13. Topics in Embeddings (flipped)

DS-GA 1015, Text as Data
Arthur Spirling

May 4, 2021

Housekeeping

Housekeeping

1 Homework coming in today May 4, 2021, at 11pm.

Housekeeping

1. Homework coming in today May 4, 2021, at 11pm.
2. Final papers come in on May 12. No extensions, no exceptions!

Housekeeping

1. Homework coming in today May 4, 2021, at 11pm.

2. Final papers come in on May 12. No extensions, no exceptions! Details will follow for uploading etc.

Housekeeping

1. Homework coming in today May 4, 2021, at 11pm.
2. Final papers come in on May 12. No extensions, no exceptions! Details will follow for uploading etc.
3. Final office hours tomorrow.

Housekeeping

1 Homework coming in today May 4, 2021, at 11pm.

2 Final papers come in on May 12. No extensions, no exceptions! Details will follow for uploading etc.

3 Final office hours tomorrow.

4 Evaluations

Self-Promotion

Self-Promotion

1 shameless.

Self-Promotion

1. shameless.
2. “Word Embeddings: What works, what doesn’t, and how to tell the difference for applied research”, Rodríguez and Arthur Spirling

Self-Promotion

1. shameless.
2. "Word Embeddings: What works, what doesn't, and how to tell the difference for applied research", Rodríguez and Arthur Spirling
3. "Embedding Regression: Models for Context-Specific Description and Inference", Pedro L. Rodríguez, Arthur Spirling and Brandon Stewart

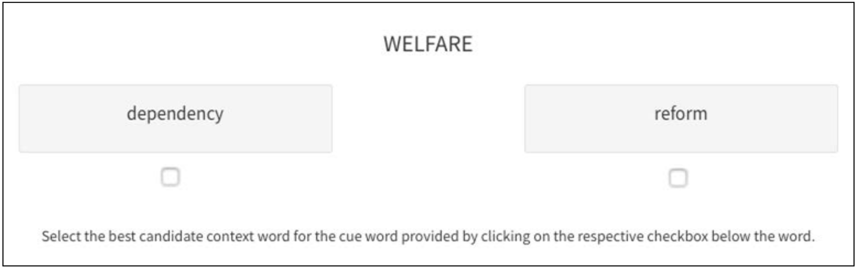
Self-Promotion

1. shameless.
2. "Word Embeddings: What works, what doesn't, and how to tell the difference for applied research", Rodríguez and Arthur Spirling
3. "Embedding Regression: Models for Context-Specific Description and Inference", Pedro L. Rodríguez, Arthur Spirling and Brandon Stewart

Big Picture(s)

Big Picture(s)

Which one of these portraits is more realistic?


WELFARE
dependency
reform
Select the best candidate context word for the cue word provided by clicking on the respective checkbox below the word.

Decisions, Decisions

Decisions, Decisions

Explosion of interest in word embeddings.

Decisions, Decisions

Explosion of interest in word embeddings. These are real valued vectors that are used for two purposes:

Decisions, Decisions

Explosion of interest in word embeddings. These are real valued vectors that are used for two purposes:

1. feature representations for downstream NLP/ML tasks.

Decisions, Decisions

Explosion of interest in word embeddings. These are real valued vectors that are used for two purposes:

1. feature representations for downstream NLP/ML tasks.

2. tools for studying word usage and meaning—"semantics".

Decisions, Decisions

Explosion of interest in word embeddings. These are real valued vectors that are used for two purposes:

1. feature representations for downstream NLP/ML tasks.
2. tools for studying word usage and meaning—"semantics".

As with all such representational strategies (topic models, TF-IDF), there are (literally thousands) of modeling options available...

Decisions, Decisions

Explosion of interest in word embeddings. These are real valued vectors that are used for two purposes:

1. feature representations for downstream NLP/ML tasks.
2. tools for studying word usage and meaning—"semantics".

As with all such representational strategies (topic models, TF-IDF), there are (literally thousands) of modeling options available...

How should political scientists choose among them?

Why do we ask?

$\boxed{\textbf{How should political scientists choose among them?}}$

The Problem

The Problem

There are **no** generally accepted downstream tasks in political science:

The Problem

There are **no** generally accepted downstream tasks in political science: 'extrinsic' evaluation criteria make no sense (e.g. analogy banks, learner accuracy).

The Problem

There are **no** generally accepted downstream tasks in political science: 'extrinsic' evaluation criteria make no sense (e.g. analogy banks, learner accuracy).

So, embeddings are **only** useful to the extent they capture semantically meaningful information about politics:

The Problem

There are **no** generally accepted downstream tasks in political science: 'extrinsic' evaluation criteria make no sense (e.g. analogy banks, learner accuracy).

So, embeddings are **only** useful to the extent they capture semantically meaningful information about politics: focus on 'intrinsic' evaluation criteria.

The Problem

There are **no** generally accepted downstream tasks in political science: 'extrinsic' evaluation criteria make no sense (e.g. analogy banks, learner accuracy).

So, embeddings are **only** useful to the extent they capture semantically meaningful information about politics: focus on 'intrinsic' evaluation criteria.

But how can we evaluate this?

The Solution

The Solution

We propose a "Turing test":

The Solution

We propose a “Turing test”: ask crowdworkers whether output from humans or machine (model) fits a cue better.

The Solution

We propose a "Turing test": ask crowdworkers whether output from humans or machine (model) fits a cue better.

We get remarkable, human-like performance from embeddings models in terms of meaning. ✓

Wait, there's more...

Wait, there's more...

Embeddings models have multiple parameters,

Wait, there's more...

Embeddings models have multiple parameters, especially window-size and embedding dimensions.

Wait, there's more. . .

Embeddings models have multiple parameters, especially window-size and embedding dimensions.

And should you use pretrained or locally fit?

Wait, there's more...

Embeddings models have multiple parameters, especially window-size and embedding dimensions.

And should you use pretrained or locally fit?

We use our technical critera on fit and stability and our Turing test to provide advice.

Wait, there's more. . .

Embeddings models have multiple parameters, especially window-size and embedding dimensions.

And should you use pretrained or locally fit?

We use our technical critera on fit and stability and our Turing test to provide advice.

Avoid small windows, few dimensions but otherwise results are robust to these parameter choices. ✓

Wait, there's more...

Embeddings models have multiple parameters, especially window-size and embedding dimensions.

And should you use pretrained or locally fit?

We use our technical critera on fit and stability and our Turing test to provide advice.

Avoid small windows, few dimensions but otherwise results are robust to these parameter choices. ✓

Pretrained embeddings work about as well as anything else. ✓

Finally . . .

Finally ...

Should you use `Word2Vec` (skip-gram) or `GloVe`?

Finally . . .

Should you use Word2Vec (skip-gram) or GloVe?

We apply our technical criteria to both architectures.

Finally . . .

Should you use Word2Vec (skip-gram) or GloVe?

We apply our technical criteria to both architectures.

GloVe appears to be more robust than Word2vec, but both are equally liked by humans. ✓

Implementing Choices

Implementing Choices

We train a series of (local) models.

Implementing Choices

We train a series of (local) models.

We focus on two hyperparameter choices (25 combinations):

Implementing Choices

We train a series of (local) models.

We focus on two hyperparameter choices (25 combinations):

window-size —1, 6, 12, 24 and 48

Implementing Choices

We train a series of (local) models.

We focus on two hyperparameter choices (25 combinations):

window-size —1, 6, 12, 24 and 48

embedding dimension —50, 100, 200, 300 and 450

Implementing Choices

We train a series of (local) models.

We focus on two hyperparameter choices (25 combinations):

- window-size —1, 6, 12, 24 and 48
- embedding dimension —50, 100, 200, 300 and 450

For each pair we estimate 10 sets of embeddings (250 in all).

Implementing Choices

We train a series of (local) models.

We focus on two hyperparameter choices (25 combinations):

window-size —1, 6, 12, 24 and 48

embedding dimension —50, 100, 200, 300 and 450

For each pair we estimate 10 sets of embeddings (250 in all).

Also compare to `GloVe` and `Word2Vec` (skip-gram) pretrained.

Evaluation Criteria

technical criteria: model loss and computation time.

model variance (stability): within-model Pearson correlation of nearest neighbor rankings across multiple initializations.

query search ranking correlation: Pearson and rank correlations of cosine similarities.

human preference: a “Turing test” assessment and rank deviations from human generated lists.

