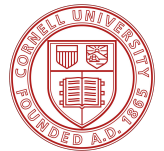


Evaluating the Stability of Embedding-based Word Similarities

Maria Antoniak & David Mimno



Thank you for the introduction! Today I'll be talking about our work on measuring and explaining the instability of cosine similarities between word vectors and what do about it.

I'll begin with a small case study of the type of copus-centered work that I'll be discussing. We'll be using embeddings in a way that you might not be familiar with. Imagine we are interested in the biases surrounding the term *marijuana* and we want to investigate how these biases across different corpora.

Most similar words to *marijuana*

NYT Sports	Reddit AskScience	4th Circuit (US Federal Court of Appeals)
cocaine procedures smoking testing addiction purposes steroid blaming suspensions positive		

First we might consider the NYT Sport section. If we find the ten words with the greatest cosine similarity to marijuana using the GloVe word embedding training algorithm, we find these results, which look like marijuana is most associated with testing, addiction, and steroids.

Most similar words to *marijuana*

NYT Sports	Reddit AskScience	4th Circuit (US Federal Court of Appeals)
cocaine procedures smoking testing addiction purposes steroid blaming suspensions positive	cannabis smoking smoke tobacco thc drug weed caffeine drugs effects	

We might compare to a Reddit corpus, where anyone can post anonymously, to see what the average person thinks about marijuana. If we train on this data, We might interpret these results as implying that Reddit users see marijuana as closer to tobacco and caffeine than other drugs.

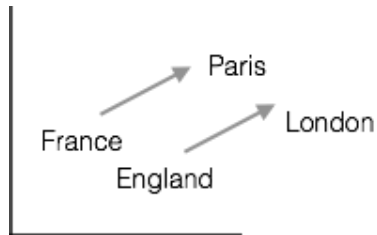
Most similar words to *marijuana*

NYT Sports	Reddit AskScience	4th Circuit (US Federal Court of Appeals)
cocaine procedures smoking testing addiction purposes steroid blaming suspensions positive	cannabis smoking smoke tobacco thc drug weed caffeine drugs effects	cocaine heroin kilograms crack distribute drugs grams smoked growing possession

Finally, if we train on the 4th Circuit of the US Federal Court of Appeals. Looking at this ranked list, it seems that the court views marijuana as closer to drugs like cocaine, heroin, and crack. This bias could (and does) have serious legal, real-world consequences.

That said, while these stories seem to make sense, and align with my previous conceptions of the authors of these corpora, we should question whether these results are significant. If we trained our model again, would we get the same results? If our corpus were altered in some seemingly innocuous way, would we get the same results? How can we be confident in these stories we're telling ourselves about these sets of ranked words?

What do embeddings represent?

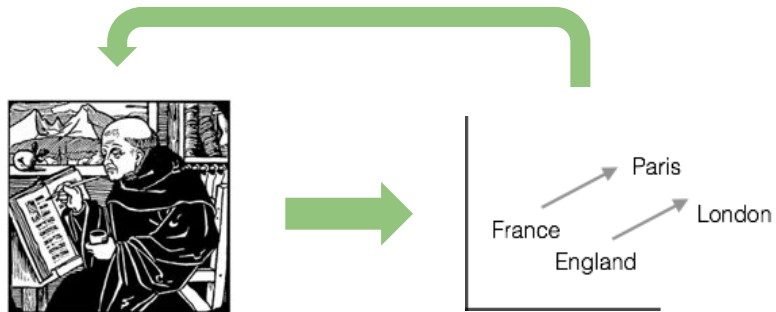


5

Ok, where do these similarities come from? Word embeddings are mappings of words to points in a K -dimensional space where K is much smaller than the size of the vocabulary. The cosine similarity between word vectors can reveal latent semantic relationships between words. These embeddings have proven very useful for downstream tasks, but they're also increasingly popular in fields such as digital humanities and computational social science.

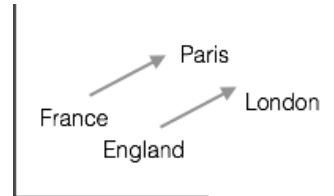
A typical view of embeddings is that they give us knowledge about fixed universal properties of language.

What do embeddings represent?



Another view is that the embeddings represent the mental models of authors in a particular time and place, like in our marijuana example. We might think that since the authors produced the text that led to the embedding, we can go backwards, and learn something about those authors.

What do embeddings represent?



However, in between the embedding and the author is a the dataset created by the authors and organized by various curators over time. Documents could be re-ordered or missing.

Word embeddings do not measure properties of language;
they measure properties of a *curated* corpus.

8

Therefore, it's important that we remember that word embeddings do not directly measure properties of language or of an author's mental model; they measure properties of a curated corpus.

This leads us to two different views of embeddings.

Two views of embeddings

Downstream-centered	Corpus-centered
Big corpus	Small corpus, difficult or impossible to expand
Source is not important	Source is the object of study
Only vectors are important	Specific, fine-grained comparisons are important
Embeddings are used in downstream tasks	Embeddings are used to learn about the mental model of word association for the authors of the corpus

9

On the one hand, we have a downstream-centered view of embeddings, which is a common point of view and will be familiar to many of us with NLP and ML backgrounds. In this view, embeddings are assumed to represent general characteristics of language. We train on a large corpus whose source is not very important, so long as the embeddings extend to our downstream task, and we often use the same model for many different tasks.

On the other hand, we have a corpus-centered view, which focuses on comparisons between smaller collections, as in our marijuana example. Fine-grained comparisons between word vectors are important, and the work seeks to learn about the mental model of the authors of the corpus.

An example to illustrate the difference between these views is recent work detecting gender bias in word embeddings. In the downstream centered view, the gender bias is viewed as harmful, as something to be filtered out before the embeddings are used downstream. In the corpus-centered view, bias in embeddings provides evidence of bias in the training corpus, and in the corpus authors.

