SOSC 4300/5500: Topic models and Word Embeddings

Han Zhang

Oct 27, 2020

Outline

Logistics

More on topic models

Word Embedding

Word Embedding Examples

Literature Review and Presentation

- Poll: do you want to do the presentation first and then hand-in the literature review?
  - That is, present in next week (Nov 3)
  - And turn in report on Nov 10
- Idea: may be better to get some feedback first

Assignment 2

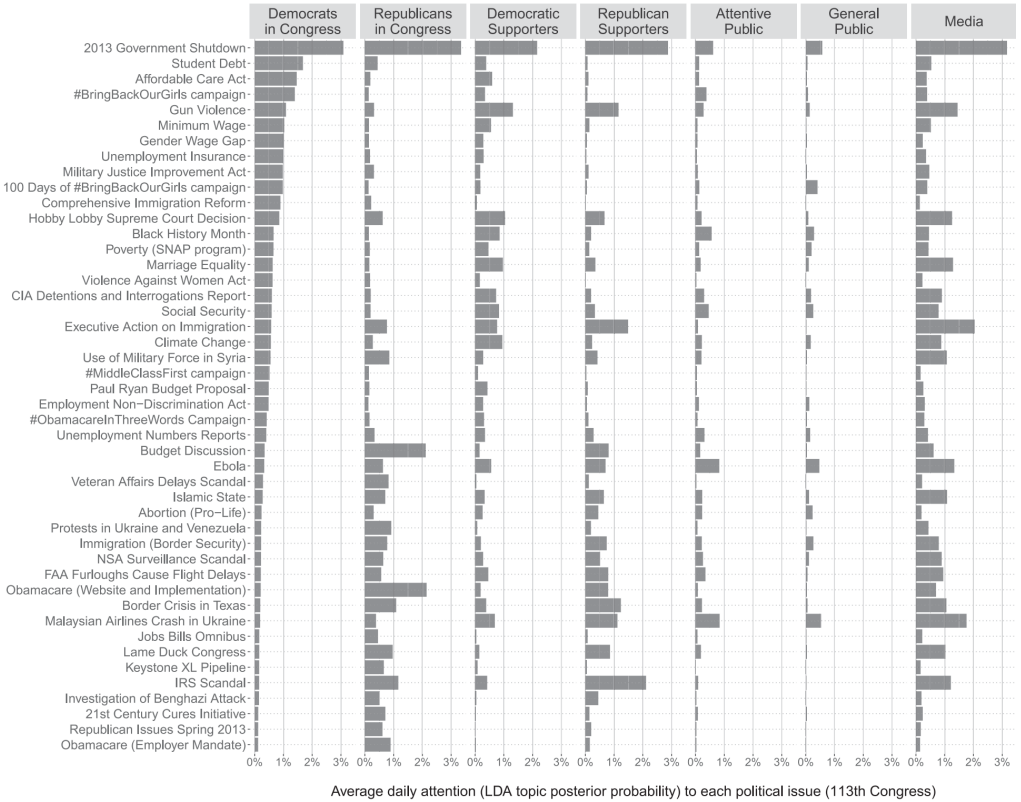
- Will be released today
- Predicting whether an article will be accepted or not for a real CS conference
- And fitting a topic model to get what these articles are about

What to do with covariates in topic models?

- LDA only finds topics
- Suppose you have covariates, and want to see how topics vary by covariates
  - That's a central question in Barbera et. al, (2019) we read in last week
  - E.g., whether a topic's proportions change over time?
  - whether a topic's proportion changes by author type?