Corpora

Corpus	Period	Docs.	Tokens	Tok/Doc.	Vocab
Congressional Record	1991-2011	1.4M	3.4×10^8	238	92k
Hansard	1935-2013	4.3M	7.2×10^8	162	79k
SOTU	1790-2018	239	2.0×10^6	8143	11k
Spanish Parl	1993-2018	1.3M	3.0×10^8	224	95k
Bundestag	1998-2018	1.2M	0.8×10^8	69	108k

Corpora

Corpus	Period	Docs.	Tokens	Tok/Doc.	Vocab
Congressional Record	1991-2011	1.4M	3.4×10^8	238	92k
Hansard	1935-2013	4.3M	7.2×10^8	162	79k
SOTU	1790-2018	239	2.0×10^6	8143	11k
Spanish Parl	1993-2018	1.3M	3.0×10^8	224	95k
Bundestag	1998-2018	1.2M	0.8×10^8	69	108k

Here we will focus on: Congressional Record and `GloVe`.

Evaluation Criteria

technical criteria: model loss and computation time.

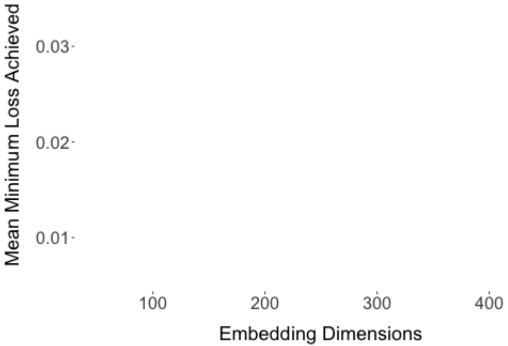
model variance (stability): within-model Pearson correlation of nearest neighbor rankings across multiple initializations.

query search correlations: Pearson and rank correlations of cosine similarities.

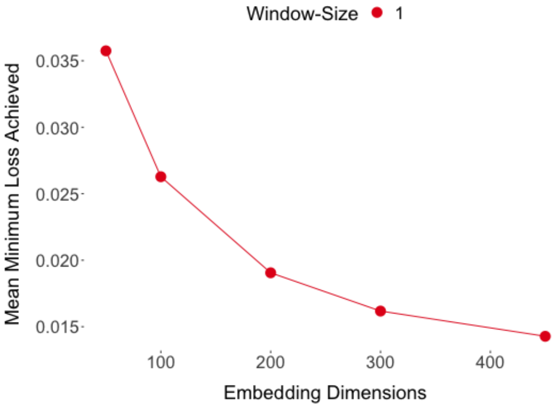
human preference: a "Turing test" assessment and rank deviations from human generated lists.

