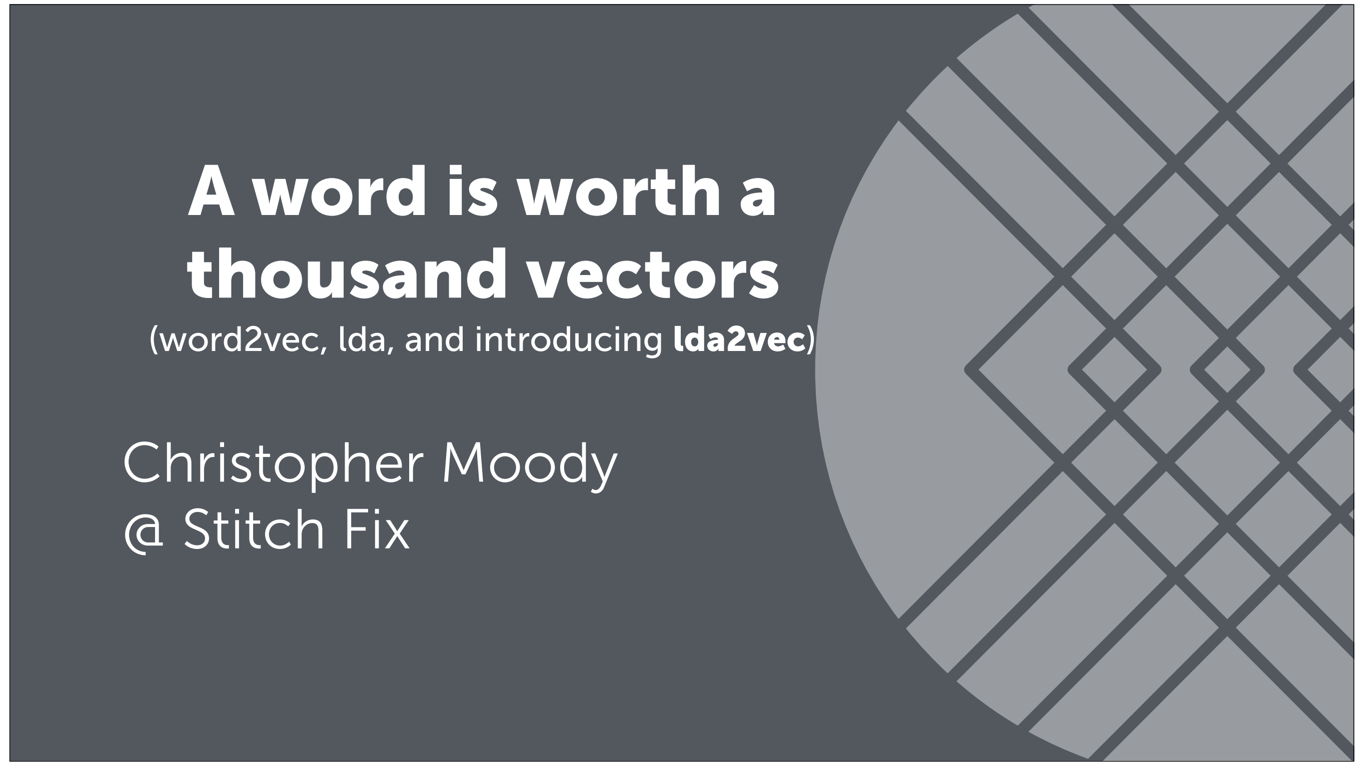


Welcome,
thanks for coming, having me, organizer

NLP can be a messy affair because you have to teach a computer about the irregularities and ambiguities of the English language in this sort of hierarchical sparse nature in all the grammar

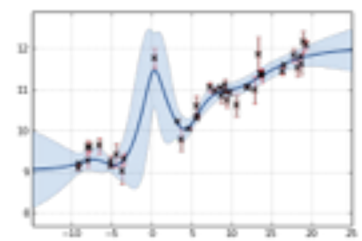
3rd trimester, pregnant
“wears scrubs” — medicine
taking a trip — a fix for vacation clothing

power of word vectors promise is to sweep away a lot of issues

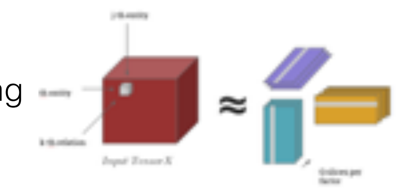
About

- @chrisemoody
- Caltech Physics
- PhD. in astrostats supercomputing
- sklearn t-SNE contributor
- Data Labs at Stitch Fix
- github.com/cemoody

Gaussian Processes



Tensor Decomposition



t-SNE

chainer
deep learning

Chainer

Credit

Large swathes of this talk are from
previous presentations by:

- Tomas Mikolov
- David Blei
- Christopher Olah
- Radim Rehurek
- Omer Levy & Yoav Goldberg
- Richard Socher
- Xin Rong
- Tim Hopper

1 word2vec

2 lda

3 lda2vec

word2vec

1. *king* - *man* + *woman* = *queen*
2. Huge splash in NLP world
3. Learns from raw text
4. Pretty simple algorithm
5. Comes pretrained

1. Learns what words *mean* — can solve analogies cleanly.
    1. Not treating words as blocks, but instead modeling relationships
2. Distributed representations form the basis of more complicated deep learning systems
3. Shallow — not deep learning!
    1. Power comes from this simplicity — super fast, lots of data
4. Get a lot of mileage out of this
    1. Don’t need to model the wikipedia corpus before starting your own

word2vec

1. Set up an objective function
2. Randomly initialize vectors
3. Do gradient descent

word2vec

word2vec: learn word vector v_{in}
from it's surrounding context

$$v_{in}$$

1. Let's talk about training first
2. In SVD and n-grams we built a co-occurence and transition probability matrices
3. Here we will learn the embedded representation directly, with no intermediates, update it w/ every example

word2vec

"The fox jumped **over** the lazy dog"

Maximize the likelihood of seeing the words given the word **over**.

$$P(the|over)$$
$$P(fox|over)$$
$$P(jumped|over)$$
$$P(the|over)$$
$$P(lazy|over)$$
$$P(dog|over)$$

...instead of maximizing the likelihood of co-occurrence counts.

1. Context — the words surrounding the training word
2. Naively assume P(*|over) is independent conditional on the training word
3. Still a pretty simple assumption!

Conditioning on just *over* no other secret parameters or anything

word2vec

What should this be?

$$P(fox|over)$$

word2vec

Should depend on the word vectors.

$$P(fox|over)$$

$$\downarrow$$

$$P(v_{fox}|v_{over})$$

Trying to learn the word vectors, so let's start with those
(we'll randomly initialize them to begin with)

word2vec

Twist: we have *two* vectors for every word.
Should depend on whether it's the input or the output.

Also a *context* window around every input word.

$$P(v_{OUT}|v_{IN})$$

"The fox jumped **over** the lazy dog"

word2vec

Twist: we have *two* vectors for every word.
Should depend on whether it's the input or the output.

Also a *context* window around every input word.

$$P(v_{OUT}|v_{IN})$$

"The fox jumped **over** the lazy dog"

v_{IN}

IN = training word

word2vec

Twist: we have *two* vectors for every word.
Should depend on whether it's the input or the output.

Also a *context* window around every input word.

$$P(v_{OUT}|v_{IN})$$

"The fox jumped **over** the lazy dog"

v_{OUT} v_{IN}

word2vec

Twist: we have *two* vectors for every word.
Should depend on whether it's the input or the output.

Also a *context* window around every input word.

$$P(v_{OUT}|v_{IN})$$

"The fox jumped **over** the lazy dog"

v_{OUT} v_{IN}

word2vec

Twist: we have *two* vectors for every word.
Should depend on whether it's the input or the output.

Also a *context* window around every input word.

$$P(v_{OUT}|v_{IN})$$

"The fox jumped **over** the lazy dog"

v_{OUT} v_{IN}

word2vec

Twist: we have *two* vectors for every word.
Should depend on whether it's the input or the output.

Also a *context* window around every input word.

$$P(v_{OUT}|v_{IN})$$

"The fox jumped **over** the lazy dog"

v_{IN} v_{OUT}

word2vec

Twist: we have *two* vectors for every word.
Should depend on whether it's the input or the output.

Also a *context* window around every input word.

$$P(v_{OUT}|v_{IN})$$

"The fox jumped **over** the lazy dog"

v_{IN} v_{OUT}

word2vec

Twist: we have *two* vectors for every word.
Should depend on whether it's the input or the output.

Also a *context* window around every input word.

$$P(v_{OUT}|v_{IN})$$

"The fox jumped **over** the lazy dog"

v_{IN} v_{OUT}

word2vec

Twist: we have *two* vectors for every word.
Should depend on whether it's the input or the output.

Also a *context* window around every input word.

$$P(v_{OUT}|v_{IN})$$

"The fox jumped over **the** lazy dog"

$\uparrow$

v_{IN}

word2vec

Twist: we have *two* vectors for every word.
Should depend on whether it's the input or the output.

Also a *context* window around every input word.

$$P(v_{OUT}|v_{IN})$$

"The fox jumped over **the** lazy dog"

v_{OUT} v_{IN}

…So that at a high level is what we want word2vec to do.

word2vec

Twist: we have *two* vectors for every word.
Should depend on whether it's the input or the output.

Also a *context* window around every input word.

$$P(v_{OUT}|v_{IN})$$

"The fox jumped over **the** lazy dog"

v_{OUT} v_{IN}

…So that at a high level is what we want word2vec to do.

word2vec

Twist: we have *two* vectors for every word.
Should depend on whether it's the input or the output.

Also a *context* window around every input word.

$$P(v_{OUT}|v_{IN})$$

"The fox jumped over **the** lazy dog"

v_{OUT} v_{IN}

…So that at a high level is what we want word2vec to do.

word2vec

Twist: we have *two* vectors for every word.
Should depend on whether it's the input or the output.

Also a *context* window around every input word.

$$P(v_{OUT}|v_{IN})$$

"The fox jumped over **the** lazy dog"

v_{OUT} v_{IN}

…So that at a high level is what we want word2vec to do.

word2vec

Twist: we have *two* vectors for every word.
Should depend on whether it's the input or the output.

Also a *context* window around every input word.

$$P(v_{OUT}|v_{IN})$$

"The fox jumped over **the** lazy dog"

v_{IN} v_{OUT}

…So that at a high level is what we want word2vec to do.

word2vec

Twist: we have *two* vectors for every word.
Should depend on whether it's the input or the output.

Also a *context* window around every input word.

$$P(v_{OUT}|v_{IN})$$

"The fox jumped over **the** lazy dog"

v_{IN} v_{OUT}

…So that at a high level is what we want word2vec to do.

two for loops

That's it! A bit disengious to make this a giant network

objective

How should we define $P(v_{OUT}|v_{IN})$?

Measure loss between
v_{IN} and v_{OUT}?

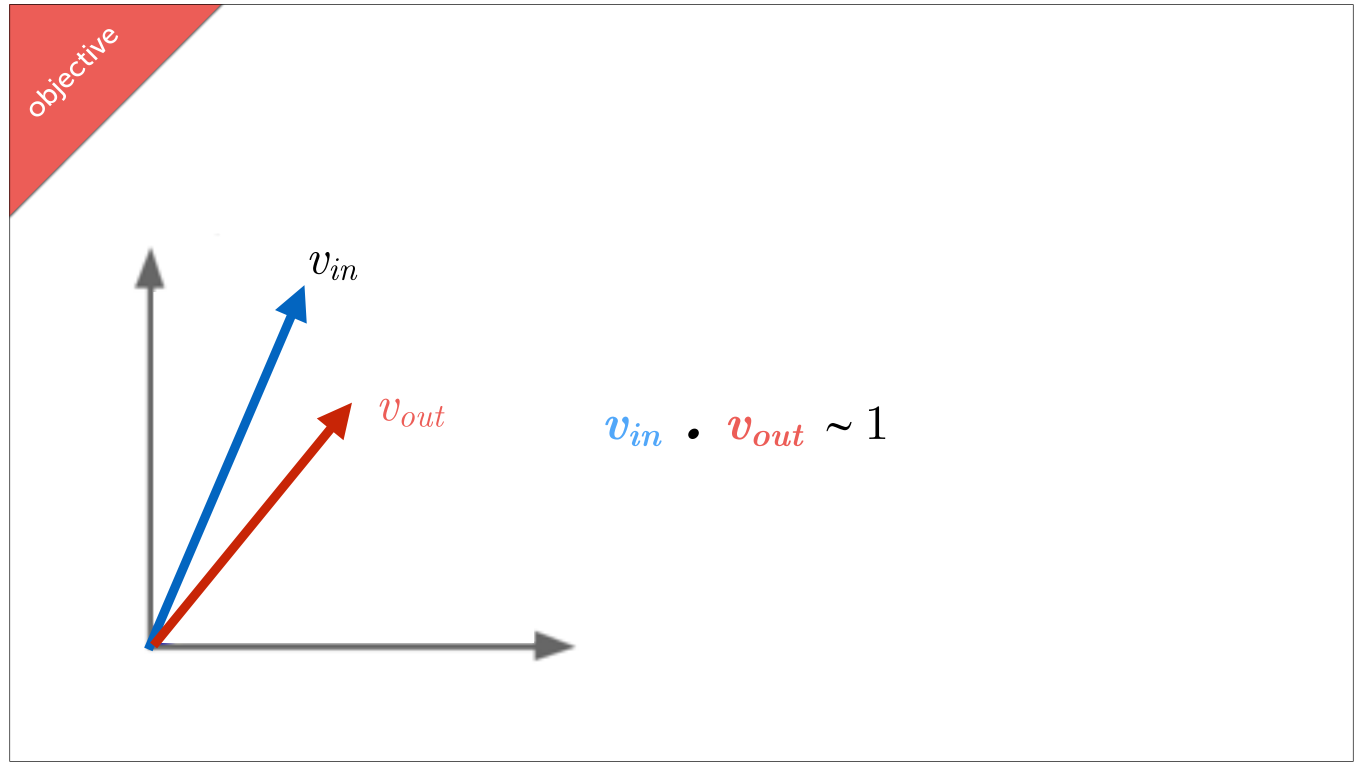
$$v_{in} \bullet v_{out}$$

Now we've defined the high-level update path for the algorithm.

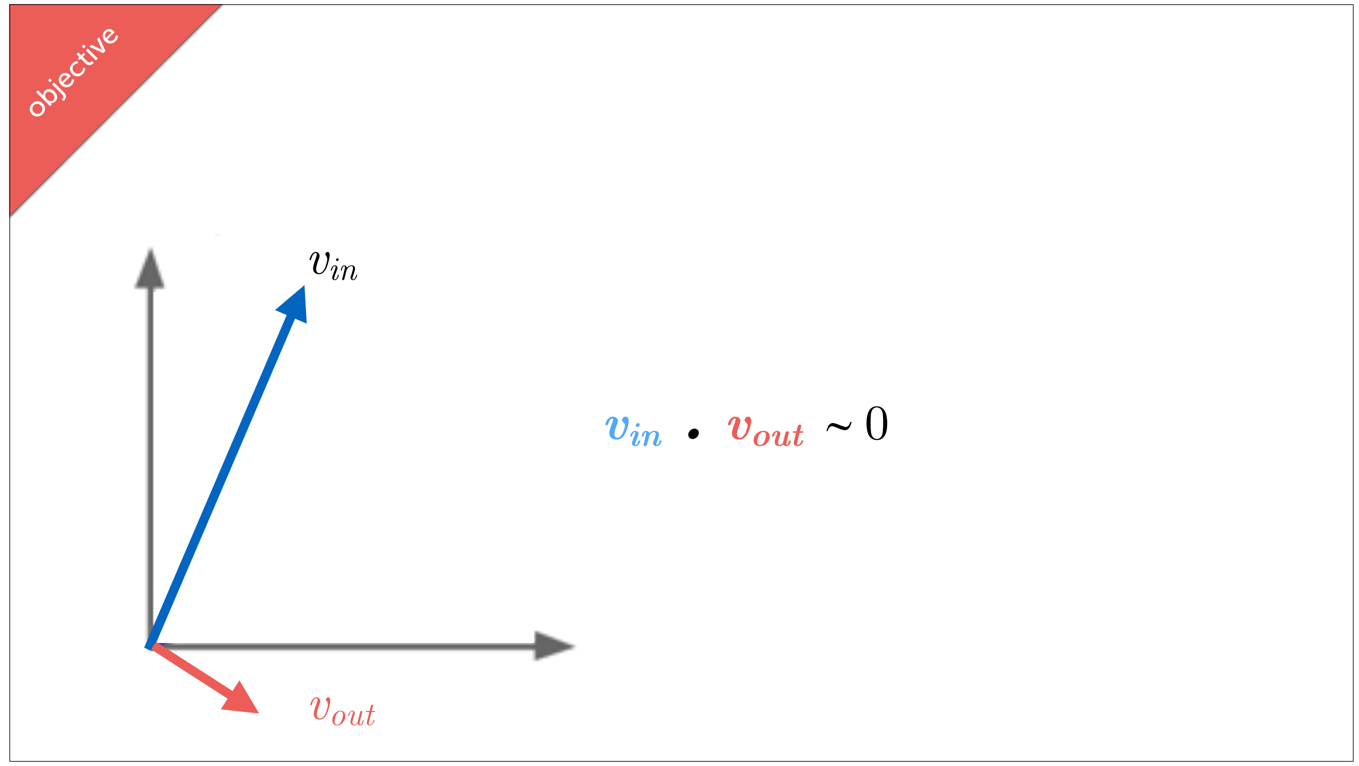
Need to define this prob exactly in order to define our updates.

Boils down to diff between in & out — want to make as similar as possible, and then the probability will go up.

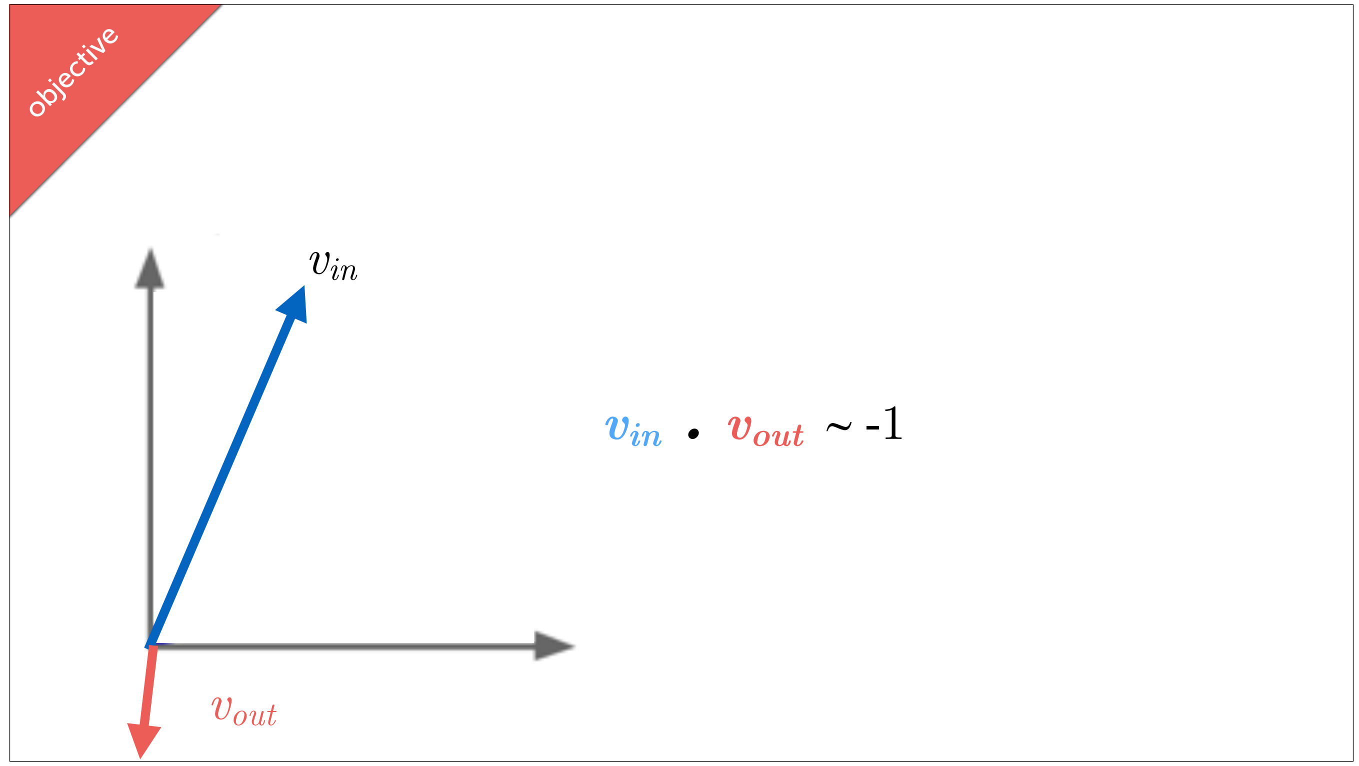
Use cosine sim.



Dot product has these properties:
Similar vectors have similarly near 1



Orthogonal vectors have similarity near 0



Orthogonal vectors have similarity near -1

objective

$$v_{in} \bullet v_{out} \in [-1,1]$$

But the inner product ranges from -1 to 1 (when normalized)
…and we'd like a probability

objective

But we'd like to measure a probability.

$$v_{in} \bullet v_{out} \in [-1,1]$$

But the inner product ranges from -1 to 1 (when normalized)
…and we'd like a probability

objective

But we'd like to measure a probability.

$$softmax(v_{in} \bullet v_{out}) \in [0,1]$$

Transform again using softmax

objective

But we'd like to measure a probability.

$$softmax(v_{in} \bullet v_{out})$$

Probability of choosing 1 of N discrete items.
Mapping from vector space to a multinomial over words.

Similar to logistic function for binary outcomes, but instead for 1 of N outcomes.

So now we're modeling the probability of a word showing up as the combination of the training word vector and the target word vector and transforming it to a 1 of N

objective

But we'd like to measure a probability.

$$softmax \sim \quad exp(v_{in} \bullet v_{out}) \in [0,1]$$

So here's the actual form of the equation — we normalize by the sum of all of the other possible pairs of word combinations

objective

But we'd like to measure a probability.

$$softmax = \frac{exp(v_{in} \cdot v_{out})}{\sum_{k \in V} exp(v_{in} \cdot v_k)}$$

Normalization term over all words

So here's the actual form of the equation — we normalize by the sum of all of the other possible pairs of word combinations

two effects
make vin and vout more similar
make vin and every other word less similar

objective

But we'd like to measure a probability.

$$softmax = \frac{exp(v_{in} \bullet v_{out})}{\sum_{k \in V} exp(v_{in} \bullet v_k)} = P(v_{out}|v_{in})$$

This is the kernel of the word2vec. We're just going to apply this operation every time we want to update the vectors.

For every word, we're going to have a context window, and then for every pair of words in that window and the input word, we'll measure this probability.

objective

Learn by gradient descent on the softmax prob.

For every example we see update v_{in}

$$v_{in} := v_{in} + \frac{\partial}{\partial v_{in}} P(v_{out}|v_{in})$$

$$v_{out} := v_{out} + \frac{\partial}{\partial v_{out}} P(v_{out}|v_{in})$$

…I won't go through the derivation of the gradient, but this is the general idea

relatively simple, fast — fast enough to read billions of words in a day

Model (training time)	Redmond	Havel	ninjutsu
Collobert (50d) (2 months)	conyers lubbock keene	plauen dzerzhinsky osterreich	reiki kohona karate
Turian (200d) (few weeks)	McCarthy Alston Cousins	Jewell Arzu Ovitz	- - -
Mnih (100d) (7 days)	Podhurst Harlang Agarwal	Pontiff Pinochet Rodionov	- - -
Skip-Phrase (1000d, 1 day)	Redmond Wash. Redmond Washington Microsoft	Vaclav Havel president Vaclav Havel Velvet Revolution	ninja martial arts swordsmanship

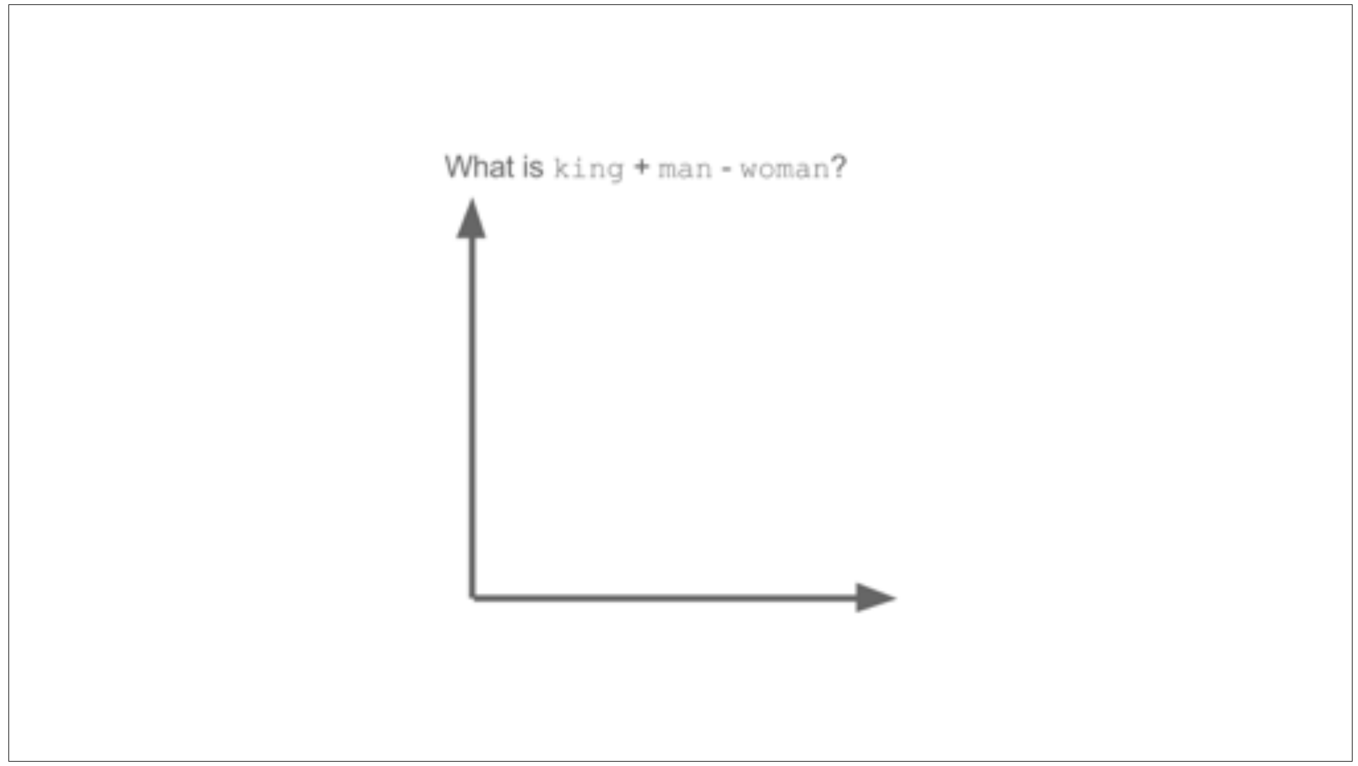
word2vec

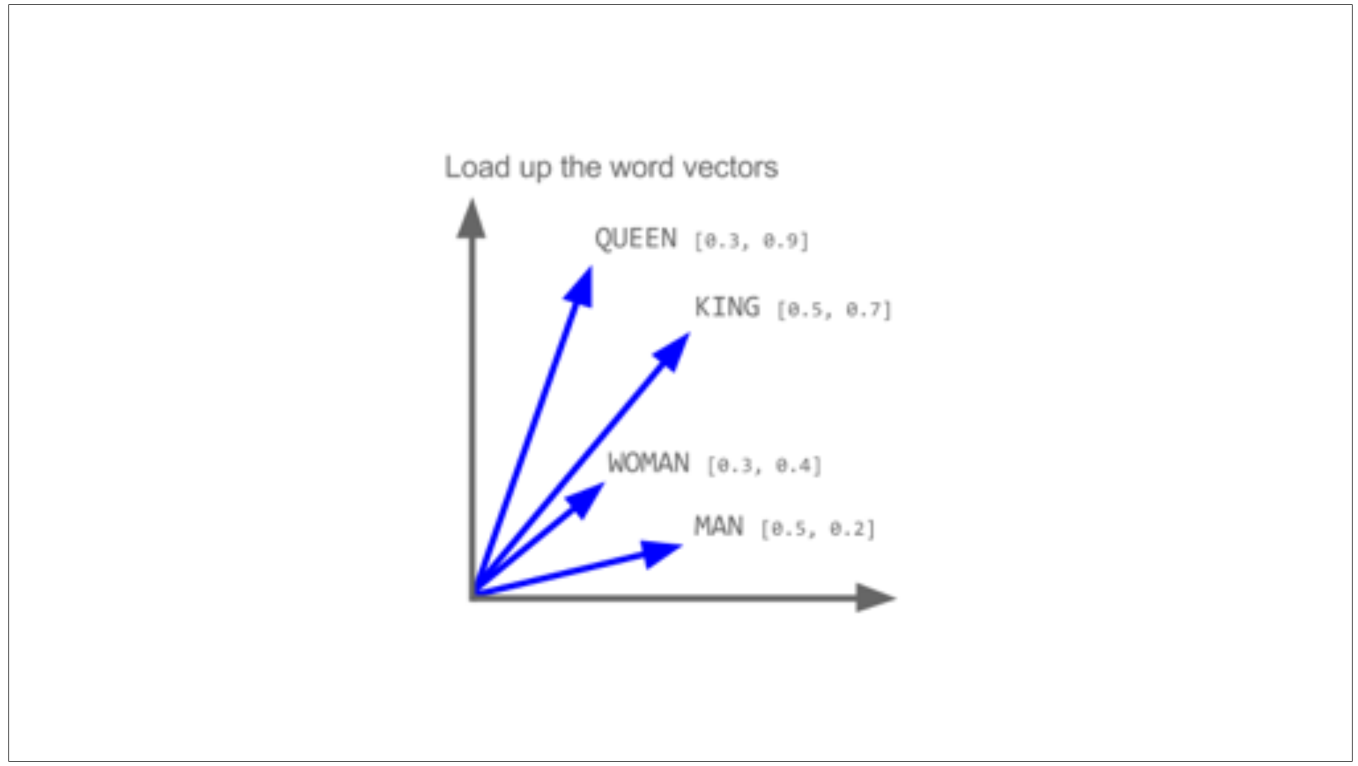
explain table

Model	Vector Dimensionality	Training words	Accuracy [%]		
			Semantic	Syntactic	Total
Collobert-Weston NNLM	50	660M	9.3	12.3	11.0
Turian NNLM	50	37M	1.4	2.6	2.1
Turian NNLM	200	37M	1.4	2.2	1.8
Mnih NNLM	50	37M	1.8	9.1	5.8
Mnih NNLM	100	37M	3.3	13.2	8.8
Mikolov RNNLM	80	320M	4.9	18.4	12.7
Mikolov RNNLM	640	320M	8.6	36.5	24.6
Huang NNLM	50	990M	13.3	11.6	12.3
CBOW	300	783M	15.5	53.1	36.1
Skip-gram	300	783M	**50.0**	55.9	**53.3**