Two views of embeddings

Downstream-centered	Corpus-centered
Big corpus	Small corpus, difficult or impossible to expand
Source is not important	Source is the object of study
Only vectors are important	Specific, fine-grained comparisons are important
Embeddings are used in downstream tasks	Embeddings are used to learn about the mental model of word association for the authors of the corpus

10

In this work, we will focus on the corpus-centered view of embeddings, where fine-grained differences between word vectors is important.

Corpus parameters

- **Order** of documents
- **Presence** of documents
- **Size** of corpus
- **Size** of documents

Since we are focused on the corpus, we can question its construction. How would the order of the documents affect the training and resulting cosine similarities between word vectors? The presence of documents? The size of the corpus? The size of the documents?

Three experimental settings

Setting	Tests...	Run 1	Run 2	Run 3
Fixed	variability due to algorithm (baseline)	A B C	A B C	A B C
Shuffled	variability due to document order	A C B	B A C	C B A
Bootstrap	variability due to document presence	B A A	C A B	B B B

12

To examine these questions, we train multiple models while manipulating the training corpus in three ways.

In the first setting, we keep the order and presence of the documents fixed. This is our baseline, which will measure variability due to the algorithm rather than the corpus.

In the second setting, we shuffle the documents, so as to test the variability due to document order.

Finally, in the third setting, we take bootstrap samples of the corpus, in which we sample randomly with replacement, to test the variability due to document presence.

Corpus	Number of documents	Number of unique words	Words per document
<i>NYT Sports (2000)</i>	8,786	12,475	708
<i>NYT Music (2000)</i>	3,666	9,762	715
<i>Reddit AskScience</i>	331,635	16,901	44
<i>Reddit AskHistorians</i>	63,578	9,384	66
<i>4th Circuit (U.S. Federal Courts of Appeals)</i>	5,368	16,639	2,281
<i>9th Circuit (U.S. Federal Courts of Appeals)</i>	9,729	22,146	2,108

13

We apply each of those three settings to a set of six datasets.

We chose these datasets to be:

- Publicly available
- Suggestive of social research questions
- Varied in corpus parameters (e.g. topic, size, vocabulary)
- Much smaller than the standard corpora typically used to train word embeddings (e.g. Wikipedia, Gigaword)

(counts are after removing words that appear fewer than 20 times)

Algorithms

- LSA (Deerwester et al., 1990; Landauer and Dumais, 1997)
- SGNS (Mikolov et al., 2013)
- GloVe (Pennington et al., 2014)
- PPMI (Levy and Goldberg, 2014)

Finally, for each of the three settings and each of the six datasets, we examine four different training algorithms. LSA, SGNS, GloVe, and PPMI.

We're interested in properties of the corpus, but there could be variation from algos that we need to account for before we can measure variability due to the corpus.

Variation in algorithms

$$X \sim N(0, \sigma^2)$$

Random initialization

First, we expect some variation due to the random initialization of word vectors in SGNS and GloVe.

Variation in algorithms

$$J_{\text{NEG}}^{(t)} = \log Q_{\theta}(D = 1 | \text{the, quick}) + \log(Q_{\theta}(D = 0 | \text{sheep, quick}))$$

<https://www.tensorflow.org/tutorials/word2vec>

randomized SVD,
stochastic gradient descent,
random construction of
negative samples

16

Second, we expect variation due to various training choices, such as using a randomized SVD solver for LSA, stochastic gradient descent in SGNS, and the random construction of negative samples for SGNS.

Variation in algorithms

Sir Walter Elliot, of Kellynch Hall, in Somersetshire, was a man who, for his own amusement, never took up any book but the Baronetage; there he found occupation for an idle hour, and consolation in a distressed one; there his faculties were roused into admiration and respect, by contemplating the limited remnant of the earliest patents



Sir Walter Elliot, _ Kellynch Hall, __ Somersetshire, was _ man who, ___ his ___ amusement, never took __ _ book but __ Baronetage; there __ found occupation for __ idle hour, ___ consolation ___ distressed __; there ___ faculties were roused ___ admiration and respect, by contemplating ___ limited remnant ___ earliest patents

frequent word subsampling

Finally, we expect some variation due the subsampling of frequent words, especially for PPML.

Methods

Train 50 models ...

... for each algorithm

[LSA, SGNS, GloVe, PPMI]

... for each setting

[Fixed, Shuffled, Bootstrap]

... for each corpus

[NYT Sports, NYT Music, AskScience, AskHistorians, 4th Circuit, 9th Circuit]

To sum up, we train 50 word embedding models for each of the four algorithms, for each of the three settings, and for each corpus.

Methods

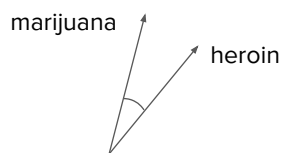
- Select **query words**
- Calculate the **cosine similarity** with all other words
- Calculate the **mean and standard deviation** of the cosine similarities across each set of 50 models

For each corpus, we train an LDA topic model and select 20 high probability query words.

For each of these words, we calculate the cosine similarity with all the other words in the vocabulary, and then we find the mean and standard deviation of the cosine similarities across each set of 50 models.

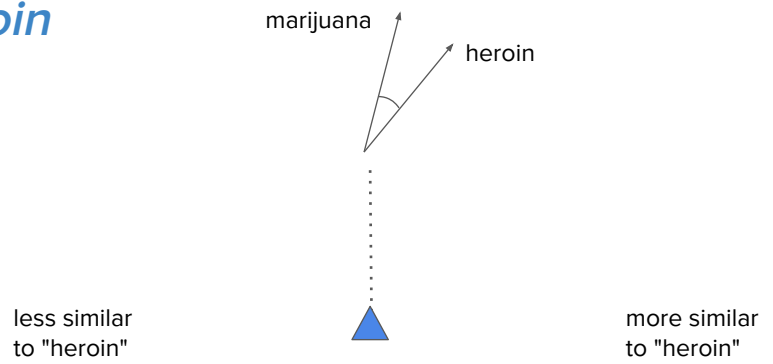
Thinking back to our marijuana example, this should allow us to compare different models and determine the significance of our word rankings.