Technical Criteria: Model Loss

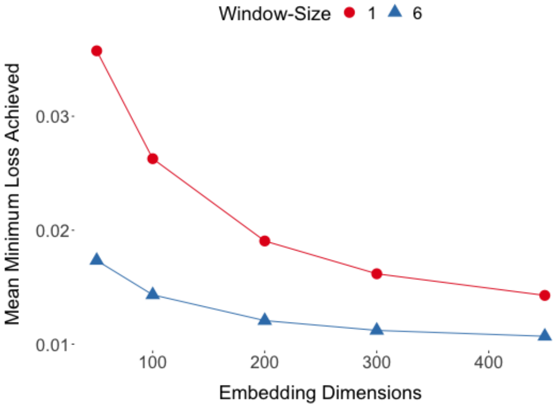
Technical Criteria: Model Loss



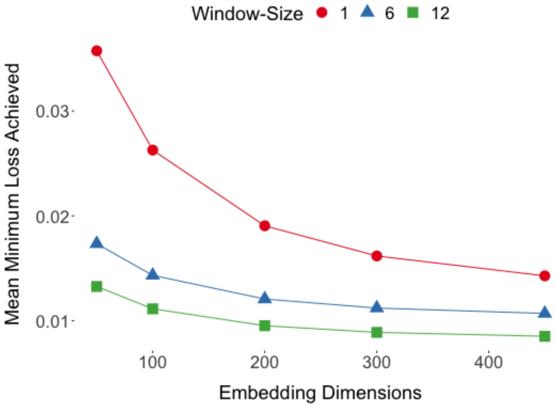
Technical Criteria: Model Loss



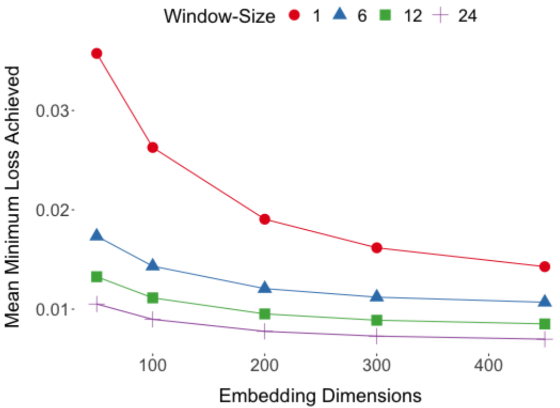
Technical Criteria: Model Loss



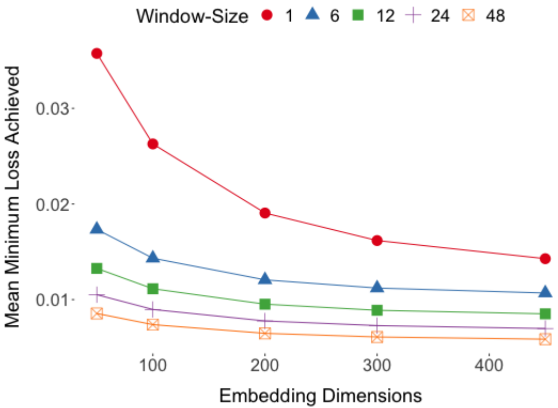
Technical Criteria: Model Loss



Technical Criteria: Model Loss

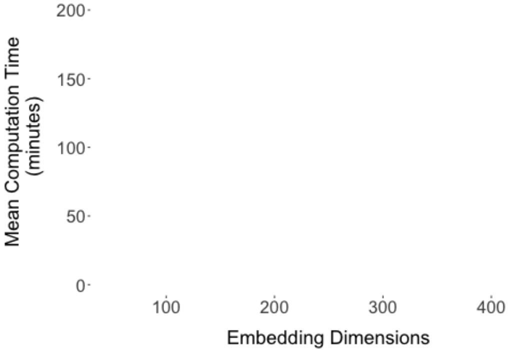


Technical Criteria: Model Loss

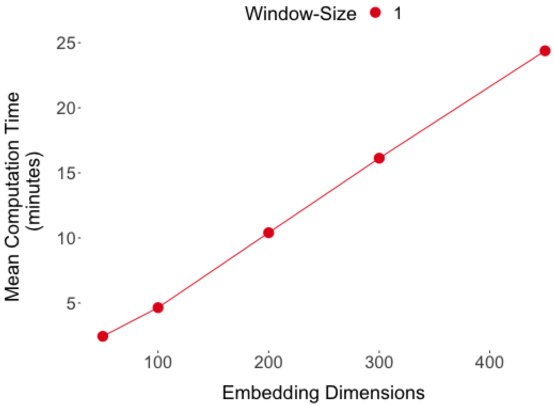


Technical Criteria: Computation Time

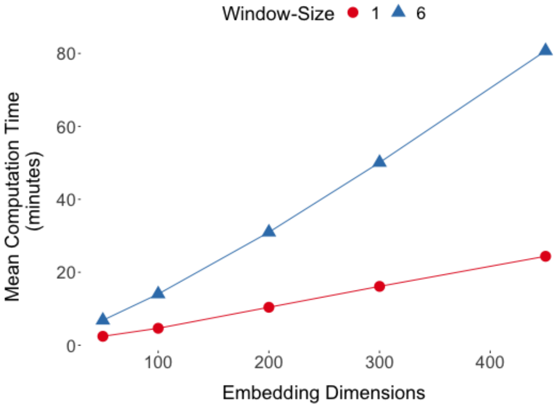
Technical Criteria: Computation Time



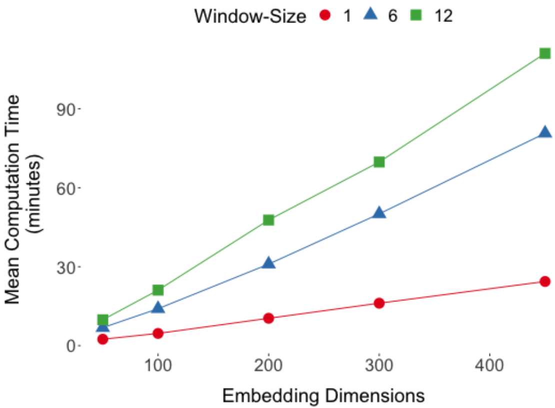
Technical Criteria: Computation Time



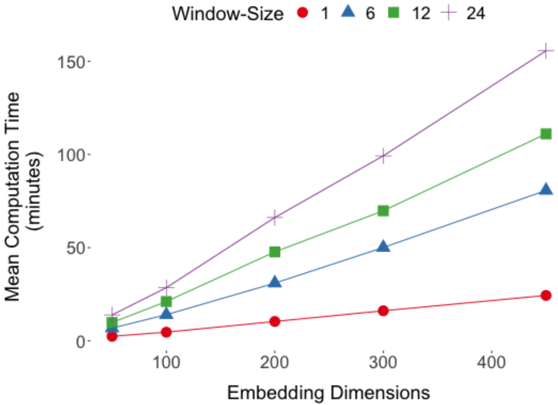
Technical Criteria: Computation Time



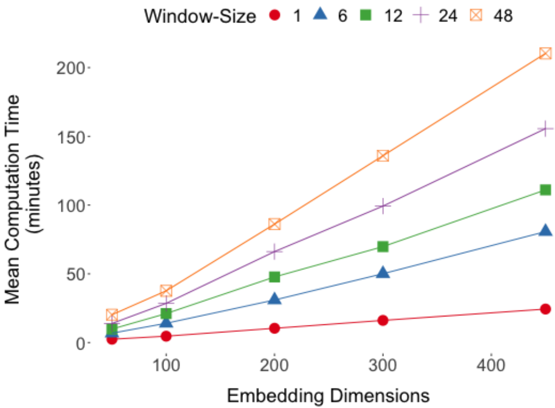
Technical Criteria: Computation Time



Technical Criteria: Computation Time



Technical Criteria: Computation Time



Evaluation Criteria

technical criteria: model loss and computation time.

model variance (stability): within-model Pearson correlation of nearest neighbor rankings across multiple initializations.

query search correlations: Pearson and rank correlations of cosine similarities.

human preference: a "Turing test" assessment and rank deviations from human generated lists.

How does one compare two models?

How does one compare two models?

Say we estimate an embeddings model with parameters 1-50:

How does one compare two models?

Say we estimate an embeddings model with parameters 1-50:

```
> local_1_50["welfare",1:5]
[1] -0.66828263 -0.02155782 -0.30138868 -0.32067144 -0.38522419
```

How does one compare two models?

Say we estimate an embeddings model with parameters 1-50:

```
> local_1_50["welfare",1:5]
[1] -0.66828263 -0.02155782 -0.30138868 -0.32067144 -0.38522419
```

And we estimate another model with parameters 6-300:

How does one compare two models?

Say we estimate an embeddings model with parameters 1-50:

```
> local_1_50["welfare",1:5]
[1] -0.66828263 -0.02155782 -0.30138868 -0.32067144 -0.38522419
```

And we estimate another model with parameters 6-300:

```
> local_6_300["welfare",1:5]
[1] -0.23670918  0.31427711  0.09471779  0.00000000  0.06700983
```

How does one compare two models?

Say we estimate an embeddings model with parameters 1-50:

```
> local_1_50["welfare",1:5]
[1] -0.66828263 -0.02155782 -0.30138868 -0.32067144 -0.38522419
```

And we estimate another model with parameters 6-300:

```
> local_6_300["welfare",1:5]
[1] -0.23670918  0.31427711  0.09471779  0.00000000  0.06700983
```

We cannot compare the two directly.

How does one compare two models?

How does one compare two models?

For each model we can compute (cosine) similarity between the query and all words in the vocabulary:

How does one compare two models?

For each model we can compute (cosine) similarity between the query and all words in the vocabulary:

```
> nearest_neighbors(cue = "welfare", embeds = local_1_50, N = 5, norm = "l2")
    reform recipients   medicare  reforming   medicaid
 0.6259441  0.6169239  0.6003502  0.5817788  0.5705987
```

How does one compare two models?

For each model we can compute (cosine) similarity between the query and all words in the vocabulary:

```
> nearest_neighbors(cue = "welfare", embeds = local_1_50, N = 5, norm = "l2")
    reform recipients   medicare  reforming   medicaid
 0.6259441  0.6169239  0.6003502  0.5817788  0.5705987
```

```
> nearest_neighbors(cue = "welfare", embeds = local_6_300, N = 5, norm = "l2")
recipients      rolls     reform       afdc dependency
 0.5953968  0.5468232  0.5333782  0.5144503  0.4794730
```

How does one compare two models?

For each model we can compute (cosine) similarity between the query and all words in the vocabulary:

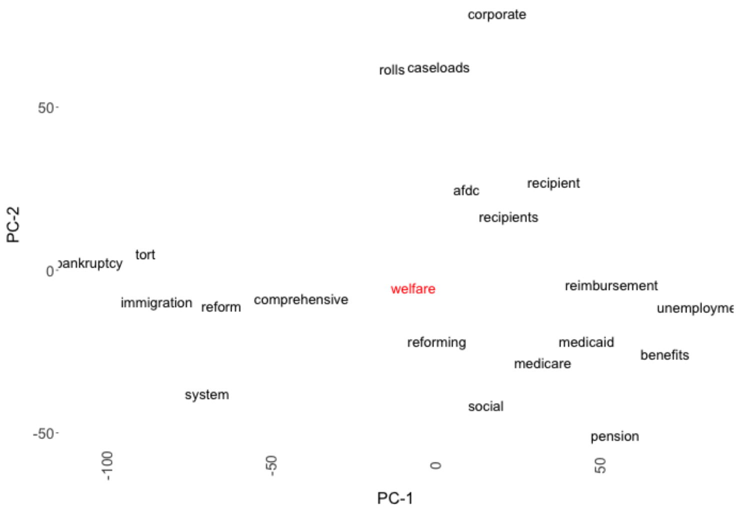
```
> nearest_neighbors(cue = "welfare", embeds = local_1_50, N = 5, norm = "l2")
    reform recipients   medicare  reforming   medicaid
 0.6259441  0.6169239  0.6003502  0.5817788  0.5705987
```

```
> nearest_neighbors(cue = "welfare", embeds = local_6_300, N = 5, norm = "l2")
recipients      rolls     reform       afdc dependency
 0.5953968  0.5468232  0.5333782  0.5144503  0.4794730
```

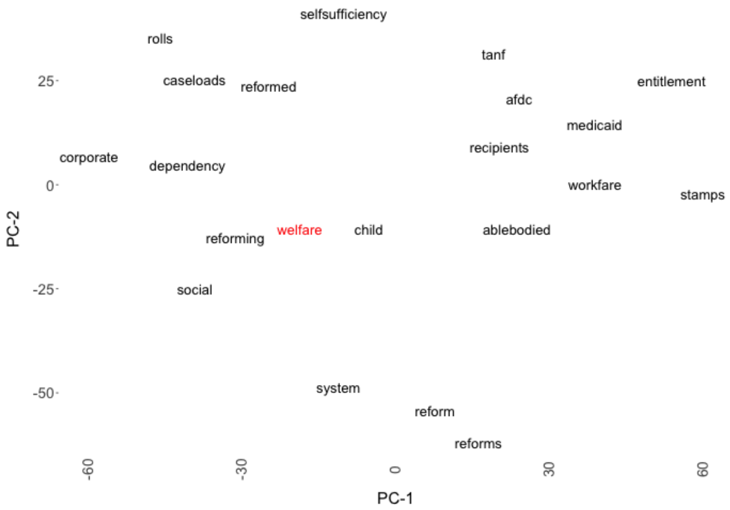
As queries we use:

- 100 randomly chosen terms from vocabulary.
- 10 curated politics terms: `democracy`, `freedom`, `equality`, `justice`, `immigration`, `abortion`, `welfare`, `taxes`, `republican`, `democrat`.

Visually (1-50)



Visually (6-300).



Evaluation Criteria

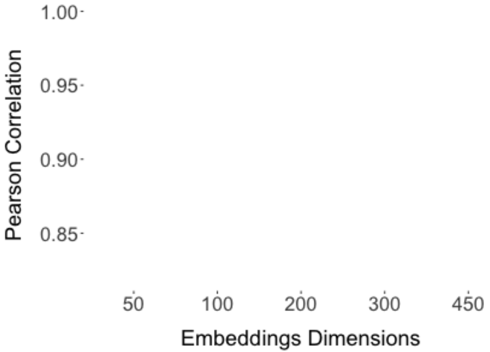
technical criteria: model loss and computation time.

