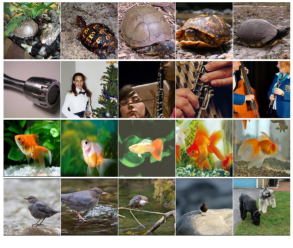


Fig. 1: Images (first column) and their nearest neighbors (other columns) [51].

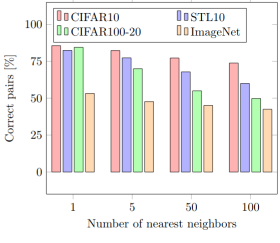


Fig. 2: Neighboring samples tend to be instances of the same semantic class.

SCAN: Semantic Clustering by Adopting Nearest-neighbors

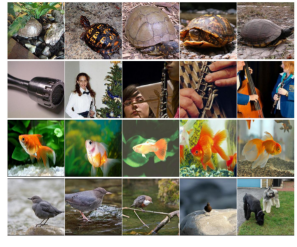


Fig. 1: Images (first column) and their nearest neighbors (other columns) [51].

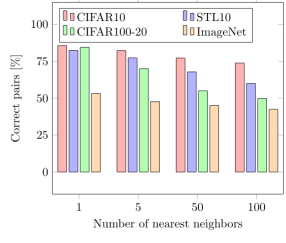
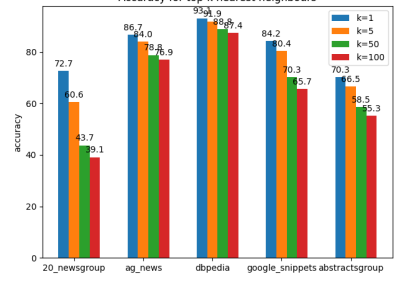


Fig. 2: Neighboring samples tend to be instances of the same semantic class.

- SCAN is an unsupervised classification method that relies on the regularity that objects in the same class tend to co-locate in embedding space.
- We find that in text classification datasets, neighboring documents in the S-BERT embedding space are usually in the same class.



SCAN Algorithm

1. form initial representations $\boldsymbol{x} \in \mathbb{R}^L$ via self-learning
   - SCAN: pre-trained image encoders (e.g. SimCLR)
   - DocSCAN: pre-trained document encoders (e.g. S-BERT, with $L = 768$ dimensions) [sentence_transformers]

SCAN Algorithm

1. form initial representations $\boldsymbol{x} \in \mathbb{R}^L$ via self-learning
   - SCAN: pre-trained image encoders (e.g. SimCLR)
   - DocSCAN: pre-trained document encoders (e.g. S-BERT, with $L = 768$ dimensions) [sentence_transformers]
2. assume K topics, each observation randomly initialized to topic shares $\boldsymbol{y} \in [0, 1]^K$, $\sum_{k \in K} y_k = 1$.

SCAN Algorithm

1. form initial representations $\boldsymbol{x} \in \mathbb{R}^L$ via self-learning
   - SCAN: pre-trained image encoders (e.g. SimCLR)
   - DocSCAN: pre-trained document encoders (e.g. S-BERT, with $L = 768$ dimensions) [sentence_transformers]
2. assume K topics, each observation randomly initialized to topic shares $\boldsymbol{y} \in [0, 1]^K$, $\sum_{k \in K} y_k = 1$.
3. positive samples = neighboring $\boldsymbol{x}$ in the initial representation space
   - for each observation i, pick n closest image or document vectors. [faiss]

SCAN Algorithm

1. form initial representations $\boldsymbol{x} \in \mathbb{R}^L$ via self-learning
   - SCAN: pre-trained image encoders (e.g. SimCLR)
   - DocSCAN: pre-trained document encoders (e.g. S-BERT, with $L = 768$ dimensions) [sentence_transformers]
2. assume K topics, each observation randomly initialized to topic shares $\boldsymbol{y} \in [0, 1]^K$, $\sum_{k \in K} y_k = 1$.
3. positive samples = neighboring $\boldsymbol{x}$ in the initial representation space
   - for each observation i, pick n closest image or document vectors. [faiss]
4. SCAN objective: learn transformation of embeddings to classes (next slide)