Example: Barbera et. al, (2019)

FIGURE 1. Average Issue Attention by Groups of Politicians, the Public, and the Media



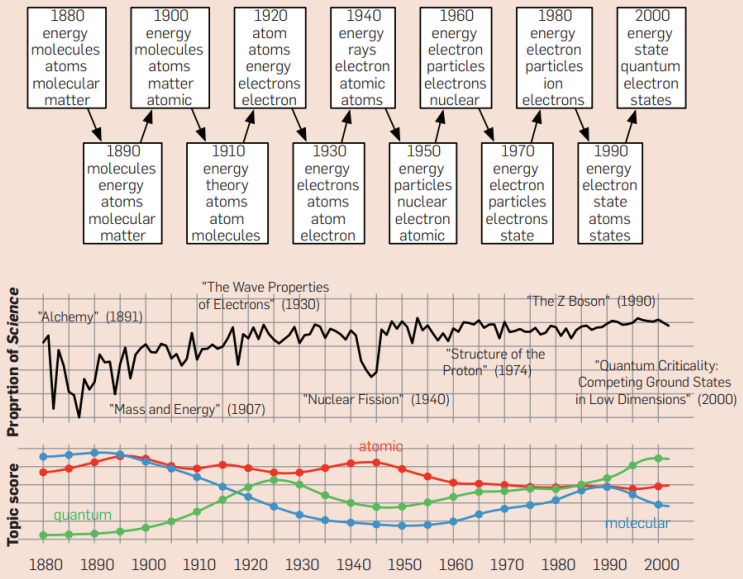
Problems

- Simple workarounds: split the documents by year/author type, and fit LDA separately for subsets
- Problems?
  - Topics are not comparable
  - Normal people's political discussions may be very different from that of politicians
  - Certain issues may only be discussed by politicians, but not by politicians, or vice versa
- How did Barbera solved this problem? Anyone remembers?
  - They fit LDA for politicians only, thus effectively fixing the allowed topics.
  - And of course, they sacrifice the possibility that normal people may come up with entirely different topics that's not in politician's discussions

Extending Topic Model (STM)

- Incorporating covariates allows us to fix the topics, but also allow it to vary by covariates
- Remember this picture in Blei 2012?
- He used Dynamic Topic Models, originally proposed in Blei, David M., and John D. Lafferty. "Dynamic topic models." In Proceedings of the 23rd international conference on Machine learning, pp. 113-120. 2006.

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.



Structure Topic Model (STM)

- Margaret E. Roberts, Brandon M. Stewart, Dustin Tingley, Christopher Lucas, Jetson Leder-Luis, Shana Kushner Gadarian, Bethany Albertson, and David G. Rand, *Structural Topic Models for Open-Ended Survey Responses*, American Journal of Political Science (2014)
- STM is particularly popular among social scientists
- One reason is that it's fairly easy to use; with a R package provided by the authors
  - Most of Blei's models are implemented in C/C++; not even in Python. Not friendly to social scientists
- The other reason is that they tried to combine LDA with regressions, which social scientists are familiar with
  - CS people do not care about regressions, of course

STM: topical prevalence

- What LDA does:
  - $$p(w_{id}) = \sum_{j=1}^{K} P(w_i|Z_i = j)P(z_i = j)$$
  - $p(w_{id})$ is the probability of observing word i in document d
  - $P(w_{id}|Z_d = j) = \phi^j$: probability of observing word i in document d, if we know d's topic is j
  - $P(Z_d = j) = \theta^d$: topic j's probability of document d
- How STM extends?
  - $P(Z_d = j) = \theta^d = LogisticNormal(\beta X, \epsilon)$
    - That is, topic's probability of document d can vary by covariates of d, X_d
    - In STM's notation, covariates can influence topical prevalence
    - Recall that linear regression looks like $P(Y|X) = Normal(\beta X_d, \epsilon)$
    - LogisticNormal extends Normal distribution, by taking the logic transformation and forcing values to be between (0,1), which is required (since we are modeling probabilities!)