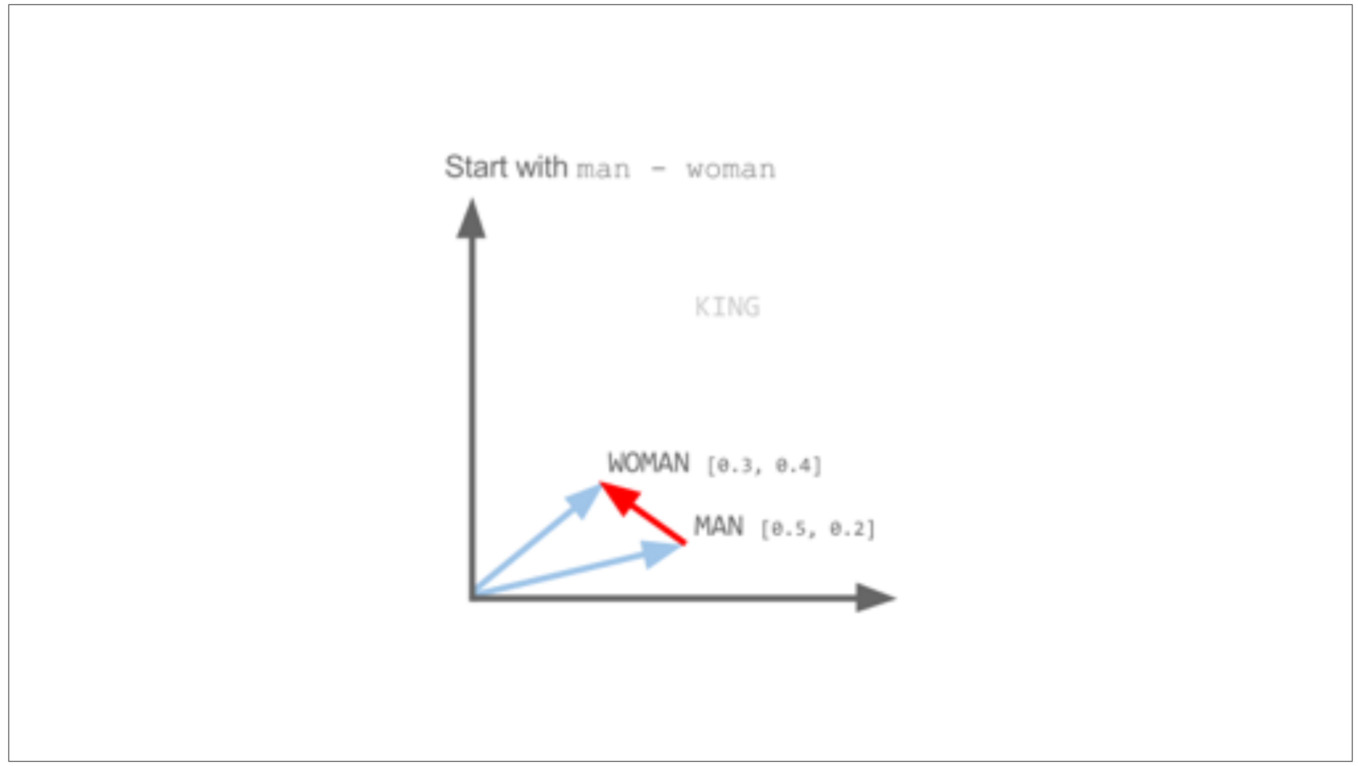
word2vec

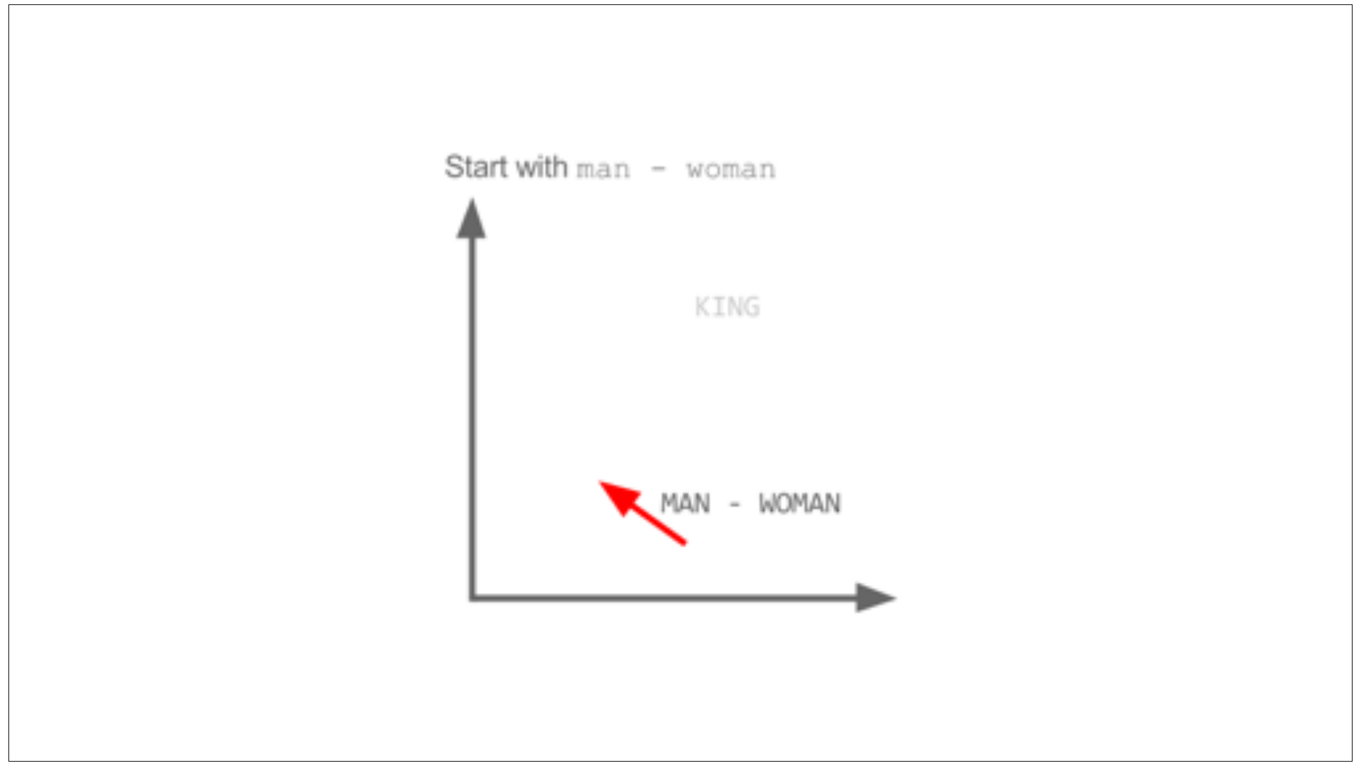
if not convinced by qualitative results….

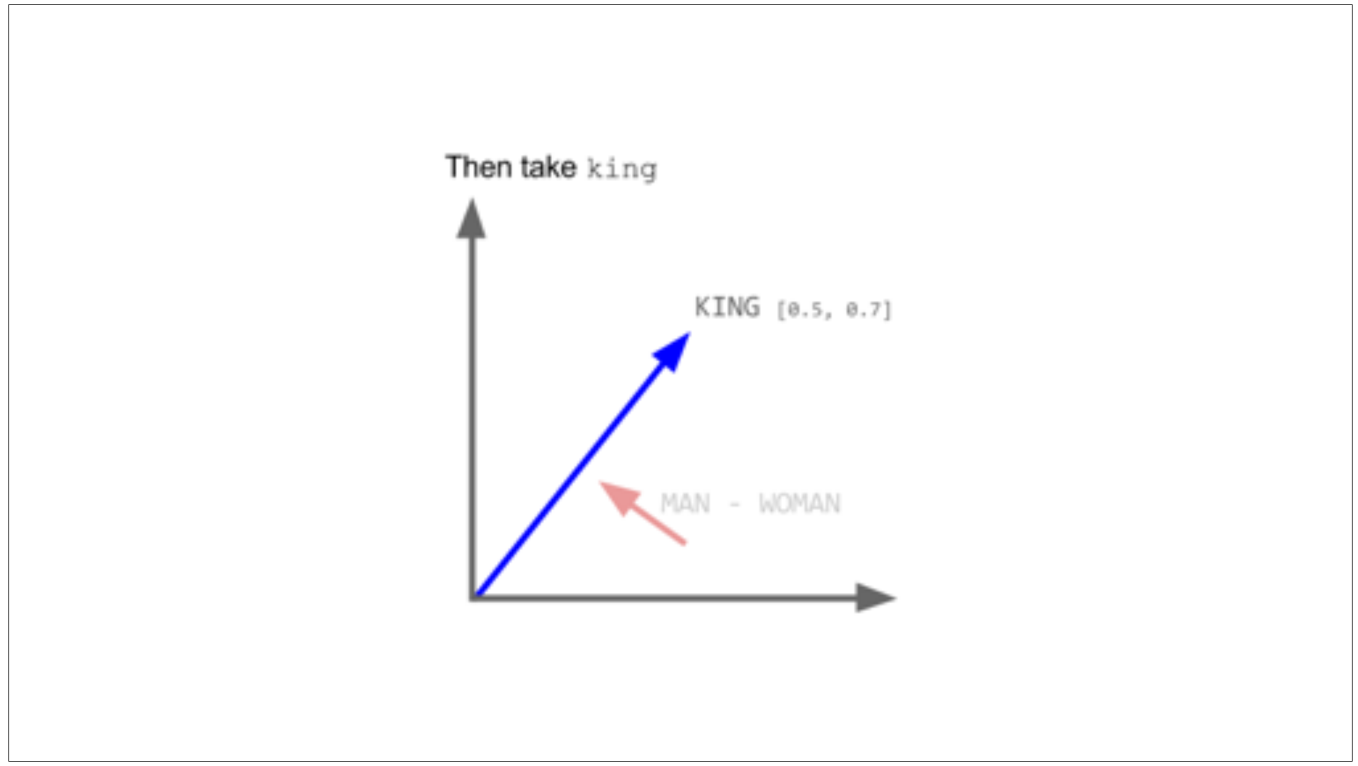

What is king + man - woman?

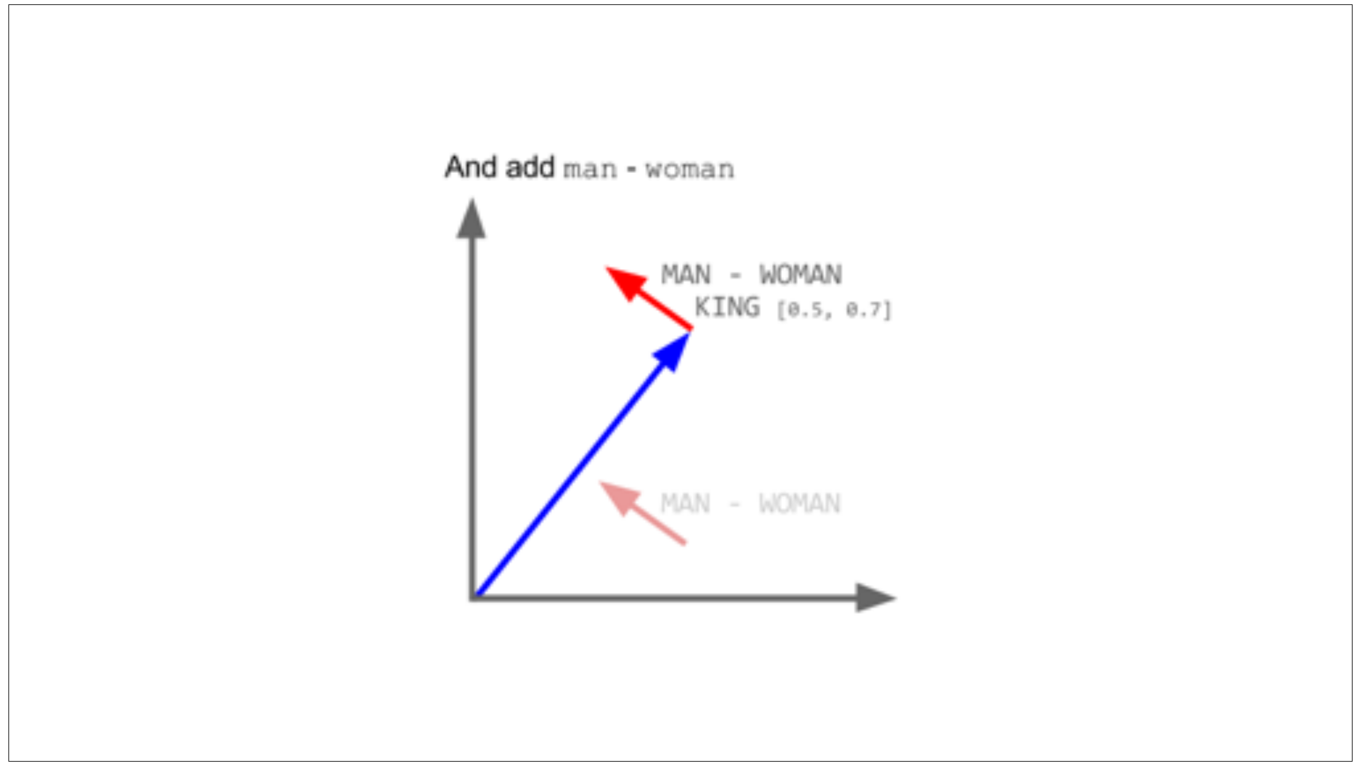


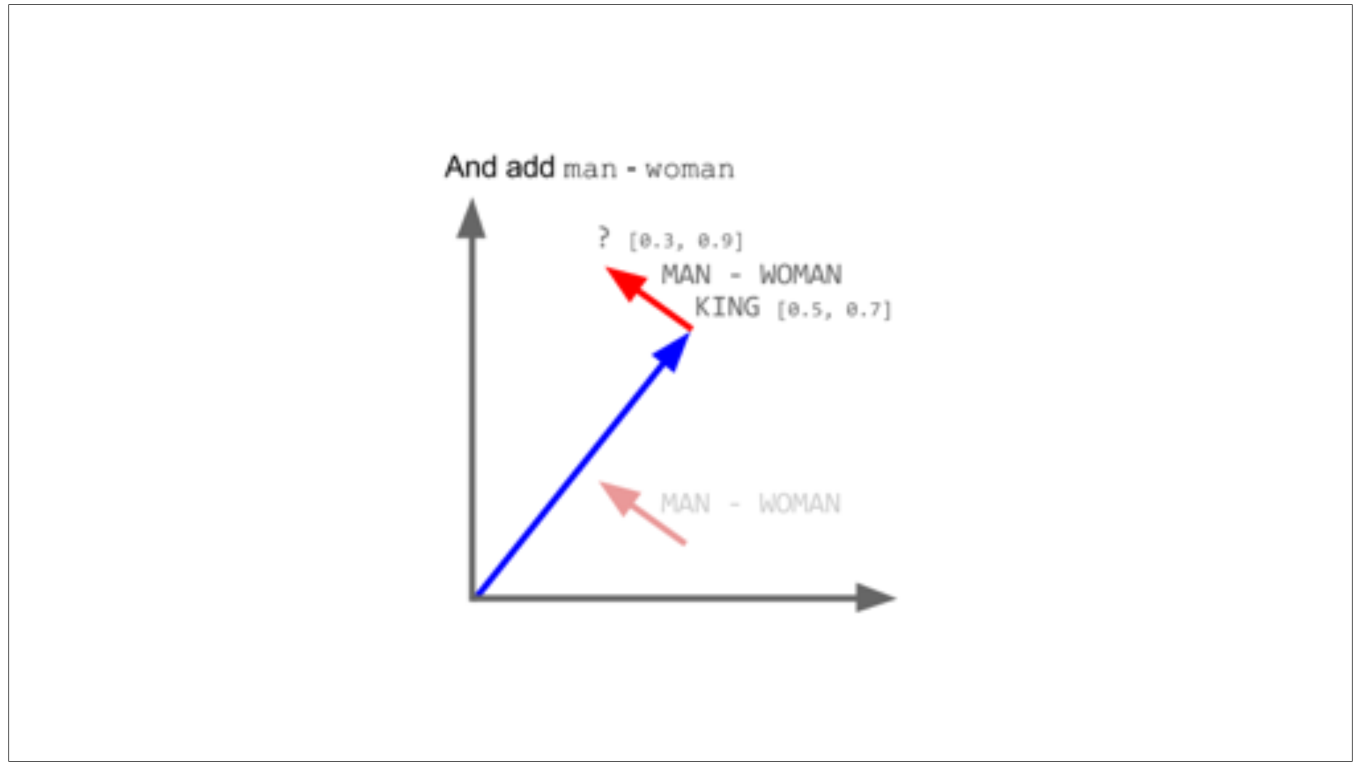
Showing just 2 of the ~500 dimensions. Effectively we’ve PCA’d it

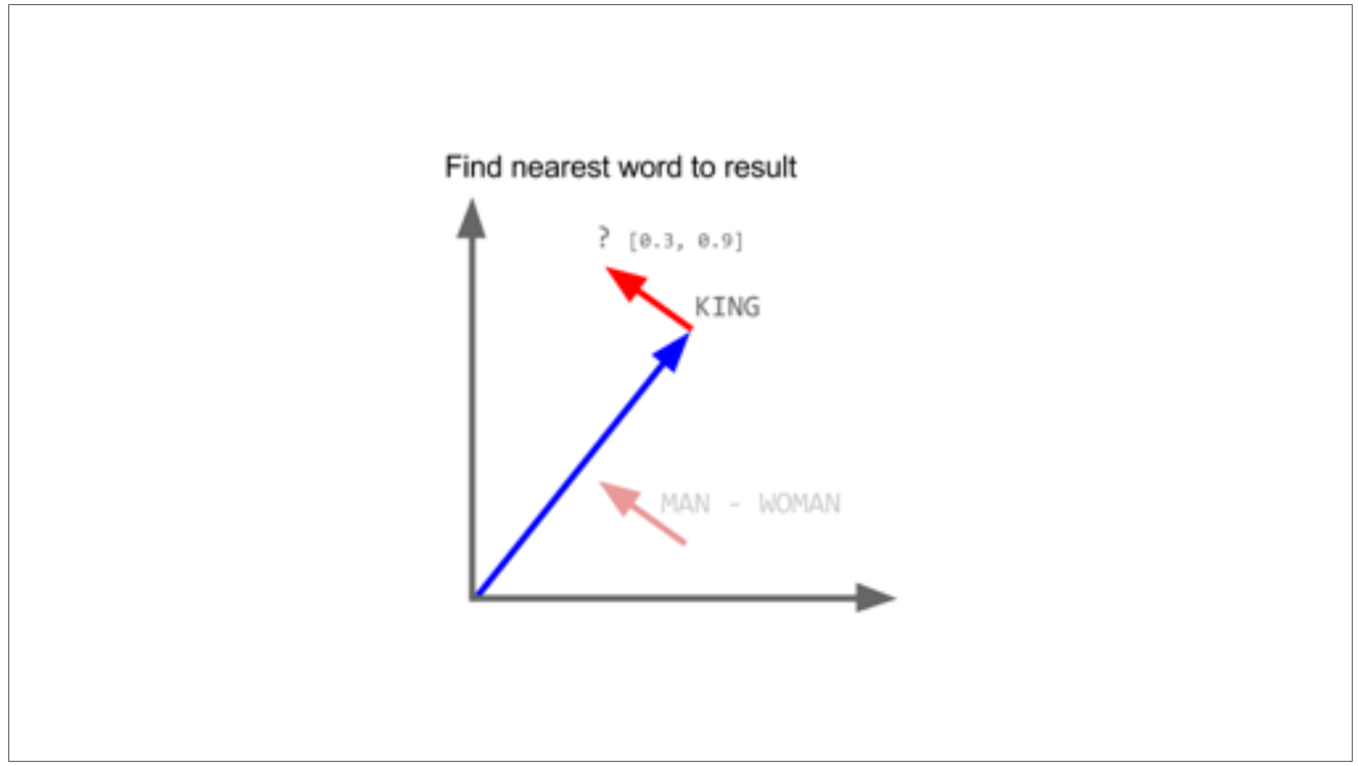

Start with man - woman
KING
WOMAN [0.3, 0.4]
MAN [0.5, 0.2]