SCAN Algorithm

1. form initial representations $\boldsymbol{x} \in \mathbb{R}^L$ via self-learning
   - SCAN: pre-trained image encoders (e.g. SimCLR)
   - DocSCAN: pre-trained document encoders (e.g. S-BERT, with $L = 768$ dimensions) [sentence_transformers]
2. assume K topics, each observation randomly initialized to topic shares $\boldsymbol{y} \in [0, 1]^K$, $\sum_{k \in K} y_k = 1$.
3. positive samples = neighboring $\boldsymbol{x}$ in the initial representation space
   - for each observation i, pick n closest image or document vectors. [faiss]
4. SCAN objective: learn transformation of embeddings to classes (next slide)
5. self-label the K clusters based on confident examples (with highest $\max_k y_k$).

SCAN Objective [implemented in PyTorch]

1. training sample: randomly picked data point $\boldsymbol{x}_i$ plus one of its neighbors $\boldsymbol{x}_i^{'}$
   - only positive samples, no negative samples (for now)

SCAN Objective [implemented in PyTorch]

1. training sample: randomly picked data point $\boldsymbol{x}_i$ plus one of its neighbors $\boldsymbol{x}_i^{'}$
   - only positive samples, no negative samples (for now)
2. learn a function $f(\cdot)$ with parameters θ to transform input vectors $\boldsymbol{x}_i, \boldsymbol{x}_i^{'}$ to output vectors $\boldsymbol{y}_i, \boldsymbol{y}_i^{'}$
   - SCAN: deep CNN
   - DocSCAN: classification layer applied to the encodings, or RoBERTa applied to the texts.

SCAN Objective [implemented in PyTorch]

1. training sample: randomly picked data point $\boldsymbol{x}_i$ plus one of its neighbors $\boldsymbol{x}_i^{'}$
   - only positive samples, no negative samples (for now)
2. learn a function $f(\cdot)$ with parameters θ to transform input vectors $\boldsymbol{x}_i, \boldsymbol{x}_i'$ to output vectors $\boldsymbol{y}_i, \boldsymbol{y}_i'$
   - SCAN: deep CNN
   - DocSCAN: classification layer applied to the encodings, or RoBERTa applied to the texts.
3. reward a higher dot product of the topic vectors $\boldsymbol{y}_i \cdot \boldsymbol{y}_i'$
   - pushes neighbors toward being in the same topic
4. reward a higher entropy over outcome shares $\boldsymbol{y}_i$
   - regularization that prevents collapse into a single topic
5. train until convergence, drop observations that don't fit into a topic (e.g. $\max_k y_k < 0.9$).

Evaluation

- SCAN:
  - image classification dataset, e.g. ImageNet
  - does SCAN recover the annotated classes?
- DocSCAN:
  - text classification dataset, e.g. 20 Newsgroups
  - does DocSCAN recover the annotated classes?

Results

- Test-set accuracy by benchmarks (columns) and classifier (rows). Cell values give the mean over 5 runs with standard error in parentheses

Experiment	20 News	AG news	DBPedia	Google Snippets	Abstracts Group

Results

- Test-set accuracy by benchmarks (columns) and classifier (rows). Cell values give the mean over 5 runs with standard error in parentheses

Experiment	20 News	AG news	DBPedia	Google Snippets	Abstracts Group
Random Baseline	7.0 (0.0)	26.1 (0.3)	7.7 (0.0)	15.04 (0.5)	23.5 (0.6)

Results

- Test-set accuracy by benchmarks (columns) and classifier (rows). Cell values give the mean over 5 runs with standard error in parentheses

Experiment	20 News	AG news	DBPedia	Google Snippets	Abstracts Group
Random Baseline	7.0 (0.0)	26.1 (0.3)	7.7 (0.0)	15.04 (0.5)	23.5 (0.6)
TF-IDF + kmeans	22.5 (3.7)	39.4 (3.5)	47.6 (3.0)	23.7 (2.1)	54.8 (4.1)

Results

- Test-set accuracy by benchmarks (columns) and classifier (rows). Cell values give the mean over 5 runs with standard error in parentheses

Experiment	20 News	AG news	DBPedia	Google Snippets	Abstracts Group
Random Baseline	7.0 (0.0)	26.1 (0.3)	7.7 (0.0)	15.04 (0.5)	23.5 (0.6)
TF-IDF + kmeans	22.5 (3.7)	39.4 (3.5)	47.6 (3.0)	23.7 (2.1)	54.8 (4.1)
SBERT embeddings + kmeans	31.7 (0.9)	55 (8.4)	64.7 (4.3)	52.7 (3.4)	50.5 (3.9)

