Lecture 6: Word embedding

Han Zhang

Outline

Dimensionality reduction

Word Embedding

Word Embedding Examples

Curse of dimensionality of Document-term Matrix

- Document-term matrix is high-dimensional
- 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
  - As we have seen from coding examples, if you allow n-grams, then easily we have more columns than observations

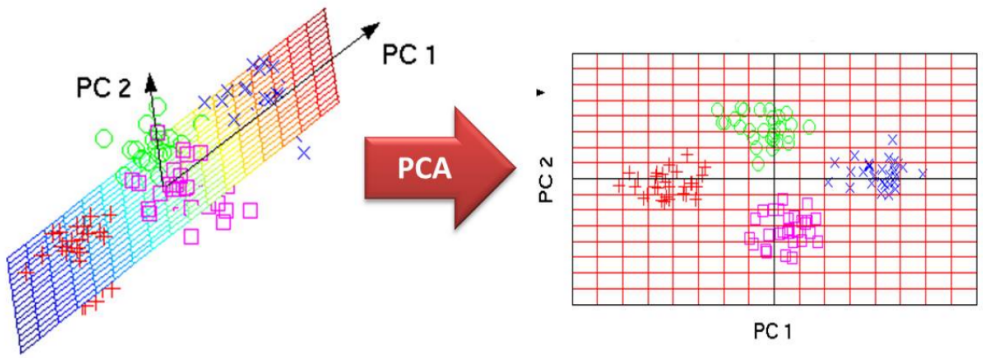
	words										
docs	made	because	had	into	get	some	through	next	where	many	irish
t06_kenny_fg	12	11	5	4	8	4	3	4	5	7	10
t05_cowen_ff	9	4	8	5	5	5	14	13	4	9	8
t14_ocaolain_sf	3	3	3	4	7	3	7	2	3	5	6
t01_lenihan_ff	12	1	5	4	2	11	9	16	14	6	9
t11_gormley_green	0	0	0	3	0	2	0	3	1	1	2
t04_morgan_sf	11	8	7	15	8	19	6	5	3	6	6
t12_ryan_green	2	2	3	7	0	3	0	1	6	0	0
t10_quinn_lab	1	4	4	2	8	4	1	0	1	2	0
t07_odonnell_fg	5	4	2	1	5	0	1	1	0	3	0
t09_higgins_lab	2	2	5	4	0	1	0	0	2	0	0
t03_burton_lab	4	8	12	10	5	5	4	5	8	15	8
t13_cuffe_green	1	2	0	0	11	0	16	3	0	3	1
t08_gilmore_lab	4	8	7	4	3	6	4	5	1	2	11
t02_bruton_fg	1	10	6	4	4	3	0	6	16	5	3

Dimensionality reduction

- One general approach of dealing with high-dimensional data is to reduce its dimensions first
- Dimensionality reduction
- This is a general idea for any type of data
- We first talk with some general strategies
- Then move to word embedding (specifically for texts)

Dimensionality reduction: PCA

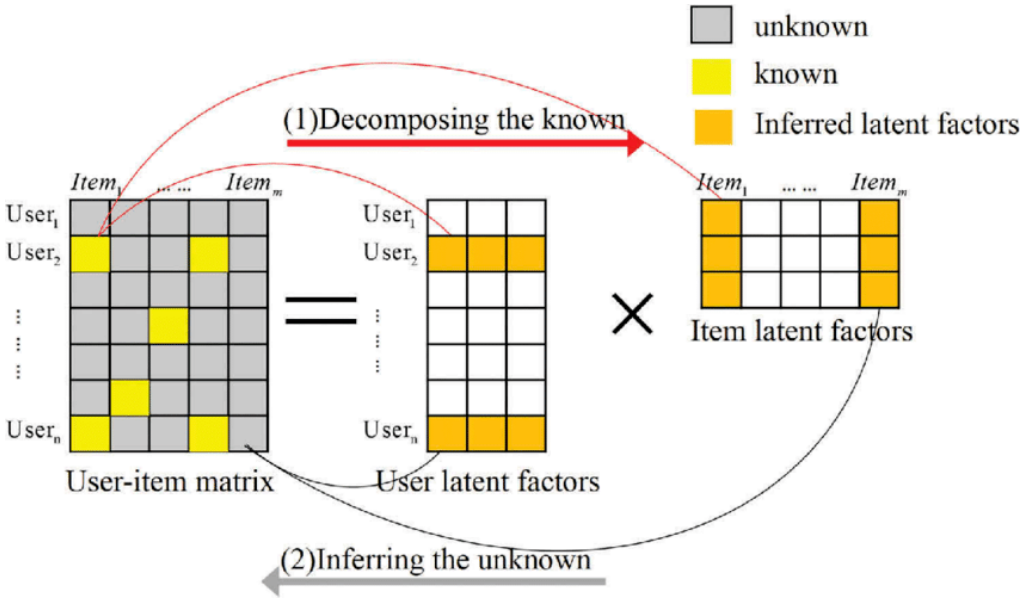
- PCA: principal component analysis
- One of the simplest and widely used dimensionality reduction techniques
- It seeks to project original data onto dimensions that best preserve the variances in the data