STM: topical contents

- What LDA does:
  - $$p(w_{id}) = \sum_{j=1}^{K} P(w_i|Z_i = j)P(z_i = j)$$
  - $p(w_{id})$ is the probability of observing word i in document d
  - $P(w_{id}|Z_d = j) = \phi^j$: probability of observing word i in document d, if we know d's topic is j
  - $P(Z_d = j) = \theta^d$: topic j's probability of document d
- How STM extends?
  - $P(w_{id}|Z_d = j) = \phi^j + \kappa_{X_d}$
    - That is, probability of using word i in document d varies by document's covariates
    - In STM's notation, covariates can influence topical content

Prevalence vs. Content

- If you have some covariates X (e.g., year, politician/normal people)
- You do not need to let them influence prevalence and content simultaneously
- Allow X to determine topical prevalence if:
  - You can about how topics vary by X
- Allow X to determine topical if:
  - You want to see how word usage varies by X, within a topic

Sparse Representations of words

- Document-term matrix is a sparse representation
- Each word is represented as a vector of length n, with n being the size of the corpus
  - Most entries in that word vector is 0
- This causes two problems:
  - Harder to compare distances between vectors: the distances are all clumped together
    - We talked about this in the previous week, on why you should do dimensionality reduction
  - Increase in size with documents: big data -> higher dimensions

Sparse representation of words

- Sparse representation tries map a word into a lower-dimensional representation, with dimensions $n' << n$
  - And of course, most entries in the lower-dimensional vector should not be 0
  - n' is usually fixed; it means sense, because we think that the vector representation of a word should means something intrinsic, with fixed size, instead of changing by corpus sizes.
  - Think about survey; it's a kind of a dense representation of a respondent

Word embedding

- In the old days, you can perform dimensionality-reduction methods we talked about in the last week
  - e.g., PCA, nonnegative matrix factorization
- Word embedding is a revolutionary approach
- word2vec: the founding work
  - Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean, *Distributed Representations of Words and Phrases and Their Compositionality*, Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2 (USA), NIPS'13, Curran Associates Inc., 2013, pp. 3111–3119
- Embed means to map a word into a dense representation
- Embedding is the vector representation of a word

Word embedding

- Algorithm: share some similarity to t-SNE
- Intuition 1: we can use the surrounding words of a word to predict the word itself
  - E.g., "use the XXX words of a word to predict the word itself"
    - words such as "nearby", "surrounding", is more reasonable guess
    - words such as "happy" are not
    - In fact, most words out of a vocabulary are not good guesses
- Intuition 2: On vector space, embeddings of surrounding words should also be able to predict the embedding of the word itself.

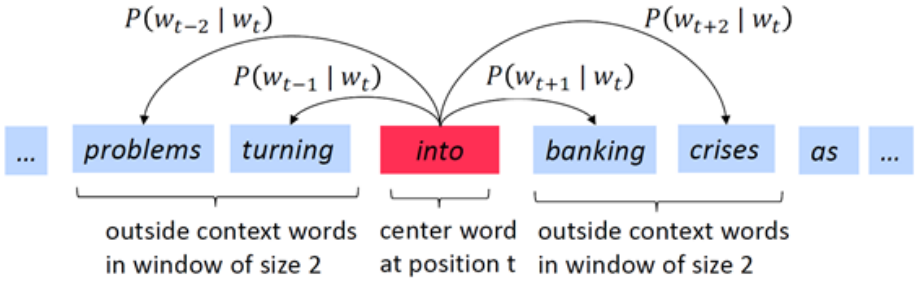
Word embedding: CBOW vs. Skip-Gram

- Two variants of word2vec (Mikolov et. al, 2013): CBOW and Skip-Gram
- Continuous Bag of Words (CBOW) model:
  - training a model to predict a word given its context words
  - This is the intuition we have given in the previous slides
- Pros:
  - CBOW is more intuitive, and much more quicker
- Cons:
  - But it does not predict rare words very well
  - E.g., `yesterday was a really [...] day`
  - The most probable words are `beautiful` or `nice`
  - Infrequent words like `delightful` will has low probability

Word embedding: CBOW vs. Skip-Gram

- Skip-Gram Model is the opposite model:
  - training a model to predict context words given a word
- Cons:
  - This is a much harder prediction task, as you can imagine
  - `[...] [...] [...] [...] delightful [...]`
  - And if your dataset is small, do not use Skip-Gram model
- Pros: deal with rare words better
  - Because you are not competing `delightful` vs. `nice`
  - But different n-grams, such as `delightful + [...]`
    - And `delightful day` is more probable than `delightful model`, for example.