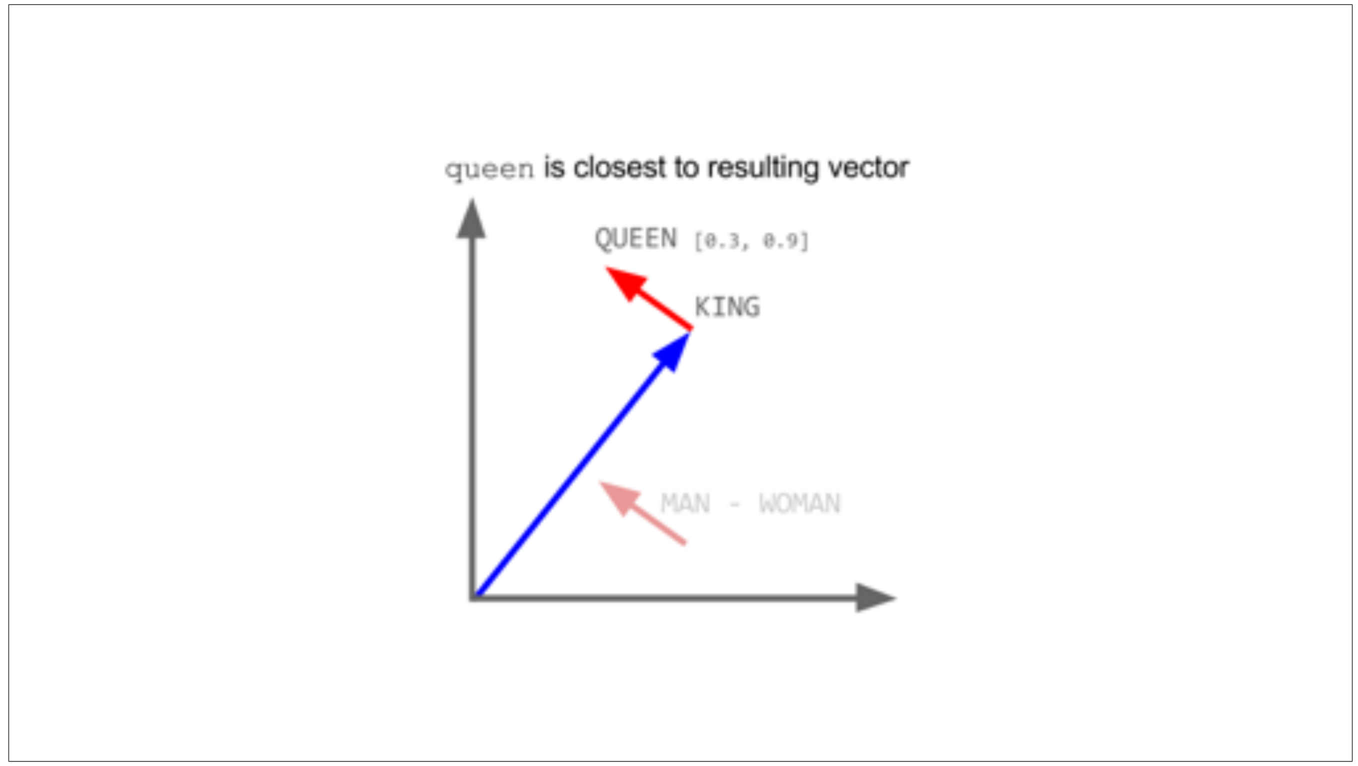

Start with man - woman
KING
MAN - WOMAN

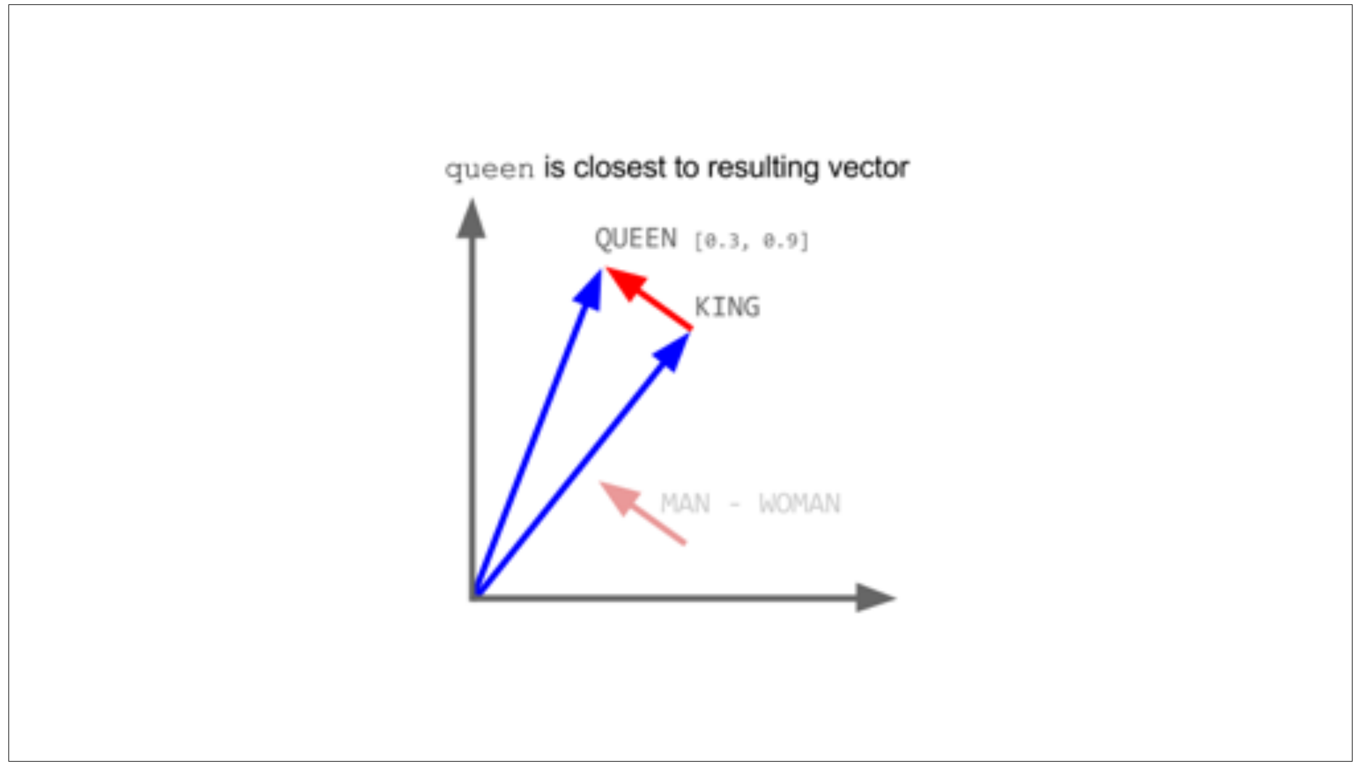

Then take king
KING [0.5, 0.7]
MAN - WOMAN


And add man - woman
MAN - WOMAN
KING [0.5, 0.7]
MAN - WOMAN

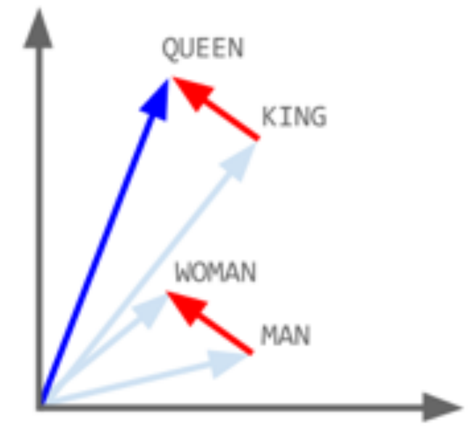

And add man - woman
? [0.3, 0.9]
MAN - WOMAN
KING [0.5, 0.7]
MAN - WOMAN

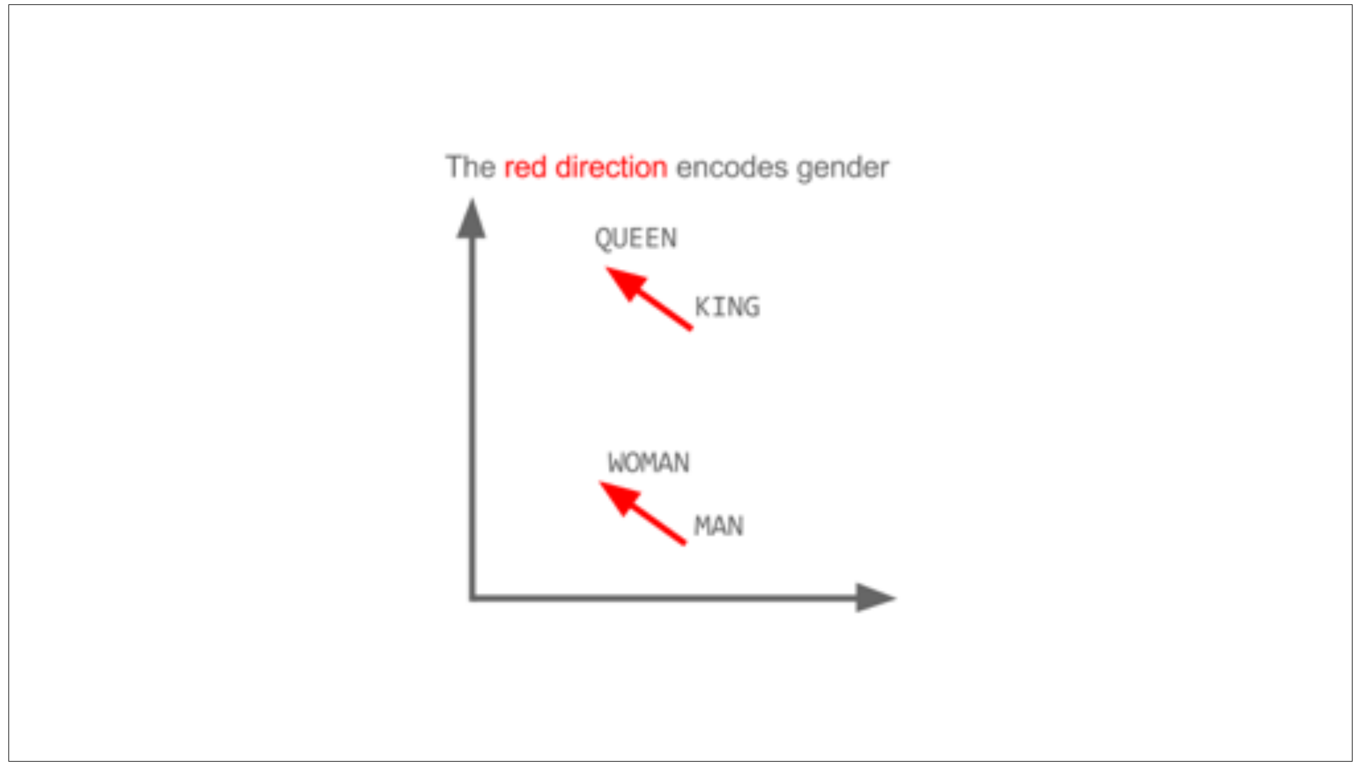

Find nearest word to result
? [0.3, 0.9]
KING
MAN - WOMAN


queen is closest to resulting vector
QUEEN [0.3, 0.9]
KING
MAN - WOMAN

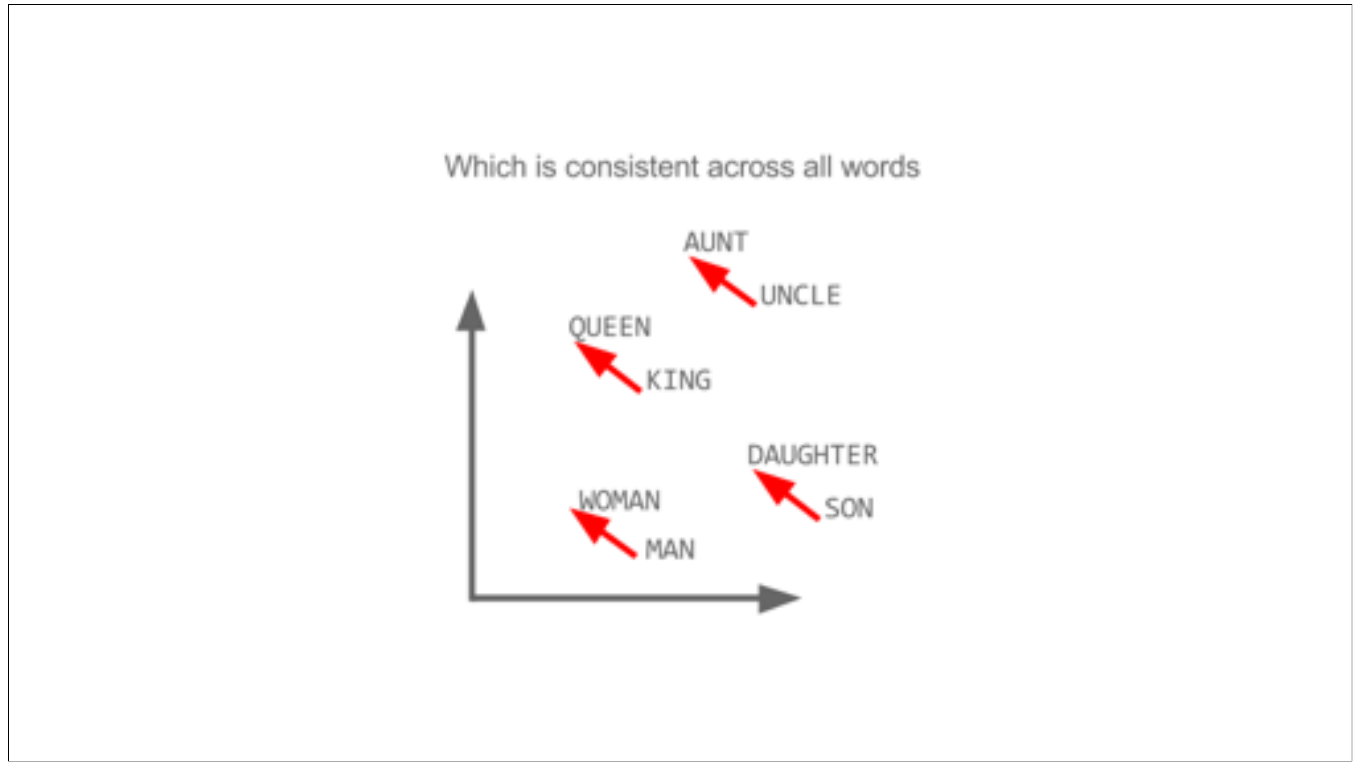

queen is closest to resulting vector
QUEEN [0.3, 0.9]
KING
MAN - WOMAN

So king + man - woman = queen!

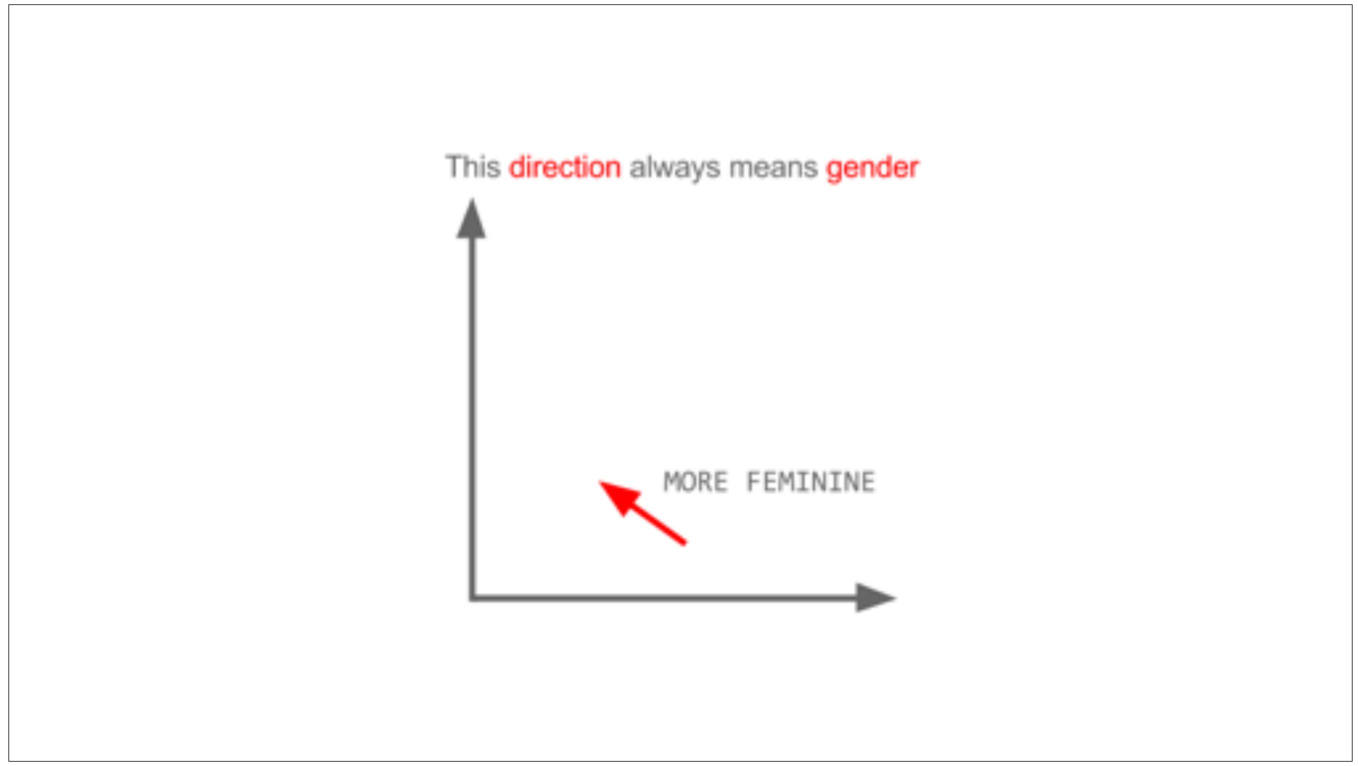


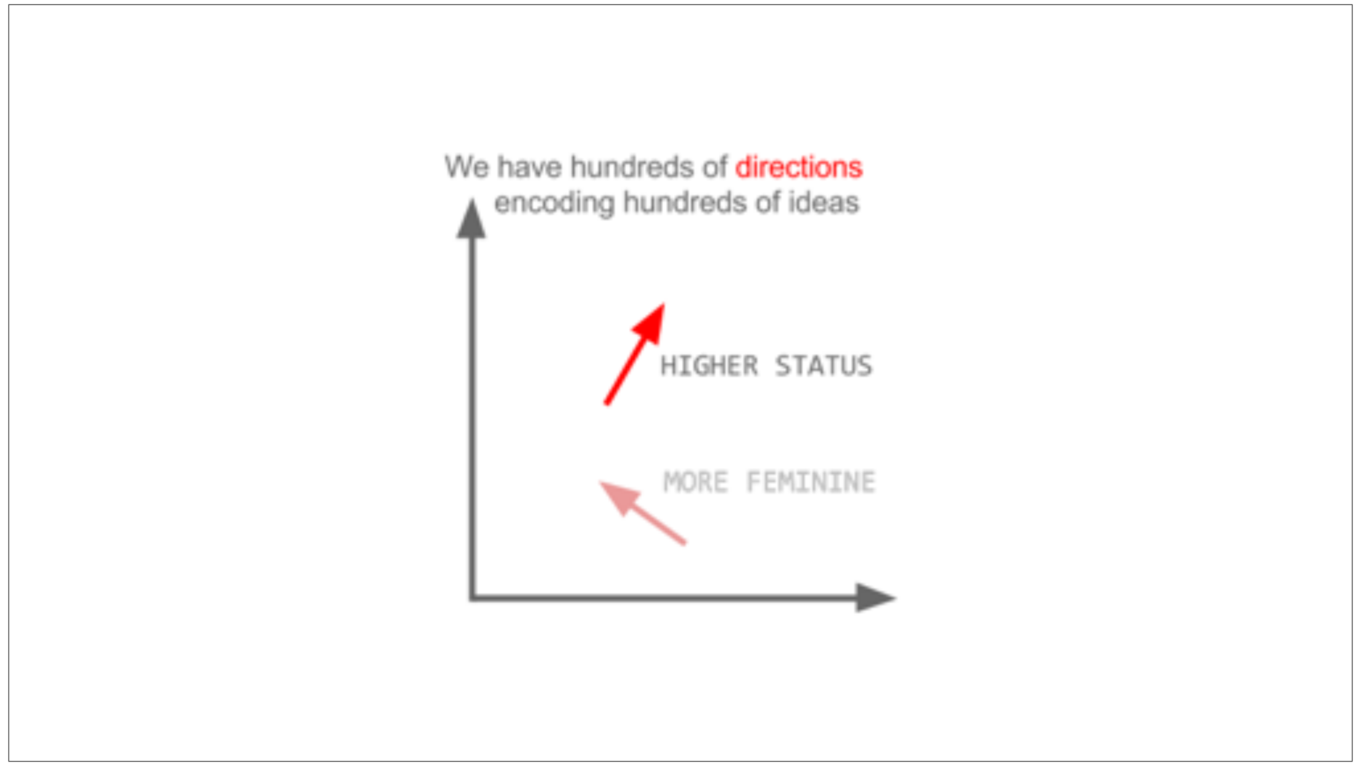


The red direction encodes gender
QUEEN
KING
WOMAN
MAN



If we only had locality and not regularity, this wouldn't necessarily be true


This direction always means gender
MORE FEMININE


We have hundreds of directions
encoding hundreds of ideas
HIGHER STATUS
MORE FEMININE

Czech + currency	Vietnam + capital	German + airlines	Russian + river	French + actress
koruna	Hanoi	airline Lufthansa	Moscow	Juliette Binoche
Check crown	Ho Chi Minh City	carrier Lufthansa	Volga River	Vanessa Paradis
Polish zolty	Viet Nam	flag carrier Lufthansa	upriver	Charlotte Gainsbourg
CTK	Vietnamese	Lufthansa	Russia	Cecile De

So we live in a vector space where operations like addition and subtraction are meaningful.

So here’s a few examples of this working.

Really get the idea of these vectors as being ‘mixes’ of other ideas & vectors

ITEM_3469 + ‘Pregnant’

SF is a person service

Box

I love the stripes and the cut around my neckline was amazing

someone else might write ‘grey and black’

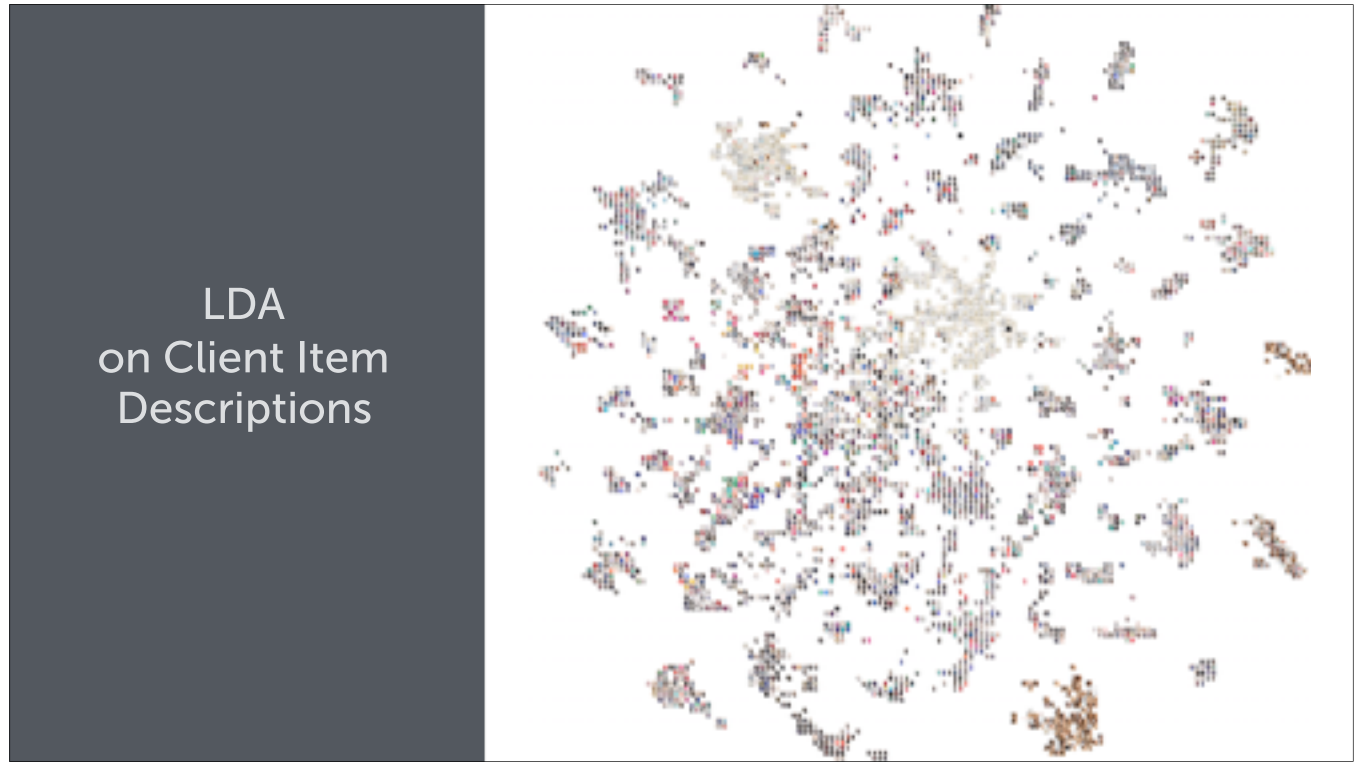
subtlety and nuance in that language

We have lots of this interaction — of order wikipedia amount — far too much to manually annotate anything

= ITEM_701333
= ITEM_901004
= ITEM_800456

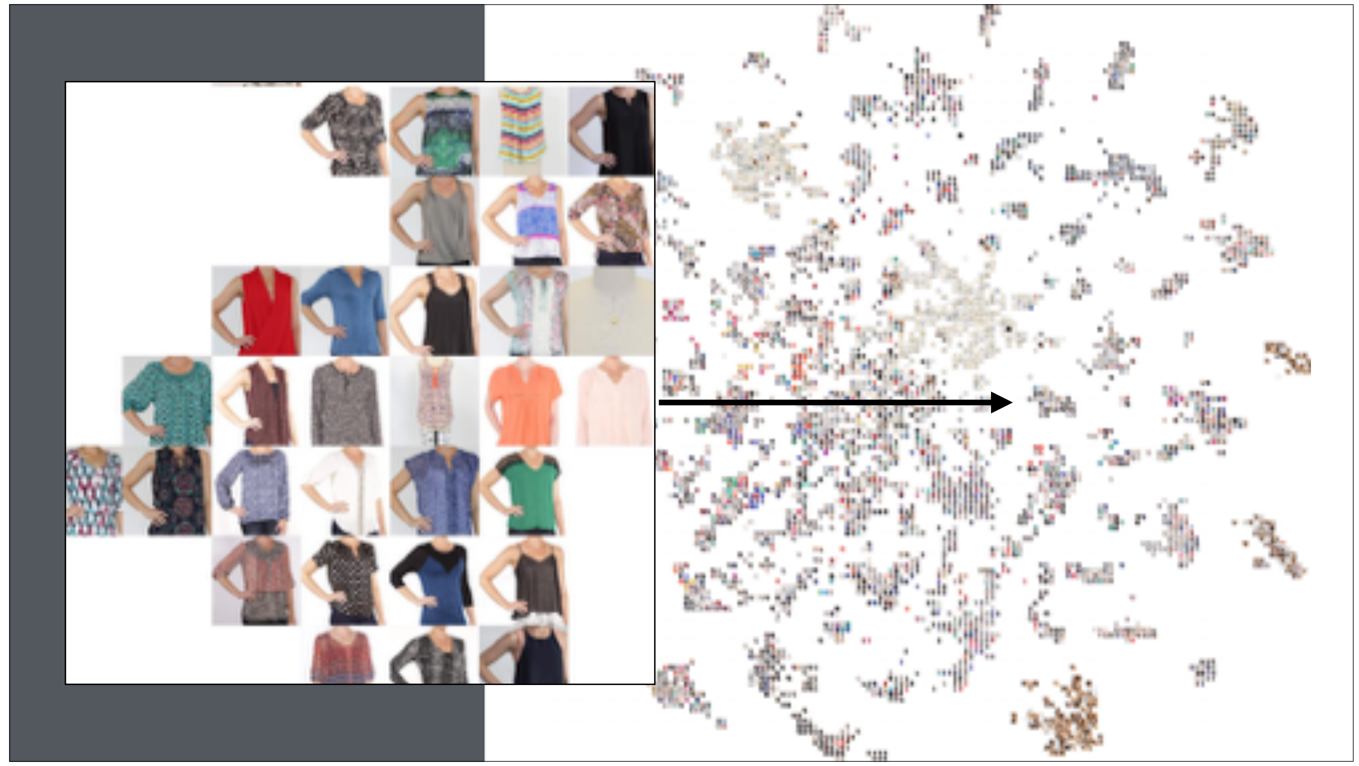
Stripes and are safe for maternity
And also similar tones and flowy — still great for expecting mothers

what about **LDA?**

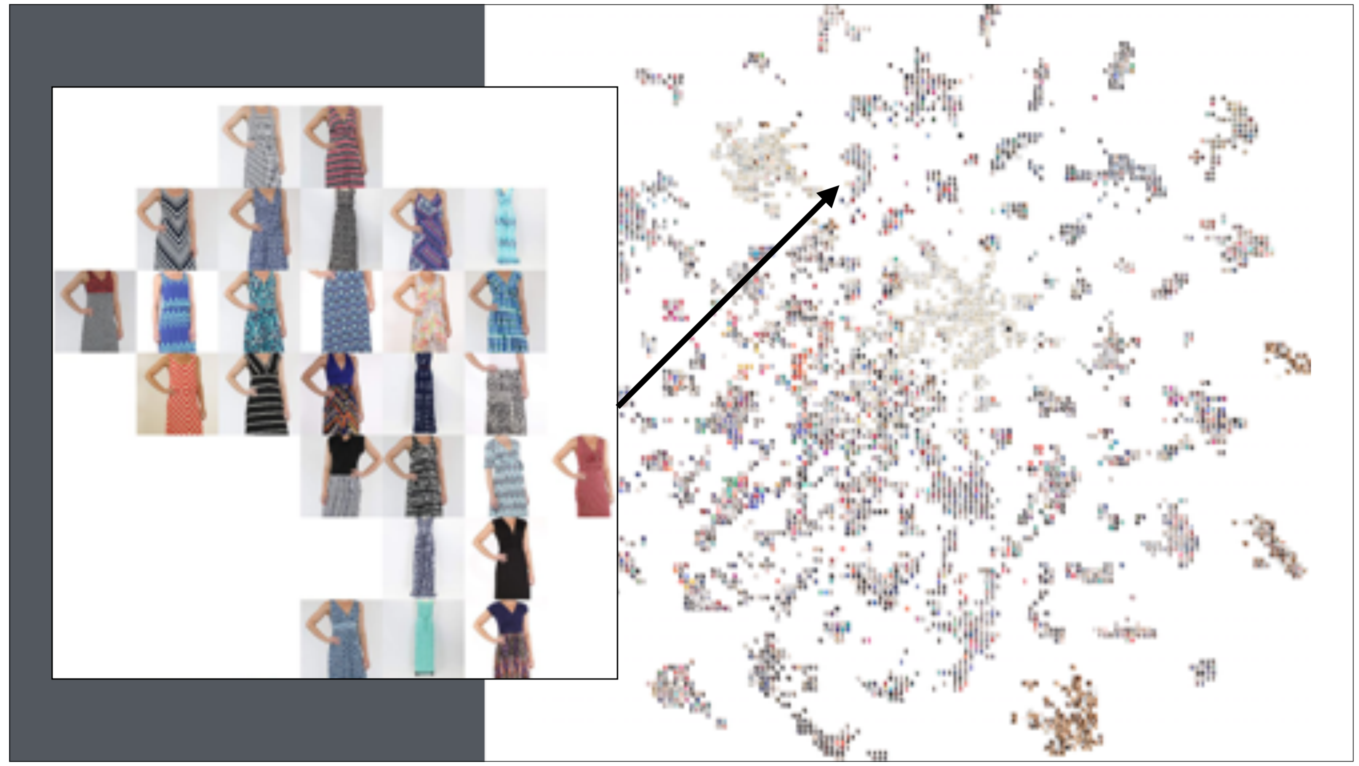


This shows the incredible amount of structure

clunky jewelry
dangling delicate jewelry elsewhere

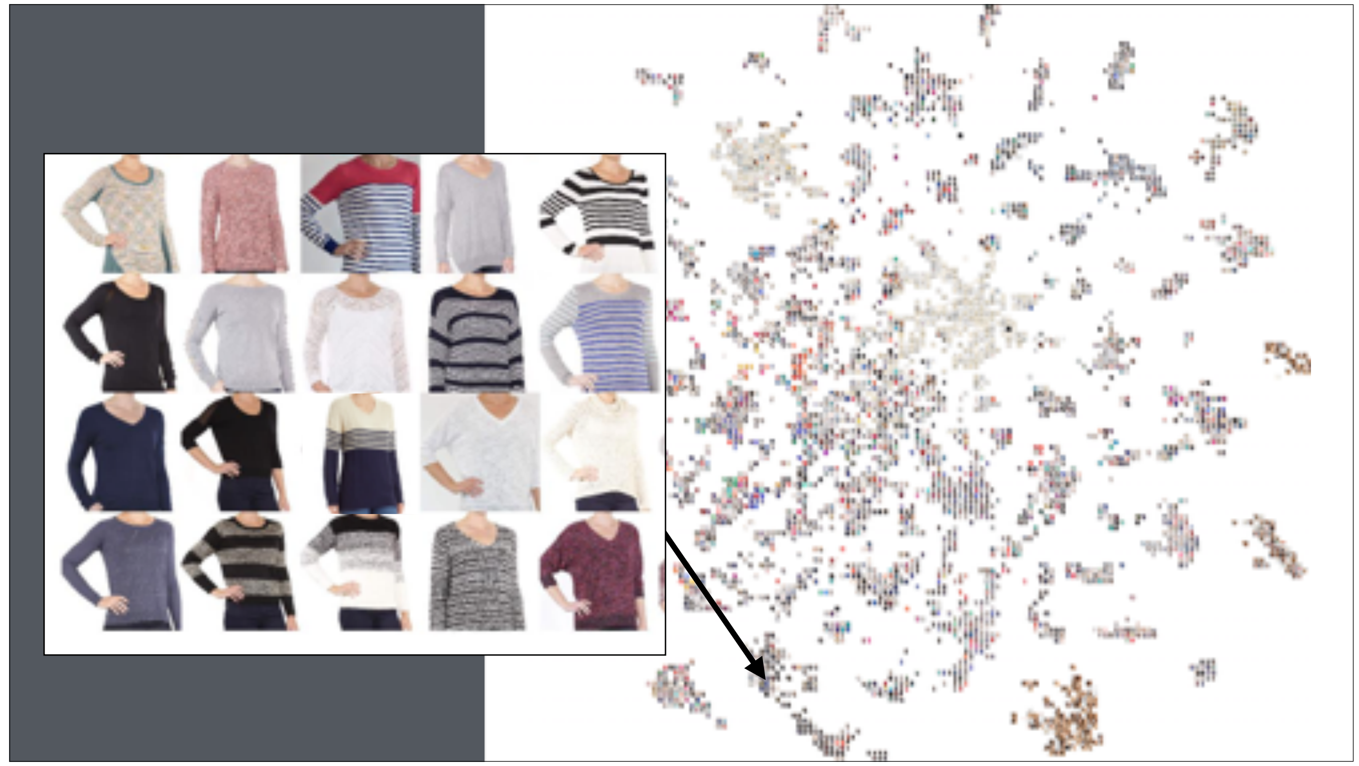


topics on patterns, styles — this cluster is similarly described as high contrast tops with popping colors