PCA

Dimensionality reduction: Nonnegative Matrix Factorization

- PCA only reduces the dimension of rows; not considering columns
- An two-dimensional extension of PCA, considering both features of rows and columns



Nonnegative matrix factorization in social sciences

- Matrix factorization is the idea behind item response theory
- This is how the scores of TOEFL, GRE, LSAT and GMAT were calculated
- Raw data:
  - Row are students
  - Columns are questions
  - each cell is student's answer to a particular question
- There is a large pool of questions and you only answer a sample of questions
- Nonnegative matrix factorization
  - Row: student ability (may be multi-dimensional)
  - Column: question difficulty

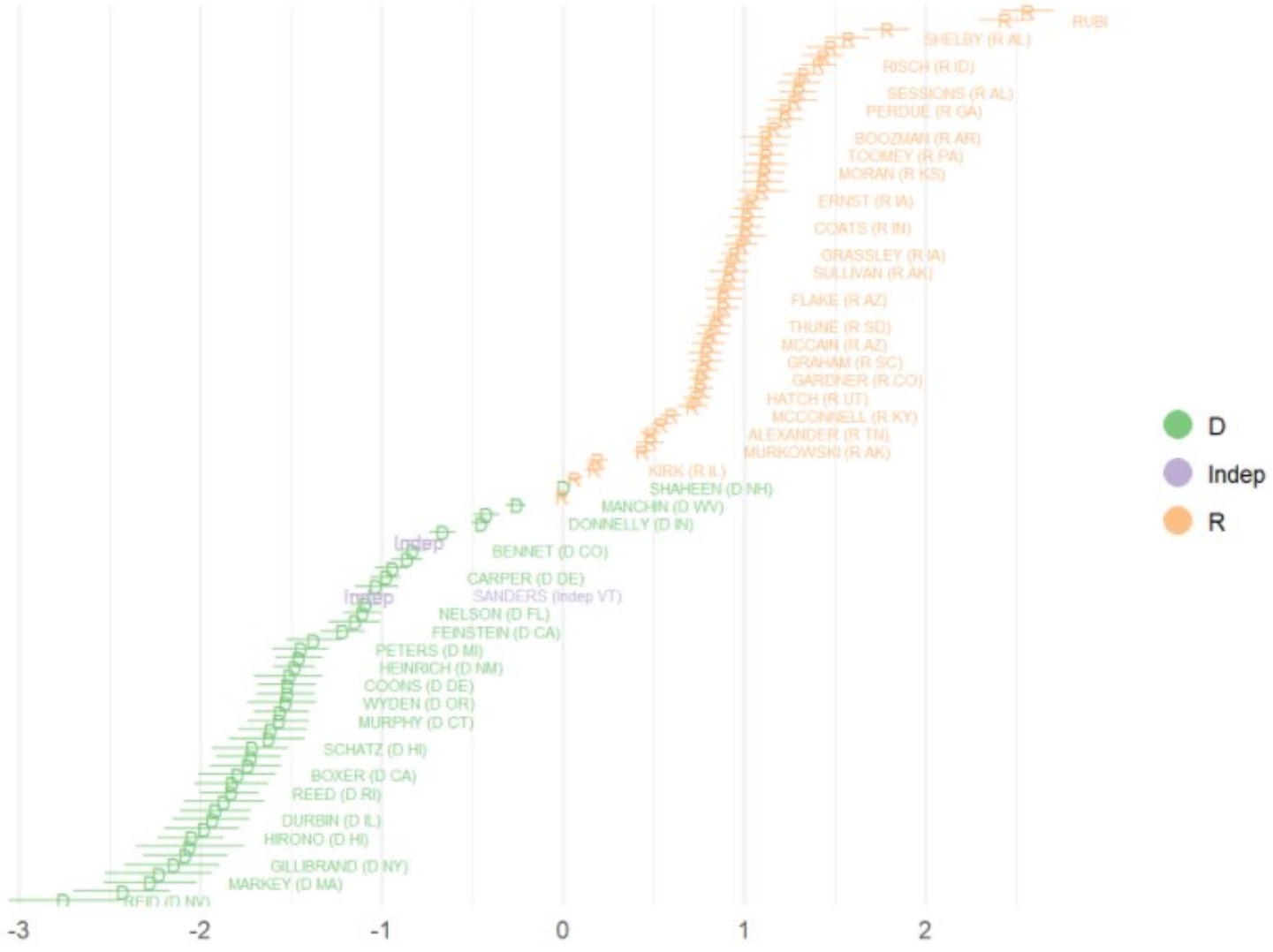
Nonnegative matrix factorization in social sciences