Results

- Test-set accuracy by benchmarks (columns) and classifier (rows). Cell values give the mean over 5 runs with standard error in parentheses

Experiment	20 News	AG news	DBPedia	Google Snippets	Abstracts Group
Random Baseline	7.0 (0.0)	26.1 (0.3)	7.7 (0.0)	15.04 (0.5)	23.5 (0.6)
TF-IDF + kmeans	22.5 (3.7)	39.4 (3.5)	47.6 (3.0)	23.7 (2.1)	54.8 (4.1)
SBERT embeddings + kmeans	31.7 (0.9)	55 (8.4)	64.7 (4.3)	52.7 (3.4)	50.5 (3.9)
Topic Modeling (LDA)	15.2	31.4	20.4	24.9	30.9

Results

- Test-set accuracy by benchmarks (columns) and classifier (rows). Cell values give the mean over 5 runs with standard error in parentheses

Experiment	20 News	AG news	DBPedia	Google Snippets	Abstracts Group
Random Baseline	7.0 (0.0)	26.1 (0.3)	7.7 (0.0)	15.04 (0.5)	23.5 (0.6)
TF-IDF + kmeans	22.5 (3.7)	39.4 (3.5)	47.6 (3.0)	23.7 (2.1)	54.8 (4.1)
SBERT embeddings + kmeans	31.7 (0.9)	55 (8.4)	64.7 (4.3)	52.7 (3.4)	50.5 (3.9)
Topic Modeling (LDA)	15.2	31.4	20.4	24.9	30.9
DocSCAN Classification	**41.1** (1.8)	80.5 (7.1)	**77.7** (5.7)	64.3 (2.5)	**63.2** (2.3)

Results

- Test-set accuracy by benchmarks (columns) and classifier (rows). Cell values give the mean over 5 runs with standard error in parentheses

Experiment	20 News	AG news	DBPedia	Google Snippets	Abstracts Group
Random Baseline	7.0 (0.0)	26.1 (0.3)	7.7 (0.0)	15.04 (0.5)	23.5 (0.6)
TF-IDF + kmeans	22.5 (3.7)	39.4 (3.5)	47.6 (3.0)	23.7 (2.1)	54.8 (4.1)
SBERT embeddings + kmeans	31.7 (0.9)	55 (8.4)	64.7 (4.3)	52.7 (3.4)	50.5 (3.9)
Topic Modeling (LDA)	15.2	31.4	20.4	24.9	30.9
DocSCAN Classification	**41.1** (1.8)	80.5 (7.1)	**77.7** (5.7)	64.3 (2.5)	**63.2** (2.3)
DocSCAN RoBERTa	5.7 (0.0)	**85.0** (1.6)	62.3 (5.8)	**74.8** (1.5)	57.2 (10.3)

Results

- Test-set accuracy by benchmarks (columns) and classifier (rows). Cell values give the mean over 5 runs with standard error in parentheses

Experiment	20 News	AG news	DBPedia	Google Snippets	Abstracts Group
Random Baseline	7.0 (0.0)	26.1 (0.3)	7.7 (0.0)	15.04 (0.5)	23.5 (0.6)
TF-IDF + kmeans	22.5 (3.7)	39.4 (3.5)	47.6 (3.0)	23.7 (2.1)	54.8 (4.1)
SBERT embeddings + kmeans	31.7 (0.9)	55 (8.4)	64.7 (4.3)	52.7 (3.4)	50.5 (3.9)
Topic Modeling (LDA)	15.2	31.4	20.4	24.9	30.9
DocSCAN Classification	**41.1** (1.8)	80.5 (7.1)	**77.7** (5.7)	64.3 (2.5)	**63.2** (2.3)
DocSCAN RoBERTa	5.7 (0.0)	**85.0** (1.6)	62.3 (5.8)	**74.8** (1.5)	57.2 (10.3)
SBERT embeddings + SVM	66	90	99	75	80