Skip-Gram



- Optimization problem: maximize the following

$$\frac{1}{\text{size of vocab}} \prod_{w \in Vocab} \prod_{c \in C(w)} P(c|w)$$

- c is a unique context of word w
  - `[...] [...] delightful [...] [...]`
  - [...] is an arbitrary word
- $C(w)$ is the set of context of word w

Skip-Gram

- How can we calculate $P(c|w)$?
- Each word w has an embedding vector v_w
  - this is the unknown embedding we want to obtain
  - v_w has dimension d, which is the critical parameter; it determines how long your embedding is.

$$P(c|w) = \frac{exp(v_c^T v_w)}{\sum_{c' \in Vocab} exp(v_{c'}^T v_w)}$$

- Essentially, it says that the probability of seeing context word x, $P(c|w)$, is proportional to the inner product between embeddings of word w and c
  - Which captures the similarity between these two embedding vectors v_w and v_c
  - If $v_c^T v_w$ is large, then $p(c|w)$ is large

Learning Skip-Gram

- The unknown parameters are thus v_w for all words w in the vocabulary
- How many parameters we have?
- For each word w and each context c of the word, we need to find their vectors
- So the total number of parameters are huge:
- vocab size * number of unique contexts * embedding vector length
- In particular, the number of possible words are tremendously large
- If window size is 4, the number of contexts is vocab size^4, a crazy number
  - e.g., with 10000 words, 10000^4

Learning Skip-Grams

- In practice, Mikolov et. al, (2013) used *negative sampling*
- That is, only keep the observed contexts, and randomly sample some unobserved contexts, perhaps with equal size
- And the probability of observed contexts should be much higher than the probability of observing an unseen context
- And this procedure drastically reduces the number of parameters you need to learn
  - From a scale of 10000 * 10000^4 * 300
  - To (10000 + 10000) * 300

Optimization

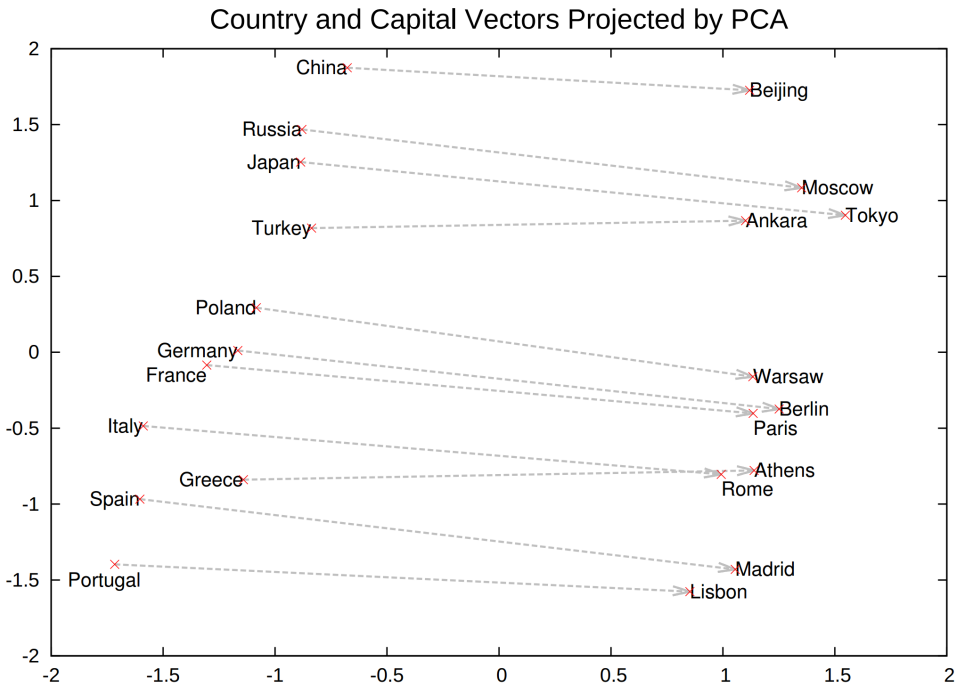
- I am going to skip how they exactly calculated these unknown parameters
- What they did is that they framed the optimization problem as a neural network, which is a type of supervised ML methods
- And people have known how to solve neural networks since 1980s
  - Based on *gradient descent* and *backpropagation*
- Now, assume that an algorithm will help you to map a word to a lower-dimensional vector
- The notations of the three previous slides is based on
- Yoav Goldberg and Omer Levy, *Word2vec Explained: Deriving Mikolov et al.'s negative-sampling word-embedding method*, arXiv:1402.3722 [cs, stat] (2014)
- Goldberg and Levy explained this problem more clearly without using neural networks.