- Matrix factorization is also behind the ideal point estimation (in political science)

Roll Call	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	ETC.
Johnson (D-US)																			N	Y	
Sparkman (D-AL)	N	Y	N	N	Y	N	Y		Y	N	N	Y	N				N	N	Y	Y	
Hill (D-AL)	N	Y	N	N	Y	N	Y	Y	Y	Y	Y			Y	Y	Y	N	N	Y	Y	
Gruening (D-AK)	N	Y	N		Y	N	Y	N	N	N	N	Y	Y				N		Y	Y	
Bartlett (D-AK)	N	Y	N	N	Y	Y	Y	N	N	N	Y	N	Y	Y	Y	Y	Y	N		Y	
Hayden (D-AZ)	N		N	Y	Y	N	Y	Y	Y	N	N	Y	N	Y	Y	Y	N	N	Y	Y	
Fannin (R-AZ)	N	Y	N	Y	Y	N	Y	Y	N	Y	Y	N	Y	Y	Y	Y	N	N	N	N	
Fulbright (D-AR)	N	Y	N		Y	N	Y	N										N	Y		
McClellan (D-AR)	N	Y	N	N	Y	N	Y	Y	Y	N	Y	Y		Y	Y	Y	N	N	Y		
Kuchel (R-CA)	Y	N	Y	Y	Y	N	Y	N	N	N	N	N		Y	N	Y		N	Y	N	
Murphy (R-CA)	N	Y	N	Y	Y	N	Y	Y	N	N	Y	Y	Y	N	Y	Y	N	N	N	N	