marijuana vs. heroin



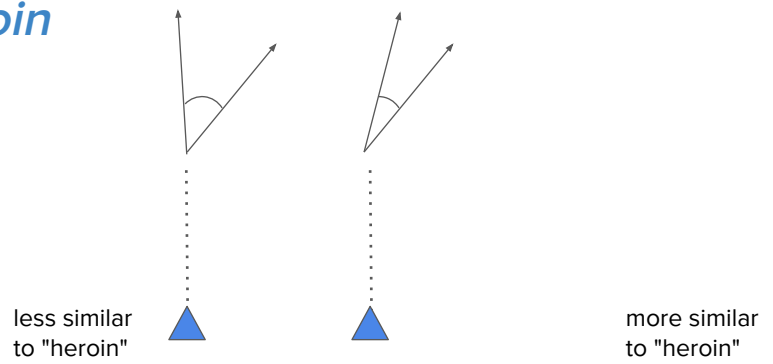
Let's return to our marijuana example. We use cosine similarity to find the similarity between two vectors, in this case, "marijuana" and "heroin."

marijuana vs. heroin



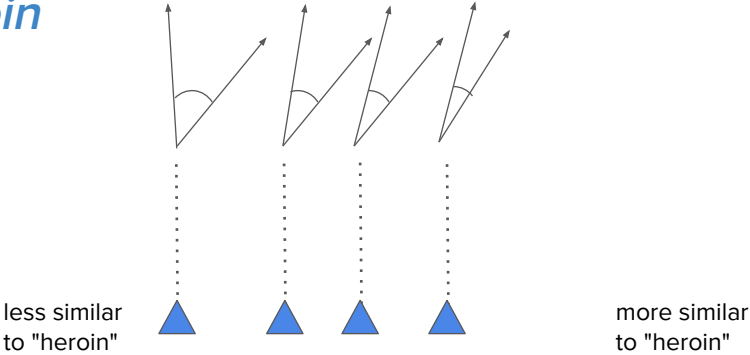
We'll plot this cosine similarity on an axis spanning from less similar to more similar to "heroin"

marijuana vs. heroin



We re-train the model and repeat the same process, resulting in a slightly difference cosine similarity.

marijuana vs. heroin



We repeat again and again, adding more training iterations.

marijuana vs. heroin

less similar
to "heroin"

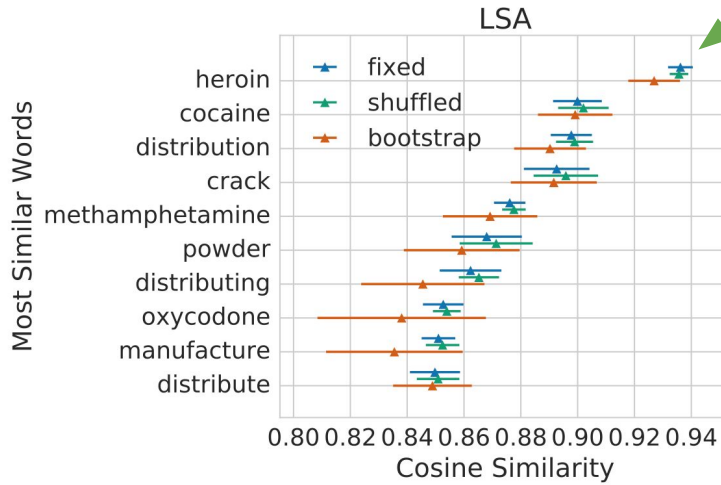


more similar
to "heroin"

Finally, across these iterations, we find the mean and standard deviation of the cosine similarities. Remember this error bar for the next slide.

Most similar words to *marijuana*

standard deviation
across 50 models



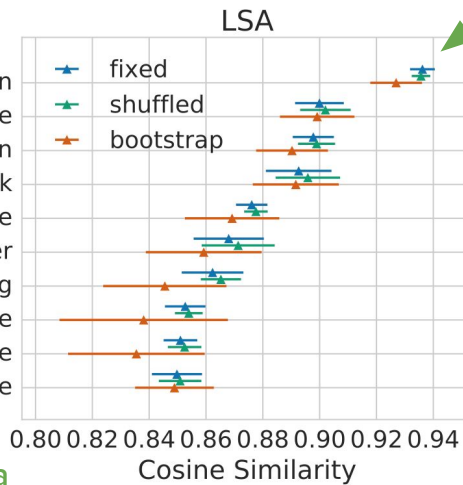
In this figure, each of the bars represents that same mean and standard deviation across 50 training iterations for a particular model, dataset, and setting.

Most similar words to *marijuana*

standard deviation across 50 models

most similar word to marijuana

Most Similar Words

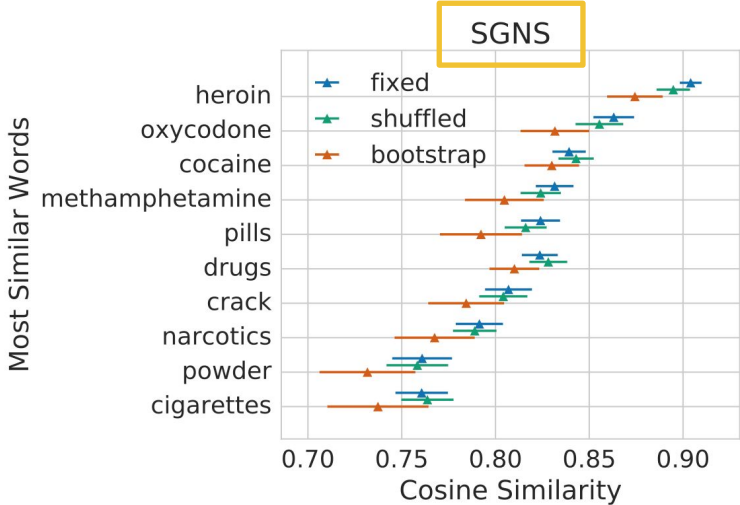


10th most similar word to marijuana

On the y-axis are the closest words ranked by average cosine similarity in the fixed setting to the query word “marijuana”.