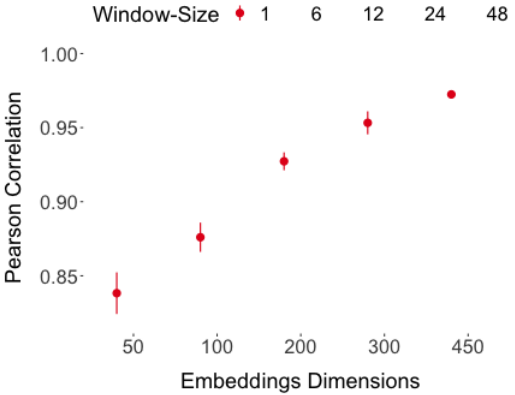
model variance (stability): within-model Pearson correlation of nearest neighbor rankings across multiple initializations.

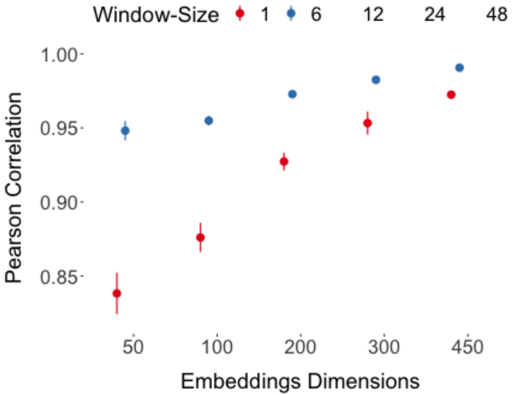
query search correlations: Pearson and rank correlations of cosine similarities.

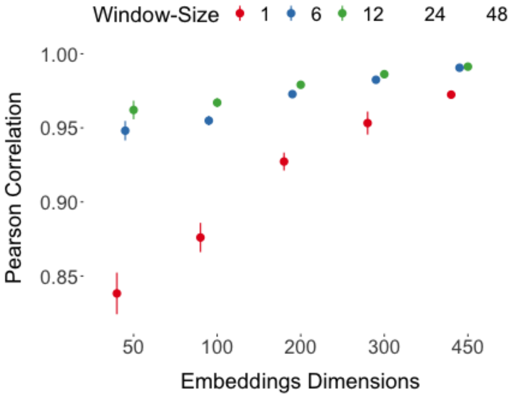
human preference: a "Turing test" assessment and rank deviations from human generated lists.

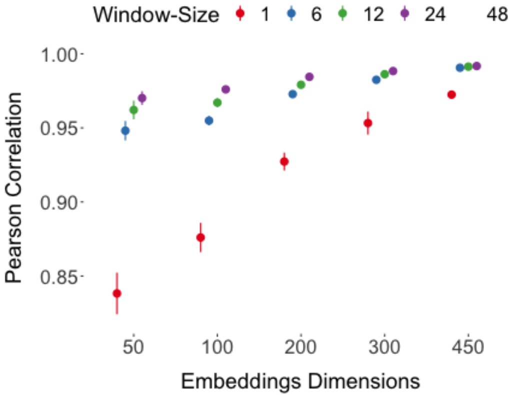

50
100
200
300
450
Embeddings Dimensions


Pearson Correlation
1.00
0.95
0.90
0.85
50
100
200
300
450
Embeddings Dimensions

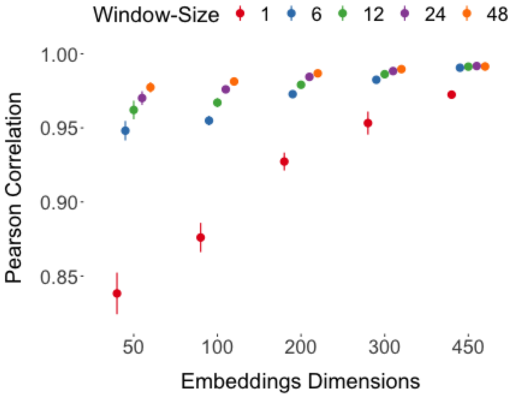

Window-Size 1 6 12 24 48
1.00
0.95
0.90
0.85
Pearson Correlation
50
100
200
300
450
Embeddings Dimensions


Window-Size 1 6 12 24 48
1.00
0.95
0.90
0.85
Pearson Correlation
50
100
200
300
450
Embeddings Dimensions


Window-Size 1 6 12 24 48
Pearson Correlation
1.00
0.95
0.90
0.85
50
100
200
300
450
Embeddings Dimensions



Window-Size 1 6 12 24 48
Pearson Correlation
1.00
0.95
0.90
0.85
50
100
200
300
450
Embeddings Dimensions



Window-Size 1 6 12 24 48
1.00
0.95
0.90
0.85
Pearson Correlation
50
100
200
300
450
Embeddings Dimensions

Evaluation Criteria

technical criteria: model loss and computation time.

model variance (stability): within-model Pearson correlation of nearest neighbor rankings across multiple initializations.

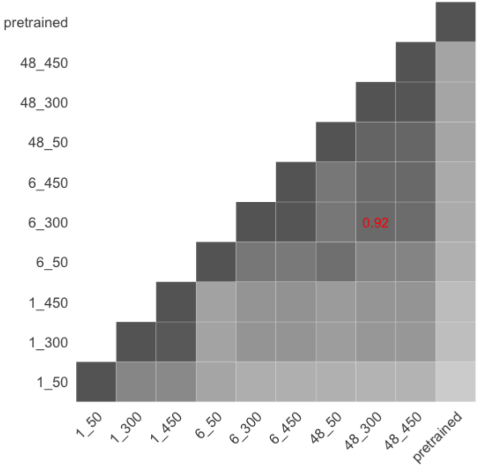
query search correlations: Pearson and rank correlations of cosine similarities.

human preference: a "Turing test" assessment and rank deviations from human generated lists.

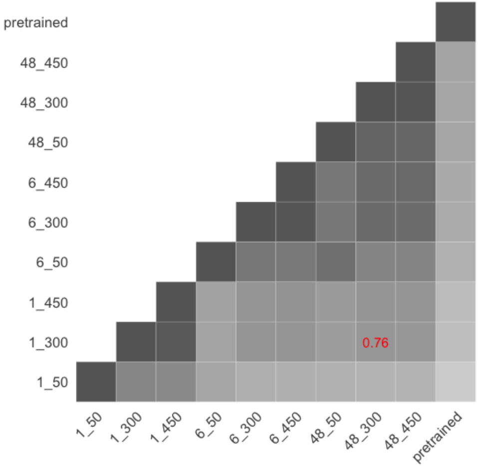
Query Search Correlations (curated queries)

	1_50	1_300	1_450	6_50	6_300	6_450	48_50	48_300	48_450	pretrained
pretrained										1
48_450									1	0.7
48_300								1	1	0.7
48_50							1	0.94	0.94	0.7
6_450						1	0.88	0.92	0.93	0.68
6_300					1	0.99	0.88	0.92	0.92	0.68
6_50				1	0.88	0.87	0.91	0.83	0.83	0.65
1_450			1	0.71	0.77	0.78	0.74	0.77	0.77	0.6
1_300		1	0.98	0.71	0.77	0.77	0.74	0.76	0.76	0.59
1_50	1	0.82	0.81	0.7	0.66	0.66	0.65	0.64	0.64	0.52

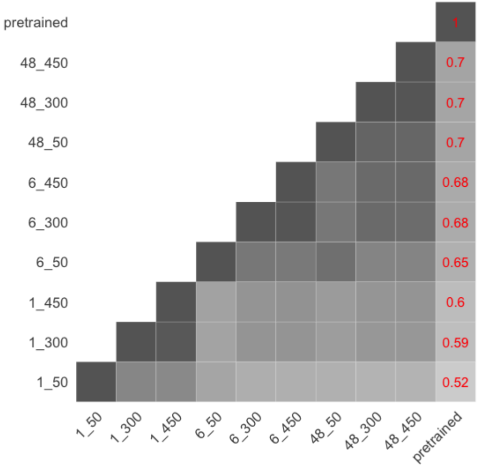
Query Search Correlations (curated queries)



Query Search Correlations (curated queries)



Query Search Correlations (curated queries)



Evaluation Criteria

technical criteria: model loss and computation time.

model variance (stability): within-model Pearson correlation of nearest neighbor rankings across multiple initializations.

query search correlations: Pearson and rank correlations of cosine similarities.

human preference: a "Turing test" assessment and rank deviations from human generated lists.

RShiny App: Definitions

Context Words

A famous maxim in the study of linguistics states that:

You shall know a word by the company it keeps. (Firth, 1957)

This task is designed to help us understand the nature of the ''company'' that words ''keep'': that is, their CONTEXT.

Specifically, for a CUE WORD, its CONTEXT WORDS include words that:

- Tend to occur in the vicinity of the CUE WORD. That is, they are words that appear close to the CUE WORD in written or spoken language.