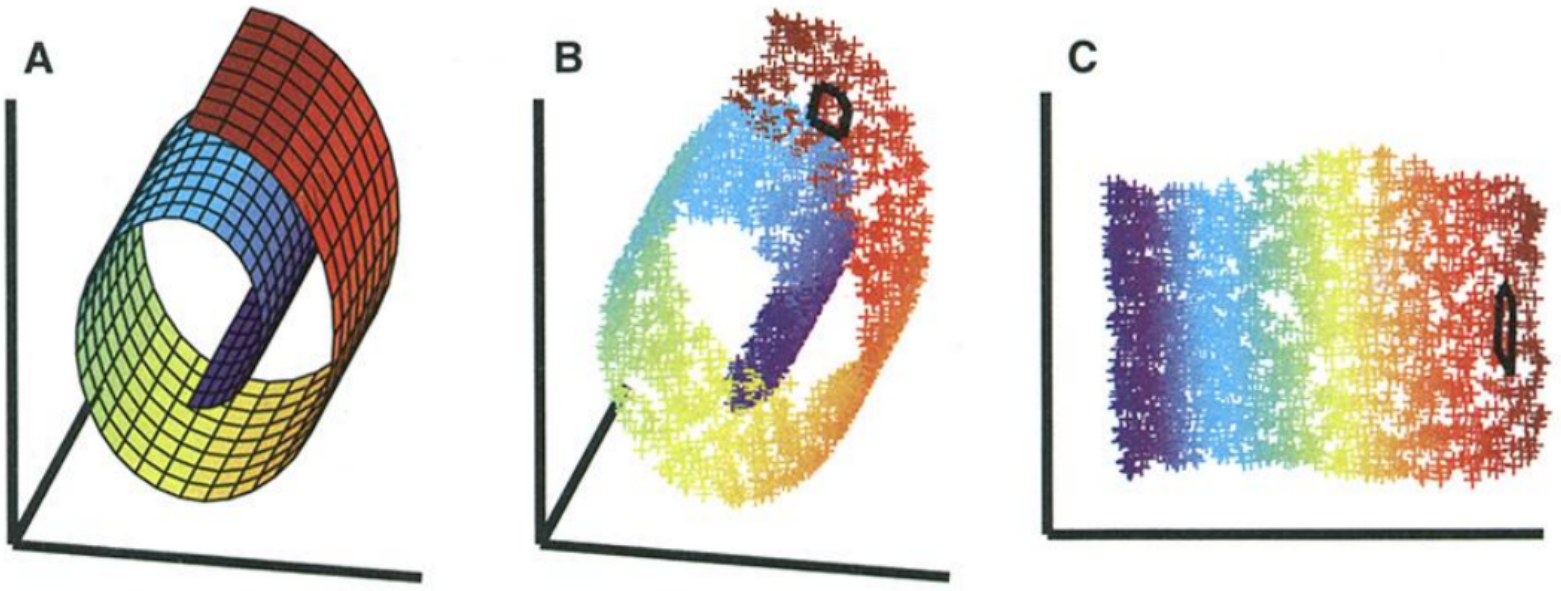
- Use matrix factorization to decompose into two matrix
  - People matrix: each individuals' ideological position
  - Bill matrix: each legislative bill's ideological position

Plotting individual's ideological positions



Dimensionality reduction: non-linear data

- PCA and nonnegative matrix factorization are linear methods
- They cannot work on the below type of non-linear data
- Sam T. Roweis and Lawrence K. Saul, *Nonlinear Dimensionality Reduction by Locally Linear Embedding*, Science **290** (2000), no. 5500, 2323–2326

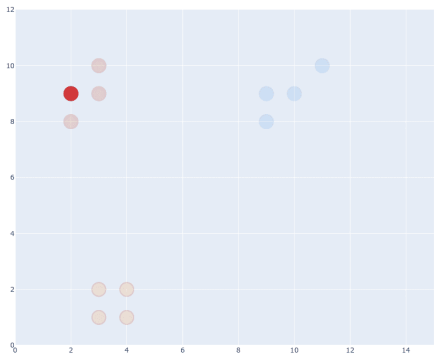


Dimensionality reduction: t-SNE

- t-SNE: t-distributed stochastic neighbor embedding
- Laurens van der Maaten and Geoffrey Hinton, *Visualizing Data using t-SNE*, Journal of Machine Learning Research **9** (2008), no. Nov, 2579–2605
- t-SNE is suited if you have nonlinear data
- And in particular, if you want to visualize them
- Note: this one can be even more slower than PCA

Intuition of t-SNE

- We can use the nearest data points to predict a single data X_i



- Similarly, if we map X_i to a lower-dimensional representation Z_i (called embedding here)
  - And do so for other data points X_j
- Then using Z_j to predict Z_i will also have a good prediction performance
- Shortly you will see that word embedding is essentially the same idea applied to texts

Sparse Representations of words

- Document-term matrix is high dimensional
- Partly because it is very sparse: most entries in the word vector is 0

docs	words made	because	had	into	get	some	through	next	where	many	irish
t06_kenny_fg	12	11	5	4	8	4	3	4	5	7	10
t05_cowen_ff	9	4	8	5	5	5	14	13	4	9	8
t14_ocaolain_sf	3	3	3	4	7	3	7	2	3	5	6
t01_lenihan_ff	12	1	5	4	2	11	9	16	14	6	9
t11_gormley_green	0	0	0	3	0	2	0	3	1	1	2
t04_morgan_sf	11	8	7	15	8	19	6	5	3	6	6
t12_ryan_green	2	2	3	7	0	3	0	1	6	0	0
t10_quinn_lab	1	4	4	2	8	4	1	0	1	2	0
t07_odonnell_fg	5	4	2	1	5	0	1	1	0	3	0
t09_higgins_lab	2	2	5	4	0	1	0	0	2	0	0
t03_burton_lab	4	8	12	10	5	5	4	5	8	15	8
t13_cuffe_green	1	2	0	0	11	0	16	3	0	3	1
t08_gilmore_lab	4	8	7	4	3	6	4	5	1	2	11
t02_bruton_fg	1	10	6	4	4	3	0	6	16	5	3