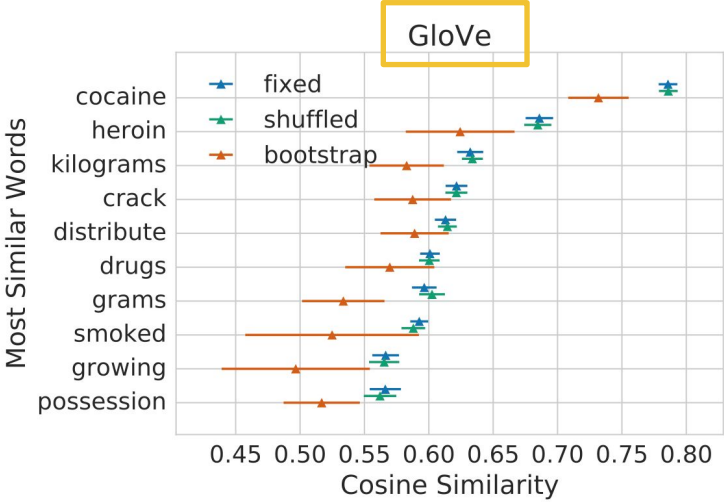
The blue bars indicate the fixed setting, the green bars indicate the shuffled setting, and the red bars indicate the bootstrap setting. The blue and green bars look very similar, indicating that shuffling the documents doesn’t introduce much variation beyond that introduced by the algorithm itself. The red bars are consistently larger, indicating that the presence of specific documents strongly affects the cosine similarities between word vectors.

Most similar words to *marijuana*



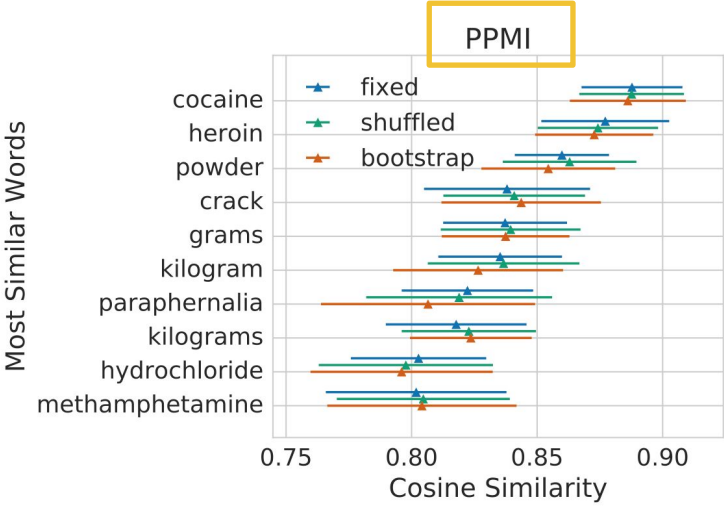
Here we see the same test, this time for SGNS.

Most similar words to *marijuana*

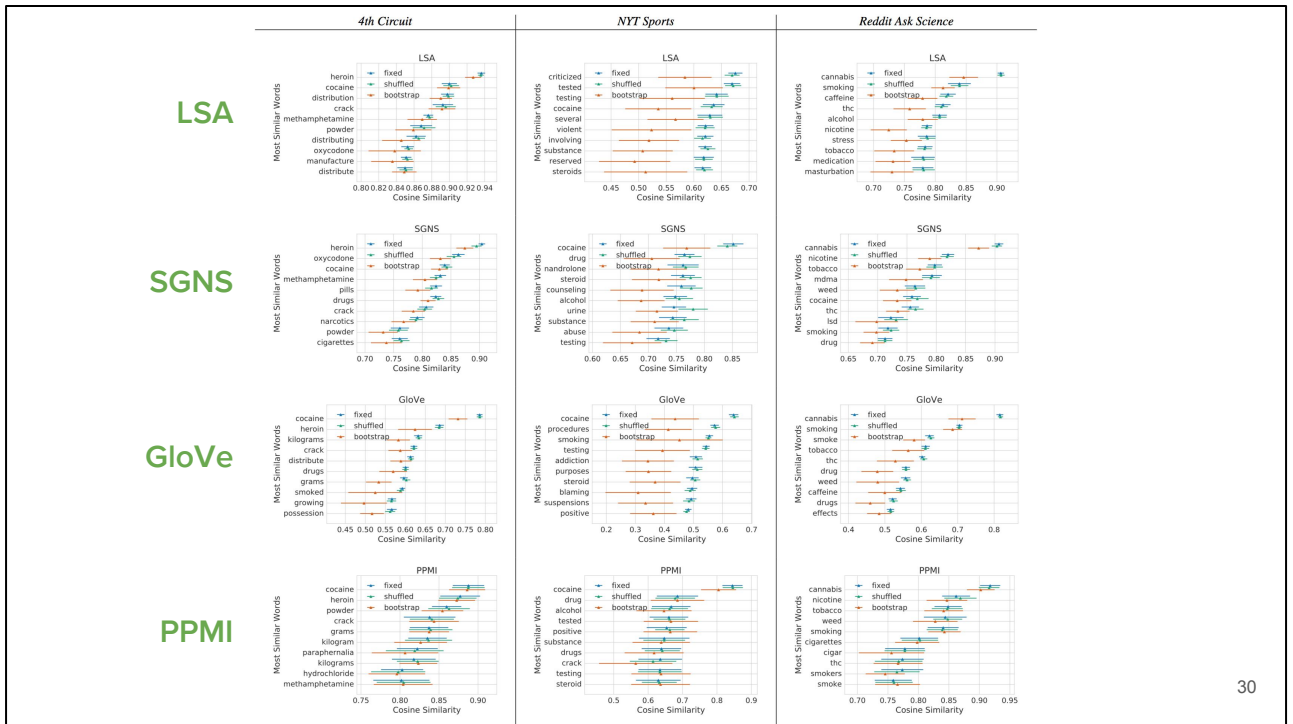


Same for GloVe.

