Quantitative text analysis: Word Embeddings

Blake Miller

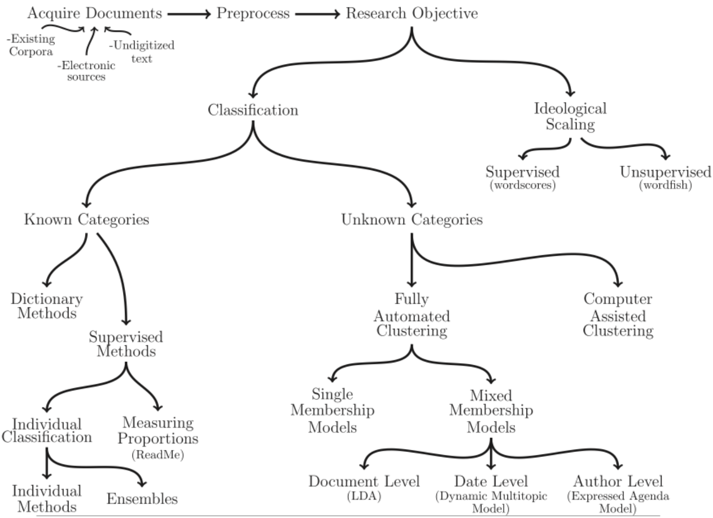
MY 459: Quantitative Text Analysis

March 23, 2019

Course website: lse-my459.github.io

1. Overview and Fundamentals
2. Descriptive Statistical Methods for Text Analysis
3. Automated Dictionary Methods
4. Machine Learning for Texts
5. Supervised Scaling Models for Texts
6. *Reading Week*
7. Unsupervised Models for Scaling Texts
8. Similarity and Clustering Methods
9. Topic models
10. Word embeddings
11. Working with Social Media

Overview of text as data methods



Outline

- Overview of topic models
- Latent Dirichlet Allocation (LDA)
- Validating the output of topic models
- Examples
- Choosing the number of topics
- Extensions of LDA

Extensions of LDA

1. Structural topic model (Roberts et al, 2014, AJPS)
2. Dynamic topic model (Blei and Lafferty, 2006, ICML; Quinn et al, 2010, AJPS)
3. Hierarchical topic model (Griffiths and Tenembaun, 2004, NIPS; Grimmer, 2010, PA)

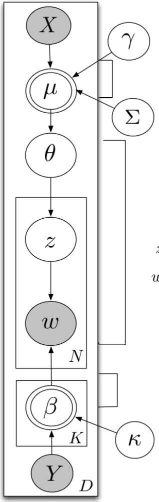
Why?

- Substantive reasons: incorporate specific elements of DGP into estimation
- Statistical reasons: structure can lead to better topics.

Structural topic model (STM)

- Basic idea: STM = LDA + Contextual Information
- STM provides two ways to include contextual information
  - Topic prevalence can vary by metadata (e.g. Democrats talk more about education than Republicans)
  - Topic content can vary by metadata (e.g. Democrats are less likely to use the word “life” when talking about abortion than Republicans)
- Including context improves the model:
  - more accurate estimation
  - better qualitative interpretability

Structural topic model (STM)



Topic Prevalence:

$$\begin{aligned} \mu_{d,k} &= X_d \gamma_k \\ \gamma_k &\sim \mathcal{N}(0, \sigma_k^2) \\ \sigma_k^2 &\sim \text{Gamma}(s^\gamma, r^\gamma) \end{aligned}$$

Language Model:

$$\begin{aligned} \theta_d &\sim \text{LogisticNormal}(\mu_d, \Sigma) \\ z_{d,n} &\sim \text{Mult}(\theta_d) \\ w_{d,n} &\sim \text{Mult}(\beta_d^{k=z_{d,n}}) \end{aligned}$$

Topical Content:

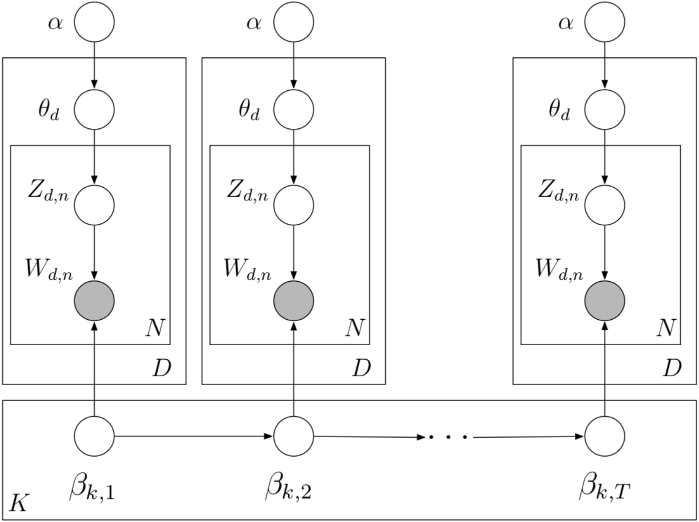
$$\begin{aligned} \beta_{d,v}^k &\propto \exp(m_v + \kappa_v^{\cdot,k} + \kappa_v^{y,\cdot} + \kappa_v^{y,k}) \\ \kappa_v^{y,k} &\sim \text{Laplace}(0, \tau_v^{y,k}) \\ \tau_v^{y,k} &\sim \text{Gamma}(s^\kappa, r^\kappa) \end{aligned}$$

- **Prevalence**: Prior on the mixture over topics is now document-specific, and can be a function of covariates (documents with similar covariates will tend to be about the same topics)
- **Content**: distribution over words is now document-specific and can be a function of covariates (documents with similar covariates will tend to use similar words to refer to the same topic)

Structural topic model (STM)

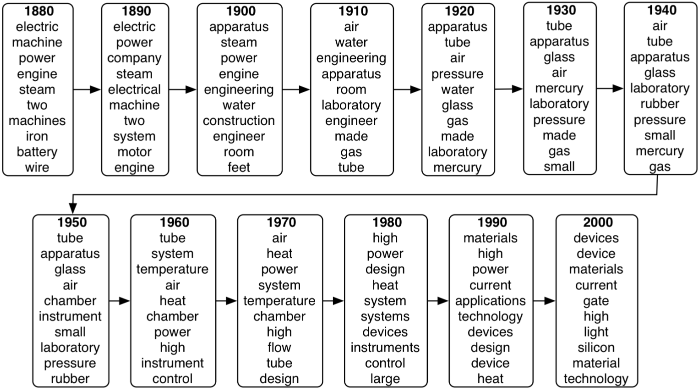
- User specifies the number of topics: K
- Observed data for standard topic models
  - Each document ($d \in 1 \dots D$) is a collection of N_d tokens
  - Each token is a particular word from a dictionary of V entries
  - Data summarized in a single matrix $D \times V$ matrix $\mathbf{W}$
- Additional data for STM
  - Topic prevalence covariates: $D \times P$ matrix $\mathbf{X}$
  - Topical content groups: D length vector $\mathbf{Y}$
- Latent variables
  - $D \times K$ matrix θ: proportion of document on each topic.
  - $K \times V$ matrix β: probability of drawing a word conditional on topic.
  - Low rank approximation to expected counts: $\underset{D \times V}{\tilde{W}} \sim \underset{D \times K}{\theta} \underset{K \times V}{\beta}$

Dynamic topic model



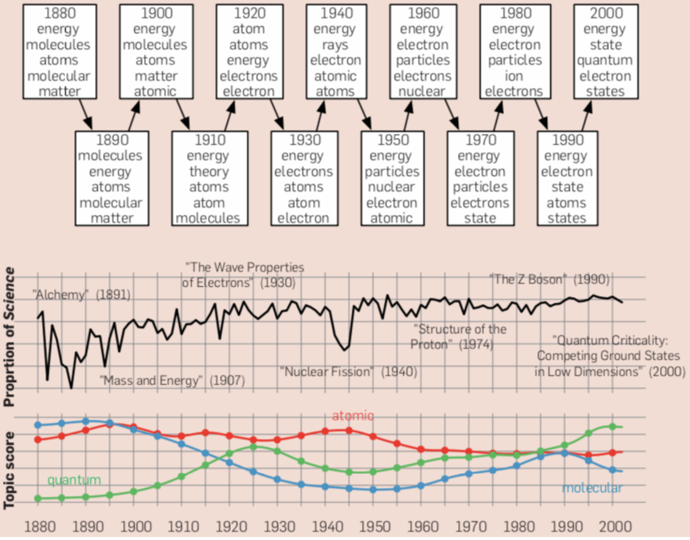
Source: Blei, "Modeling Science"