Problems of sparse representation

- Sparse representation has some problems:
  - Often waste of computer resources/memory. Most elements are 0 but we still have to store them in computer
  - To make it worse, when the number of documents increases, dimension can also increases
  - E.g., there may be only 1000 words used in 200 documents, but may be 10,000 words in 2,000 documents
  - When the length of word vector increases, it's also getting complex to do any calculation

Dense representation of words

- We can perform dimensionality reduction to obtain dense representation of a word
  - The reduced word vector length is usually fixed
  - This is different from document-term matrix which is a sparse representation
    - when the number of observation increases, dimension can also increases
- Think the the vector representation of a word should means an intrinsic meaning of a word which does not change with respect to how many documents you choose
  - And naturally, most entries in the lower-dimensional vector should not be 0

Word embedding

- In the old days, you can perform dimensionality-reduction method we just introduced
  - 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.

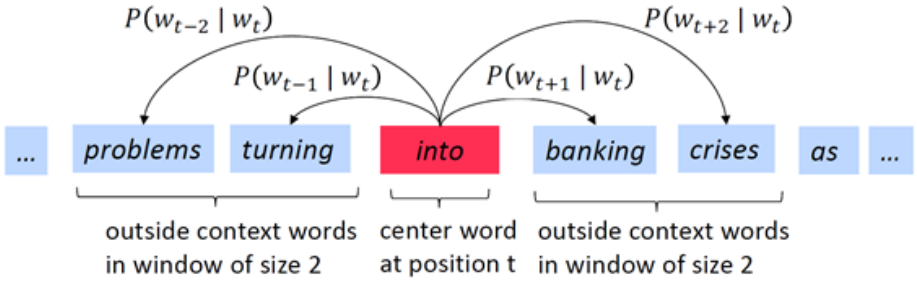
Word embedding: CBOW vs. Skip-Gram

- 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

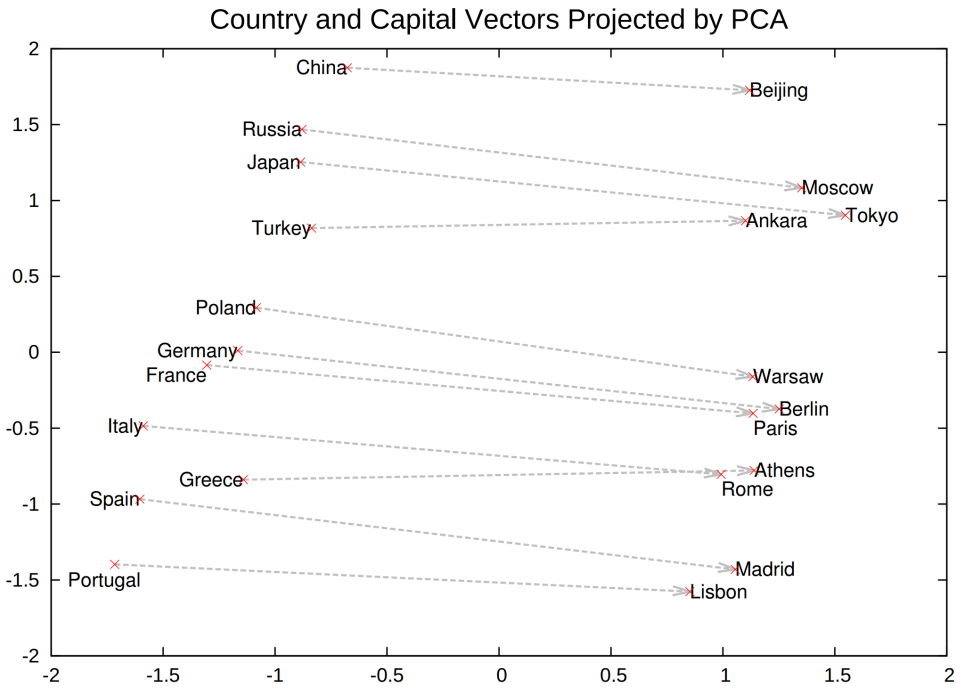
- I am going to skip how they exactly calculated these unknown parameters
- What they did is that they framed the optimization problem as a neural network, which is a type of supervised ML methods
- And people have known how to solve neural networks since 1980s
  - Based on *gradient descent* and *backpropagation*
- 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