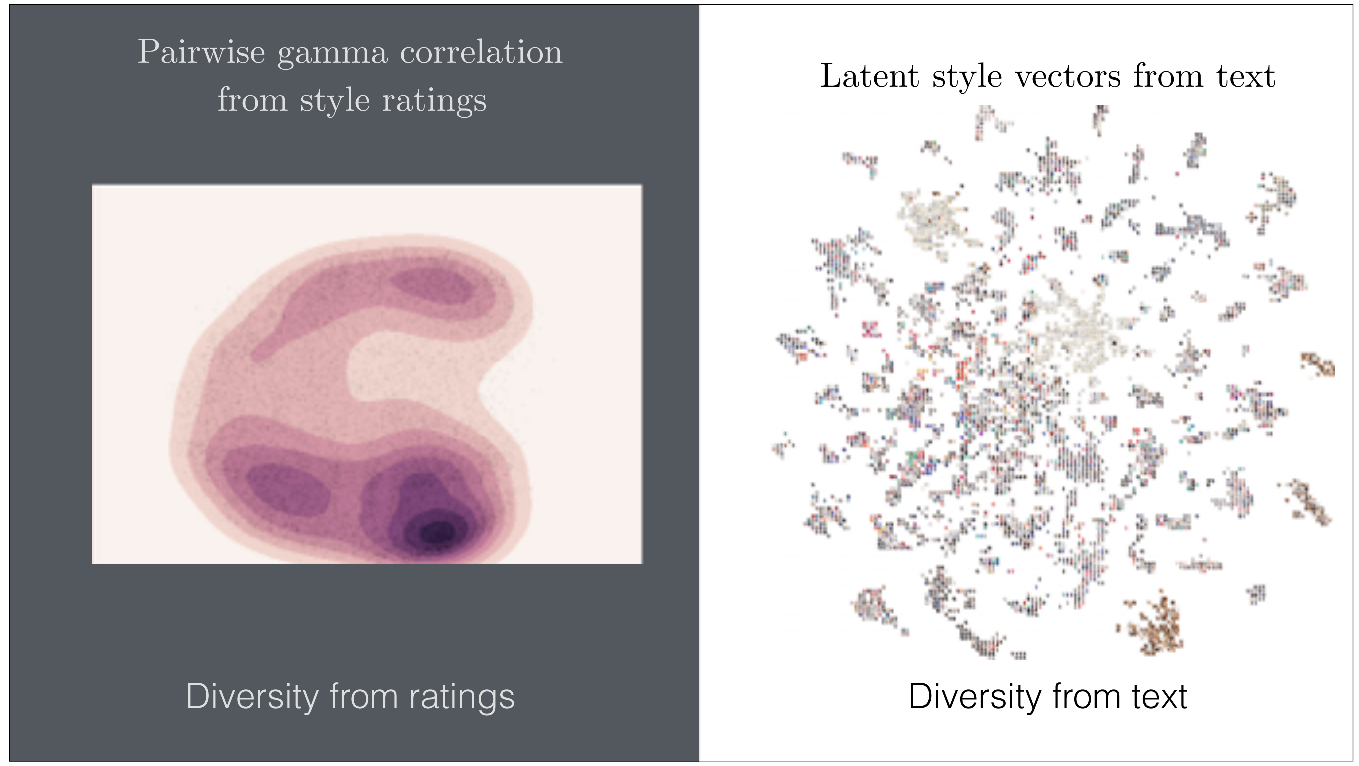


bright dresses for a warm summer

maternity line clothes



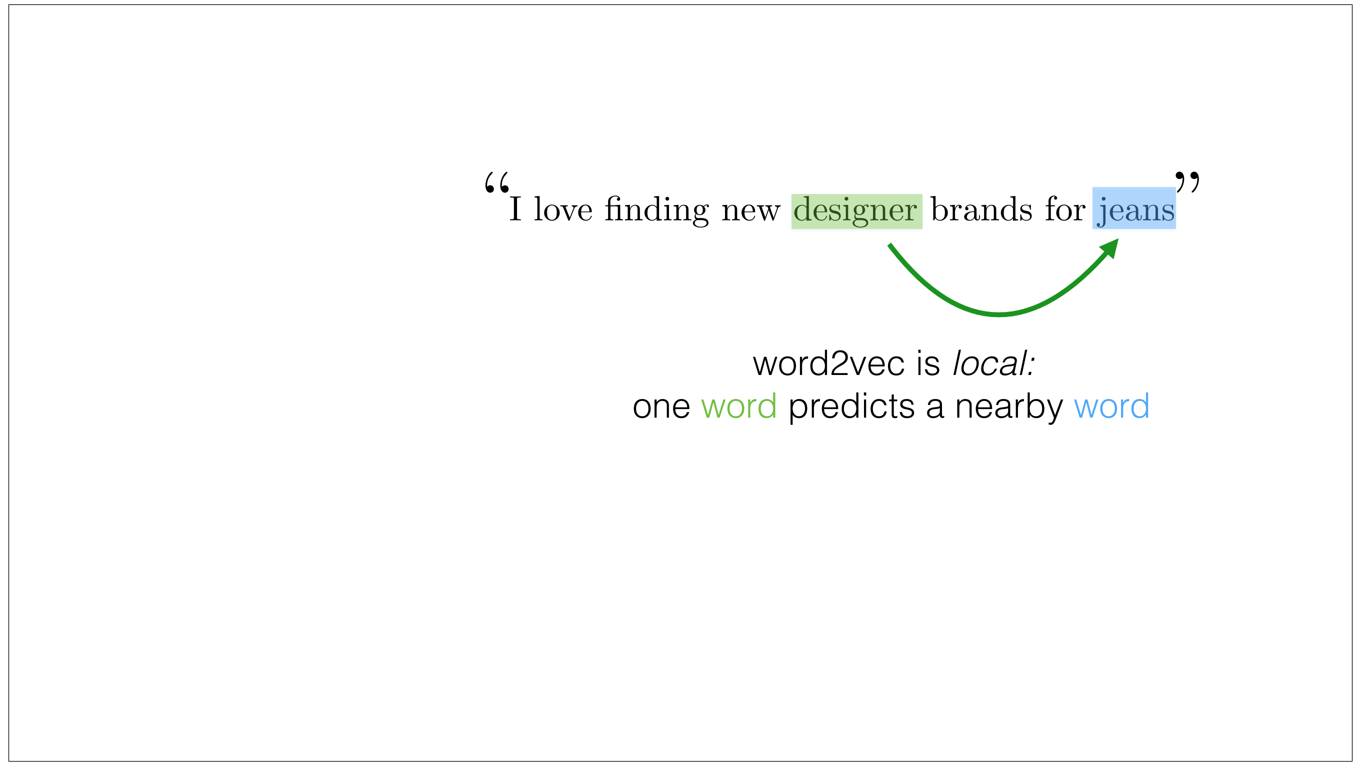
not just visual topics, but also topics about fit



Lots of structure in both — but the diversity much higher in the text

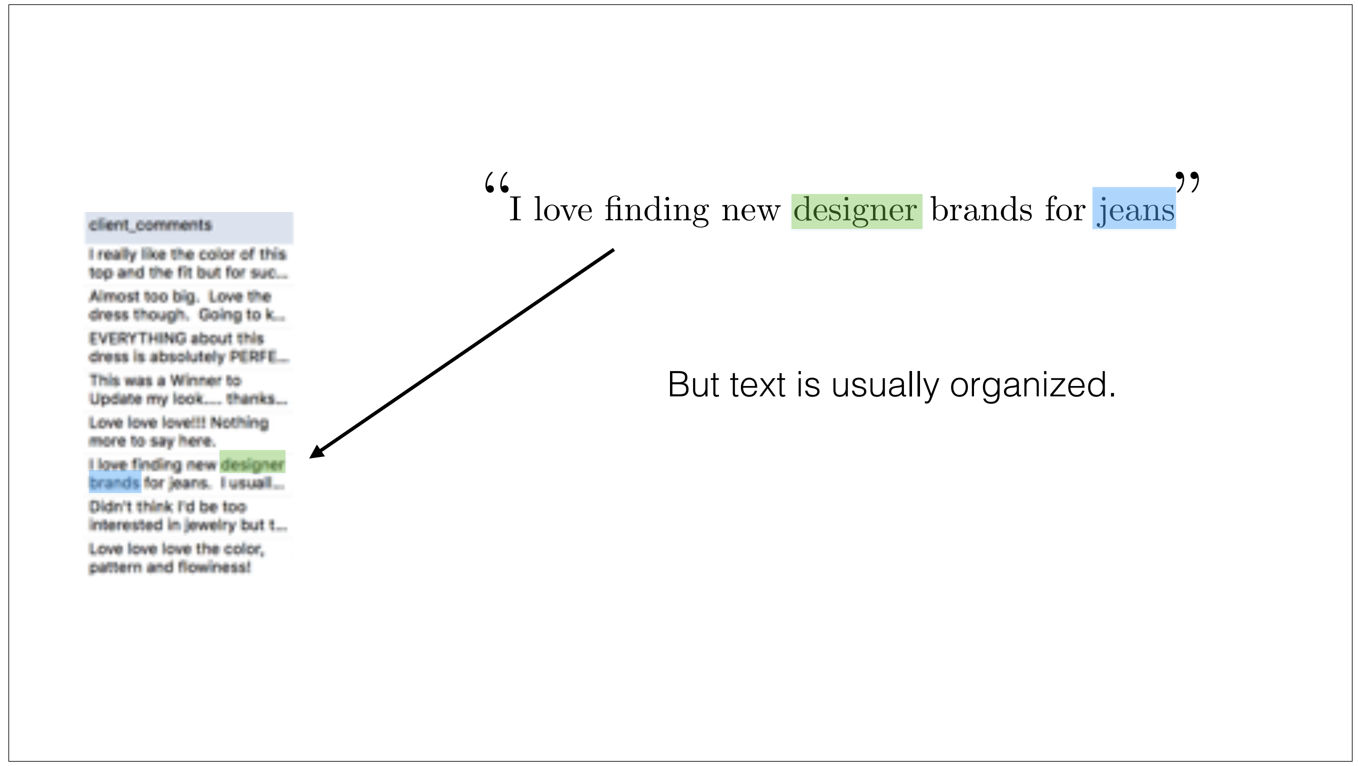
Maybe obvious: but the way people describe items is fundamentally richer than the style ratings

lda vs **word2vec**

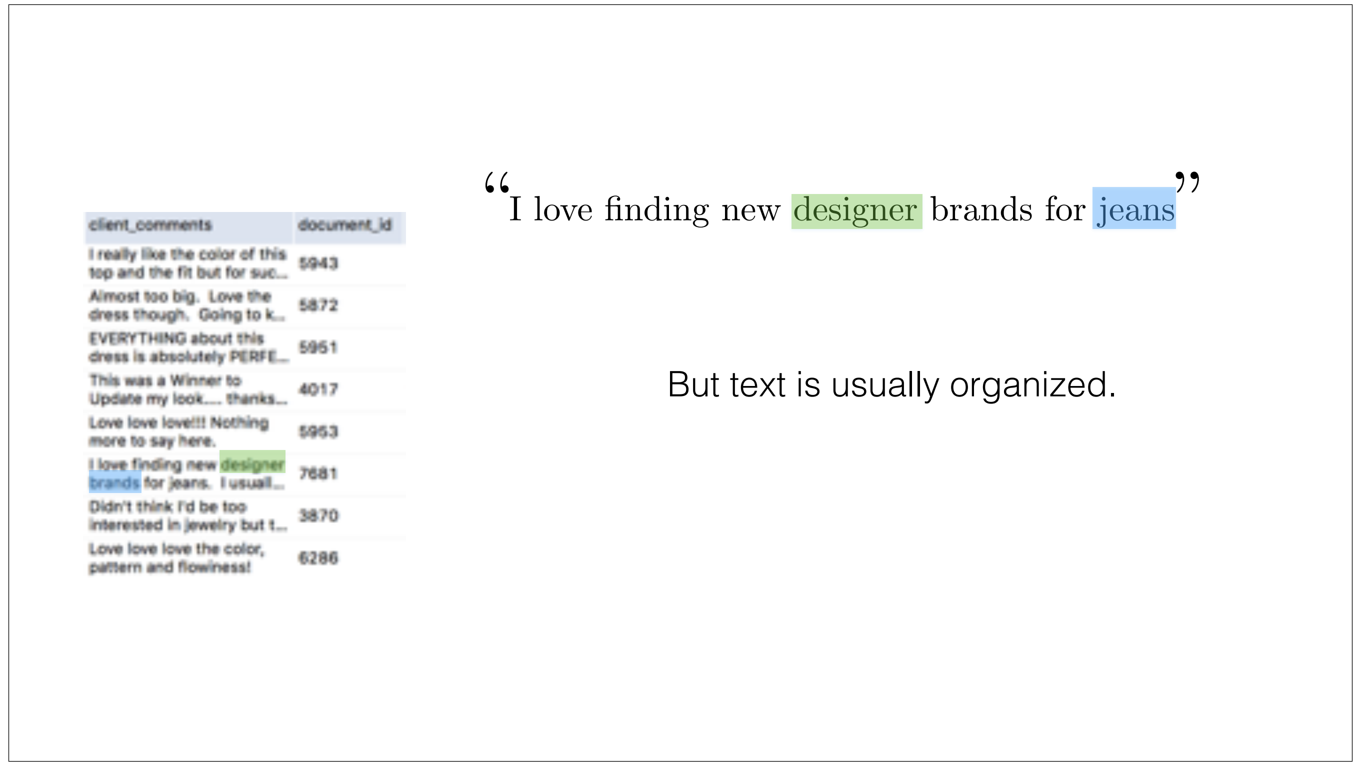


as if the world where one very long text string. no end of documents, no end of sentence, etc.

and a window across words

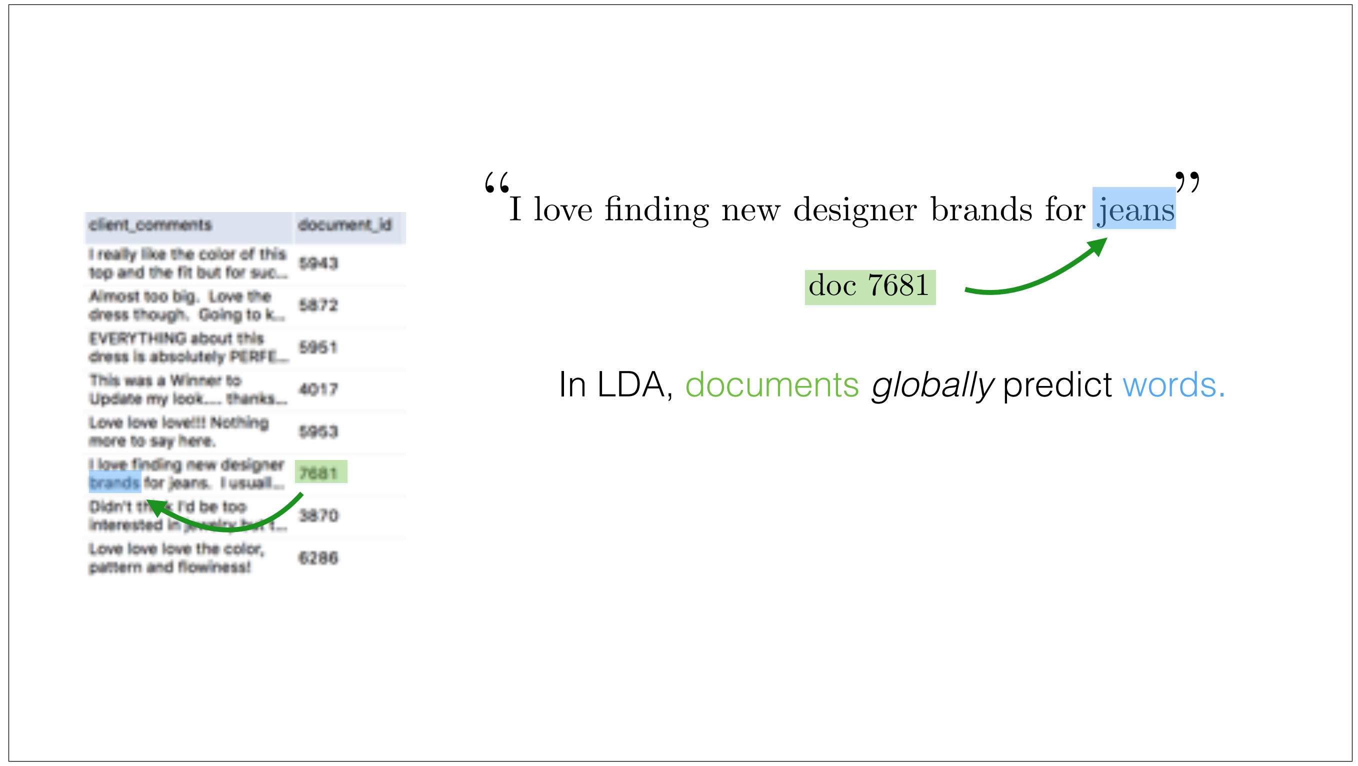


as if the world where one very long text string. no end of documents, no end of sentence, etc.



client_comments	document_id
I really like the color of this top and the fit but for suc...	5943
Almost too big. Love the dress though. Going to k...	5872
EVERYTHING about this dress is absolutely PERFE...	5951
This was a Winner to Update my look..... thanks...	4017
Love love love!!! Nothing more to say here.	5953
I love finding new designer brands for jeans. I usuall...	7681
Didn't think I'd be too interested in jewelry but t...	3870
Love love love the color, pattern and flowiness!	6286

as if the world where one very long text string. no end of documents, no end of sentence, etc.

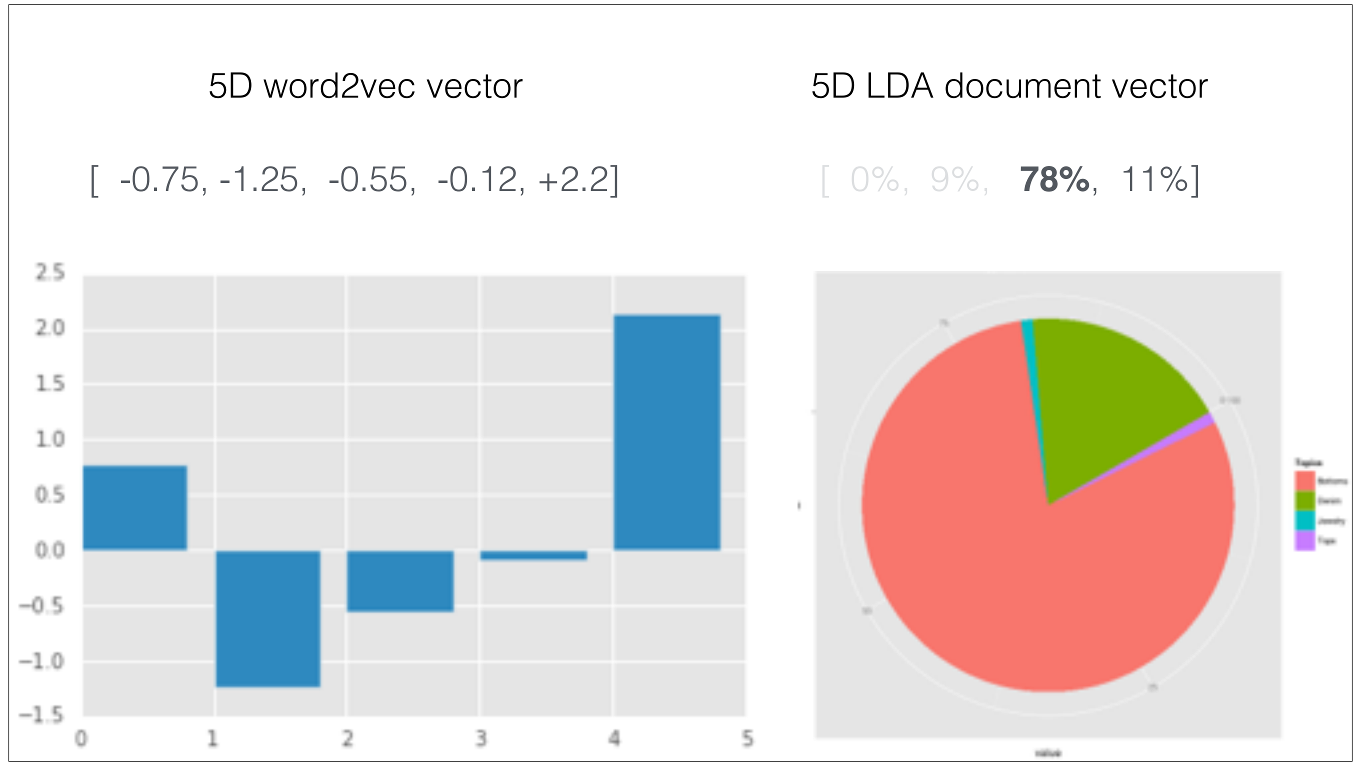


these are client comment which are short, only predict dozens of words

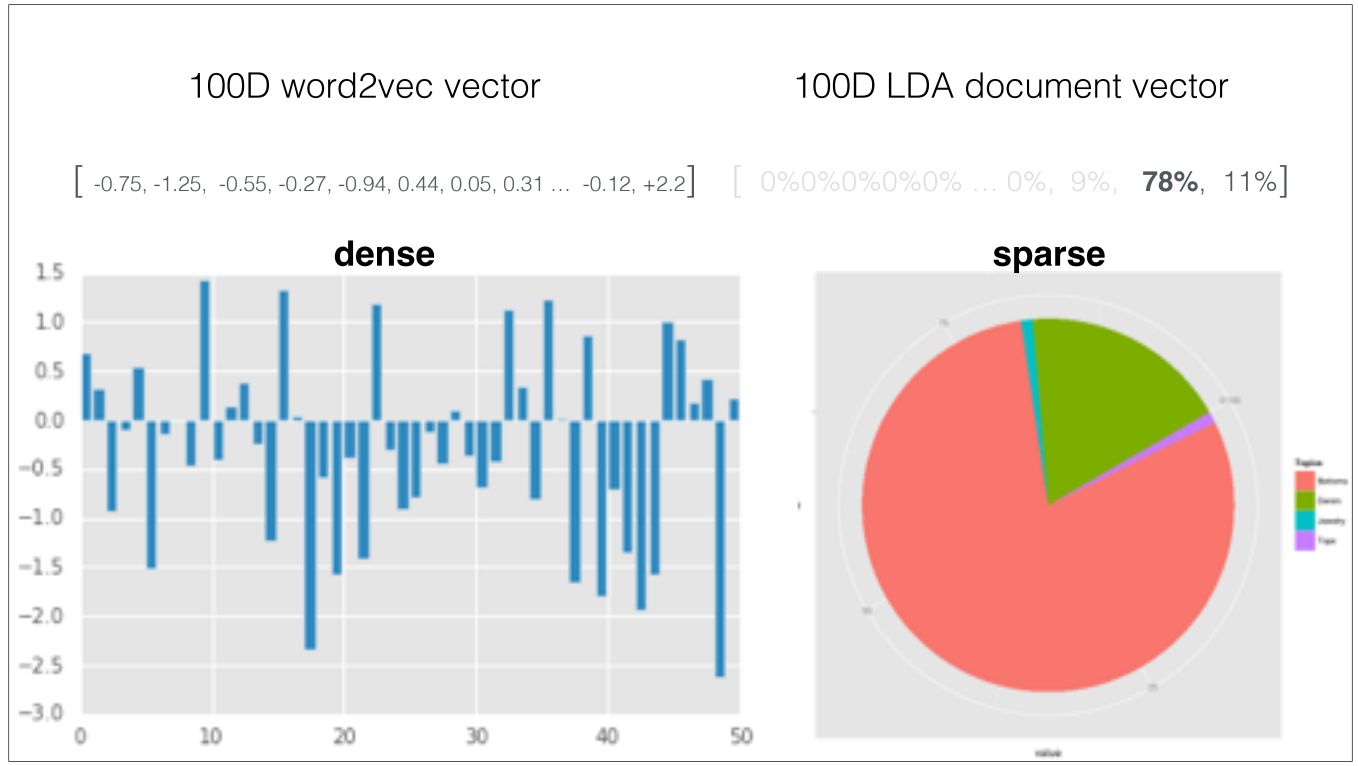
but could be legal documents, or medical documents, 10k words — here the difference between global and local algorithms is much more important

typical word2vec vector	typical LDA document vector
[-0.75, -1.25, -0.55, -0.12, +2.2]	[0%, 9%, 78%, 11%]

typical word2vec vector	typical LDA document vector
[-0.75, -1.25, -0.55, -0.12, +2.2]	[0%, 9%, **78%**, 11%]
All real values	All sum to 100%



LDA is a *mixture*
w2v is a bunch of real numbers — more like and *address*
much easier to say to another human 78% of something rather than it is +2.2 of something and -1.25 of something else


100D word2vec vector
100D LDA document vector
[-0.75, -1.25, -0.55, -0.27, -0.94, 0.44, 0.05, 0.31 … -0.12, +2.2]
[0%0%0%0%0% … 0%, 9%, 78%, 11%]
dense
sparse
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
−3.0
0
10
20
30
40
50

100D word2vec vector

[-0.75, -1.25, -0.55, -0.27, -0.94, 0.44, 0.05, 0.31 … -0.12, +2.2]

Similar in 100D ways
(very **flexible**)

100D LDA document vector

[0%0%0%0%0% … 0%, 9%, **78%**, 11%]

Similar in fewer ways
(more **interpretable**)

+mixture
+sparse

can we do both? **lda2vec**

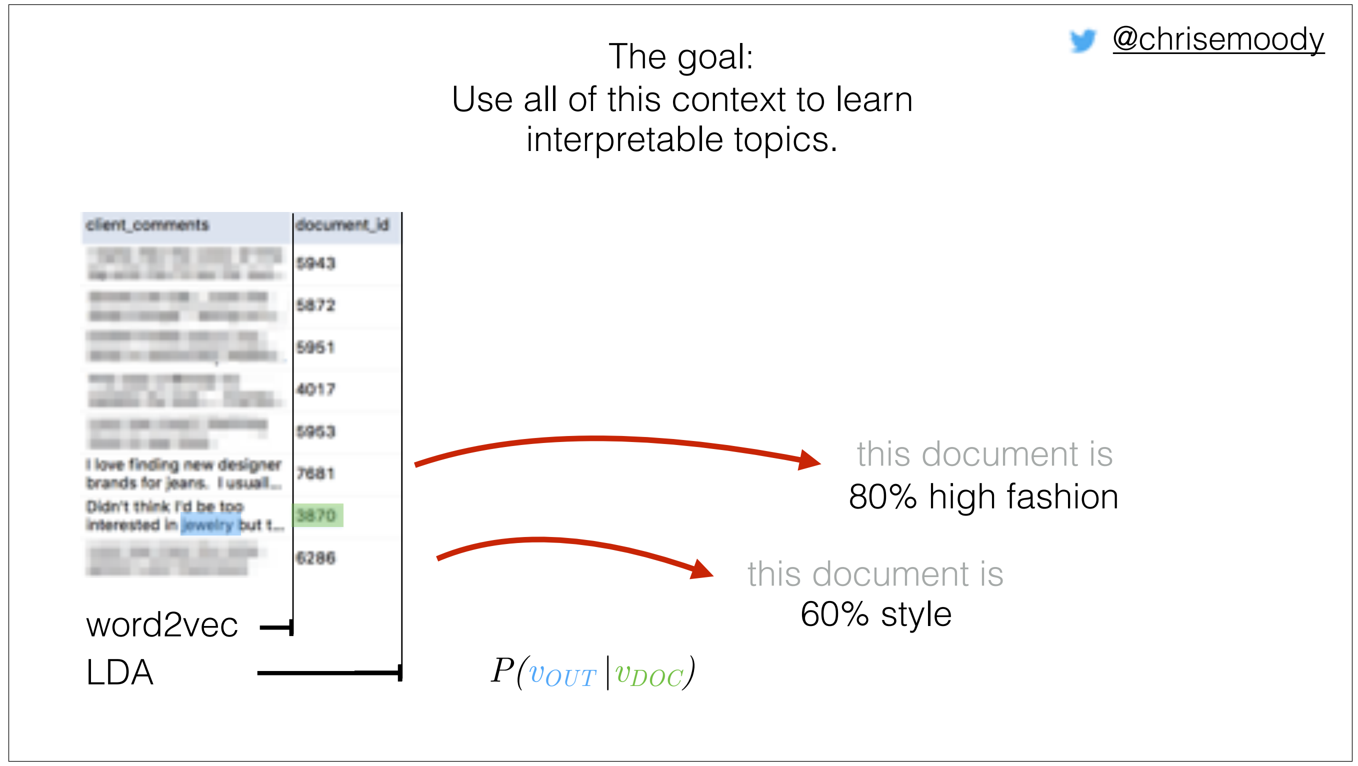
series of exp
grain of salt
very new — no good quantitative results only qualitative (but promising!)