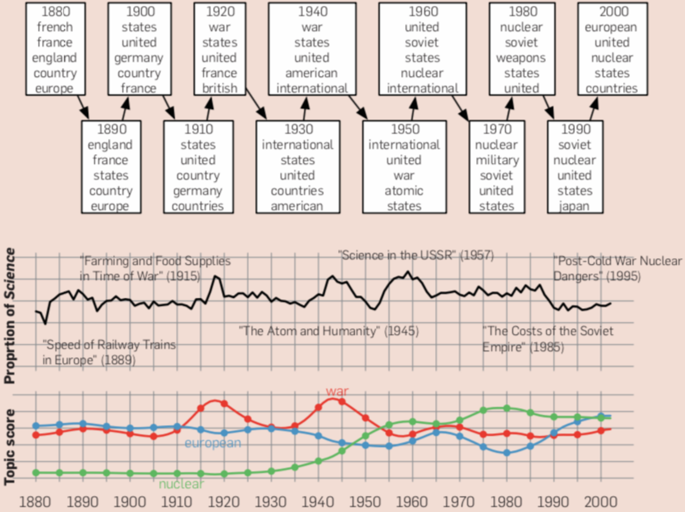
Dynamic topic model



Source: Blei, “Modeling Science”

Figure 5. Two topics from a dynamic topic model. This model was fit to *Science* from 1880 to 2002. We have illustrated the top words at each decade.




1880 french france england country europe
1890 england france states country europe
1900 states united germany country france
1910 states united country germany countries
1920 war states united france british
1930 international states united countries american
1940 war states united american international
1950 international united war atomic states
1960 united soviet states nuclear international
1970 nuclear military soviet united states
1980 nuclear soviet weapons states united
1990 soviet nuclear united states japan
2000 european united nuclear states countries
"Farming and Food Supplies in Time of War" (1915)
"Science in the USSR" (1957)
"Post-Cold War Nuclear Dangers" (1995)
"Speed of Railway Trains in Europe" (1889)
"The Atom and Humanity" (1945)
"The Costs of the Soviet Empire" (1985)
Proprtion of Science
Topic score
war
european
nuclear
1880
1890
1900
1910
1920
1930
1940
1950
1960
1970
1980
1990
2000

Outline

- Extensions of LDA
- Word embeddings:
  - Overview
  - Applications
  - Bias
  - Embeddings demo
- Encoding issues

Beyond bag-of-words

Most applications of text analysis rely on a bag-of-words representation of documents

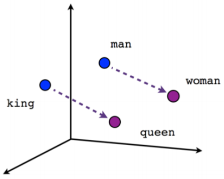
- Only relevant feature: frequency of features
- Ignores context, grammar, word order...
- Wrong but often irrelevant

An alternative: Word Vectors

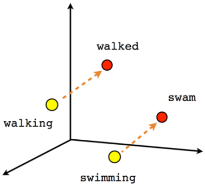
- One alternative to the bag-of-words approach is word embeddings
- Word embeddings represent words as real-valued vector in a multidimensional space (often 100–500 dimensions)
- Central logic:
  - "You shall know a word by the company it keeps" (Rupert Firth)
  - Synonyms like oculist and eye-doctor tend to occur in the same environment.
  - The difference in meaning between two words corresponds "roughly to the amount of difference in their environments" (Harris, 1954, 157).
- Word vectors learn vector representations of words from the context in which they appear.

Word embeddings example

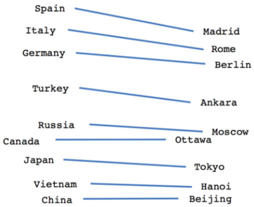
word	D_1	D_2	D_3	$\dots$	D_N
man	0.46	0.67	0.05	$\dots$	$\dots$
woman	0.46	-0.89	-0.08	$\dots$	$\dots$
king	0.79	0.96	0.02	$\dots$	$\dots$
queen	0.80	-0.58	-0.14	$\dots$	$\dots$



Male-Female



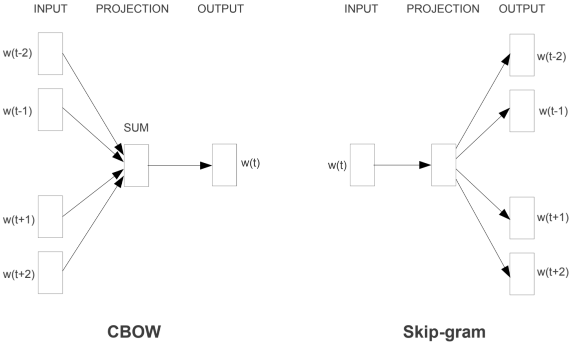
Verb tense



Country-Capital

word2vec (Mikolov 2013)

- Statistical method to efficiently learn word embeddings from a corpus, developed by Google engineer
- Most popular, in part because pre-trained vectors are available.
- Two models to learn word embeddings:



How word2vec works

```
... lemon,  a [tablespoon of apricot jam,      a] pinch ...
               c1         c2    t     c3       c4
```