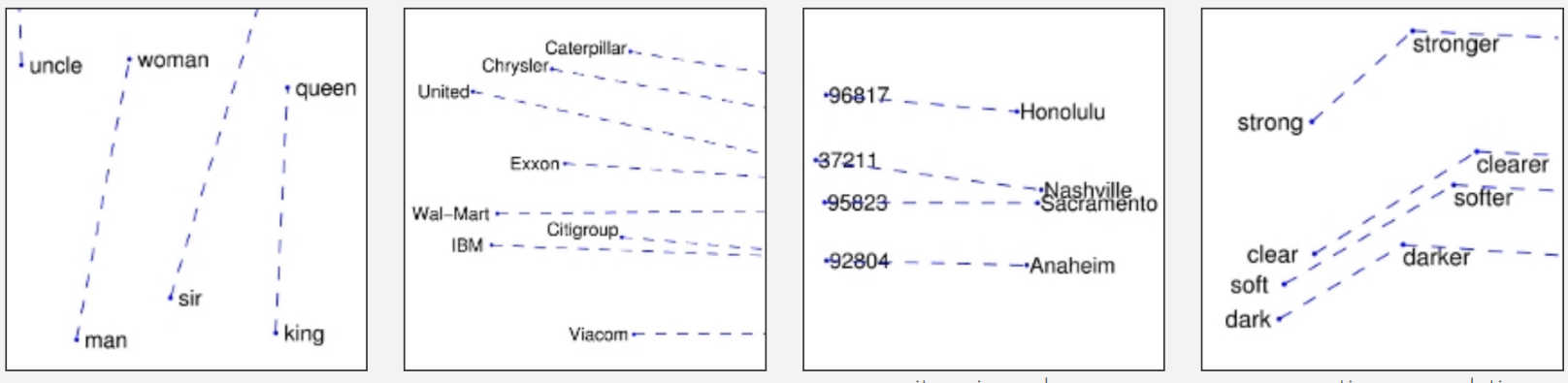
Word embedding capture meanings

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

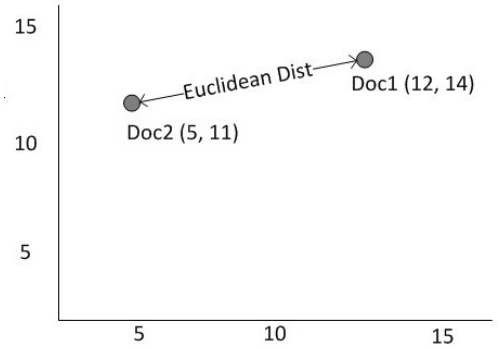
- If your documents are not too far away from the set of documents that word vectors were trained on
  - E.g., you have a bunch of English tweets,
  - And GloVe also released a version of their word vectors trained on Twitters
- Then you can directly take their word embedding as yours
- Instead of training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

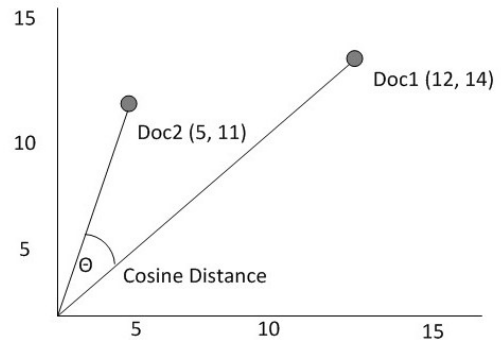
Why?