Most similar words to *marijuana*

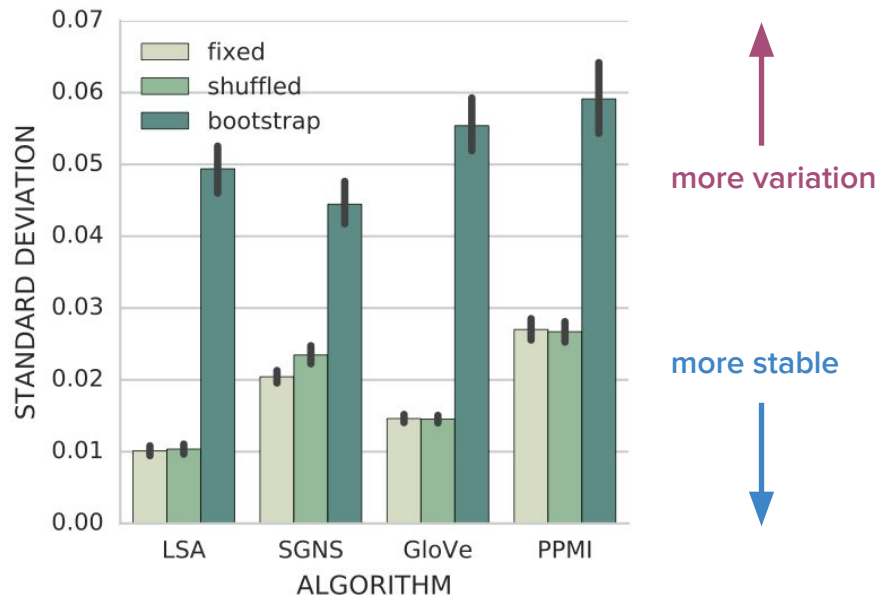


And similar for PPMI, where the red bars aren't as different from the other settings, but are slightly larger.



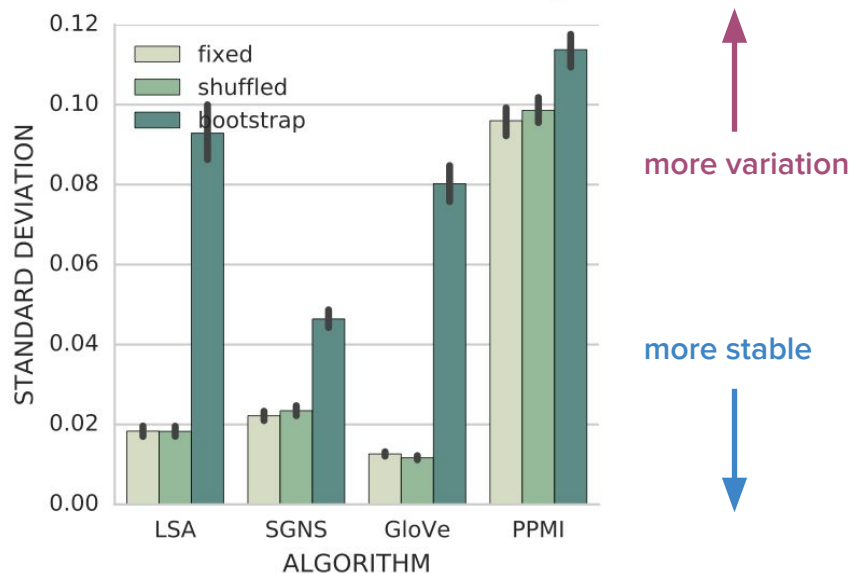
Now, I know that you can't read the words on this slide, but you can read paper for details, and I'd only like you to note that in aggregate, we see that across algorithms, and datasets, the red bars are bigger.

Standard Deviation in the 9th Circuit Corpus



Let's take another view of those results. For the 9th Circuit Corpus, we see that the average standard deviation is greater for the bootstrap setting across all algorithms.

Standard Deviation in the NYT Music Corpus

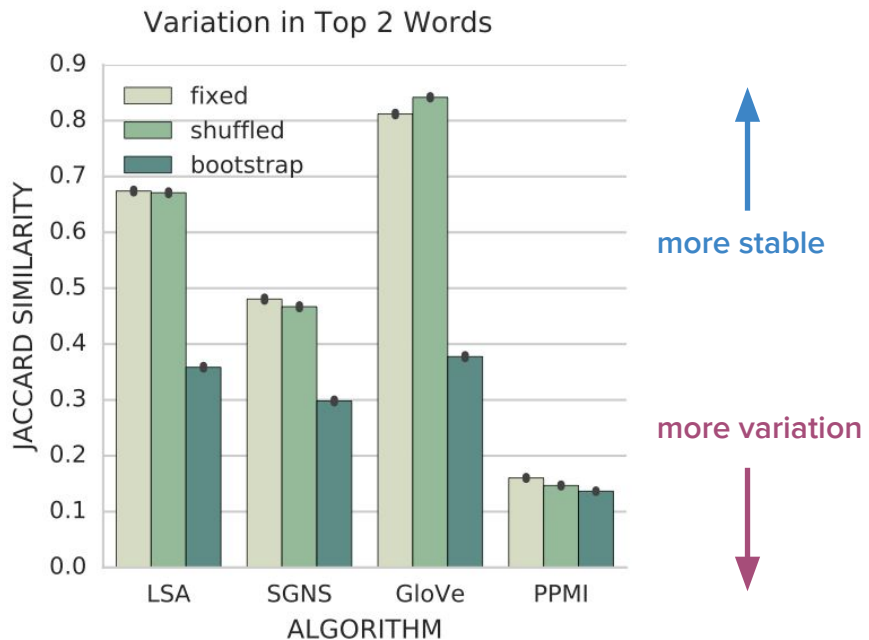


32

Similarly for the NYT Music Corpus, the bootstrap setting has a larger average standard deviation of cosine similarities, which means it introduces more variation, though here the effect is smaller for PPMI.