How word2vec works

- Main idea: Train a classifier on a binary prediction task:
  - Is w likely to show up near "apricot"?
- We don't actually care about this task, but we'll take the learned classifier weights as the word embeddings

How word2vec works

- A word s near apricot acts as "correct answer" to the question
  - "Is word w likely to show up near apricot?"
- No need for hand-labeled supervision
- The idea comes from neural language modeling
  - Bengio et al. (2003)
  - Collobert et al. (2011)

Skip-gram algorithm

1. Treat the target word and a neighboring context word as positive examples.
2. Randomly sample other words in the lexicon to get negative samples
3. Use logistic regression to train a classifier to distinguish those two cases
4. Use the weights as the embeddings

Setup

```
... lemon,  a [tablespoon of apricot jam,     a] pinch ...
               c1          c2    t     c3        c4
```

- Let's represent words as vectors of some length (say 300), randomly initialized.
- So we start with $300 \times V$ random parameters
- Over the entire training set, we'd like to adjust those word vectors such that we
  - Maximize the similarity of the target word, context word pairs (t,c) drawn from the positive data
  - Minimize the similarity of the (t,c) pairs drawn from the negative data.

Training Data

positive examples +

t	c
apricot	tablespoon
apricot	of
apricot	preserves
apricot	or

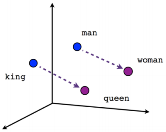
negative examples -

t	c	t	c
apricot	aardvark	apricot	twelve
apricot	puddle	apricot	hello
apricot	where	apricot	dear
apricot	coaxial	apricot	forever

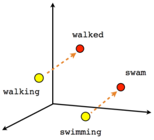
How to learn word2vec (skip-gram) embeddings

- Start with V random 300-dimensional vectors as initial embeddings
- Use logistic regression, the second most basic classifier used in machine learning after naive bayes
- Take a corpus and take pairs of words that co-occur as positive examples
- Take pairs of words that don't co-occur as negative examples
- Train the classifier to distinguish these by slowly adjusting all the embeddings to improve the classifier performance
- Throw away the classifier and keep the embeddings.

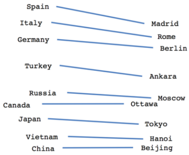
Vectors and Semantic Relationships



Male-Female



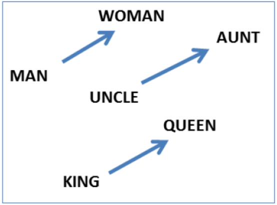
Verb tense

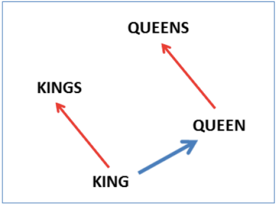


Country-Capital

Vectors capture some general semantic information about words and their relationships to one another.

Analogy: Embeddings capture relational meaning!





- vector('king') - vector('man') + vector('woman') $\approx$ vector('queen')
- vector('Paris') - vector('France') + vector('Italy') $\approx$ vector('Rome')

Outline

- Extensions of LDA
- Word embeddings:
  - Overview
  - Applications
  - Bias
  - Embeddings demo
- Encoding issues

Application: Pomeroy et al 2018

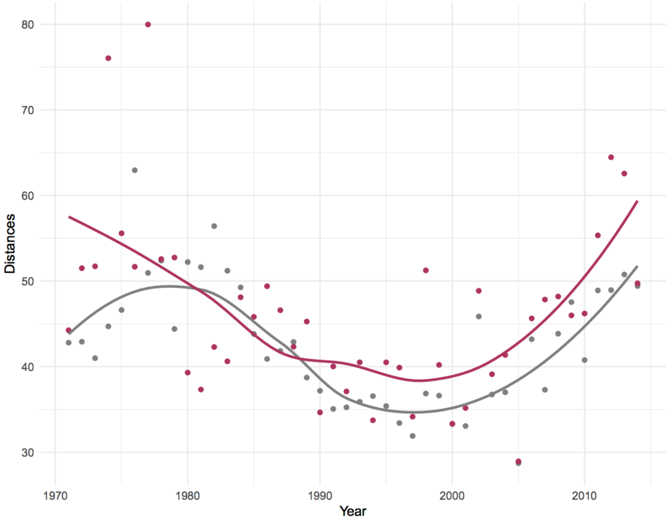
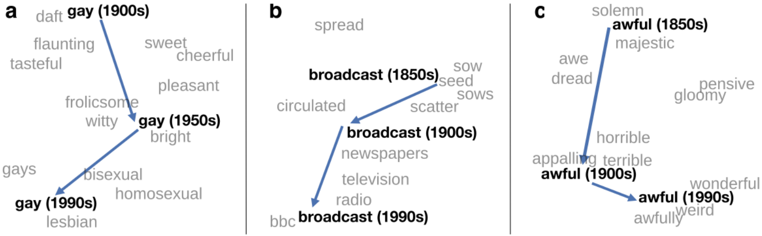


Figure 4: *Distances by core countries*. Plot of Euclidian distances between US and Russia (gray), and US and China (maroon).