AND/OR

- Tend to occur in similar situations to the CUE WORD in spoken and written language. That is, they are words that regularly appear with other words that are closely related to the CUE WORD.

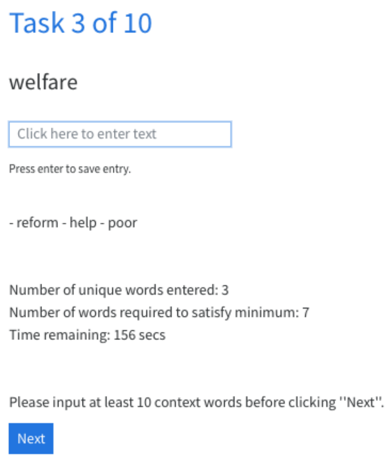
For example, CONTEXT WORDS for the cue word COFFEE include:

1. *cup* (tends to occur in the vicinity of COFFEE).
2. *tea* (tends to occur in similar situations to COFFEE, for example when discussing drinks).

Click ''Next'' to continue

Next

RShiny App-1: Context Word Generation



RShiny App-2: "Turing" Context Word Evaluation

RShiny App-2: “Turing” Context Word Evaluation

WELFARE

dependency

reform

Select the best candidate context word for the cue word provided by clicking on the respective checkbox below the word.

Click "Next" to continue

Next

RShiny App-2: “Turing” Context Word Evaluation

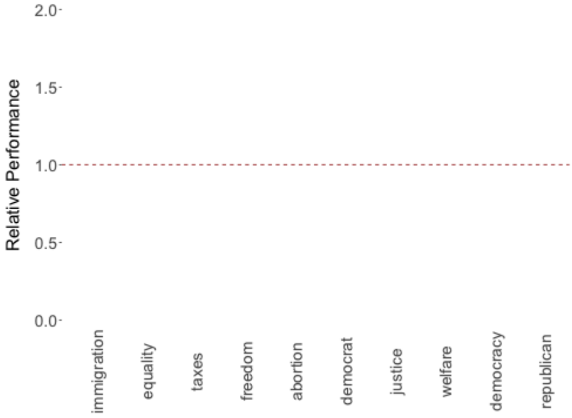
RShiny App-2: “Turing” Context Word Evaluation

Machine: 1 - Human: 0

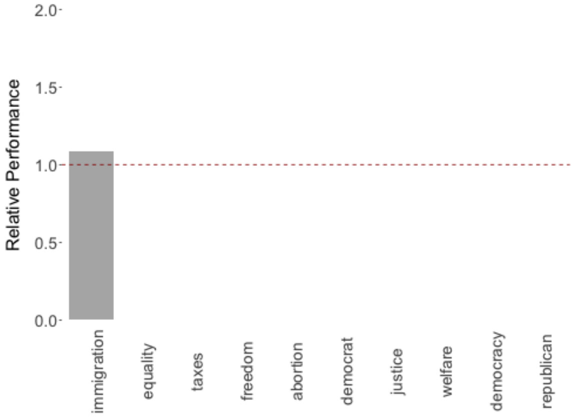
Candidate: Local 6-300
Baseline: Human

Candidate: Local 6-300
Baseline: Human

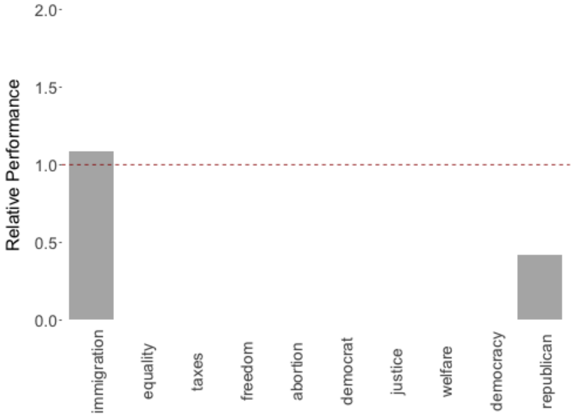
Candidate: Local 6-300
Baseline: Human



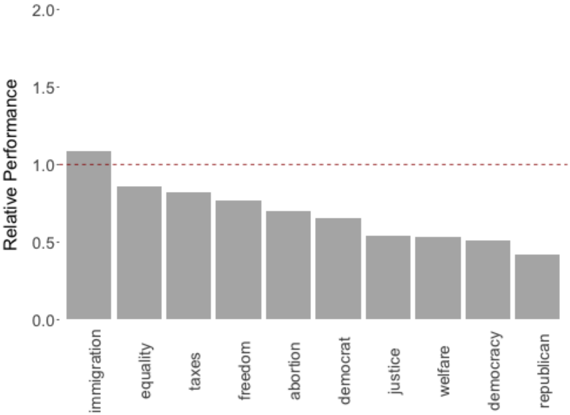
Candidate: Local 6-300
Baseline: Human



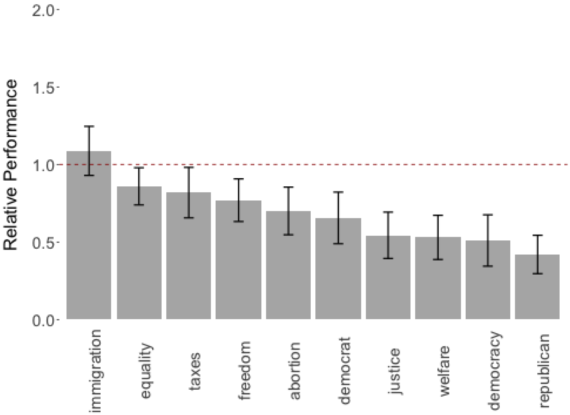
Candidate: Local 6-300
Baseline: Human



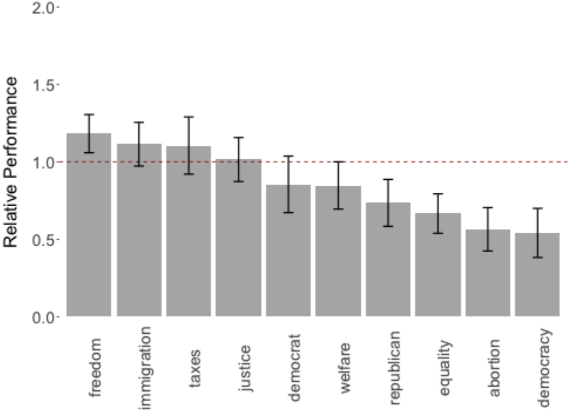
Candidate: Local 6-300
Baseline: Human



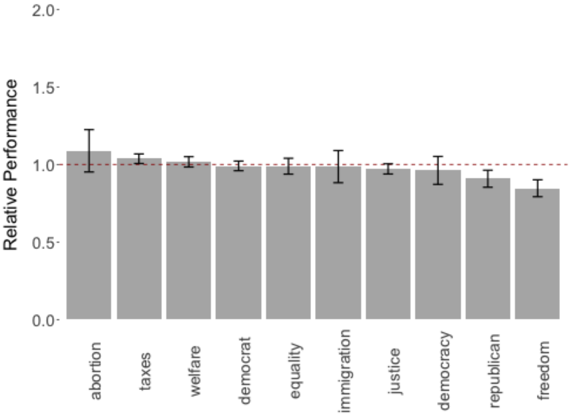
Candidate: Local 6-300
Baseline: Human



Candidate: GloVe pretrained
Baseline: Human



Candidate: Local 48-300
Baseline: Local 6-300



How about `Word2Vec`?

How about Word2Vec?

GloVe is mathematically very similar to Word2Vec's skip-gram algorithm.

How about `Word2Vec`?

`GloVe` is mathematically very similar to `Word2Vec`'s skip-gram algorithm.

Suggests algorithms will produce similar embeddings when trained on the same corpus.

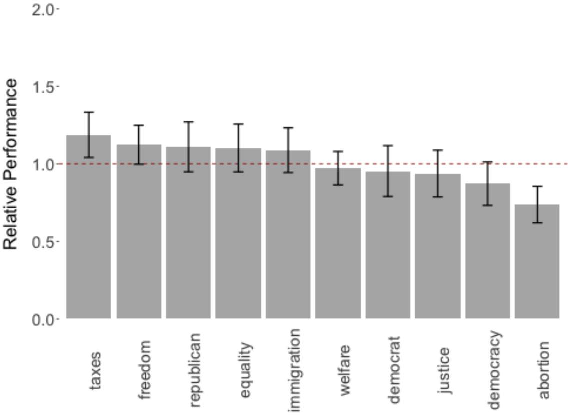
How about `Word2Vec`?

`GloVe` is mathematically very similar to `Word2Vec`'s skip-gram algorithm.