Other popular word embedding models

- Skip-Gram and CWOB: Mikolov et. al, 2013
- GloVe: Jeffrey Pennington, Richard Socher, and Christopher Manning, *Glove: Global Vectors for Word Representation*, Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, 2014, pp. 1532–1543
- FastText: Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov, *Enriching Word Vectors with Subword Information*, Transactions of the Association for Computational Linguistics **5** (2017), 135–146

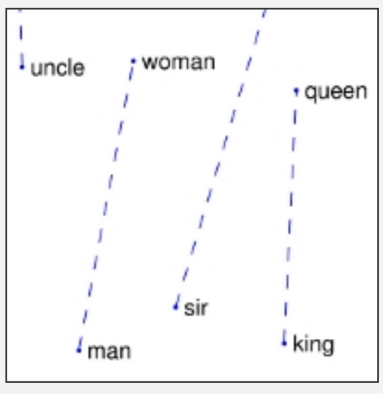
Word embedding capture meanings

The figure illustrates ability of the model to automatically organize concepts and learn implicitly the relationships between them, as during the training we did not provide any supervised information about what a capital city means

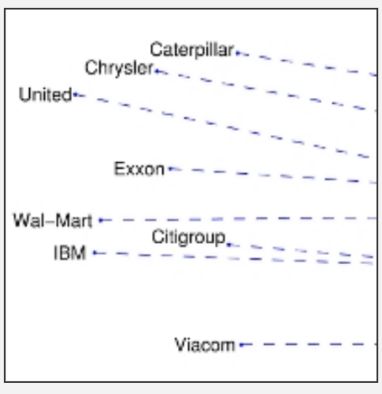


Word embedding capture meanings

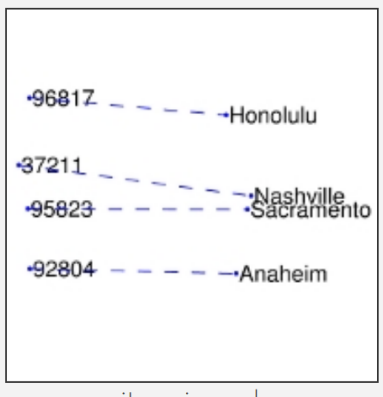
- GloVe: `https://nlp.stanford.edu/projects/glove/images/company_ceo.jpg`
- In particular the ZIP example; they may not often appear together in the same document.
  - Document-term matrix cannot capture this similarity



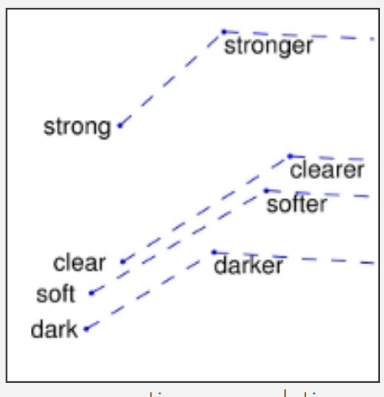
man - woman



company - ceo



city - zip code



comparative - superlative

Word embedding is more efficient

- And surely, word embedding is lower-dimensional
- It eases any subsequent analysis
  - E.g., calculate similarities, clustering, supervised methods

Word embedding is transferable

- Perhaps the greatest advantage of word embedding is that it's transferrable across corpus
- You can take word embedding learned on other corpus, and use that for your task
- This is called transfer learning
- Document-term matrix is not transferable; the word vector loses meanings once orders of documents changed