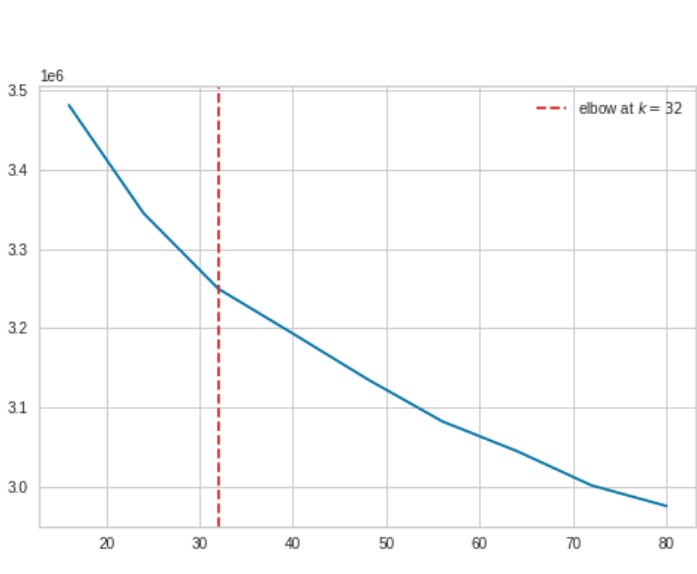
Application to Trump Tweets Archive

- Source code at `github.com/dominiksinsaarland/DocSCAN`
- See Notebook linked at `https://bit.ly/Ash-SciPy`.

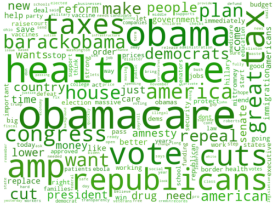
```
from DocSCAN import DocSCANPipeline
import argparse

# some default arguments
parser = argparse.ArgumentParser()
parser.add_argument("--infile", type=str, default="sample.txt", help="path to infile/dataframe")
parser.add_argument("--data_format", type=str, default="from_txt", help="whether infile points to a txt file where each line is a sentence or to a dataframe")
parser.add_argument("--outpath", type=str, default="test", help="path to output path where output of docscan gets saved")
parser.add_argument("--sbert_model", default="bert-base-nli-mean-tokens", type=str, help="SBERT model to use to embedd sentences")
parser.add_argument("--max_seq_length", default=128, type=int, help="max seq length of sbert model, sequences longer than this get truncated at this value")
parser.add_argument("--topk", type=int, default=5, help="numbers of neighbors retrieved to build SCAN training set")
parser.add_argument("--num_classes", type=int, default=10, help="numbers of clusters")
parser.add_argument("--min_clusters", default=16, type=int, help="lower bound of clusters to search through to generate elbow")
parser.add_argument("--max_clusters", default=88, type=int, help="upper bound of clusters to search through to generate elbow")
parser.add_argument("--stepsize", default=8, type=int, help="2")
parser.add_argument("--batch_size", default=64, type=int, help="Batch size per GPU/CPU for training.")
parser.add_argument("--wordcloud_frequencies", default="tf-idf", type=str, help="wordcount weighting, either tf-idf or raw counts")
parser.add_argument("--dropout", default=0.1, type=float, help="dropout for DocSCAN model")
parser.add_argument("--num_epochs", default=3, type=int, help="number of epochs to train DocSCAN model")
args, unknown = parser.parse_known_args()
```

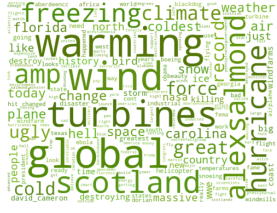
```
docscan = DocSCANPipeline(args)
docscan.run_main(sentences=df["doc"].tolist())
```

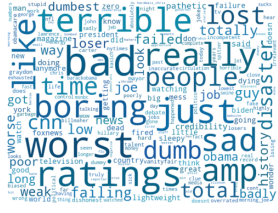
How to pick number of topics (Elbow Method)