Suggests algorithms will produce similar embeddings when trained on the same corpus.

We find this **not** to be the case but humans seem to not care.

Candidate: `GloVe` pretrained
Baseline: `W2V` pretrained



Embedding Regression

Embeddings are good, actually

Embeddings are good, actually

But social scientists want more, we want the ability to:

Embeddings are good, actually

But social scientists want more, we want the ability to:

Quantify systematic differences in meaning across groups and time and get measures of the associated uncertainty.

Embeddings are good, actually

But social scientists want more, we want the ability to:

Quantify systematic differences in meaning across groups and time and get measures of the associated uncertainty.

We want embedding regression.

Embeddings are good, actually

But social scientists want more, we want the ability to:

Quantify systematic differences in meaning across groups and time and get measures of the associated uncertainty.

We want embedding regression.

Something like a multiple regression:

Embeddings are good, actually

But social scientists want more, we want the ability to:

Quantify systematic differences in meaning across groups and time and get measures of the associated uncertainty.

$\boxed{\text{We want embedding regression.}}$

Something like a multiple regression:

$$\text{Embedding}_i = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots$$

Embeddings are good, actually

But social scientists want more, we want the ability to:

Quantify systematic differences in meaning across groups and time and get measures of the associated uncertainty.

We want embedding regression.

Something like a multiple regression:

$$\text{Embedding}_i = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots$$

where X_1 and X_2 are group memberships (or something more continuous)

Problem Statement

Problem Statement

$$\text{Embedding}_i = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots$$

Problem Statement

$$\text{Embedding}_i = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots$$

will not work, simply because each Embedding_i is not a scalar:

Problem Statement

$$\text{Embedding}_i = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots$$

will not work, simply because each Embedding_i is not a scalar: it's a vector, and perhaps of dimension 100 or 300.

Problem Statement

$$\text{Embedding}_i = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots$$

will not work, simply because each Embedding_i is not a scalar: it's a vector, and perhaps of dimension 100 or 300.

So, a natural alternative is a multivariate regression, like:

Problem Statement

$$\text{Embedding}_i = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots$$

will not work, simply because each Embedding_i is not a scalar: it's a vector, and perhaps of dimension 100 or 300.

So, a natural alternative is a multivariate regression, like:

$$\underbrace{\mathbf{Y}}_{n \times D} = \underbrace{\mathbf{X}}_{n \times p+1} \underbrace{\boldsymbol{\beta}}_{p+1 \times D} + \underbrace{\mathbf{E}}_{n \times D}$$

Problem Statement

$$\text{Embedding}_i = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots$$

will not work, simply because each Embedding_i is not a scalar: it's a vector, and perhaps of dimension 100 or 300.

So, a natural alternative is a multivariate regression, like:

$$\underbrace{\mathbf{Y}}_{n \times D} = \underbrace{\mathbf{X}}_{n \times p+1} \underbrace{\boldsymbol{\beta}}_{p+1 \times D} + \underbrace{\mathbf{E}}_{n \times D}$$

where D is the dimension of the vector, and we have n such incidences of that word in our corpus (and each one is embedded).

That is

That is

$$y_i = \left[-0.62, \quad -1.46, \quad 0.58, \quad \cdots \quad -1.82, \quad 0.31, \quad -0.27\right]$$

That is

$$y_i = \begin{bmatrix} -0.62, & -1.46, & 0.58, & \cdots & -1.82, & 0.31, & -0.27 \end{bmatrix}$$

$$\boldsymbol{Y} = \underbrace{\begin{bmatrix} -0.62, & -1.46, & 0.58, & \cdots & -1.82, & 0.31, & -0.27 \\ & & & & & & \\ & & & & & & \end{bmatrix}}_{1 \times D}$$

That is

$$y_i = \begin{bmatrix} -0.62, & -1.46, & 0.58, & \cdots & -1.82, & 0.31, & -0.27 \end{bmatrix}$$

$$\boldsymbol{Y} = \underbrace{\begin{bmatrix} 0.78, & -0.26, & 0.63, & \cdots & 0.06, & 0.99, & 1.78 \\ -1.37, & 0.72, & -1.82, & \cdots & 0.51, & -0.08, & 0.28 \\ -0.62, & -1.46, & 0.58, & \cdots & -1.82, & 0.31, & -0.27 \\ \vdots, & \vdots, & \vdots, & \dots & \vdots, & \vdots, & \vdots \\ -1.17, & -0.44, & -0.43, & \cdots & 1.29, & 1.20, & 1.41 \\ 1.51, & -0.65, & -0.58, & \cdots & 0.68, & -1.47, & -0.86 \\ -2.26, & 0.57, & -1.62, & \cdots & 0.18, & -0.73, & -2.30 \end{bmatrix}}_{n \times D}$$

Problem Statement II

Problem Statement II

But that setup requires that we can embed each single instance of a word.

Problem Statement II

But that setup requires that we can embed each single instance of a word. This is generally very challenging.

Problem Statement II

But that setup requires that we can embed each single instance of a word. This is generally very challenging.

So turn to Arora et al.

Problem Statement II

But that setup requires that we can embed each single instance of a word. This is generally very challenging.

So turn to Arora et al. Key theoretical result:

Problem Statement II

But that setup requires that we can embed each single instance of a word. This is generally very challenging.

So turn to Arora et al. Key theoretical result: if you have a word, you can form an embedding for it by taking the mean of the embeddings of the words around it ($\mathbf{u}_w$).

Problem Statement II

But that setup requires that we can embed each single instance of a word. This is generally very challenging.

So turn to Arora et al. Key theoretical result: if you have a word, you can form an embedding for it by taking the mean of the embeddings of the words around it ($\mathbf{u}_w$).

e.g. *The debate lasted hours, but finally we* **[**voted on the $\boxed{\text{bill}}$ and it passed**]** *with a large majority.*

Problem Statement II

But that setup requires that we can embed each single instance of a word. This is generally very challenging.

So turn to Arora et al. Key theoretical result: if you have a word, you can form an embedding for it by taking the mean of the embeddings of the words around it ($\mathbf{u}_w$).

e.g. *The debate lasted hours, but finally we* **[**voted on the $\boxed{\text{bill}}$ and it passed**]** *with a large majority.*

taking the mean of the embeddings for `voted`, `on`, `the`, `and`, `it`, `passed` will yield an embedding for `bill`.

Actually. . .

Actually. . .

That *doesn't* quite work:

Actually...

That *doesn't* quite work: need to weight down embeddings of function words.

Actually...

That *doesn't* quite work: need to weight down embeddings of function words. So, Khodak et al (2018) suggest multiplying the mean by a transformation matrix , A:

Actually...

That *doesn't* quite work: need to weight down embeddings of function words. So, Khodak et al (2018) suggest multiplying the mean by a transformation matrix , A:

$$\hat{\mathbf{A}} = \arg\min_{\mathbf{A}} \sum_{w=1}^{W} \alpha(n_w) \| \mathbf{v}_w - \mathbf{A}\mathbf{u}_w \|_2^2$$

Actually...

That *doesn't* quite work: need to weight down embeddings of function words. So, Khodak et al (2018) suggest multiplying the mean by a transformation matrix , A:

$$\hat{\mathbf{A}} = \arg\min_{\mathbf{A}} \sum_{w=1}^{W} \alpha(n_w) \|\mathbf{v}_w - \mathbf{A}\mathbf{u}_w\|_2^2$$

Can be learned quite easily from a large corpus,

Actually...

That *doesn't* quite work: need to weight down embeddings of function words. So, Khodak et al (2018) suggest multiplying the mean by a transformation matrix , A:

$$\hat{\mathbf{A}} = \arg\min_{\mathbf{A}} \sum_{w=1}^{W} \alpha(n_w) \|\mathbf{v}_w - \mathbf{A}\mathbf{u}_w\|_2^2$$

Can be learned quite easily from a large corpus, and only needs to be estimated once.

Actually...

That *doesn't* quite work: need to weight down embeddings of function words. So, Khodak et al (2018) suggest multiplying the mean by a transformation matrix , A:

$$\hat{\mathbf{A}} = \arg\min_{\mathbf{A}} \sum_{w=1}^{W} \alpha(n_w)\|\mathbf{v}_w - \mathbf{A}\mathbf{u}_w\|_2^2$$

Can be learned quite easily from a large corpus, and only needs to be estimated once. Then each new embedding ($\mathbf{v}_w$) is obtained **a la carte**, assuming you have the context.

Who Cares?

Who Cares?

Well, if we can produce embeddings for any word,

Who Cares?