Transfer Learning in Practice

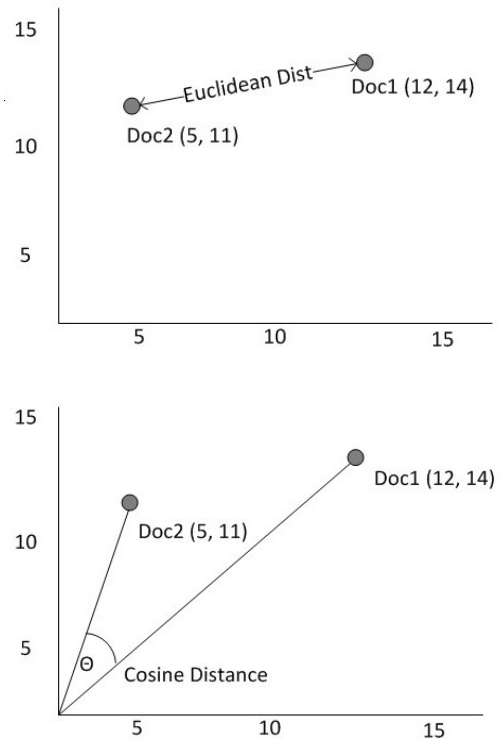
- If your documents are not too far away from the set of documents that word vectors were trained on
  - E.g., you have a bunch of English tweets,
  - And GloVe also released a version of their word vectors trained on Twitters
- Then you can directly take their word embedding as yours
- Instead of training from scratches

Why?

- Why do not use your own data for training?
- Your data is often way smaller than the datasets of some famous word embedding models
  - E.g., the original word2vec used 100 billion words in Google News
- And if you do not have enough data, the algorithm easily over fits
- Furthermore, if the dataset is too small, it cannot captures meanings of words that's not in your corpus
  - E.g., the ZIP address example.

Usage 1: calculate word similarity

- With word embedding, it becomes fairly simple to calculate word similarities
- So that you can find similar words of a given word
- [In-class question]: How can you find similar words, if you do not have word embeddings? What are their shortcomings?
- Typically, people use Euclidean distance or cosine similarity



Using similarity to construct dictionaries

- William L. Hamilton, Kevin Clark, Jure Leskovec, and Dan Jurafsky, *Inducing Domain-Specific Sentiment Lexicons from Unlabeled Corpora*, Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, 2016, pp. 595–605
- Intuition: human have recall biases when constructing dictionaries
- Snow-ball sampling:
  - Start with a set of seed words
  - Find similar words in the corpus, and decide whether to add them to dictionary
    - And similarities are calculated based on word embeddings
  - Iterate the above process until you reach a satisfactory dictionary

Usage 2: supervised learning

- Previously we have seen how you can run a supervised machine model on document-term matrix to predict some outcome Y (e.g., sentiment analysis)
- The only one step we need to do:
- Assume a document has m words in it
- Then a simple vector representation for that document is then the mean of word vectors
- Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov, *Bag of Tricks for Efficient Text Classification*, Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers, 2017

Usage 3: quantify word meaning changes

- Nikhil Garg, Londa Schiebinger, Dan Jurafsky, and James Zou, *Word Embeddings Quantify 100 Years of Gender and Ethnic Stereotypes*, Proceedings of the National Academy of Sciences **115** (2018), no. 16, E3635–E3644
- Question: quantifying changes in stereotypes and attitudes toward women and ethnic minorities in the US in 20th and 21th centuries

Quantify word meaning changes

- Neutral words: e.g., occupations
- Gender words: she/female vs. he/male
- Measure of gender bias:

  (average embedding distance between female words and neutral words) - (average embedding distance between male words and neutral words)