@chrisemoody

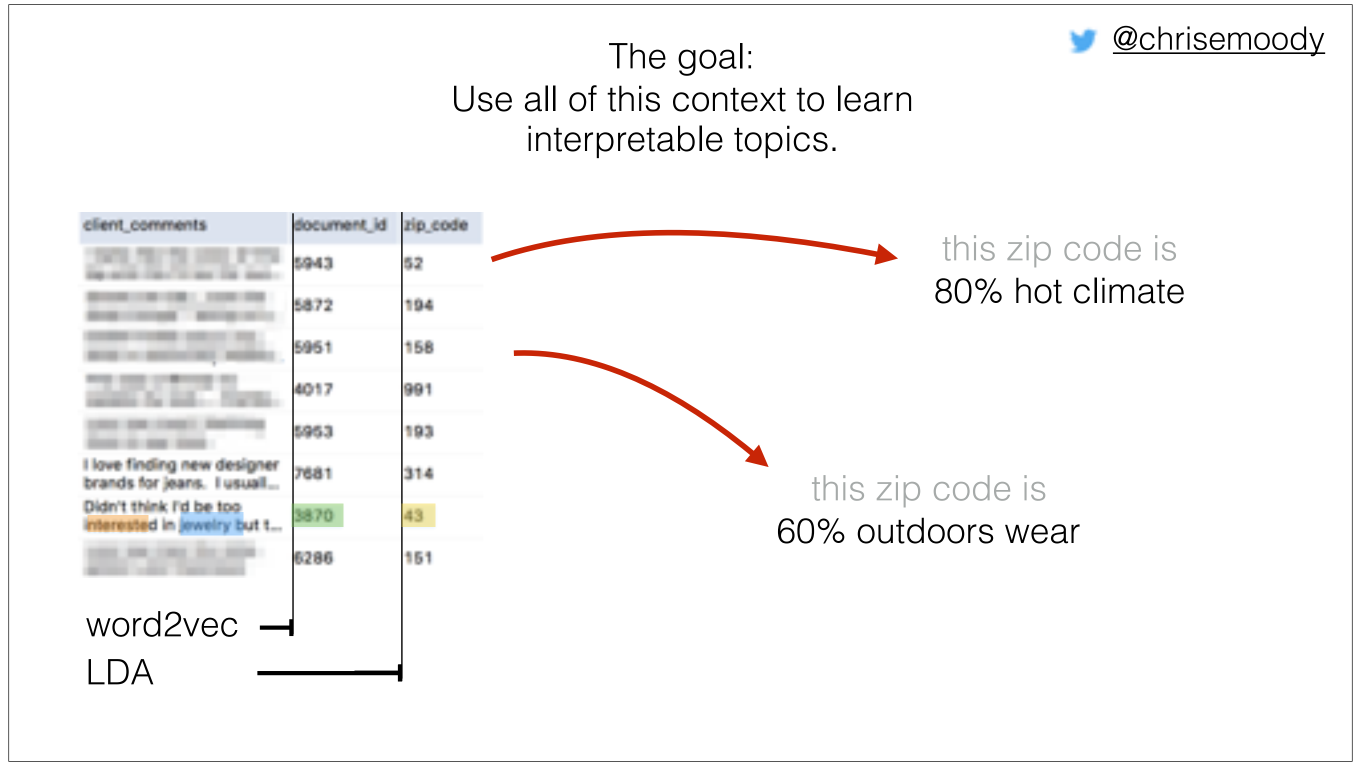
The goal:
Use all of this context to learn
interpretable topics.

word2vec → $P(v_{OUT} | v_{IN})$

Use this at SF.
typical table
w2v will use w-w

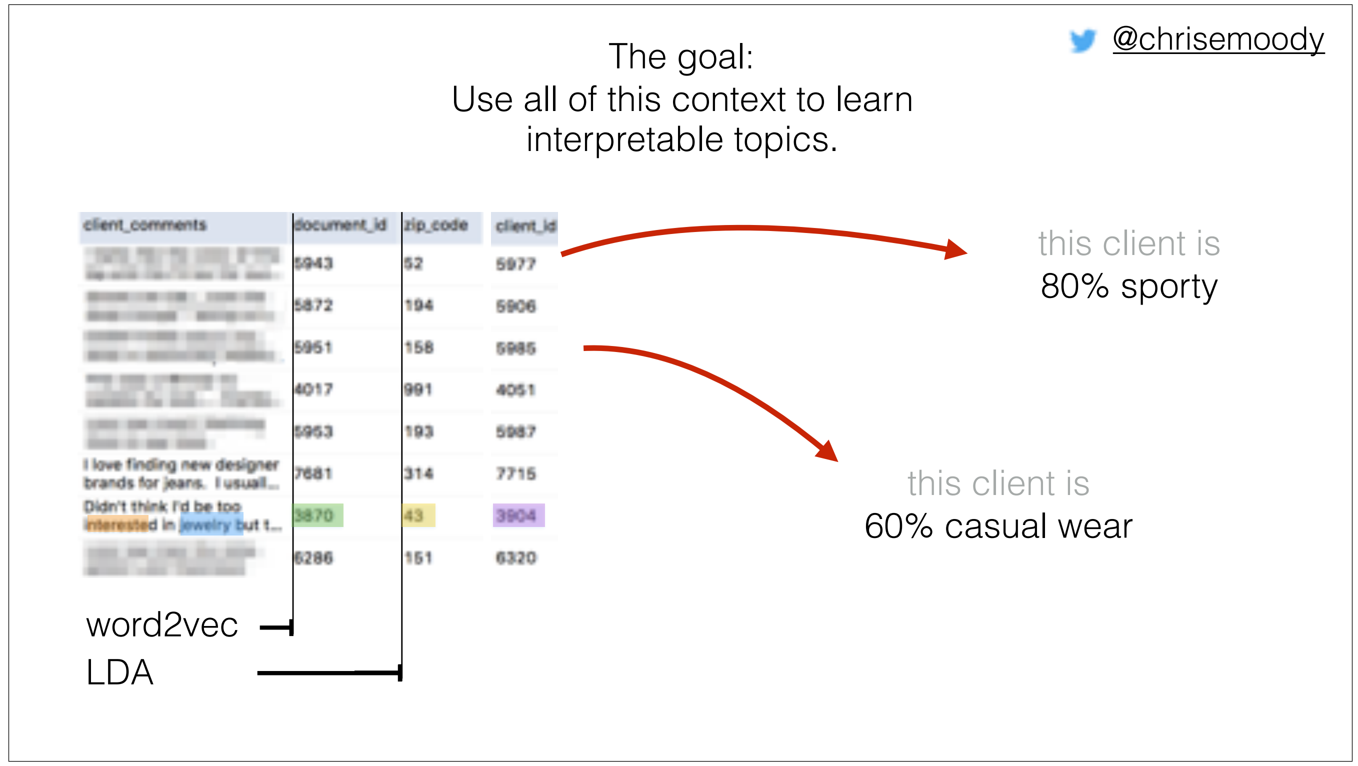


LDA will use that doc ID column
you can use this to steer the business as a whole



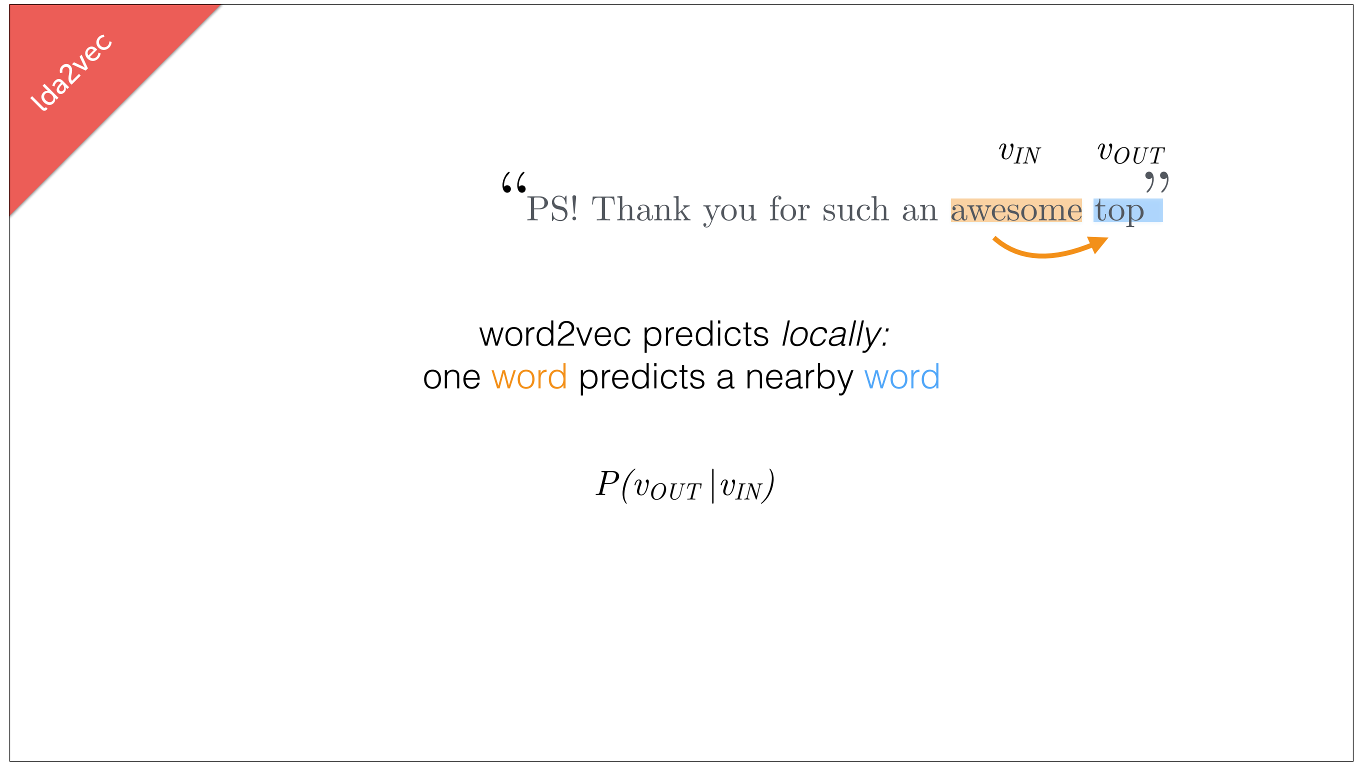
But doesn't predict word-to-word relationships.

in texas, maybe i want more lonestars & stirrup icons
in austin, maybe i want more bats

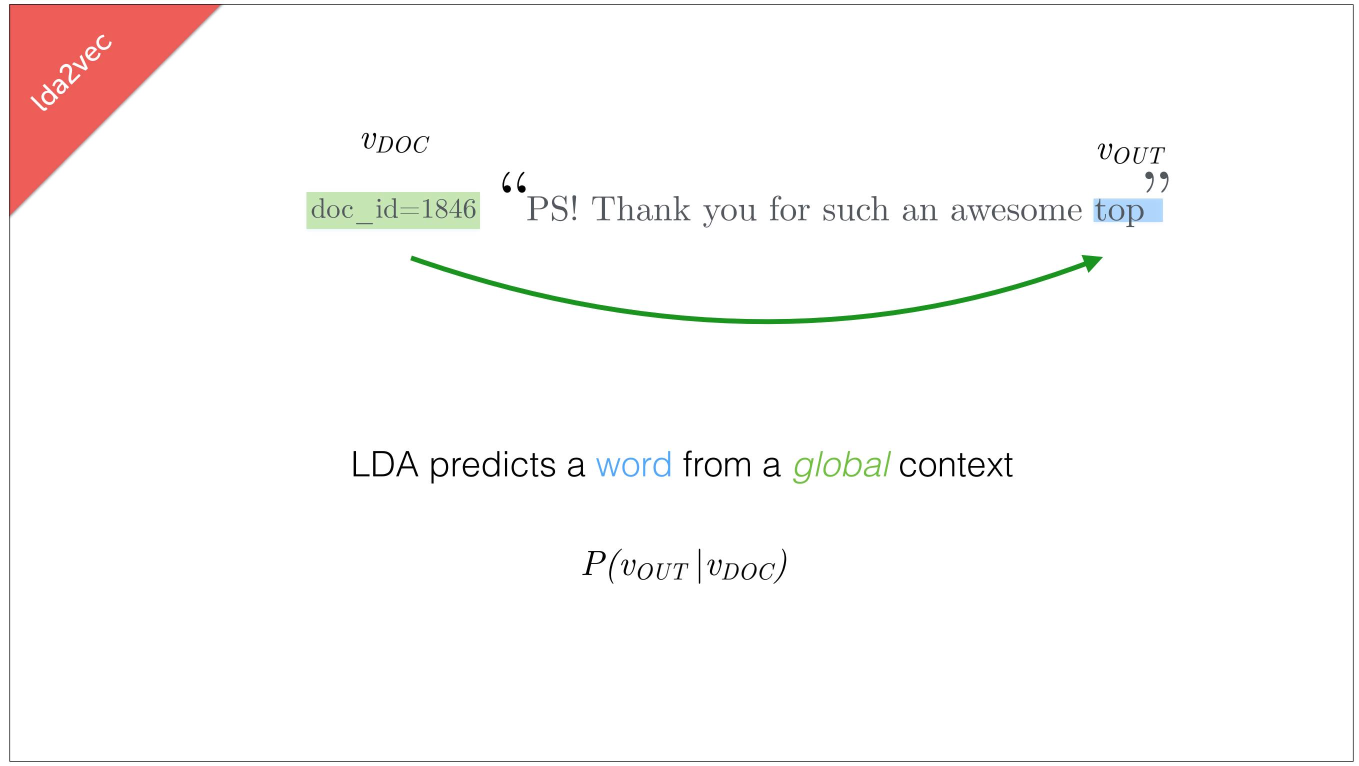


love to learn client topics
are there ‘types’ of clients? q every biz asks

so this is the promise of lda2vec



But doesn't predict word-to-word relationships.



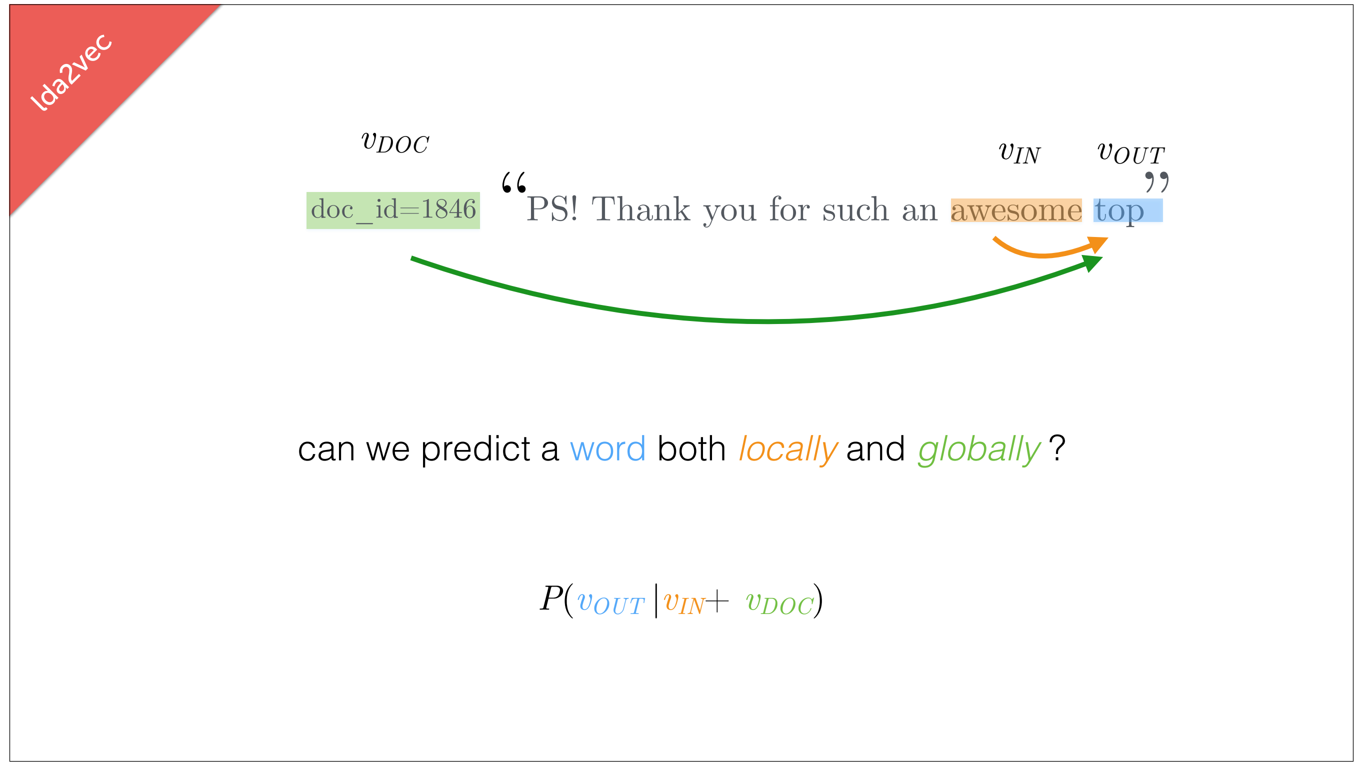
But doesn't predict word-to-word relationships.

lda2vec

v_{DOC} v_{IN} v_{OUT}

doc_id=1846 "PS! Thank you for such an awesome top"

can we predict a word both *locally* and *globally* ?



doc vector captures long-distance dependencies

word vector captures short-distance

lda2vec

v_{DOC} doc_id=1846 "PS! Thank you for such an awesome top" v_{IN} v_{OUT}

can we predict a word both *locally* and *globally* ?

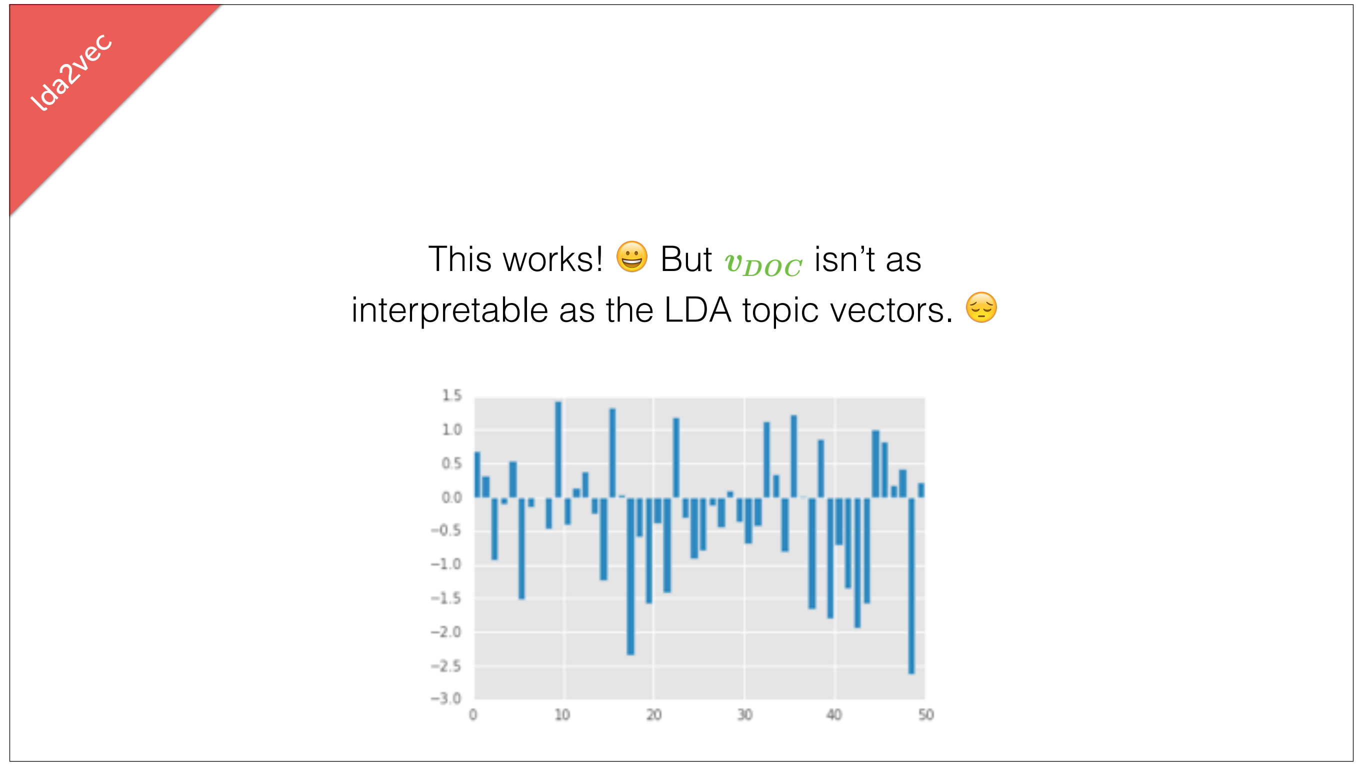
$$P(v_{OUT} \,|\, v_{IN} + \boxed{v_{DOC}})$$

*very similar to the Paragraph Vectors / doc2vec

lda2vec

This works! 😀 But v_{DOC} isn't as interpretable as the LDA topic vectors. 😔

Too many documents. I really like that document X is 70% in topic 0, 30% in topic1, …

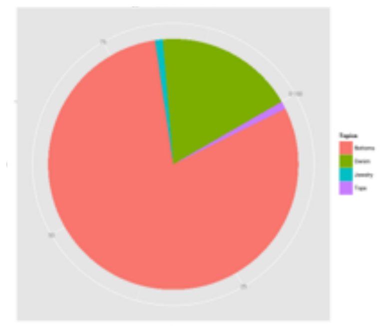


Too many documents. I really like that document X is 70% in topic 0, 30% in topic1, …

about as interpretable a hash

lda2vec

This works! 😀 But v_{DOC} isn't as interpretable as the LDA topic vectors. 😔



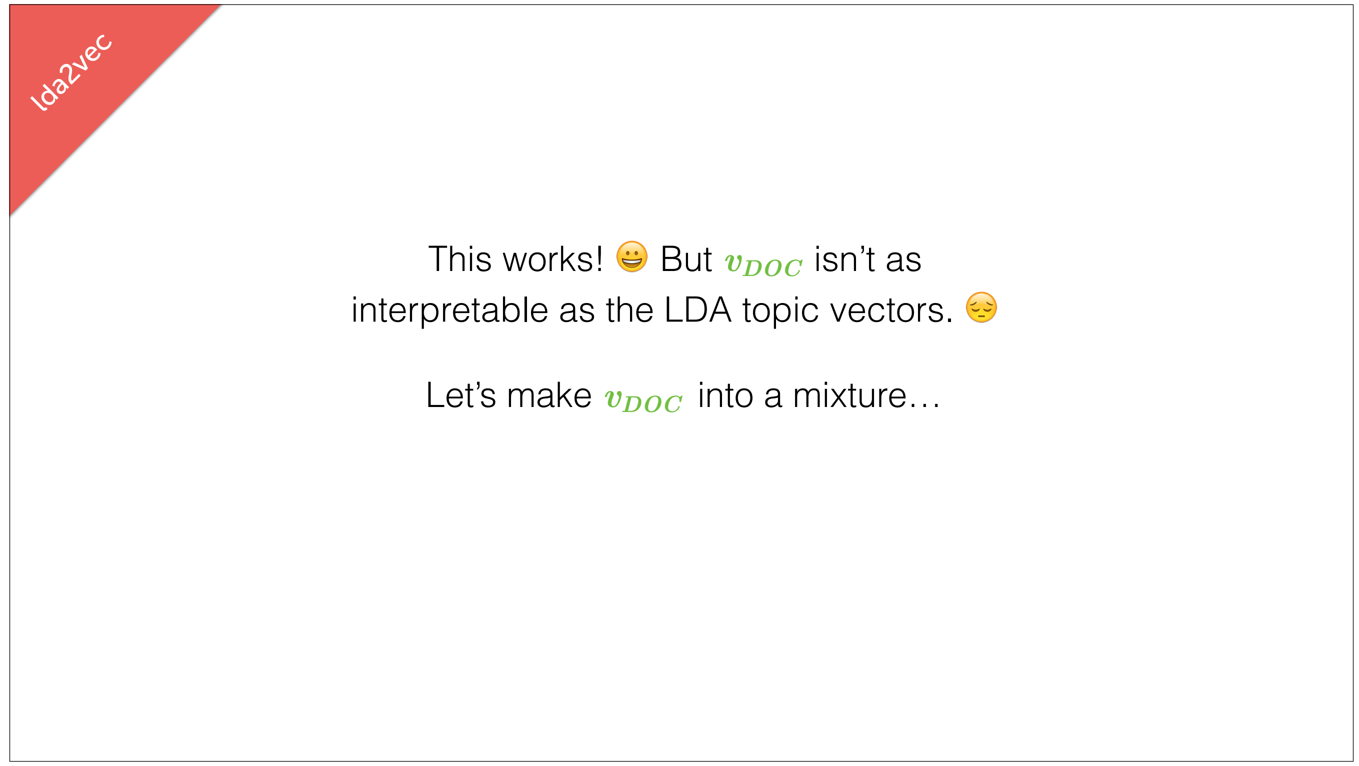
Too many documents. I really like that document X is 70% in topic 0, 30% in topic1, …

lda2vec

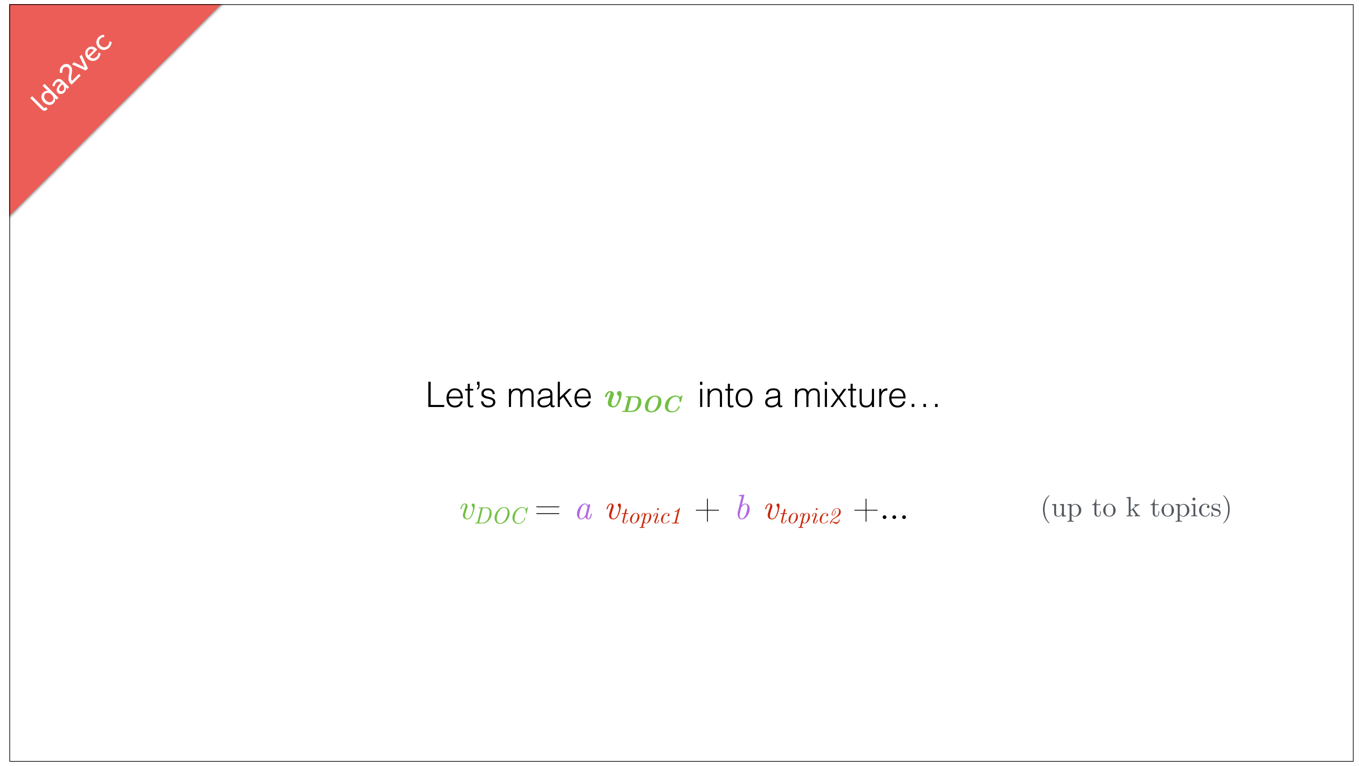
This works! 😀 But v_{DOC} isn't as interpretable as the LDA topic vectors. 😔