Well, if we can produce embeddings for any word, we can produce them from any instance of any word.

Who Cares?

Well, if we can produce embeddings for any word, we can produce them from any instance of any word. And that means we can use the multivariate regression set up.

Who Cares?

Well, if we can produce embeddings for any word, we can produce them from any instance of any word. And that means we can use the multivariate regression set up.

Does it work?

Who Cares?

Well, if we can produce embeddings for any word, we can produce them from any instance of any word. And that means we can use the multivariate regression set up.

Does it work?

Look at *NYT* corpus of mentions of `trump` vs `Trump`.

Who Cares?

Well, if we can produce embeddings for any word, we can produce them from any instance of any word. And that means we can use the multivariate regression set up.

Does it work?

Look at *NYT* corpus of mentions of `trump` vs `Trump`. Provide ALC embeddings for every one.

Who Cares?

Well, if we can produce embeddings for any word, we can produce them from any instance of any word. And that means we can use the multivariate regression set up.

Does it work?

Look at *NYT* corpus of mentions of `trump` vs `Trump`. Provide ALC embeddings for every one. Are they different across meanings, but similar within?

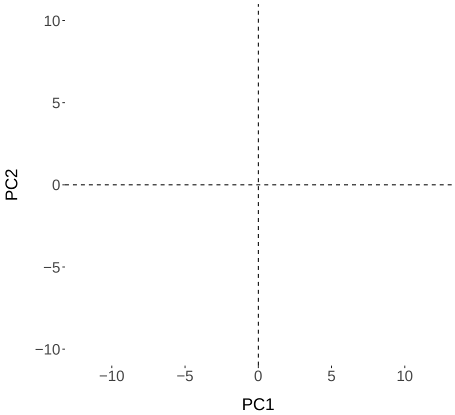
Who Cares?

Well, if we can produce embeddings for any word, we can produce them from any instance of any word. And that means we can use the multivariate regression set up.

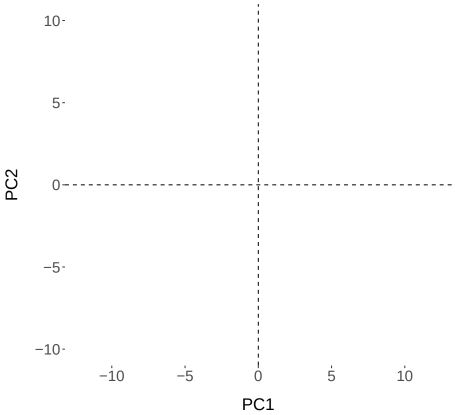
Does it work?

Look at *NYT* corpus of mentions of `trump` vs `Trump`. Provide ALC embeddings for every one. Are they different across meanings, but similar within?

Yes

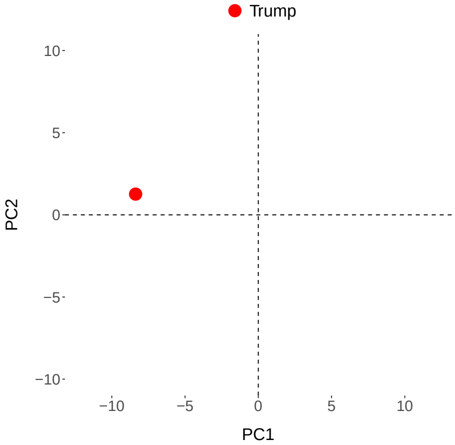


Yes



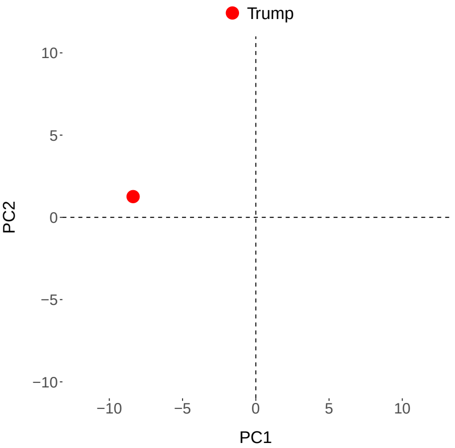
"...that painted *a* deeply unflattering picture *of* president Trump but stopped short *of* accusing him *of* criminal..."

Yes



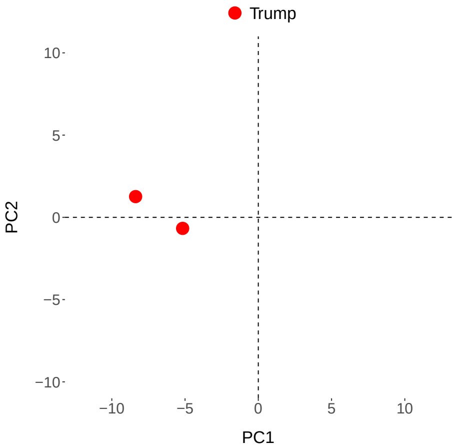
"...that painted *a* deeply unflattering picture *of* president Trump but stopped short *of* accusing him *of* criminal..."

Yes



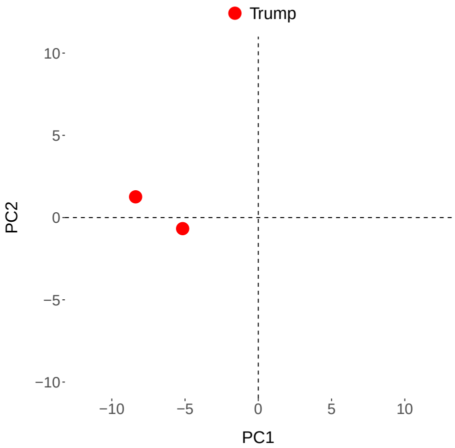
"...friday with americans focused *on* president Trump's impeachment trial, the coronavirus and the..."

Yes



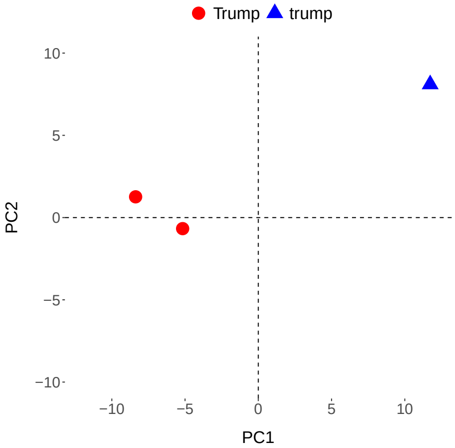
"...friday with americans focused *on* president Trump's impeachment trial, the coronavirus and the..."

Yes



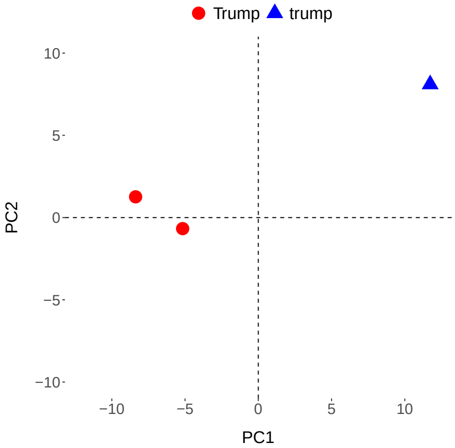
"...raise two spades superior rebid `trump` one since north has three strong spades..."

Yes



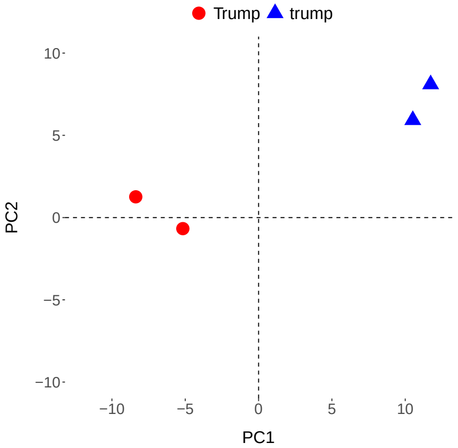
"...raise two spades superior rebid `trump` one since north has three strong spades..."

Yes



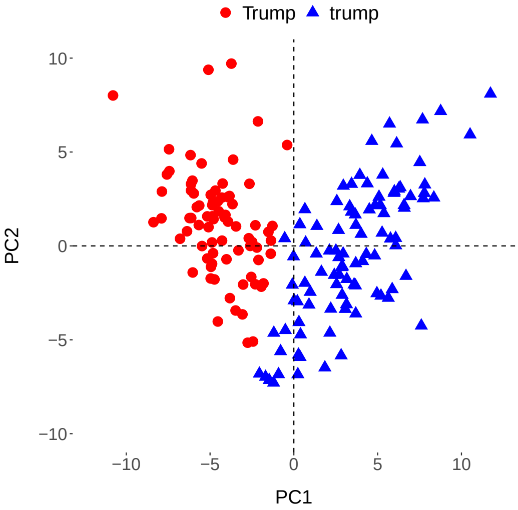
"...indicate *a* four card suit and two `trump` rebid consequently denies four *of* spades..."