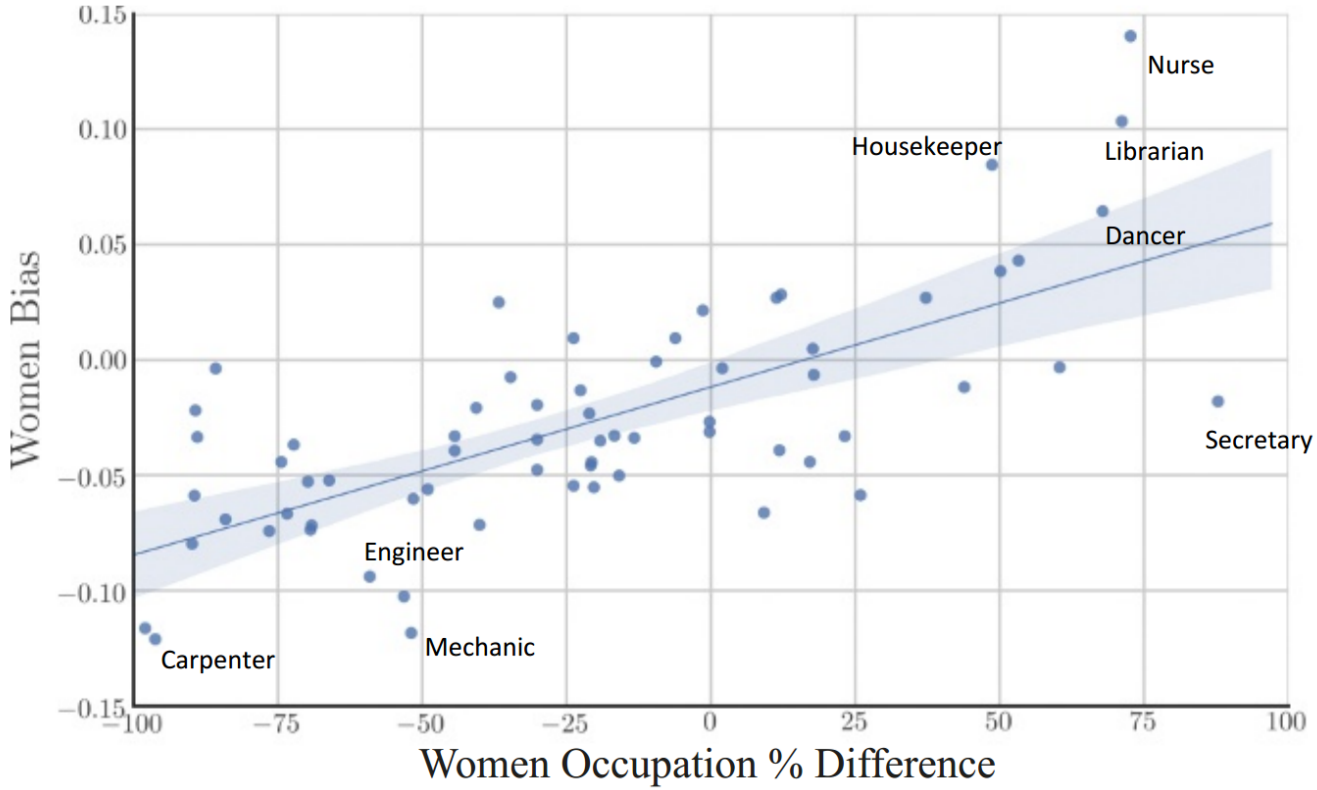
- Negative measure -> more gender bias
- Is this measure reasonable? What are potential problems?

Embeddings

- Did the authors train their own embeddings, or they use transfer learning and just borrow existing embeddings?
- Contemporary: word2vec, Google News
- Historical: word2vc, Google Books/Corpus of Historical American English (COHA)
  - The original authors trained embedding for each decade
- Validation: GloVe, trained on New York Times articles from 1988 to 2005
  - Only this one is trained by the authors; trained over a three year window

Validations

- The authors did some validations to argue that their measure of gender bias is convincing
- E.g, correlating their gender bias with (women occupation % difference) from US census