We're missing *mixtures* & *sparsity.*

Too many documents. I really like that document X is 70% in topic 0, 30% in topic1, …



Too many documents. I really like that document X is 70% in topic 0, 30% in topic1, …



sum of other word vectors

intuition here is that ‘hanoi = vietnam + capital’ and lufthansa = ‘germany + airlines’

so we think that document vectors should also be some word vector + some word vector

lda2vec

Let’s make v_{DOC} into a mixture…

Trinitarian
baptismal
Pentecostals
Bede
schismatics
excommunication

$$v_{DOC} = a\ v_{topic1} + b\ v_{topic2} + \ldots$$

twenty newsgroup dataset, free, canonical

lda2vec

$\mathbf{topic\ 1 = \text{"religion"}}$

Trinitarian
baptismal
Pentecostals
Bede
schismatics
excommunication

Let's make v_{DOC} into a mixture…

$$v_{DOC} = a\ v_{topic1} + b\ v_{topic2} + \ldots$$

lda2vec

Let's make v_{DOC} into a mixture…

topic 1 = "religion"

Trinitarian
baptismal
Pentecostals
Bede
schismatics
excommunication

$$v_{DOC} = a \; v_{topic1} + b \; v_{topic2} + \ldots$$

Milosevic
absentee
Indonesia
Lebanese
Isrealis
Karadzic

lda2vec

Let's make v_{DOC} into a mixture…

$$v_{DOC} = a\, v_{topic1} + b\, v_{topic2} + \ldots$$

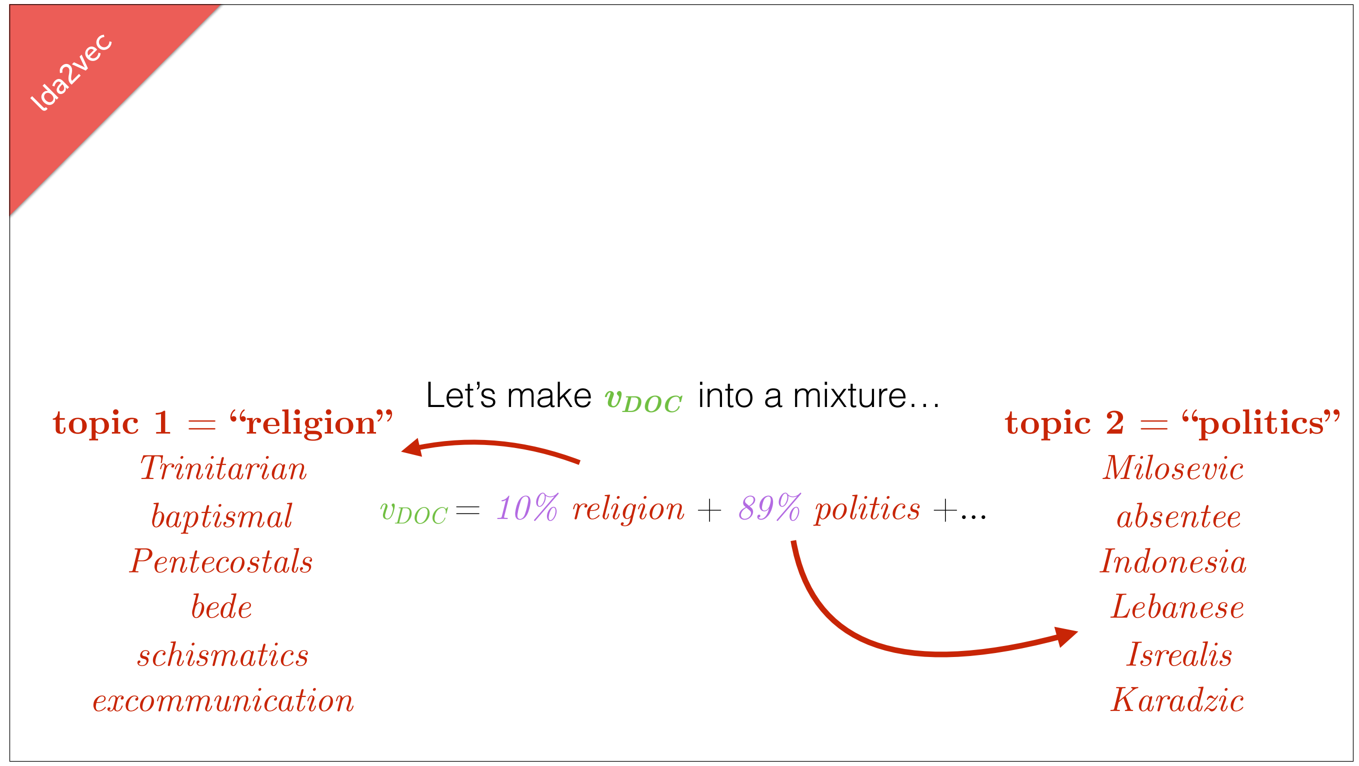
topic 1 = "religion"

Trinitarian
baptismal
Pentecostals
bede
schismatics
excommunication

topic 2 = "politics"

Milosevic
absentee
Indonesia
Lebanese
Isrealis
Karadzic

purple a,b coefficients tell you how much it is that topic



Doc is now 10% religion 89% politics

mixture models are powerful for interpretability

lda2vec

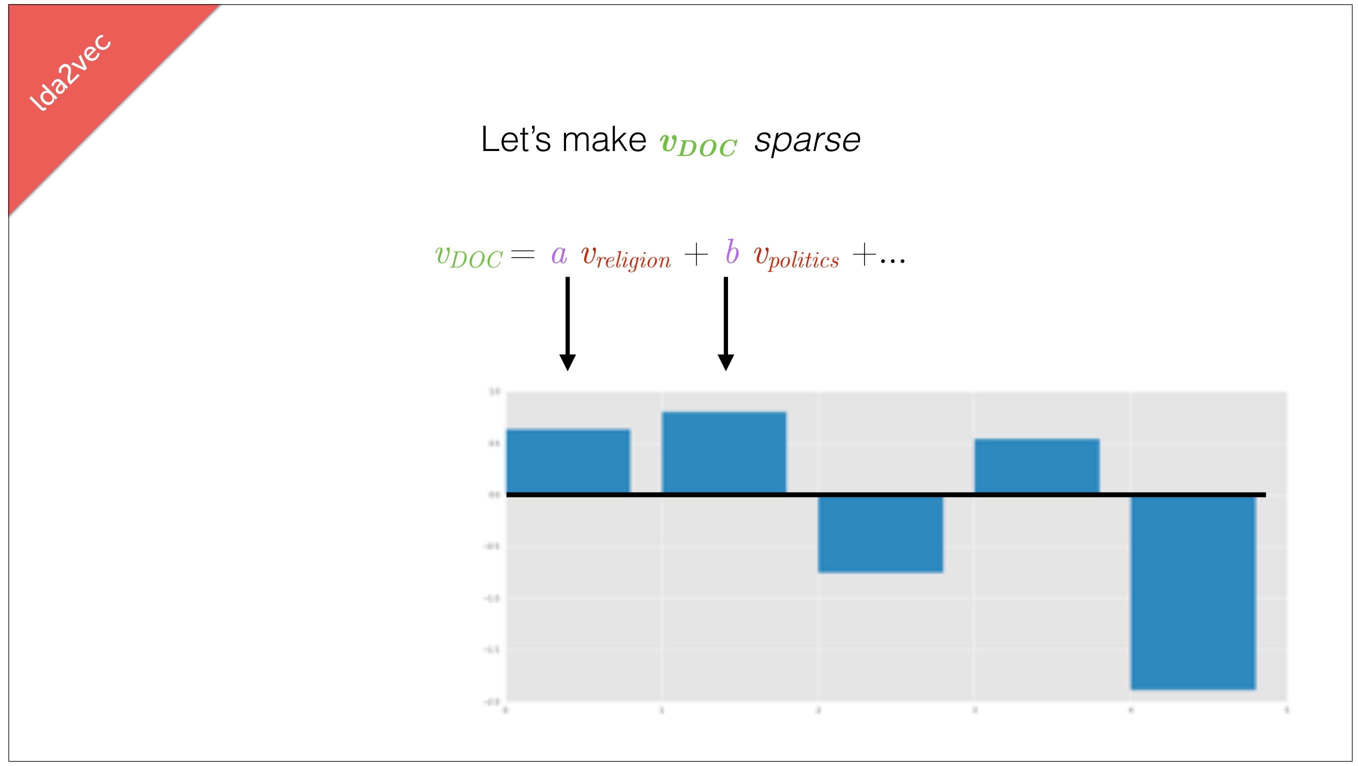
Let's make v_{DOC} *sparse*

$$v_{DOC} = a\ v_{religion} + b\ v_{politics} + \ldots$$

[-0.75, -1.25, …]

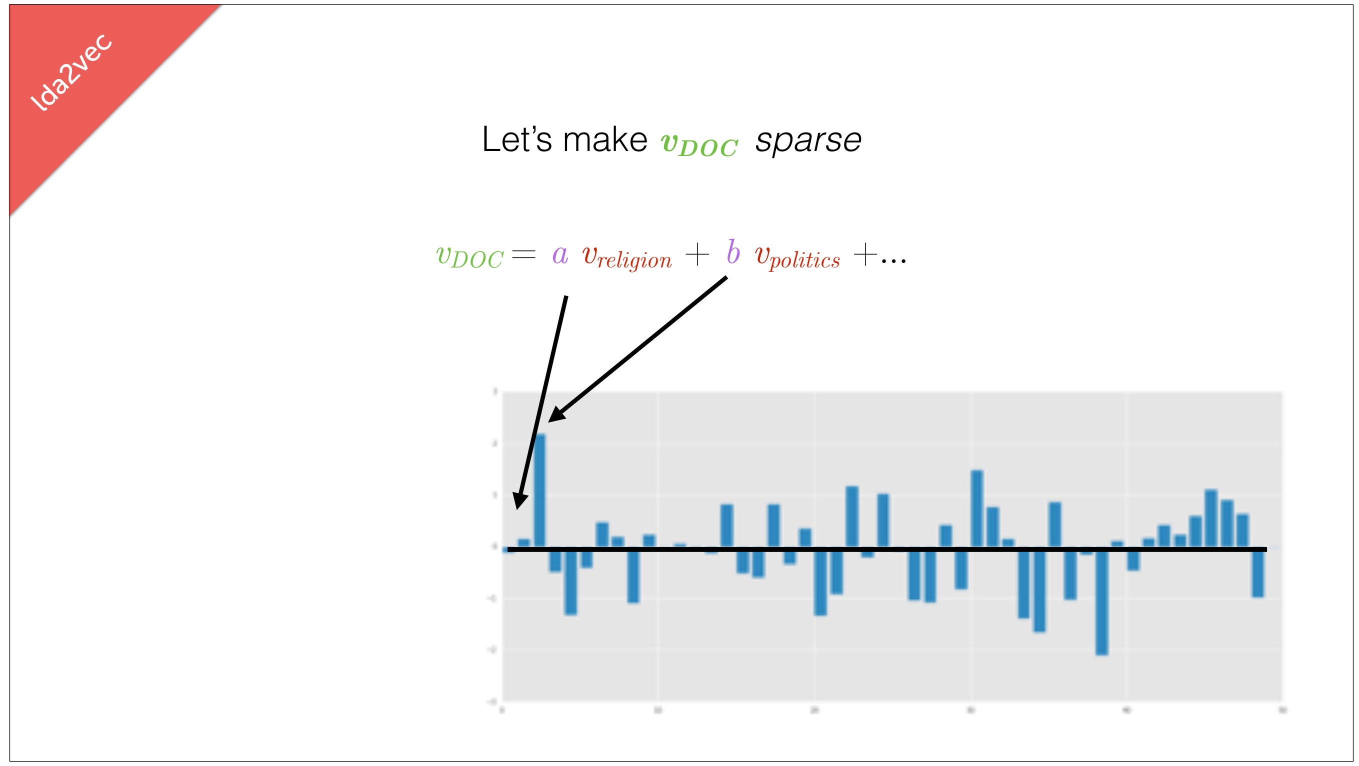
Now 1st time I did this…

Hard to interpret. What does -1.2 politics mean? math works, but not intuitive



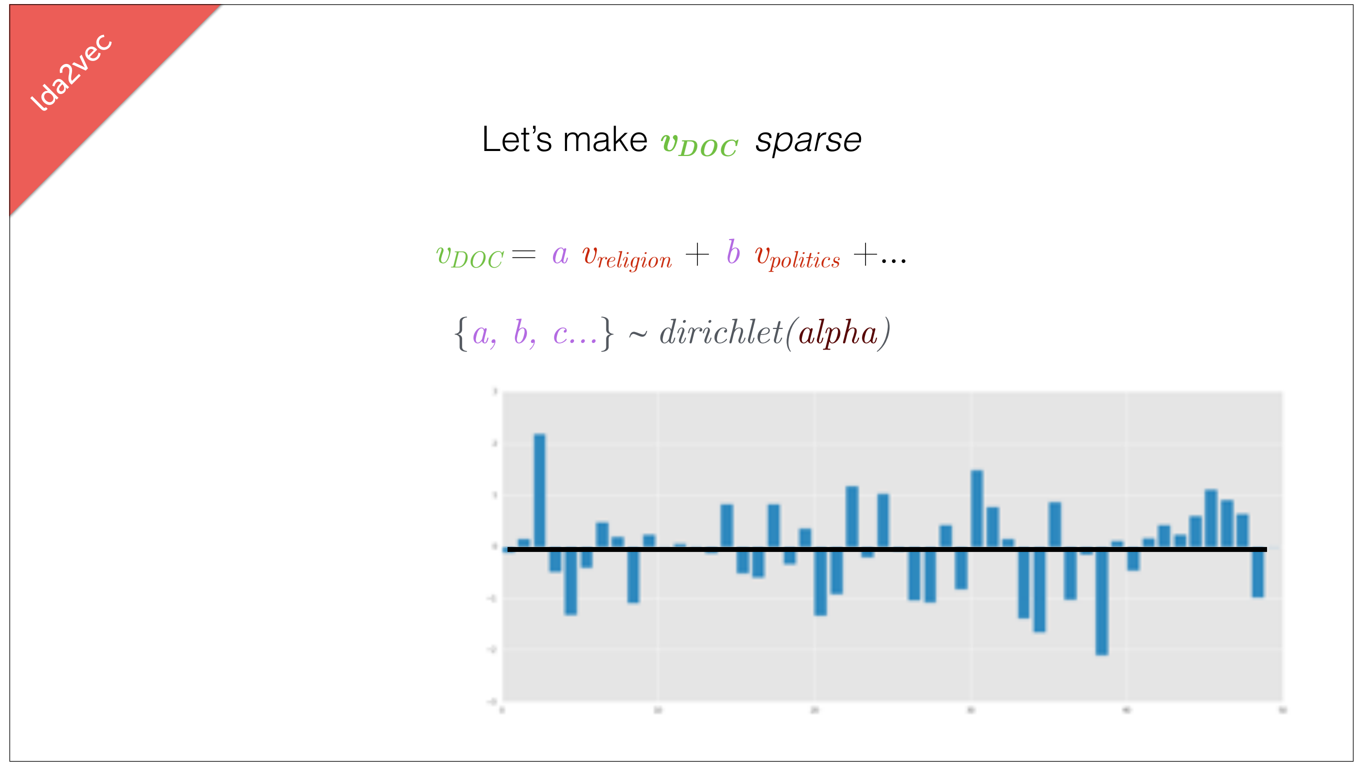
How much of this doc is in religion, how much in poltics

but doesn't work when you have more than a few



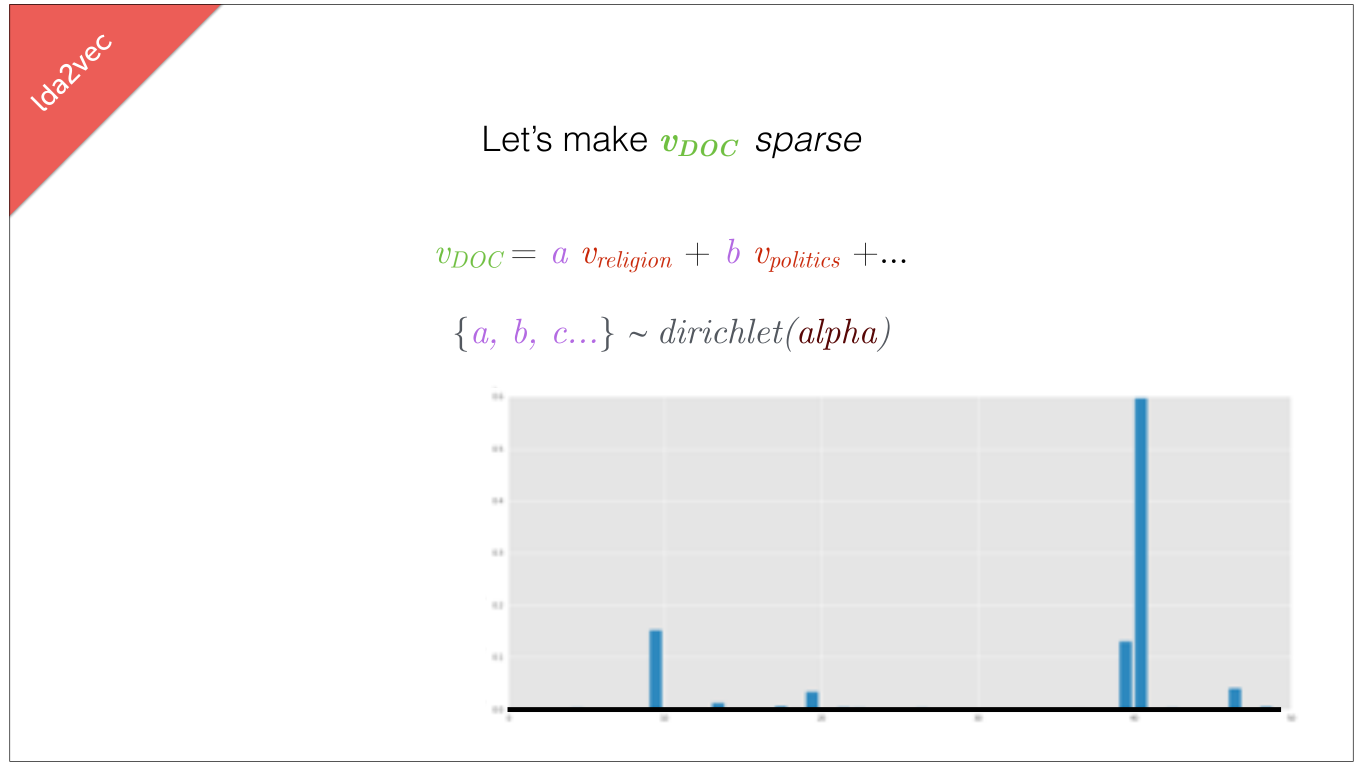
How much of this doc is in religion, how much in cars

but doesn't work when you have more than a few

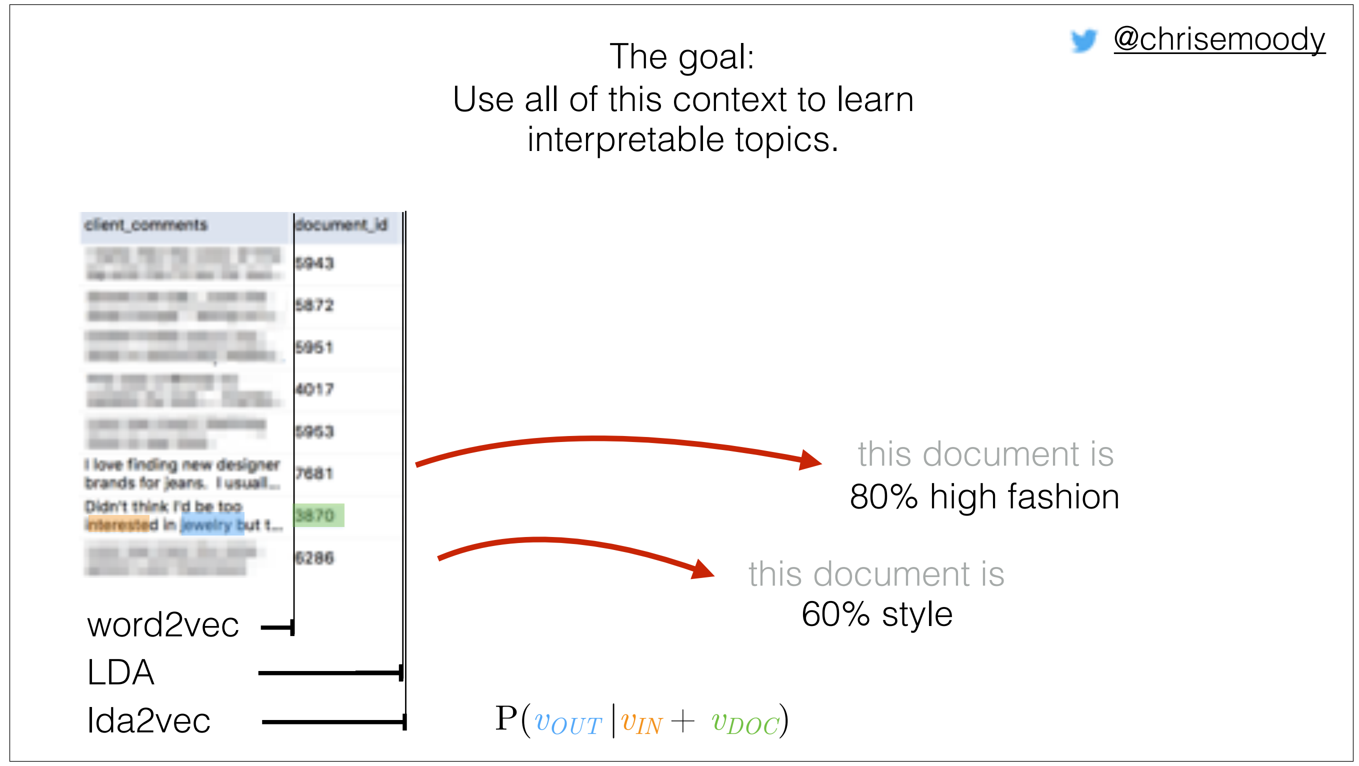


trick we can steal from bayesian

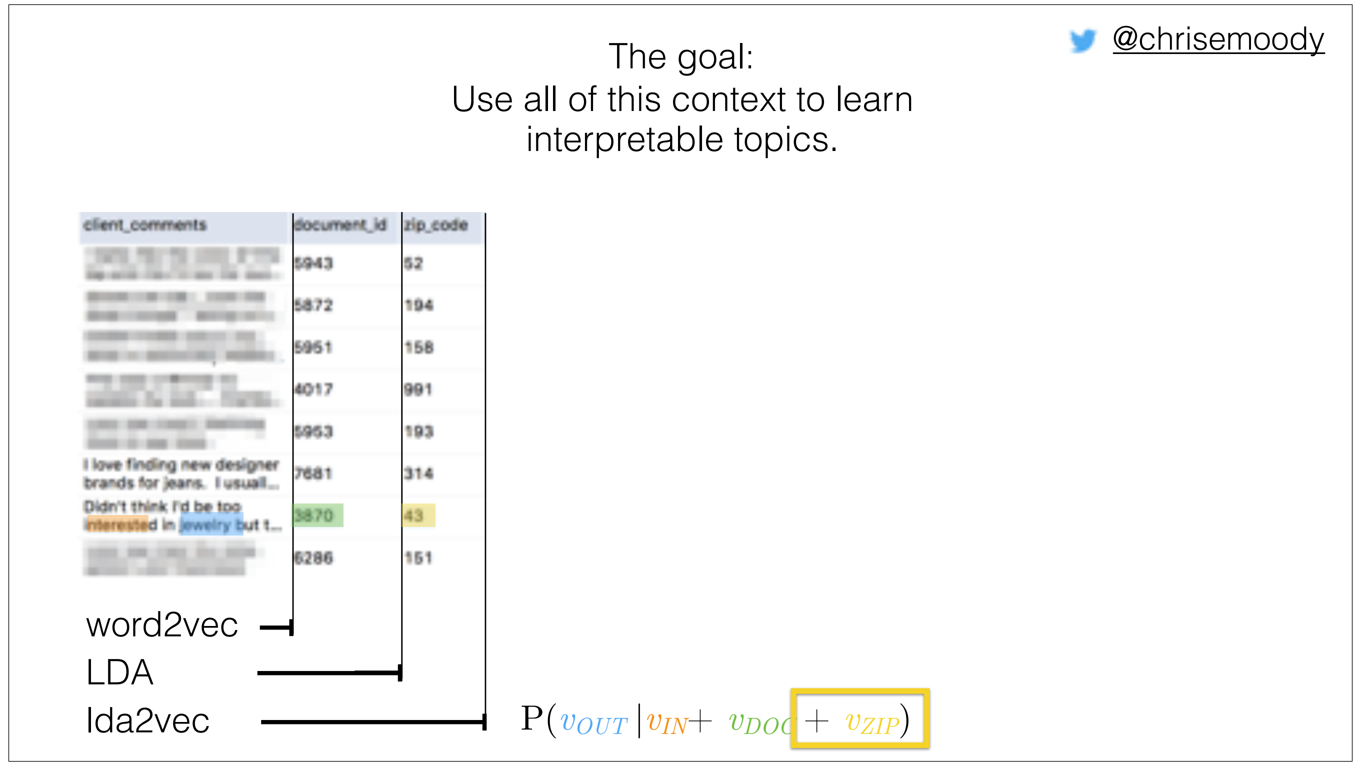
make it dirichlet
skipping technical details
make everything sum to 100%
penalize non-zero
force model to only make it non-zero w/ lots of evidence



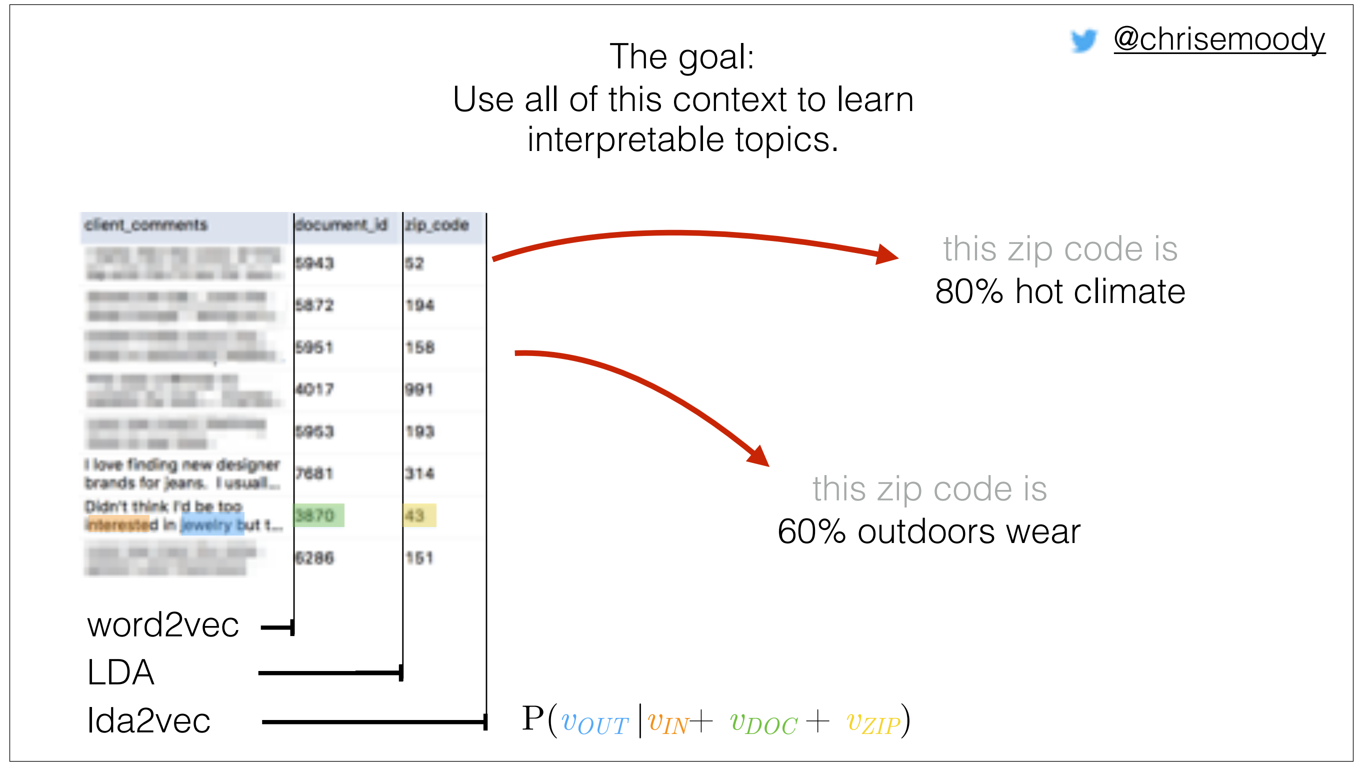
sparsity-inducing effect.
similar to the lasso or l1 reg, but dirichlet
few dimensions, sum to 100%
I can say to the CEO, set of docs could have been in 100 topics, but we picked only the best topics