In sum,

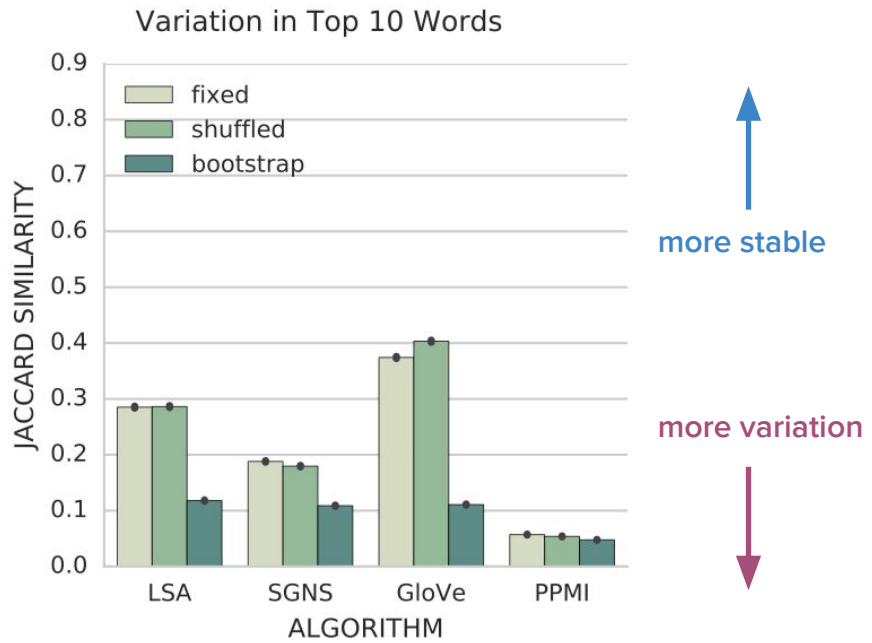
- None of the documents aren't sensitive to document order
- PPMI token subsampling produces results similar to the Bootstrap setting
- All algorithms are sensitive to presence of specific documents (Bootstrap setting)



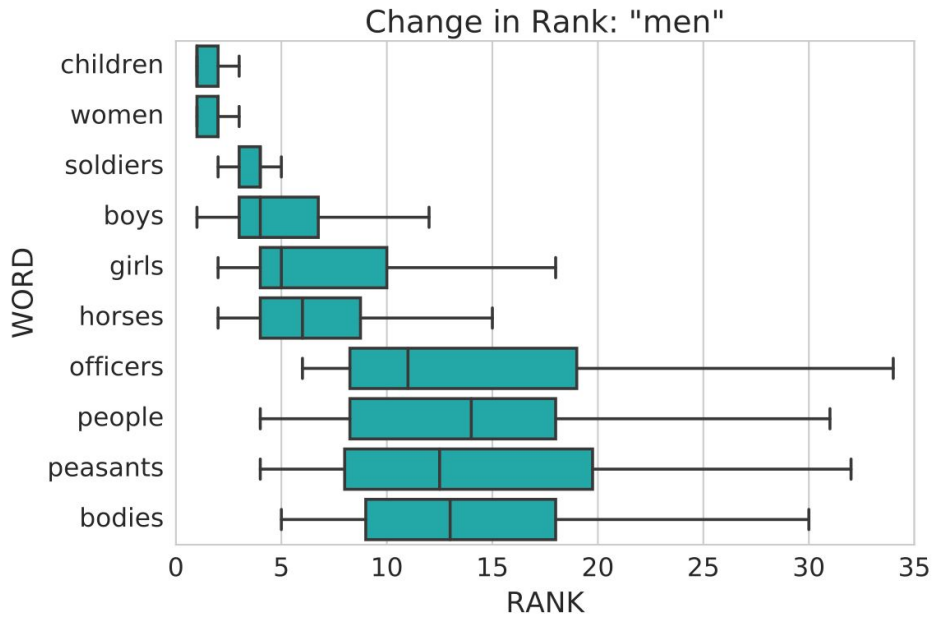
Let's look beyond cosine similarities and consider the rank of the words. Do the cosine similarities vary consistently, e.g. if all the similarities are lower by 0.1, we get the same ranked list, or do the words actually change rank across training iterations?

In the paper, we show examples of specific queries that return entirely different words for each bootstrap iteration.

Here we show the mean jaccard similarities for the top 2 words in the Reddit AskHistorians corpus. In this case, a lower score indicates less stable embeddings, we see again that the bootstrap settings introduces the most variation, with a smaller effect for PPMI.



Here we see the same results but for the top 10 words. Again, the bootstrap setting is the lowest.

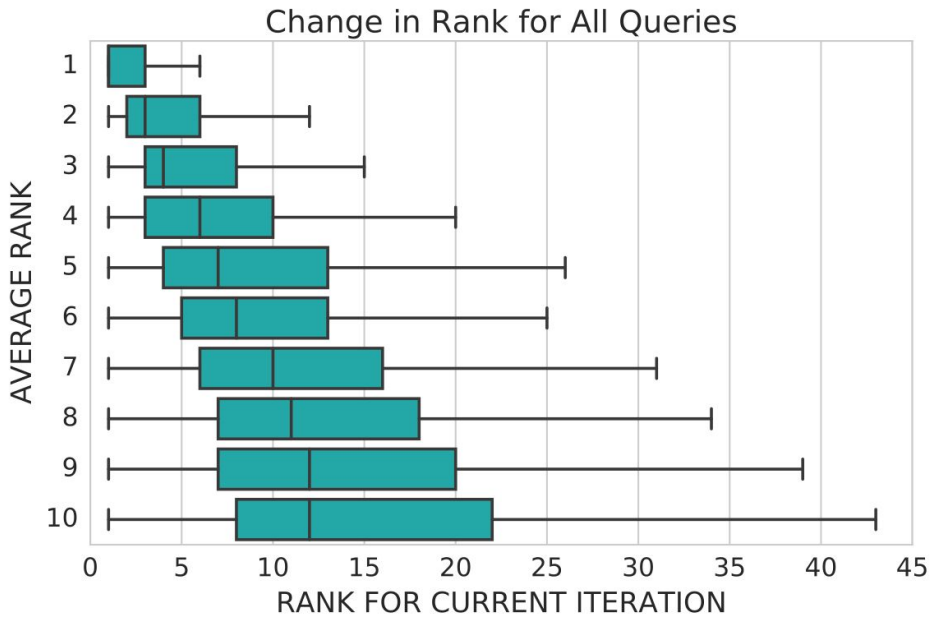


35

Here we again see the change the rank, this time for a single query, “men” in the Reddit AskHistorians corpus. Even the top two words can change rank, and by the time we reach the fourth word, the rank can change significantly, dropping all the way out of the top ten list.