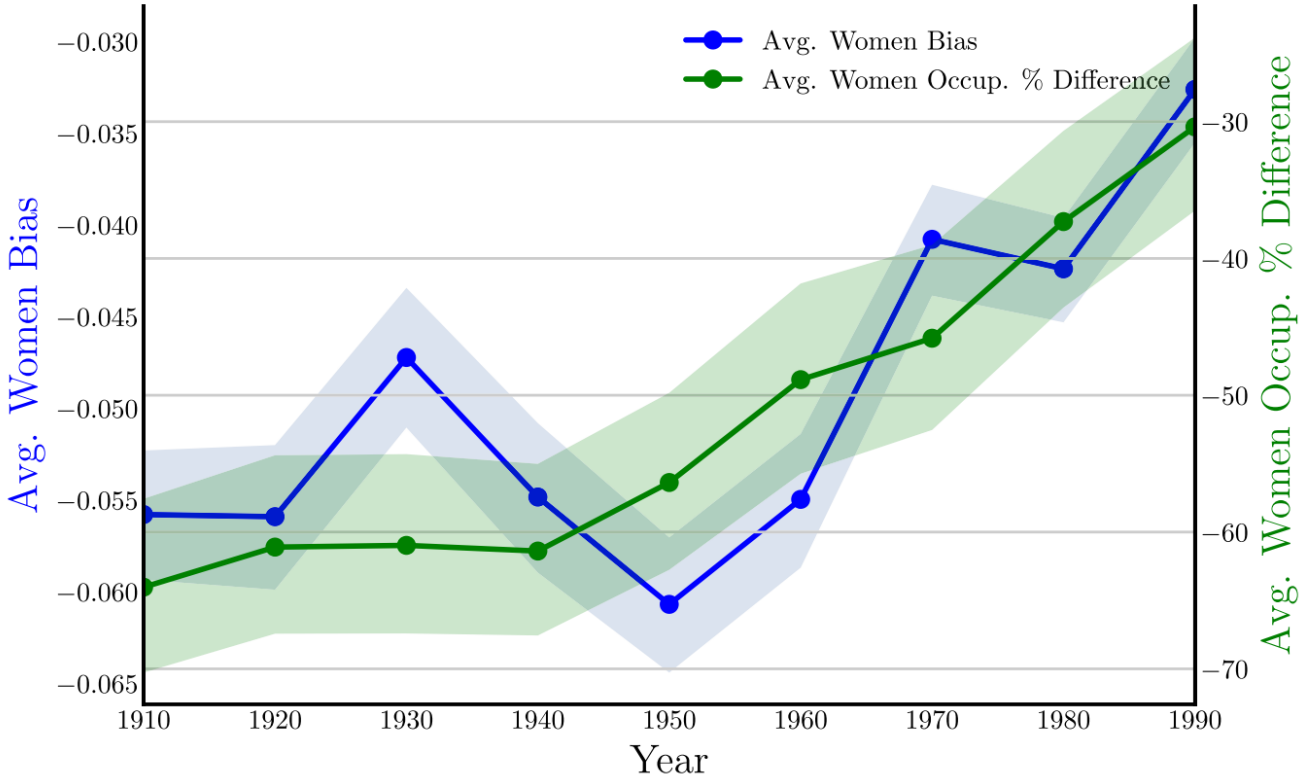
- (I personally is not convinced, this is more like result)

Relating adjectives to gender words

Table 2. Top adjectives associated with women in 1910, 1950, and 1990 by relative norm difference in the COHA embedding

1910	1950	1990
Charming	Delicate	Maternal
Placid	Sweet	Morbid
Delicate	Charming	Artificial
Passionate	Transparent	Physical
Sweet	Placid	Caring
Dreamy	Childish	Emotional
Indulgent	Soft	Protective
Playful	Colorless	Attractive
Mellow	Tasteless	Soft
Sentimental	Agreeable	Tidy

Trends in gender bias



- What happened in 1920s? The authors did not discuss, but I guess it's related to the fact that women were allowed to vote since 1920
- 1930s? Perhaps the Great Depression pushed working women back home, and then followed by WWII.

Discussions

- Good: simple; intuitive
- Shortcomings:
  - Relying on how you choose words
  - Embeddings can be unstable. So can we take their measure of gender bias and use as another variable in other contexts?
  - What do you think?