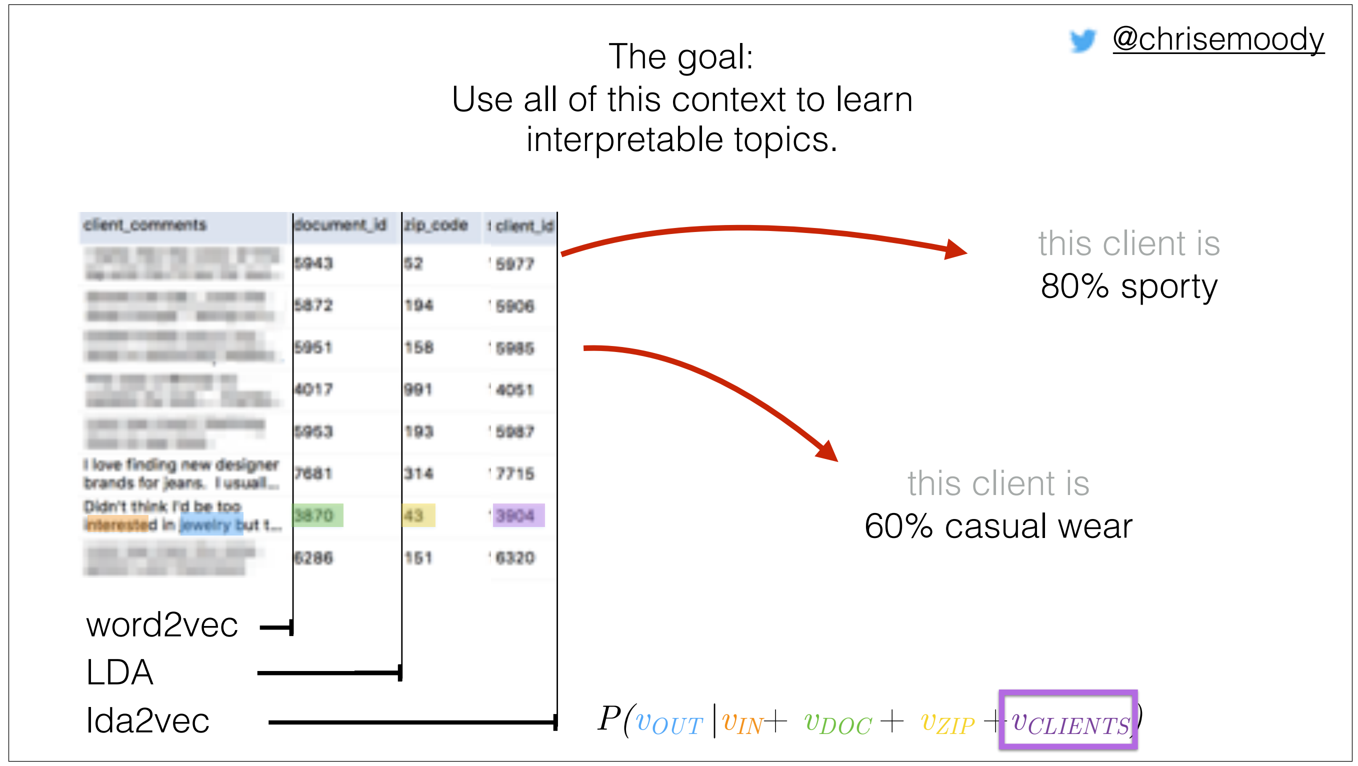
go back to our problem lda2vec is going to use all the info here



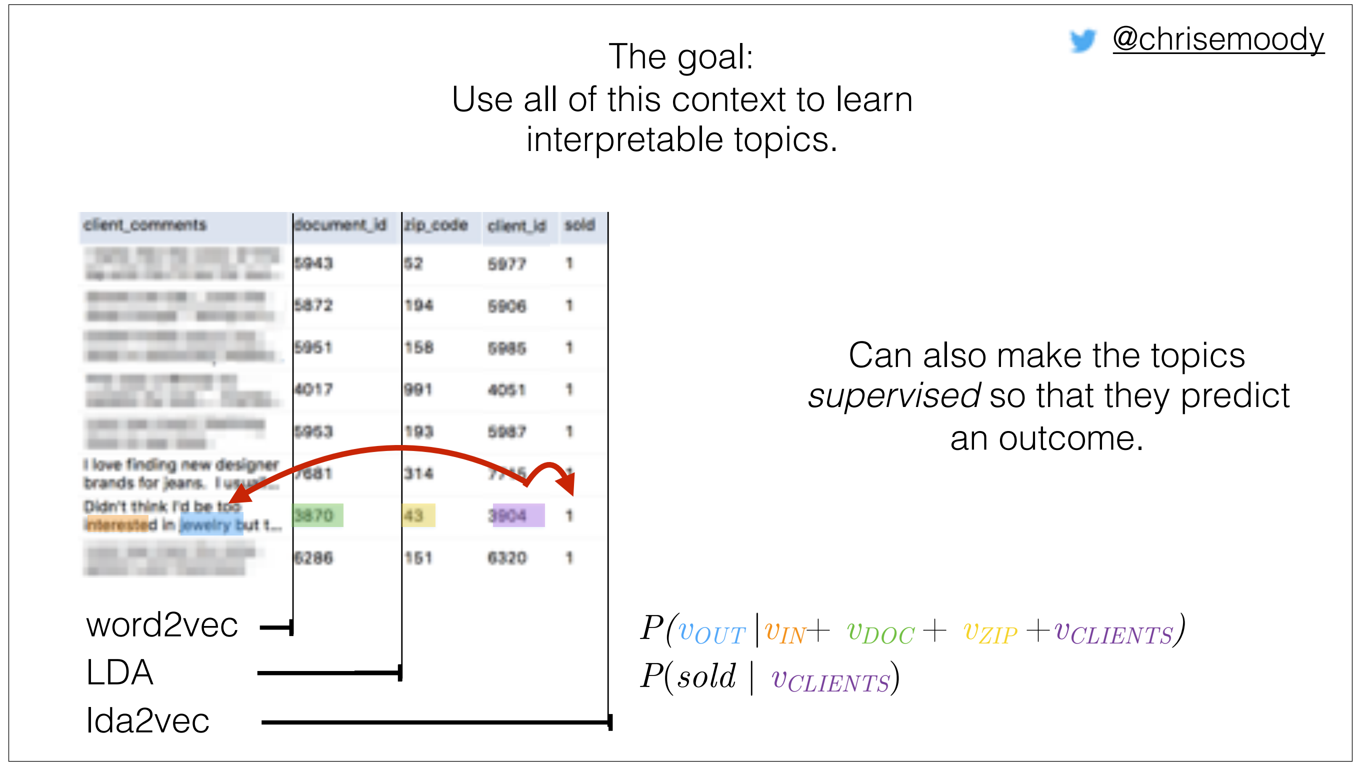
add column = adding a term
add features in an ML model



in addition to doc topics, like 'rec SF'



client topics — sporty, casual,
this is where if she says '3rd trimester' — identify a future mother
'scrubs' — medicine

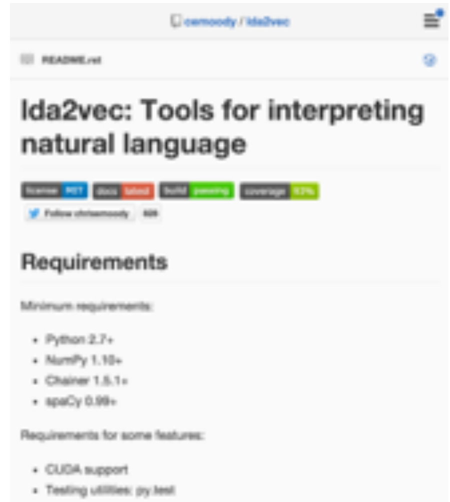


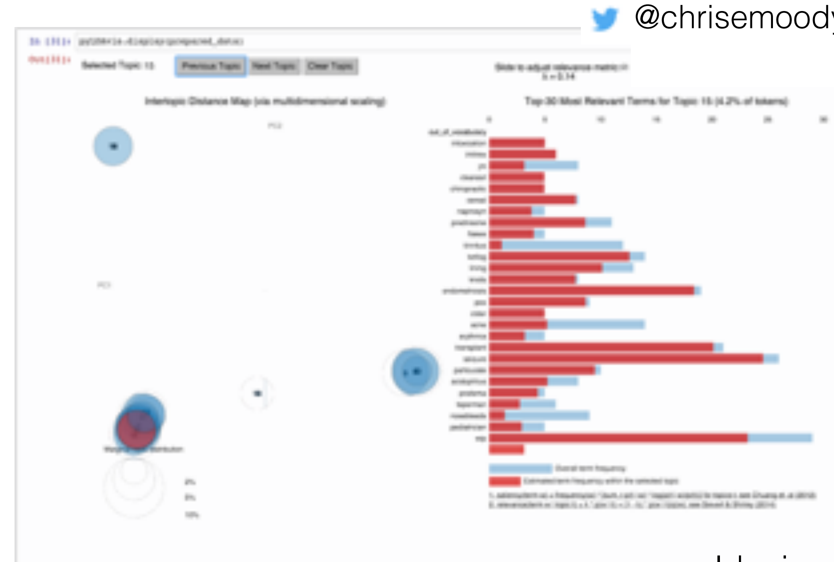
helps fine-tune topics so that correlate with your favorite business metric

align topics w/ expectations

helps us guess when revenue goes up what the leading causes are

@chrisemoody

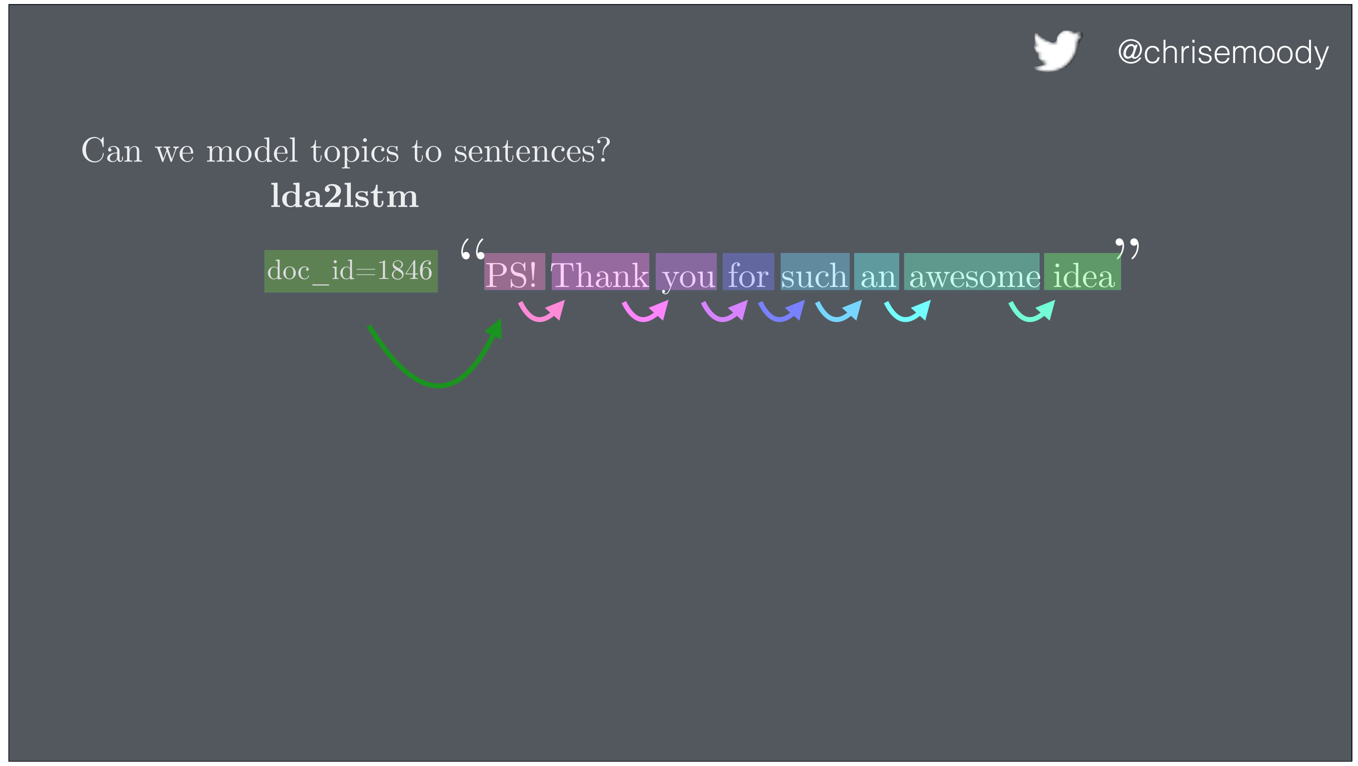




uses pyldavis

API Ref docs (no narrative docs)
GPU
Decent test coverage

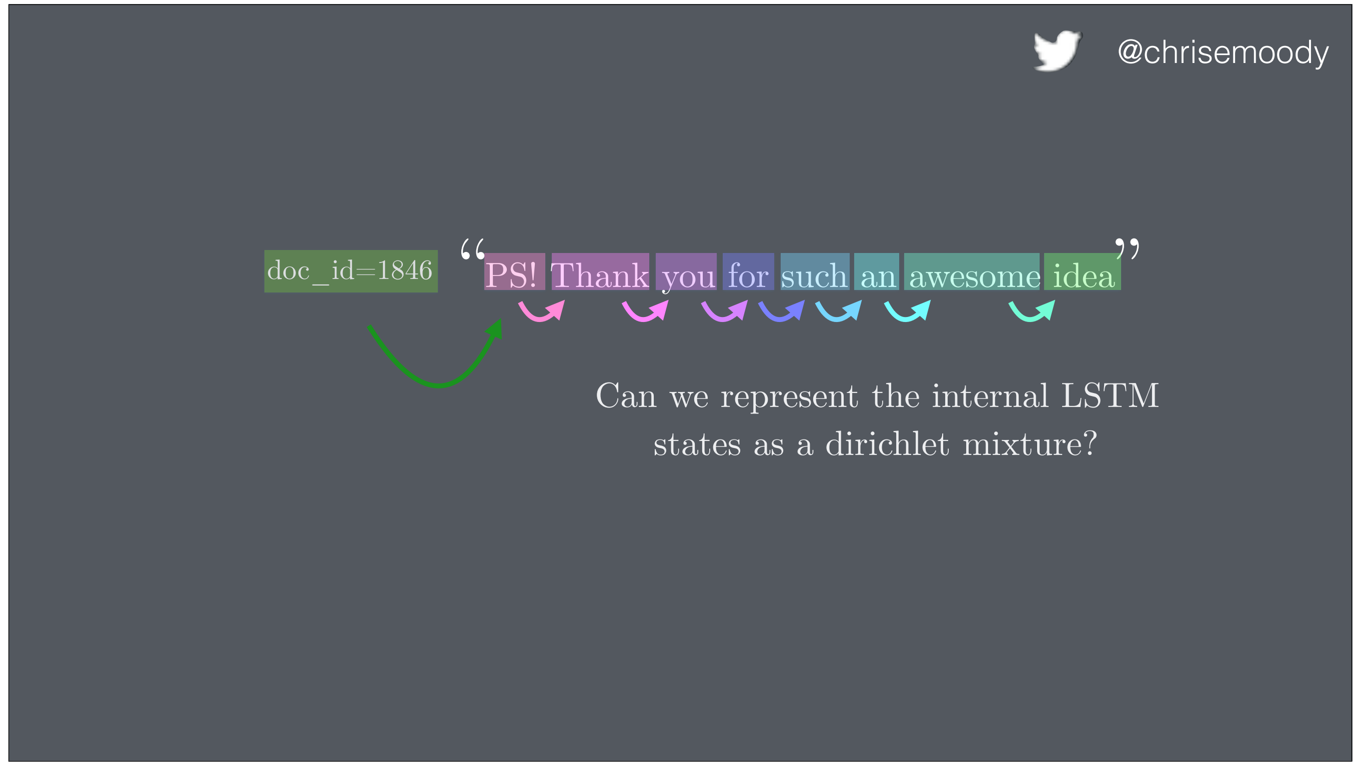
github.com/cemoody/lda2vec



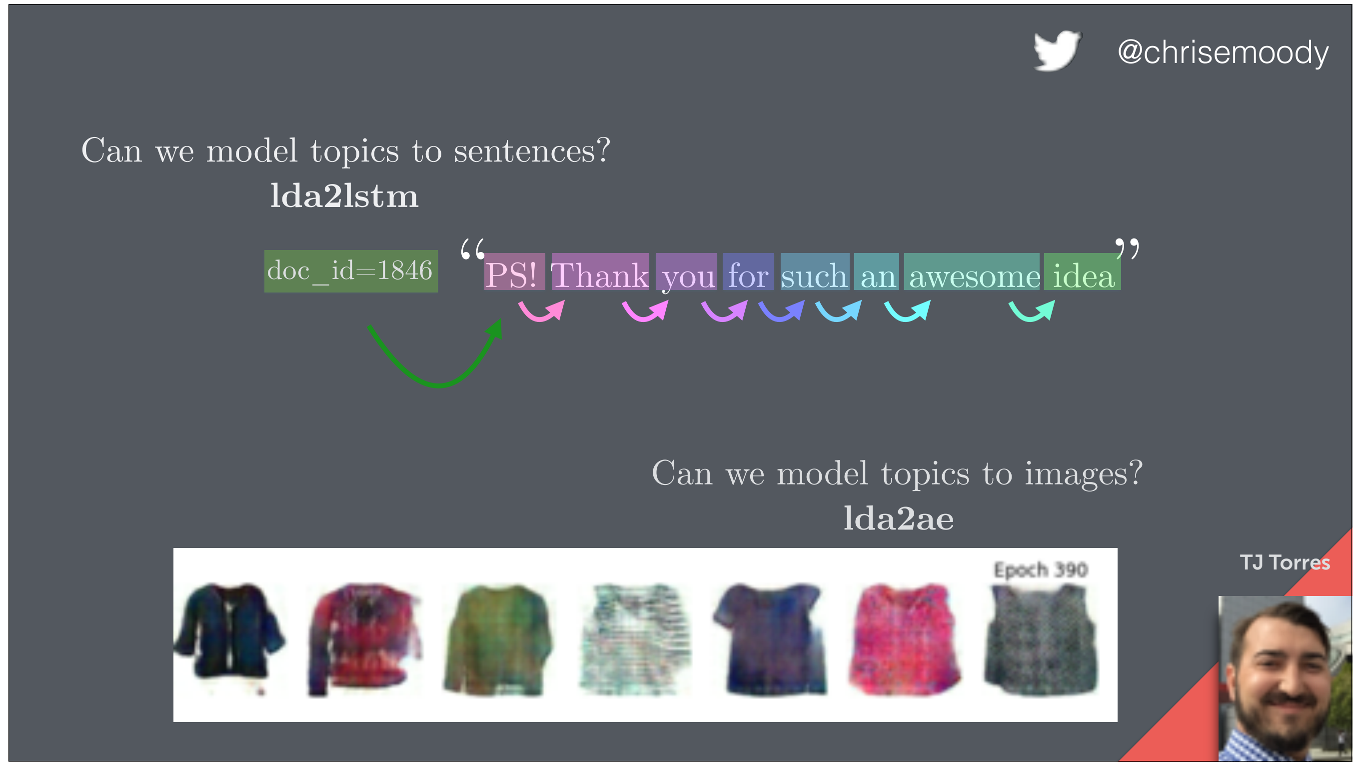
SF is all about mixing cutting edge algorithms but we absolutely need interpretability. human component to algos is not negotiable

Could we demand the model make us a sentence that is 80% religion, 10% politics?

classify word level, LSTM on sentence, LDA on document level



Dirichlet-squeeze internal states and manipulations, that maybe will help us understand the science of LSTM dynamics — because seriously WTF is going on there



Can we also extend this to image generation? TJ is working on a ridiculous VAE/GAN model… can we throw in a topic model? Can we say make me an image that is 80% sweater, and 10% zippers, and 10% elbow patches?

?

@chrisemoody

Multithreaded
Stitch Fix

Bonus slides

Paragraph Vectors

(Just extend the context window)

Content dependency

(Change the window grammatically)

Social word2vec (deepwalk)

(Sentence is a walk on the graph)

Spotify

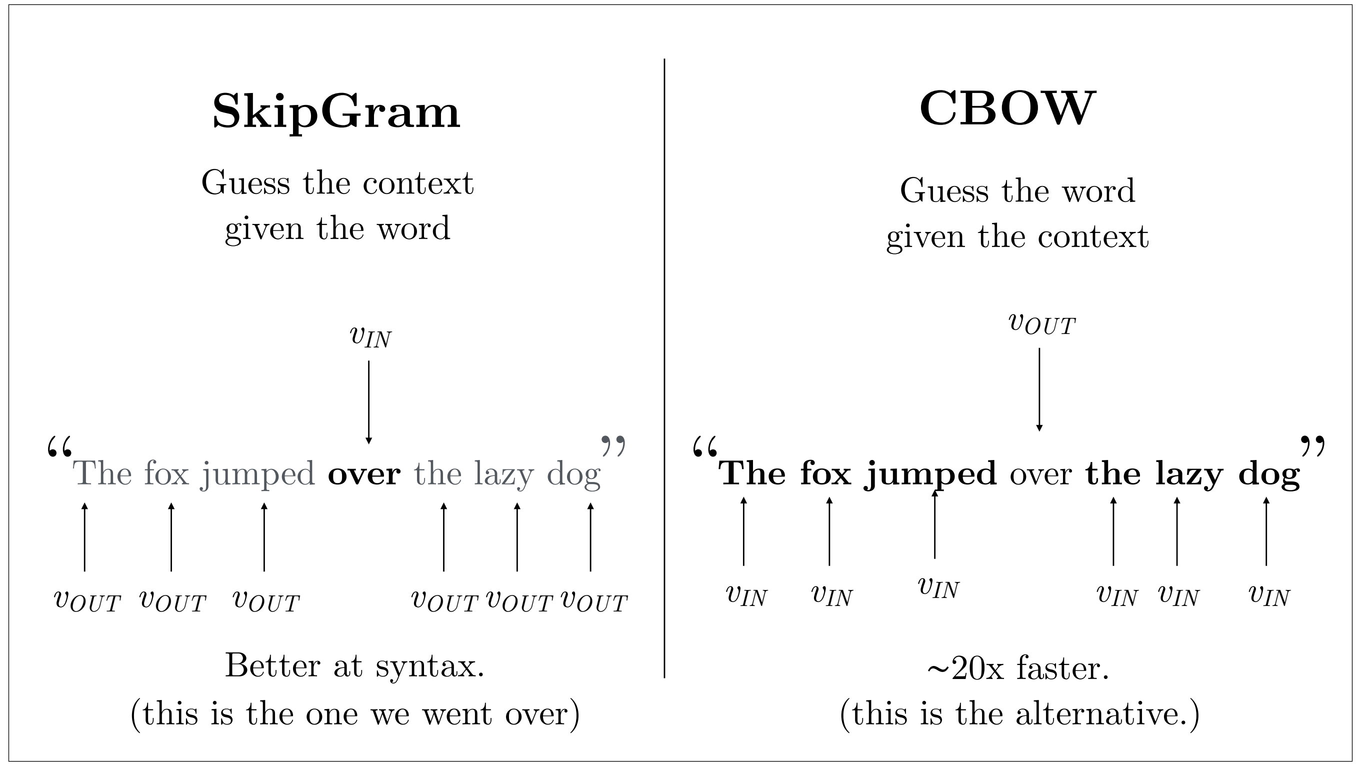
(Sentence is a playlist of song_ids)

Stitch Fix

(Sentence is a shipment of five items)

Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

See previous



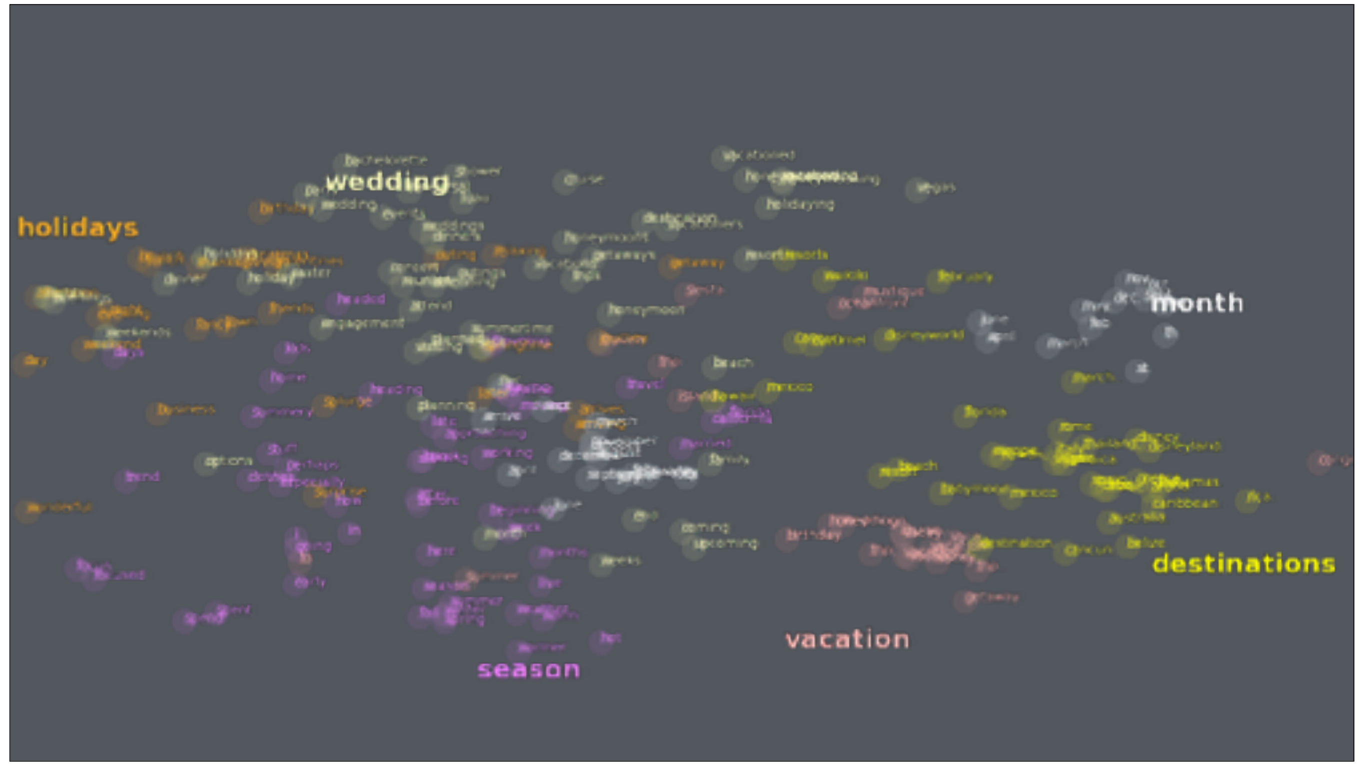
CBOW sums words vectors, loses the order in the sentence

Both are good at semantic relationships

- Child and kid are nearby
- Or gender in man, woman
- If you blur words over the scale of context — 5ish words, you lose a lot grammatical nuance

But skipgram preserves order

- Preserves the relationship in pluralizing, for example



Shows that are many words similar to vacation actually come in lots of flavors
— wedding words (bachelorette, rehearsals)
— holiday/event words (birthdays, brunch, christmas, thanksgiving)
— seasonal words (spring, summer,)
— trip words (getaway)
— destinations

LDA
Results

Great Stylist

Perfect

I loved every choice in this fix!! Great job!

History

LDA
Results

Body Fit

My measurements are 36-28-32. If that helps.
I like wearing some clothing that is fitted.
Very hard for me to find pants that fit right.