Application: semantic shifts

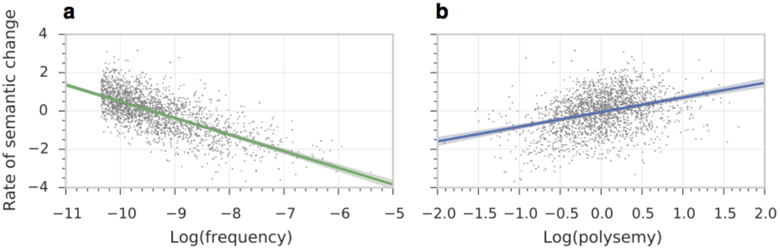
Using word embeddings to visualize changes in word meaning:



Source: Hamilton et al, 2016 ACL.
https://nlp.stanford.edu/projects/histwords/

Application: semantic shifts

Using word embeddings to visualize changes in word meaning:



Source: Hamilton et al, 2016 ACL.
https://nlp.stanford.edu/projects/histwords/

Application: dictionary expansion

Using word embeddings to expand dictionaries (e.g. incivility)

```
> distance(file_name = "FBvec.bin",
+          search_word = "libtard",
+          num = 10)
Entered word or sentence: libtard

Word: libtard  Position in vocabulary: 5753
         word              dist
1         lib 0.798957586288452
2       lefty 0.771853387355804
3     libturd 0.762575328350067
4   teabagger 0.744283258914948
5    teabilly 0.715277075767517
6     liberal 0.709996342658997
7      retard 0.690707504749298
8     dumbass 0.690422177314758
9        rwnj 0.684058785438538
10 republitard 0.678197801113129
```

```
> distance(file_name = "FBvec.bin",
+          search_word = "idiot",
+          num = 10)
Entered word or sentence: idiot

Word: idiot  Position in vocabulary: 646
         word              dist
1    imbecile 0.867565214633942
2     asshole 0.848560094833374
3       moron 0.781079053878784
4      asshat 0.772150039672852
5      a-hole 0.765781462192535
6       ahole 0.760824918746948
7     asswipe 0.742586553096771
8   ignoramus 0.735219776630402
9    arsehole 0.732272684574127
10      idoit 0.720151424407959
```

Outline

- Extensions of LDA
- Word embeddings:
  - Overview
  - Applications
  - Bias
  - Embeddings demo
- Encoding issues

Bias in word embeddings

Semantic relationships in embeddings space capture stereotypes:

- Neutral example: man – woman ≈ king – queen
- Biased example: man – woman ≈ computer programmer – homemaker

Gender stereotype *she-he* analogies.

sewing-carpentry	register-nurse-physician	housewife-shopkeeper
nurse-surgeon	interior designer-architect	softball-baseball
blond-burly	feminism-conservatism	cosmetics-pharmaceuticals
giggle-chuckle	vocalist-guitarist	petite-lanky
sassy-snappy	diva-superstar	charming-affable
volleyball-football	cupcakes-pizzas	hairdresser-barber

Gender appropriate *she-he* analogies.

queen-king	sister-brother	mother-father
waitress-waiter	ovarian cancer-prostate cancer	convent-monastery

Source: Bolukbasi et al, 2016. arXiv:1607.06520
See also Garg et al, 2018 PNAS and Caliskan et al, 2017 Science.

Outline

- Extensions of LDA
- Word embeddings:
  - Overview
  - Applications
  - Bias
  - Embeddings demo
- Encoding issues

Outline

- Extensions of LDA
- Word embeddings:
  - Overview
  - Applications
  - Bias
  - Embeddings demo
- Encoding issues

Character encodings

- **Encoding:** how digital binary signals are translated into human-readable characters.

→ e.g. `0100100` is displayed as 'd'

- This also includes characters such as á, ç, ü, etc.
- **Problem:** many different translation tables, sometimes hard to know which one is used
- R works with the default encoding scheme in your system:

```
> Sys.getlocale(category = "LC_CTYPE")
[1] "en_US.UTF-8"
```

- For English Mac and Linux systems, generally UTF-8. For Windows systems, Windows-1252.
- UTF-8 (part of Unicode standard) is most popular scheme and used on many websites.