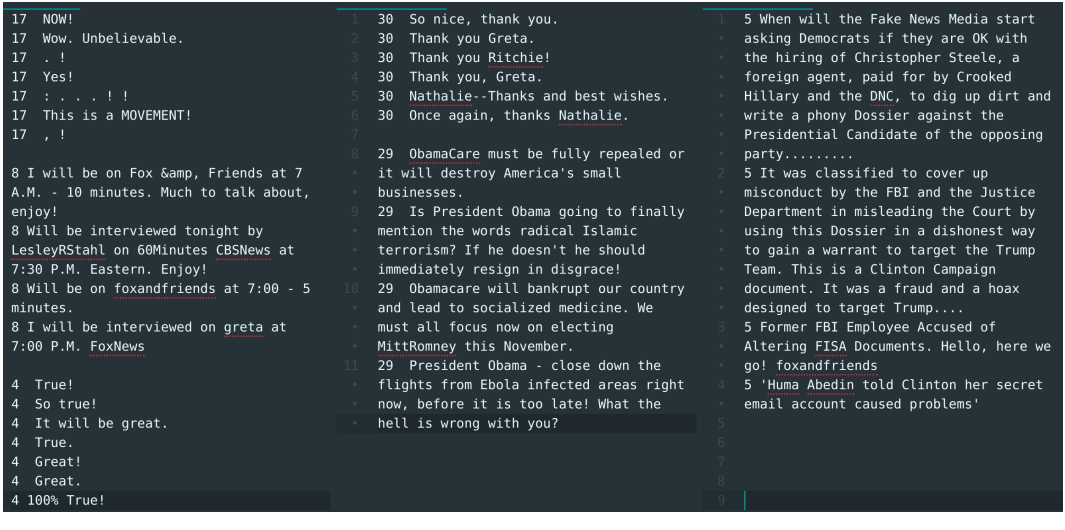
Selection of Trump Tweet Topics (Distinctive Words)







Representative Tweets by Cluster



Outline

Ash, Gauthier, and Widmer (2021)

Higher taxes will hurt the economy."

"Health insurance saves lives."

'Immigrants steal our jobs.'

Ash, Gauthier, and Widmer (2021)

Higher taxes will hurt the economy."

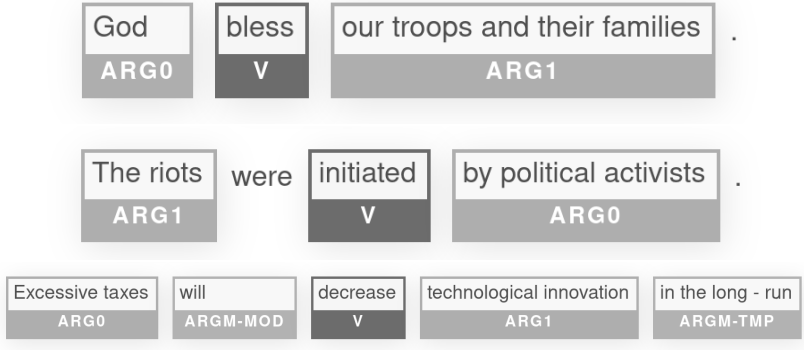
"Health insurance saves lives."

'Immigrants steal our jobs.'

- **How do narratives influence and/or reflect political and economic outcomes?**
- **A preliminary challenge: How to *identify* and *quantify* narratives.**

Semantic roles capture narrative accounts

Figure: Examples of SRL Annotations



Note: This figure presents examples of semantic role annotations based on `allennlp`'s programmatic implementation. For interactive tests, see: `https://demo.allennlp.org/semantic-role-labeling`.

[Allen NLP Semantic Role Labeller]

Semantic roles capture narrative accounts.

- We focus on the following main arguments:
  - `ARG0`: *Who?* $\rightarrow$ Agents $\mathcal{A}_0$
  - `V`: *Does what?* $\rightarrow$ verbs $\mathcal{V}$
  - `ARG1`: *To whom?* $\rightarrow$ Patients $\mathcal{A}_1$
- We decompose sentences in the corpus into role sequences:

$$\texttt{ARG0} \xrightarrow{\texttt{V}} \texttt{ARG1}$$

Two Complementary Approaches for Dimension Reduction

- **Named Entity Recognition (NER)**
  - We identify the named entities such as events, organizations, places and people in the corpus. [spaCy Named Entity Recognizer]
- **Phrase embedding clustering**
  - For the remaining roles, we embed them in a dense vector space (i.e., SIF-weighted phrase embeddings based on pre-trained GloVe vectors [from spaCy]).
    - also experiment with universal sentence encoder via tensorflow_hub and sentence_transformers encoders (sbert.net)
  - We cluster them with the K-means algorithm. [sklearn.cluster.Kmeans]