Yes



"...indicate *a* four card suit and two `trump` rebid consequently denies four *of* spades..."

Yes



And they make sense.

And they make sense.

Trump		trump	
untransformed	**transformed**	**untransformed**	**transformed**
but	biden	but	declarer
that	upbraided	only	spades
even	barack	that	trumps
because	obama	one	rebid
the	presidential	they	overcall
not	rhetoric	not	bidder
would	bellicose	well	bids
what	inauguration	same	bidding
when	elect	this	anthropocentrism
this	impeachment	both	obeyed

A regression framework for ALC

A regression framework for ALC

Suppose we are comparing the usage of a word by two different groups of people, group R and group D.

A regression framework for ALC

Suppose we are comparing the usage of a word by two different groups of people, group R and group D. Our model is,

A regression framework for ALC

Suppose we are comparing the usage of a word by two different groups of people, group R and group D. Our model is,

$$Y_i = \beta_0 + \beta_1(\text{Group } R)_i + \epsilon_i$$

A regression framework for ALC

Suppose we are comparing the usage of a word by two different groups of people, group R and group D. Our model is,

$$Y_i = \beta_0 + \beta_1(\text{Group } R)_i + \epsilon_i$$

where Y_i is a $1xD$ dimensional vector representing the ALC embedding for a single instance of the word...

A regression framework for ALC

Suppose we are comparing the usage of a word by two different groups of people, group R and group D. Our model is,

$$Y_i = \beta_0 + \beta_1 (\text{Group } R)_i + \epsilon_i$$

where Y_i is a $1xD$ dimensional vector representing the ALC embedding for a single instance of the word...

...and Group R is an indicator variable capturing group membership, equal to 1 if i is a member of group R, 0 otherwise.

A regression framework for ALC

A regression framework for ALC

We then have that:

A regression framework for ALC

We then have that:

$\beta_0 = \mathbb{E}[\mathbf{A}\mathbf{u}_{w|D}] = \mathbf{A}\mathbb{E}[\mathbf{u}_{w|D}] = \mathbf{v}_{w|D} =$ ALC embedding for D.

A regression framework for ALC

We then have that:

$\beta_0 = \mathbb{E}[\mathbf{A}\mathbf{u}_{w|D}] = \mathbf{A}\mathbb{E}[\mathbf{u}_{w|D}] = \mathbf{v}_{w|D} =$ ALC embedding for D.

$\beta_0 + \beta_1 = \mathbb{E}[\mathbf{A}\mathbf{u}_{w|R}] = \mathbf{A}\mathbb{E}[\mathbf{u}_{w|R}] = \mathbf{v}_{w|R} =$ ALC embedding for R.

A regression framework for ALC

We then have that:

$\beta_0 = \mathbb{E}[\mathbf{Au}_{w|D}] = \mathbf{A}\mathbb{E}[\mathbf{u}_{w|D}] = \mathbf{v}_{w|D} =$ ALC embedding for D.

$\beta_0 + \beta_1 = \mathbb{E}[\mathbf{Au}_{w|R}] = \mathbf{A}\mathbb{E}[\mathbf{u}_{w|R}] = \mathbf{v}_{w|R} =$ ALC embedding for R.

The "magnitude" of the difference is captured by norm of β

$\rightarrow$ actual value of embedding components has no natural interpretation, so normalize.

A regression framework for ALC

We then have that:

$\beta_0 = \mathbb{E}[\mathbf{A}\mathbf{u}_{w|D}] = \mathbf{A}\mathbb{E}[\mathbf{u}_{w|D}] = \mathbf{v}_{w|D} =$ ALC embedding for D.

$\beta_0 + \beta_1 = \mathbb{E}[\mathbf{A}\mathbf{u}_{w|R}] = \mathbf{A}\mathbb{E}[\mathbf{u}_{w|R}] = \mathbf{v}_{w|R} =$ ALC embedding for R.

The "magnitude" of the difference is captured by norm of β

$\rightarrow$ actual value of embedding components has no natural interpretation, so normalize.

Can quantify uncertainty with permutation test.

Application: Partisan Differences

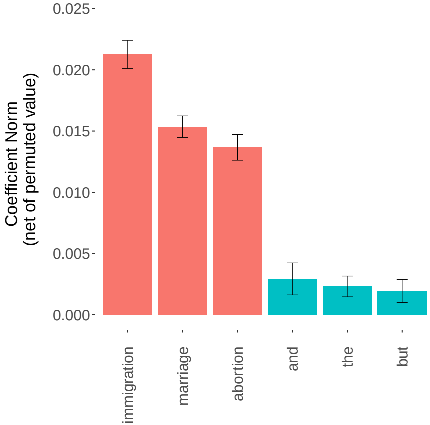


Figure: Partisan semantic differences using a corpus of the Congressional Record (Sessions 111 - 114).

Case Study 2: The Meaning of Empire

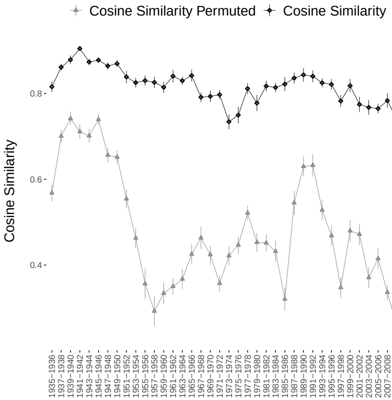


Figure: Distance between British and American understanding of `Empire`, 1935–2010. As corpora we use Congressional Record and Parliamentary Speeches (Hansard).

Application: The Meaning of Empire

US	UK
colonial	british
germany	colonial
france	australia
german	britain
italy	kingdom
british	territories
australia	overseas
french	colonies
colonies	countries
territory	whole

(a) 1945–46

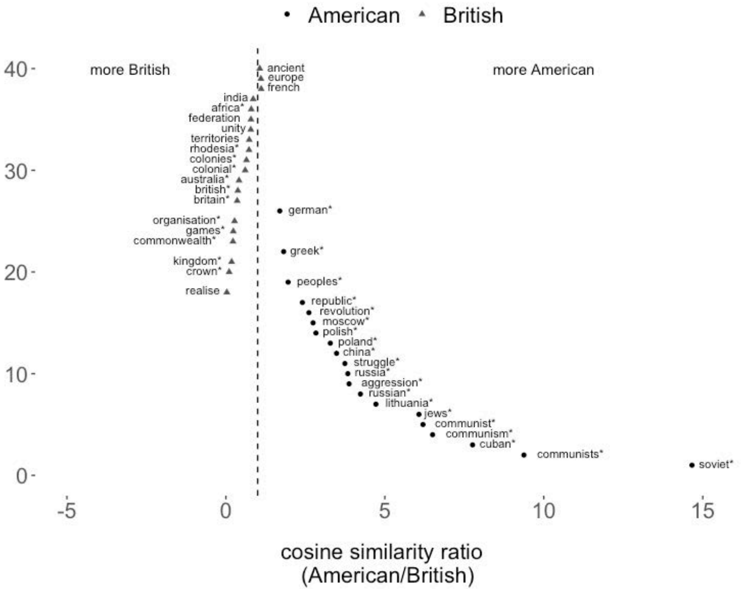
US	UK
communist	british
russian	colonial
soviet	commonwealth
lithuania	colonies
communism	territories
russia	britain
moscow	australia
republic	india
revolution	rhodesia
communists	countries

(b) 1957–58

Table: Meaning of `Empire`: post-war and 1950s

‘Most’ UK vs US NNs (1950s)

'Most' UK vs US NNs (1950s)



Software: `ConText`

Try to follow generic R `lm( )` and `glm( )` syntax in terms of operator.

Software: `ConText`

Try to follow generic R `lm( )` and `glm( )` syntax in terms of operator.

So, something like. . .

Software: ConText

Try to follow generic R `lm( )` and `glm( )` syntax in terms of operator.

So, something like. . .

```
model1 <- conText(target ~ party + gender, corpus = cr_corpus,
     bootstrap = TRUE, num_bootstraps = 20, groupby = c("party
    ", "gender"), permute = TRUE, num_permutations = 100,
    window = 6, valuetype = "fixed", case_insensitive = TRUE,
    hard_cut = FALSE, verbose = FALSE)
```

Thank You!