- 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 overfits
- 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.
- Word embedding learns the intrinsic meanings of a word and should stays the same even you have a different article

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)
- We can also use the word embedding to represent each document
- 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)

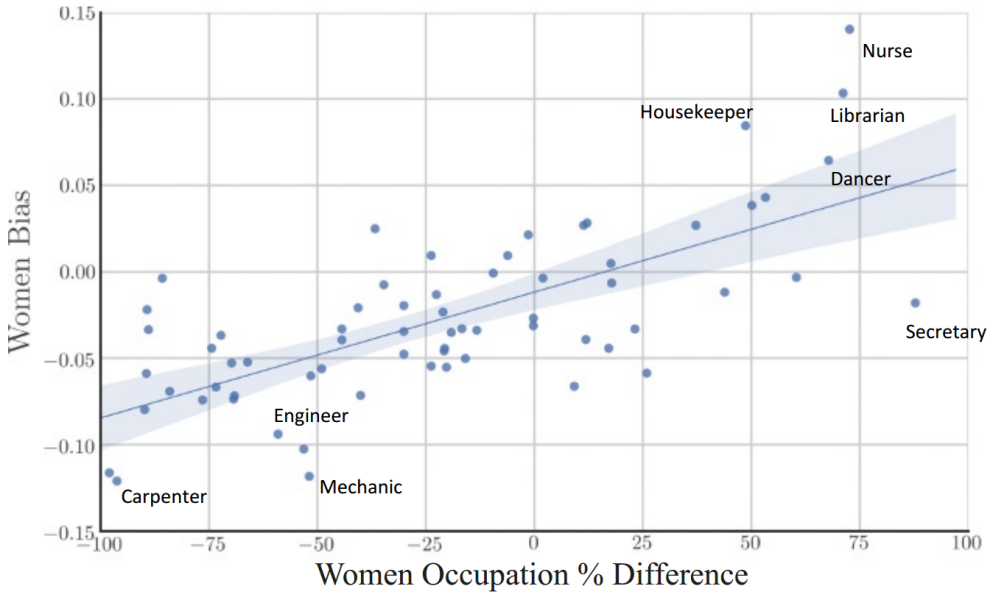
- If this gender bias measure is positive, then the words are disproportionally more likely to be associated with women
- Is this measure reasonable? What are potential problems?

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



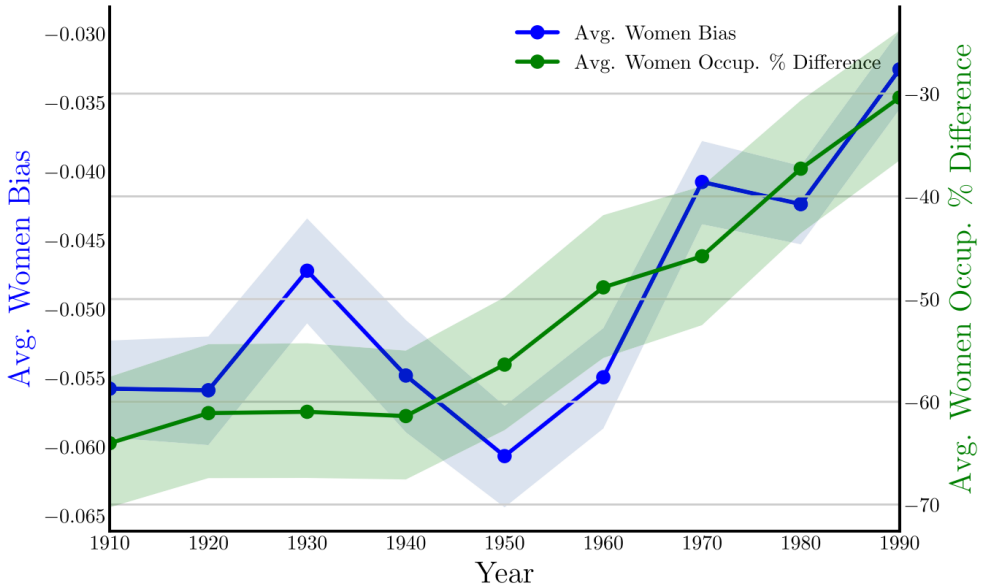
- Are you convinced?

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.
- 1970s? Civil Rights Movements?

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?