Not surprising that increases as we move down, but surprising how much it can vary!

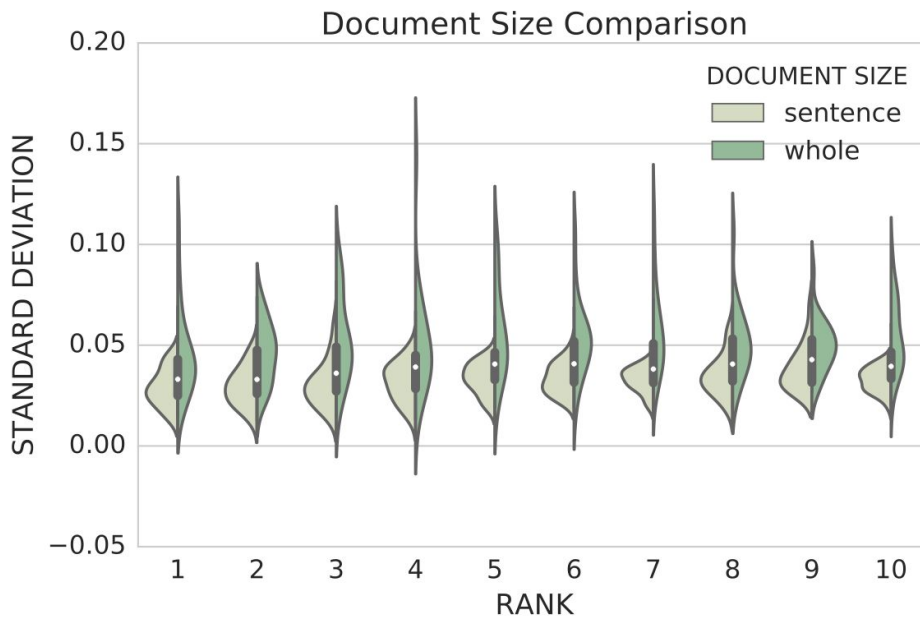
Explain “average rank” for next slide. Let’s look at all the queries instead of just one



Here we see the same results averaged across the words whose average rank falls within the top 10 words.

Even words very highly ranked in one iteration can drop out of the top-N list in the next iteration

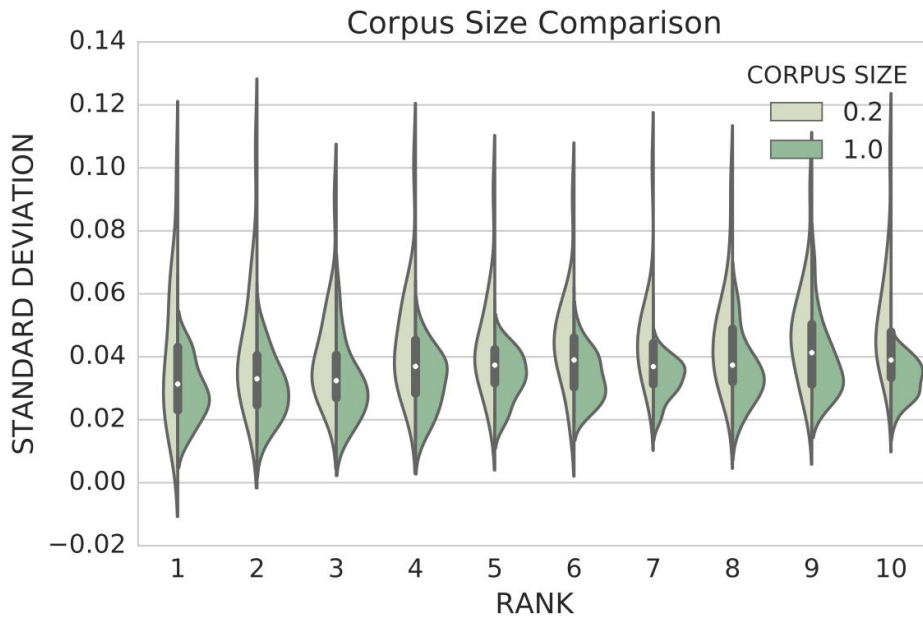
Review this and have good explanation (write down exactly what to say)
 Up to now, each row has represented a specific word, now we're zooming out to ranks



Start concluding...

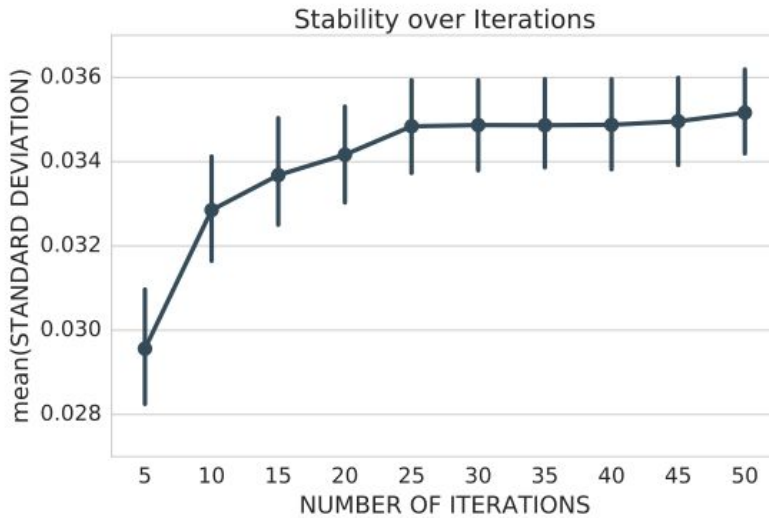
"If you're bootstrapping smaller segments, you'll get smaller variation"

Finally, we can look at two more corpus parameters: document and corpus size. Here we see that using a larger document size (the whole document instead of a single sentence) for the bootstrap setting, results in a larger spread of standard deviations, indicating greater variability.