Some Examples of Entities

Table: Examples of Mined Entities

Entity	**Associated Phrases**
people	'people', 'many', 'others', 'many people', 'number'
american	'american', 'american people', 'million american', 'american taxpayer', 'many american'
government	'government', 'local government', 'foreign government', 'government official', 'russian government'
healthcare	'health insurance', 'health care', 'health', 'health care coverage', 'health insurance coverage'
job	'job', 'work', 'good job', 'job do', 'create job'
money	'money', 'benefit', 'cost', 'pay', 'taxpayer'

Resulting Narrative Statements

Table: Most Frequent Non-procedural Narratives

Narrative	Frequency
people lose job	606
american lose job	448
citizen abide law	420
american have healthcare	414
government run healthcare	359
god bless american	357
president sign law	355
people need help	338
god bless troop	329
small business create job	304

Sentences Associated to "people lose job"

Mr. Speaker, I urge my colleagues to defeat the previous question so that we can immediately bring up H.R. 4415, which would restore unemployment benefits to 2.8 million Americans, people who have lost their job and are simply trying to find their next job and want to prevent their families from losing everything they have worked for in that period.

Our colleague from Ohio just said this is an opportunity for people who have lost their jobs.

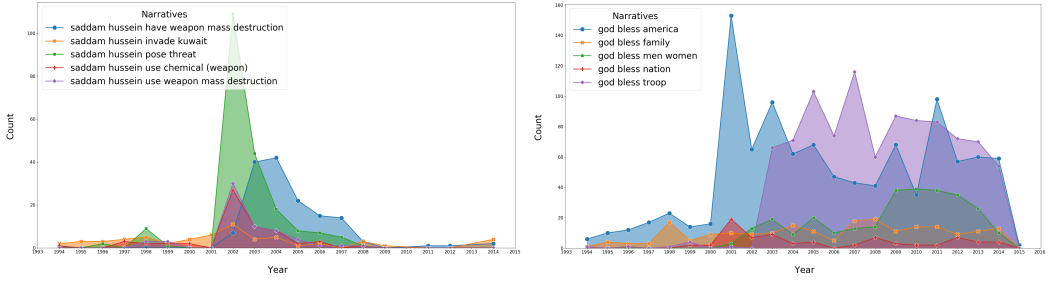
Providing health care benefits–again, none of us subscribes to the notion that people who are unemployed or lose their jobs are anywhere near as much a victim as those victims on September 11, at the World Trade Center, or the Pentagon, or aboard that airplane in Pennsylvania.

It appears, we've been hearing over and over from the Democratic leadership, and even from the President, people are losing jobs every day.

People shouldn't have to lose their jobs to pay for the New York fund.

Popular narratives reflect key historical events.

Figure: U.S. Congress Speeches: Narratives on Terrorism and War



Note: Left panel displays the number of yearly occurrences of the five most frequent narratives related to the entity "saddam hussein". One notices a distinctive spike in 2002, after the 9/11 terror attacks and before the United States' military intervention in Iraq. Right panel displays the number of yearly occurrences of the five most frequent narratives related to the action "god bless". One notices a distinctive bump in the "god bless america" narrative in 2001 following the 9/11 terror attacks. Later on, from 2003 on-wards the narrative "god bless troop" gains traction in the context of the Iraq war.

Narratives have emotional resonance [nltk.vader]

Table: Most Negative Narratives

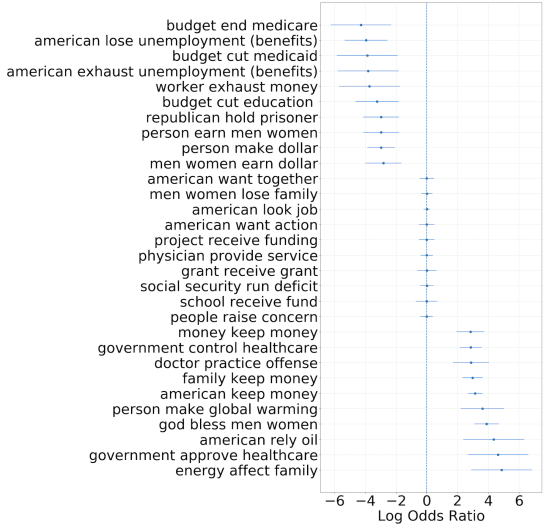
Narrative	Sentiment
corrective action save family	-0.9614
child have healthcare	-0.9522
program provide help	-0.9392
child brought gift	-0.9360
terrorist kill american	-0.9337
firefighter open fire	-0.9325
iraq pose jeopardy	-0.9313
people do harm	-0.9300
saddam hussein pose threat	-0.9237

Table: Most Positive Narratives

Narrative	Sentiment
people establish constitution	0.9843
people promote welfare	0.9843
founder endow men women	0.9769
american take healthcare	0.9643
veteran have healthcare	0.9599
people establish justice	0.9565
kid have healthcare	0.9564
family afford healthcare	0.9531
small business provide job	0.9531

Partisan narratives map ideological disagreements.

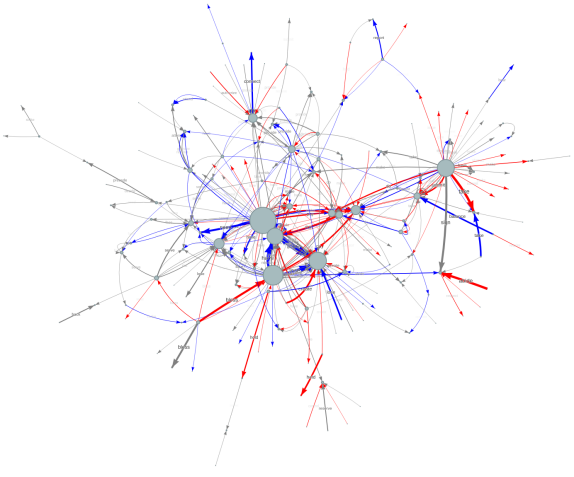
Figure: Partisan vs. Neutral Narratives



Note: This figure presents the most partisan and neutral narratives in the U.S. Congressional Record. For each narrative, we present the associated log-odds ratio and 95% confidence interval. A high log-odds ratio indicates narratives that are pronounced more by Republicans relative to Democrats.

Narratives uncover the broader structure of political debates.

Figure: Narrative Graph of Political Debates [networkx, pyvis]



Note: This figure displays the 500 most frequent narratives in the U.S. Congressional Record. We represent our narrative tuples in a directed multigraph, in which the nodes are entities and the edges are verbs. The color of edges is determined by a log-odds ratio, which measures whether a narrative is more pronounced by Republicans relative to Democrats. Statistically significant log-odds ratios are colored in red for Republicans and in blue for Democrats. Statistically insignificant log-odds ratios are colored in grey. All tests are performed at the 95% confidence level. For interactive narrative graphs, see: `https://sites.google.com/view/trump-narratives/u-s-congressional-record`.

Yarn (*Yes! Automatically Reads Narratives*)

- Beta version available here:

  `https://github.com/elliottash/narrative-nlp`

- Works in Google Colab (`https://bit.ly/Ash-SciPy`), should also work locally on Mac and Linux (not windows yet).

Recap: Telling the (Scientific) Tale, with Python

Recap: Telling the (Scientific) Tale, with Python

- How we used Python for computational social science with text:
  - measuring and analyzing emotional intensity
  - learning topics without any labels
  - extracting and analyzing core narratives

Recap: Telling the (Scientific) Tale, with Python

- How we used Python for computational social science with text:
  - measuring and analyzing emotional intensity
  - learning topics without any labels
  - extracting and analyzing core narratives
- Shout-out to co-authors Gloria Gennaro, Dominik Stammbach, Germain Gauthier, and Philine Widmer.
- More info and links here: `https://bit.ly/Ash-SciPy`

Recap: Telling the (Scientific) Tale, with Python

- How we used Python for computational social science with text:
  - measuring and analyzing emotional intensity
  - learning topics without any labels
  - extracting and analyzing core narratives
- Shout-out to co-authors Gloria Gennaro, Dominik Stammbach, Germain Gauthier, and Philine Widmer.
- More info and links here: `https://bit.ly/Ash-SciPy`

- **Thanks!**

`elliottash.com | ashe@ethz.ch`