Similarly, using a smaller corpus (in this case, 20% of the corpus), also results in greater variability.

What should we do?



39

Ok we've shown that there is significant variability in the ranked lists of words. What can we do about it? When reporting results, cosine similarities should be averaged over multiple bootstrap sampled of the dataset. We found that ~20 iterations is sufficient.

Takeaways

- The corpus is itself only a *sample*
- Fine-grained distinctions between cosine similarities are not reliable
- Smaller corpora and larger documents are more susceptible to variation
- Variation can be quantified by averaging over ~ 20 bootstrap samples

Note on vocabulary size and word frequency

- We tested four parameters of the corpus (order of documents, presence of documents, size of documents, size of corpus).
- That said, our datasets varied in terms of vocabulary size, and the bootstrap setting does measure some effects of vocabulary size and distribution.
- Please see concurrent work at the poster session from University of Michigan!

Most similar words to *abortion*

Run 1	Run 2	Run 3	Run 4	Run 5	Run 6	Run 7
viability pregnancies abortion abortions fetus gestation surgery expiration sudden fetal	fetus pregnancies gestation kindergarten viability headaches pregnant abortion pain bladder	trimester surgery visit tenure workday abortions hernia summer suicide abortion	surgery visit therapy pain hospitalization neck headaches trimester experiencing medications	trimester surgery incarceration visit arrival pain headaches birthday neck tenure	pregnancies occupation viability abortion tenure visit abortions pregnant birthday fetus	abdomen tenure stepfather wife groin throat grandmother daughter panic jaw

We've been looking at a lot of plots, now let's look at some words.

Let's look beyond cosine similarities and consider the rank of the words. Do the cosine similarities vary consistently or do the words actually change rank across training iterations?

Let's look at another example. I normally try to choose innocuous examples for talks, but here I'd really like to convince you that the results of these studies can be vital.

Let's look at the closest words to "abortion" in the corpus for the 9th Circuit US Federal Court of Appeals.

Most similar words to *abortion*

Run 1	Run 2	Run 3	Run 4	Run 5	Run 6	Run 7
viability pregnancies abortion abortions fetus gestation surgery expiration sudden fetal	fetus pregnancies gestation kindergarten viability headaches pregnant abortion pain bladder	trimester surgery visit tenure workday abortions hernia summer suicide abortion	surgery visit therapy pain hospitalization neck headaches trimester experiencing medications	trimester surgery incarceration visit arrival pain headaches birthday neck tenure	pregnancies occupation viability abortion tenure visit abortions pregnant birthday fetus	abdomen tenure stepfather wife groin throat grandmother daughter panic jaw

In the first run, we see words associated with dates, viability, and surgeries.

Most similar words to *abortion*

Run 1	Run 2	Run 3	Run 4	Run 5	Run 6	Run 7
viability pregnancies abortion abortions fetus gestation surgery expiration sudden fetal	fetus pregnancies gestation kindergarten viability headaches pregnant abortion pain bladder	trimester surgery visit tenure workday abortions hernia summer suicide abortion	surgery visit therapy pain hospitalization neck headaches trimester experiencing medications	trimester surgery incarceration visit arrival pain headaches birthday neck tenure	pregnancies occupation viability abortion tenure visit abortions pregnant birthday fetus	abdomen tenure stepfather wife groin throat grandmother daughter panic jaw

In the another run, trained on a bootstrapped example of the same dataset using the same algorithm, we see almost entirely different results. Here, the closest words are related to jails, visits, and pain.

Most similar words to *abortion*

Run 1	Run 2	Run 3	Run 4	Run 5	Run 6	Run 7
viability pregnancies abortion abortions fetus gestation surgery expiration sudden fetal	fetus pregnancies gestation kindergarten viability headaches pregnant abortion pain bladder	trimester surgery visit tenure workday abortions hernia summer suicide abortion	surgery visit therapy pain hospitalization neck headaches trimester experiencing medications	trimester surgery incarceration visit arrival pain headaches birthday neck tenure	pregnancies occupation viability abortion tenure visit abortions pregnant birthday fetus	abdomen tenure stepfather wife groin throat grandmother daughter panic jaw

In another training iteration, we again find very different results which now seem related to family members and body parts.

Significantly, across these iterations, words aren't just being reordered; membership in the top-10 list changes substantially.

Algorithms

- LSA
- SGNS
- GloVe
- PPMI

Sources of variation?

- Randomized SVD solver

term-document matrix

dense, low-rank approximation

$$X \approx X_K = U_K \Sigma_K V_K^T$$

(Deerwester et al., 1990; Landauer and Dumais, 1997)

Algorithms

- LSA
- **SGNS**
- GloVe
- PPMI

Sources of variation?

- Random initialization
- Random construction of negative samples
- Stochastic gradient descent

the quick brown fox jumped over the lazy dog

([the, brown], quick), ([quick, fox], brown), ([brown, jumped], fox), ...

(quick, the), (quick, brown), (brown, quick), (brown, fox), ...

$$J_{\text{NEG}}^{(t)} = \log Q_{\theta}(D = 1 | \text{the, quick}) + \log(Q_{\theta}(D = 0 | \text{sheep, quick}))$$

<https://www.tensorflow.org/tutorials/word2vec>

(Mikolov et al., 2013)

Algorithms

- LSA
- SGNS
- GloVe
- PPMI

Sources of variation?

- Random initialization

$$\mathcal{L} = \sum_{w,c} f(x_{wc}) \vec{w} \cdot \vec{c} + b_w + b_c - \log(x_{wc})$$

word ↓ context ↓

↑ co-occurrence count of word and context

↑ bias terms


(Pennington et al., 2014)

Algorithms

- LSA
- SGNS
- GloVe
- **PPMI**

Sources of variation?

- Random subsampling of frequent tokens


$$PMI(w, c) = \log \frac{P(w, c)}{P(w)P(c)}$$

$$PPMI(w, c) = \max(PMI(w, c), 0)$$

(Levy and Goldberg, 2014)