History

LDA
Results

Sizing

Excited for next

Really enjoyed the experience and the pieces, sizing for tops was too big. Looking forward to my next box!

History

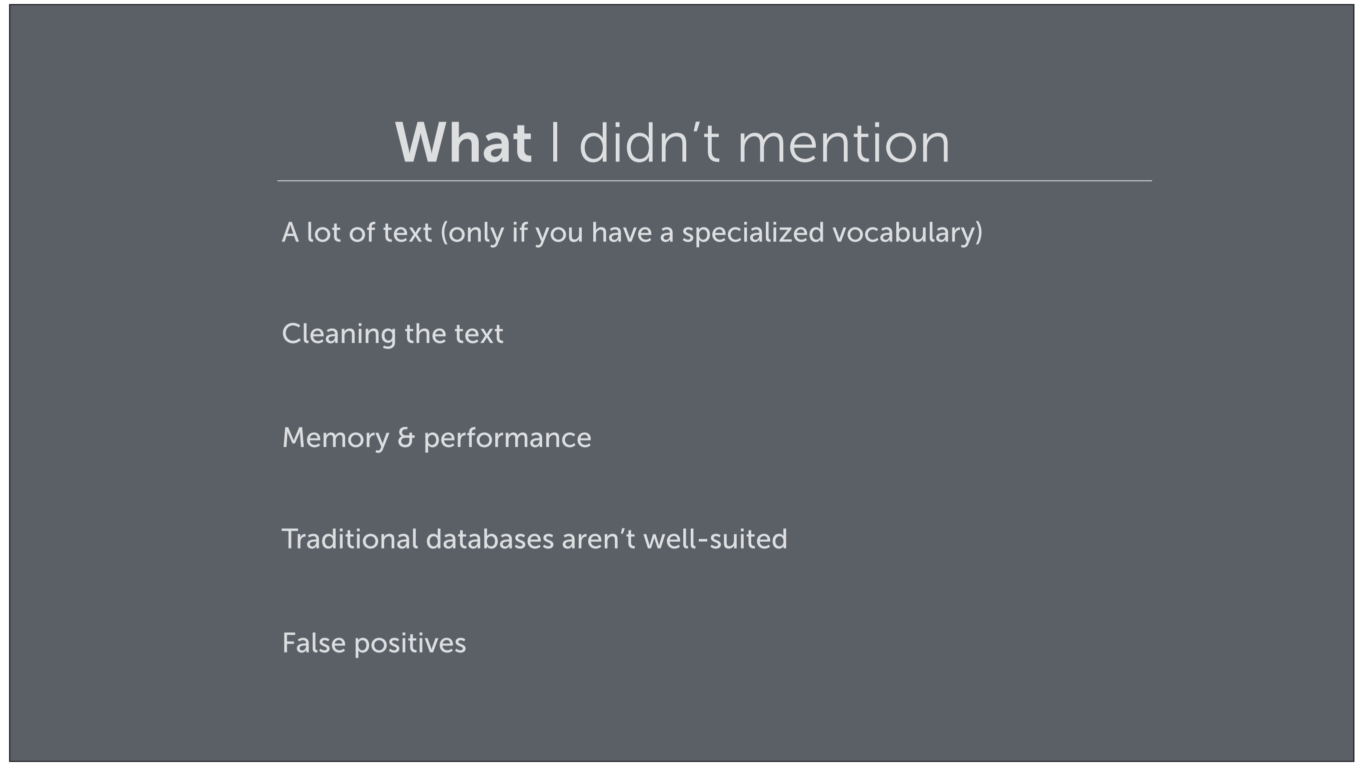
LDA
Results

Almost Bought

Perfect

It was a great fix. Loved the two items I kept and the three I sent back were close!

History



hundreds of millions of words, 1,000 books, 500,000 comments, or 4,000,000 tweets

high-memory and high-performance multicore machine.
Training can take several hours to several days but shouldn't need frequent retraining.

If you use pretrained vectors, then this isn't an issue.

Databases. Modern SQL systems aren't well-suited to performing the vector addition, subtraction and multiplication searching in vector space requires. There are a few libraries that will help you quickly find the most similar items12: annoy, ball trees, locality-sensitive hashing (LSH) or FLANN.

False-positives & exactness. Despite the impressive results that come with word vectorization, no NLP technique is perfect. Take care that your system is robust to results that a computer deems relevant but an expert human wouldn't.

and now for something **completely crazy**

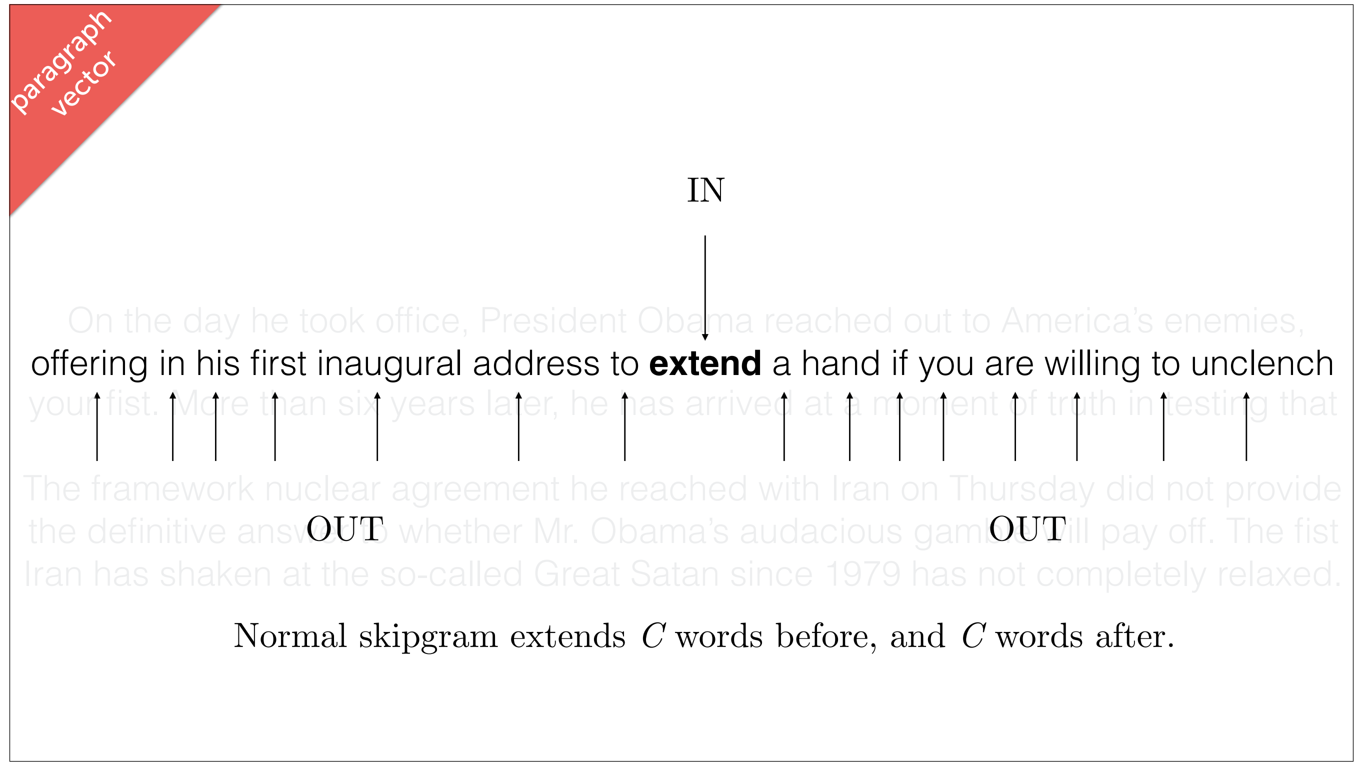
All of the following ideas will change what 'words' and 'context' represent.

But we'll still use the same w2v algo

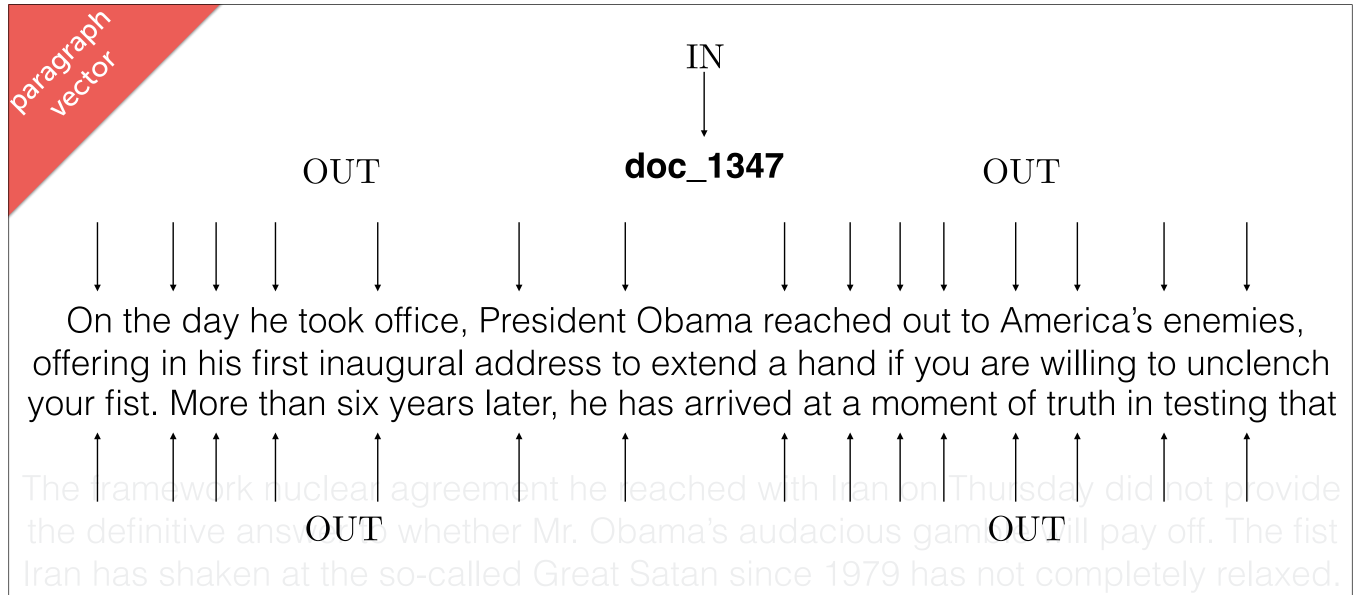
paragraph vector

What about summarizing documents?

On the day he took office, President Obama reached out to America's enemies, offering in his first inaugural address to **extend** a hand if you are willing to unclench your fist. More than six years later, he has arrived at a moment of truth in testing that



Except we stay inside a sentence

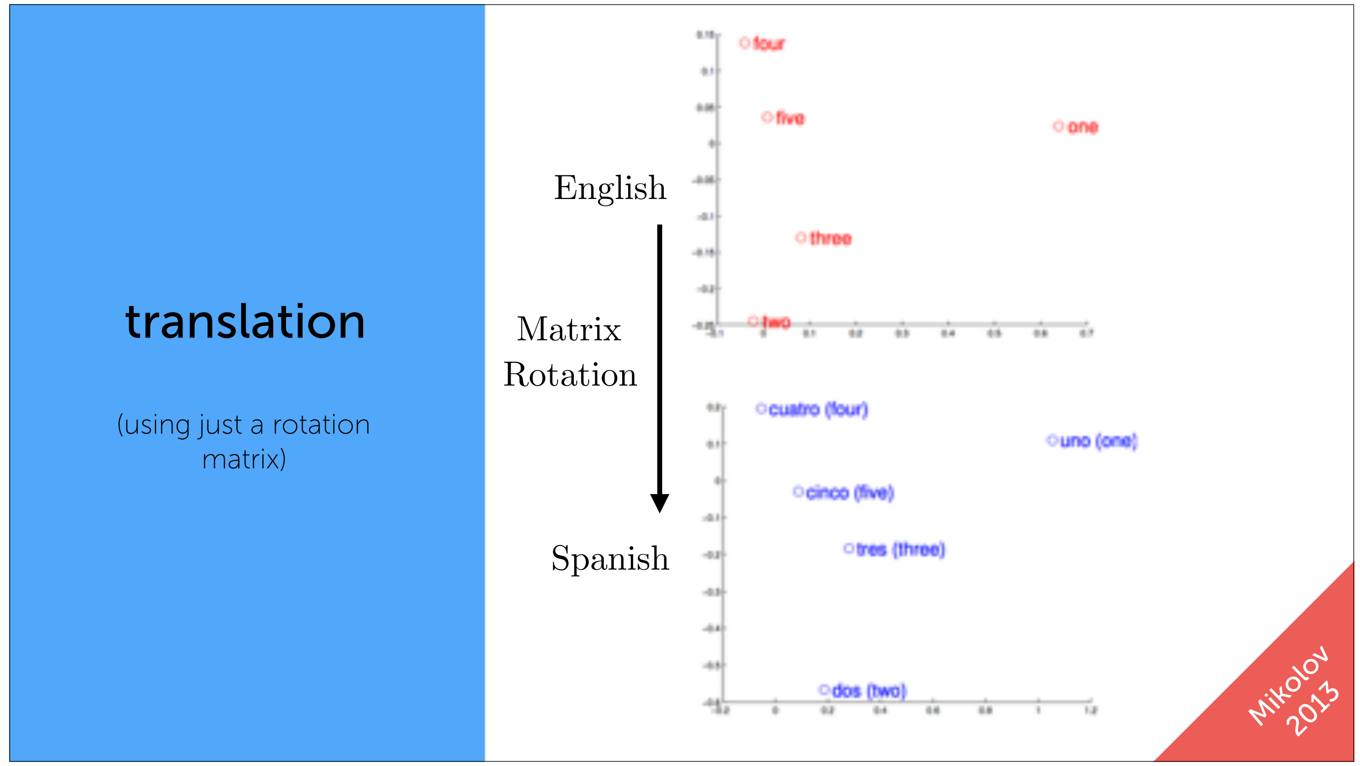


A document vector simply extends the context to the whole document.

```
from gensim.models import Doc2Vec
fn = “item_document_vectors”
model = Doc2Vec.load(fn)
model.most_similar('pregnant')
matches = list(filter(lambda x: 'SENT_' in x[0], matches))

# ['...I am currently 23 weeks pregnant...',
#  '...I'm now 10 weeks pregnant...',
#  '...not showing too much yet...',
#  '...15 weeks now. Baby bump...',
#  '...6 weeks post partum!...',
#  '...12 weeks postpartum and am nursing...',
#  '...I have my baby shower that...',
#  '...am still breastfeeding...',
#  '...I would love an outfit for a baby shower...']
```

sentence search



Blows my mind

Explain plot

Not a complicated NN here

Still have to learn the rotation matrix — but it generalizes very nicely.

Have analogies for every linalg op as a linguistic operator: + and - and matrix multiplies

Robust framework and new tools to do science on words

context dependent

Australian scientist **discovers** star with telescope

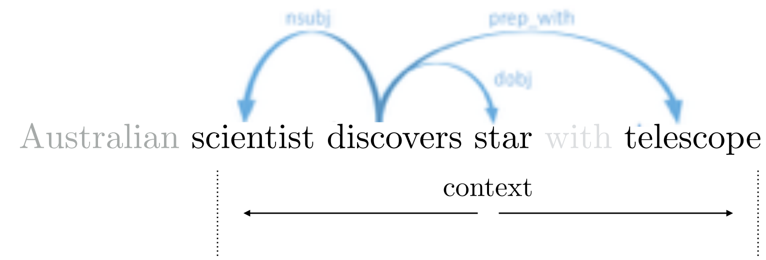
context +/- 2 words

Levy & Goldberg 2014


context dependent
nsubj
prep_with
dobj
Australian scientist discovers star with telescope
Levy & Goldberg 2014

What if we

context dependent



Levy & Goldberg 2014

context dependent

	BoW	DEPS
hogwarts	dumbledore	sunnydale
	hallows	collinwood
	half-blood	calarts
	malfoy	greendale
	snape	millfield

topically-similar vs 'functionally' similar

Levy & Goldberg 2014

context dependent

Also show that SGNS is simply factorizing:

$$w * c = PMI(\text{w, c}) - \log \text{k}$$

This is **completely** amazing!

Intuition: positive associations (canada, snow) stronger in humans than negative associations (what is the opposite of Canada?)

Levy & Goldberg 2014

Also means we can do SVD-like techniques to get a convex w2v, uses fast lining libs, uses compressed word count matrix so also better storage…. but not online

word2vec

learn word vectors from sentences

“The fox jumped over **the** lazy dog”

v_{OUT} v_{OUT} v_{OUT} v_{OUT} v_{OUT} v_{OUT}

deepwalk

‘words’ are graph vertices
‘sentences’ are random walks on the graph

$$v_{46} \rightarrow v_{45} \rightarrow v_{71} \rightarrow v_{24} \rightarrow v_5$$

Perozzi et al 2014

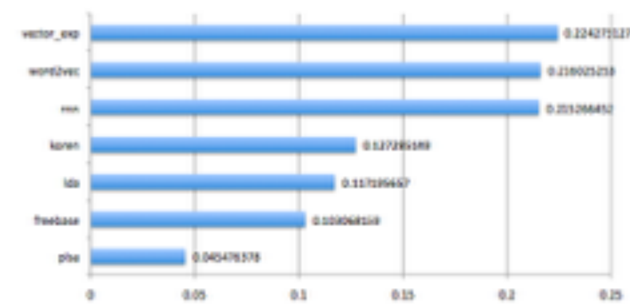
Playlists at Spotify

‘words’ are songs
‘sentences’ are playlists

sequence learning

Playlists at Spotify

Great performance on ‘related artists’



Erik Bernhardsson

Fixes at Stitch Fix

Let's try:
'words' are styles
'sentences' are fixes

sequence learning

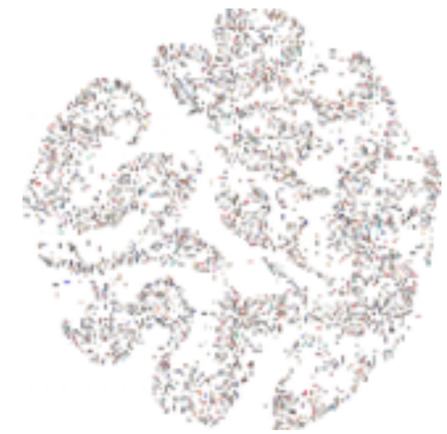
Fixes at Stitch Fix

Learn similarity between styles because they co-occur

Learn 'coherent' styles

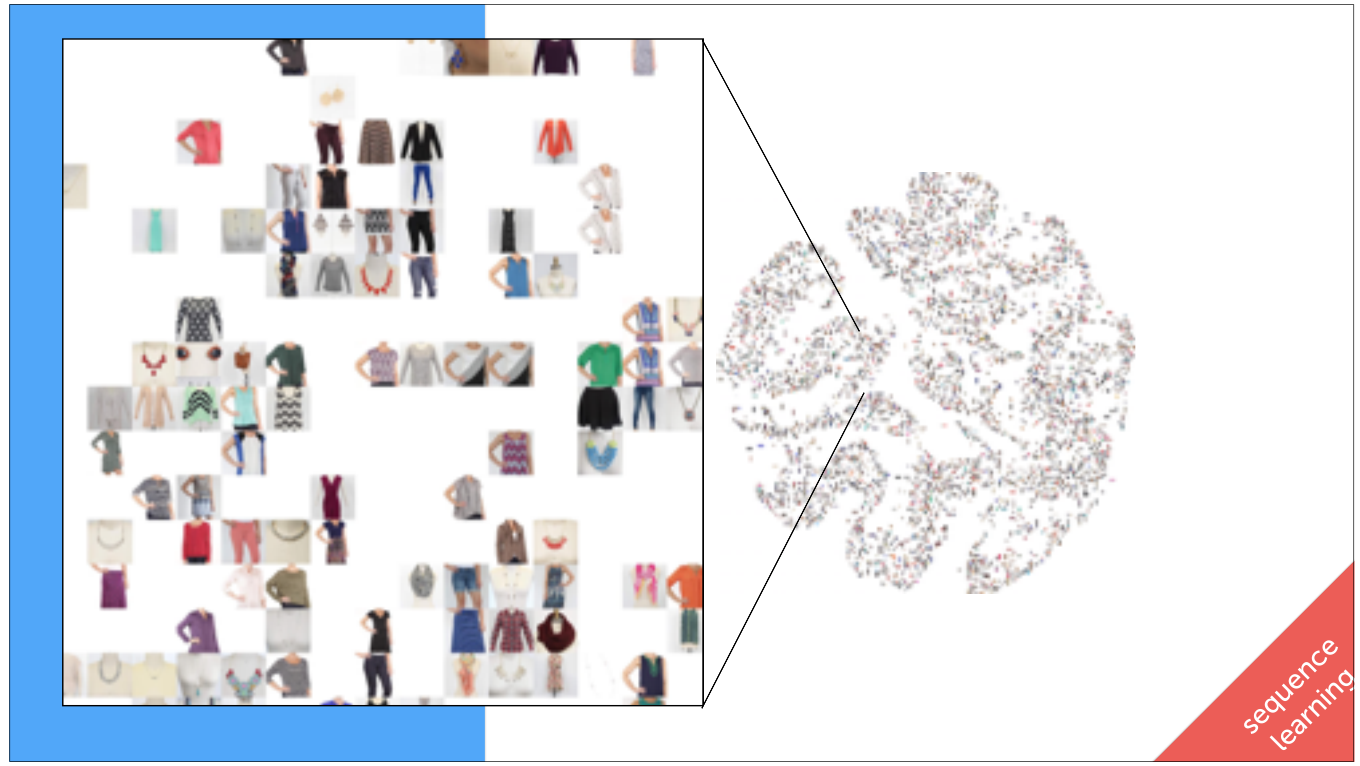
sequence learning

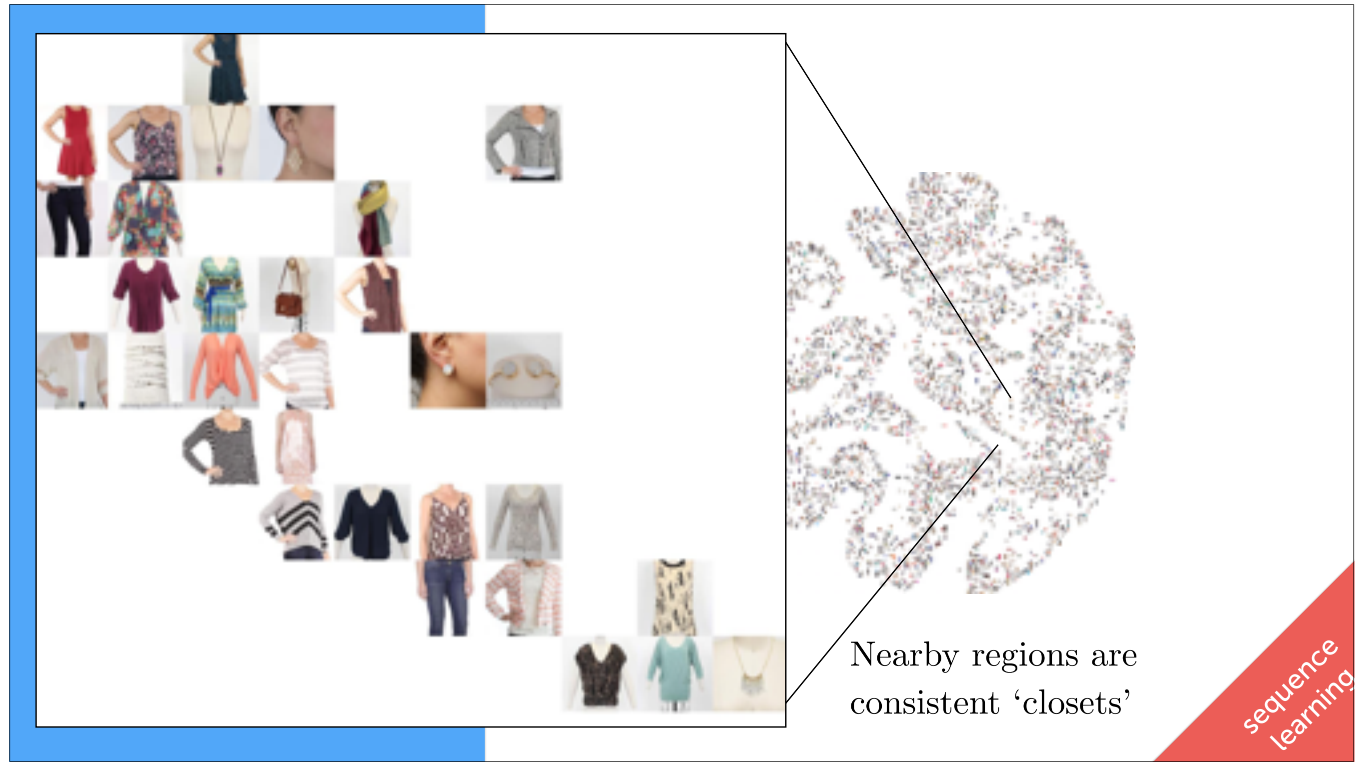
Fixes at Stitch Fix?



Got lots of structure!

sequence learning


sequence
learning


Nearby regions are
consistent 'closets'
sequence
learning

A specific lda2vec model

Our text blob is a comment that comes from a region_id and a style_id

$$L = \sigma(c * w) + \sigma(-c * w_{neg})$$

$$context = \vec{c_{ij}} = \vec{region_i} + \vec{style_j}$$

$$\vec{region_i} = \Sigma_{k=0}^{n\_topics} u_{ik} \cdot \vec{m_k}$$

$$\vec{style_j} = \Sigma_{l=0}^{n\_topics} u_{jl} \cdot \vec{n_l}$$

$$\vec{u} \sim dirichlet(\alpha_1)$$

$$\vec{v} \sim dirichlet(\alpha_2)$$

$$take\_rate\_in\_region \sim 5.0 * \sigma(W \cdot \vec{u})$$

The full likelihood model

$$L = \sigma(c * w) + \sigma(-c * w_{neg})$$

$$context = \vec{c_{ij}} = \vec{region}_i + \vec{style}_j$$

$$\vec{region}_i = \Sigma_{k=0}^{n\_topics} u_{ik} \cdot \vec{m_k}$$

$$\vec{style}_j = \Sigma_{l=0}^{n\_topics} u_{jl} \cdot \vec{n_l}$$

$$\vec{u} \sim dirichlet(\alpha_1)$$

$$\vec{v} \sim dirichlet(\alpha_2)$$

$$take\_rate\_in\_region \sim 5.0 * \sigma(W \cdot \vec{u})$$

$$L = \sigma(c * w) + \sigma(-c * w_{neg})$$

First part of the loss function is given context predict word.

Don't predict a negative word. These are words that are in our vocabulary somewhere, but not in our example.

We get negative samples **not** uniformly, but proportional to the word frequency^¾ (yes, the ¾ power is weird and ad hoc but totally works awesomely for word2vec)

$$L = \sigma(c \cdot w) + \sigma(-c \cdot w_{neg})$$

$$context = \vec{c_{ij}} = \vec{region_i} + \vec{style_j}$$

Context is made up from more than one part -- many 'contexts' available.

In this case, instead of one document, we can have many regions, or styles.

In LDA, this context is a single term: the latent document vector that 'generates' words.

In word2vec, this context is the 'pivot' word. Word2vec picks a random 'context' word in the corpus, centers a window around it, and tries to predict other words within that context.

In both word2vec and LDA context is one term, either a document or a word. For lda2vec, we can more than one term, we can have as many contexts as we like!

$$L = \sigma(c \cdot w) + \sigma(-c \cdot w_{neg})$$

$$context = \vec{c_{ij}} = \vec{region_i} + \vec{style_j}$$

$$\vec{region_i} = \Sigma_{k=0}^{n\_topics} u_{ik} \cdot \vec{m_k}$$

$$\vec{style_j} = \Sigma_{l=0}^{n\_topics} u_{jl} \cdot \vec{n_l}$$

Each context (e.g., region or style) is decomposed into topics vectors and weights on those common topics vectors. One context has one shared set of topic vectors (think of these as cluster centroids) and every 'document' in that context (think of 1 of 50 states, 1 of 20k styles) has a weight/membership onto each of those topic vectors (think topics like northeast, midwest for region or tops, bottoms, boho, romantic for style topics)

This forces the context vectors onto **a limited set of basis vectors**. Interpret this set, and you can generalize what each region vector and style vector means. For example, one topics vector might be close to the word vector for 'hand_bag', 'purse', 'bag' indicating that that topic is a handbags topic. And then anything with big weight in that topic might be a handbag.

$$L = \sigma(c \cdot w) + \sigma(-c \cdot w_{neg})$$

$$context = \vec{c_{ij}} = \vec{region_i} + \vec{style_j}$$

$$\vec{region_i} = \Sigma_{k=0}^{n\_topics} u_{ik} \cdot \vec{m_k}$$

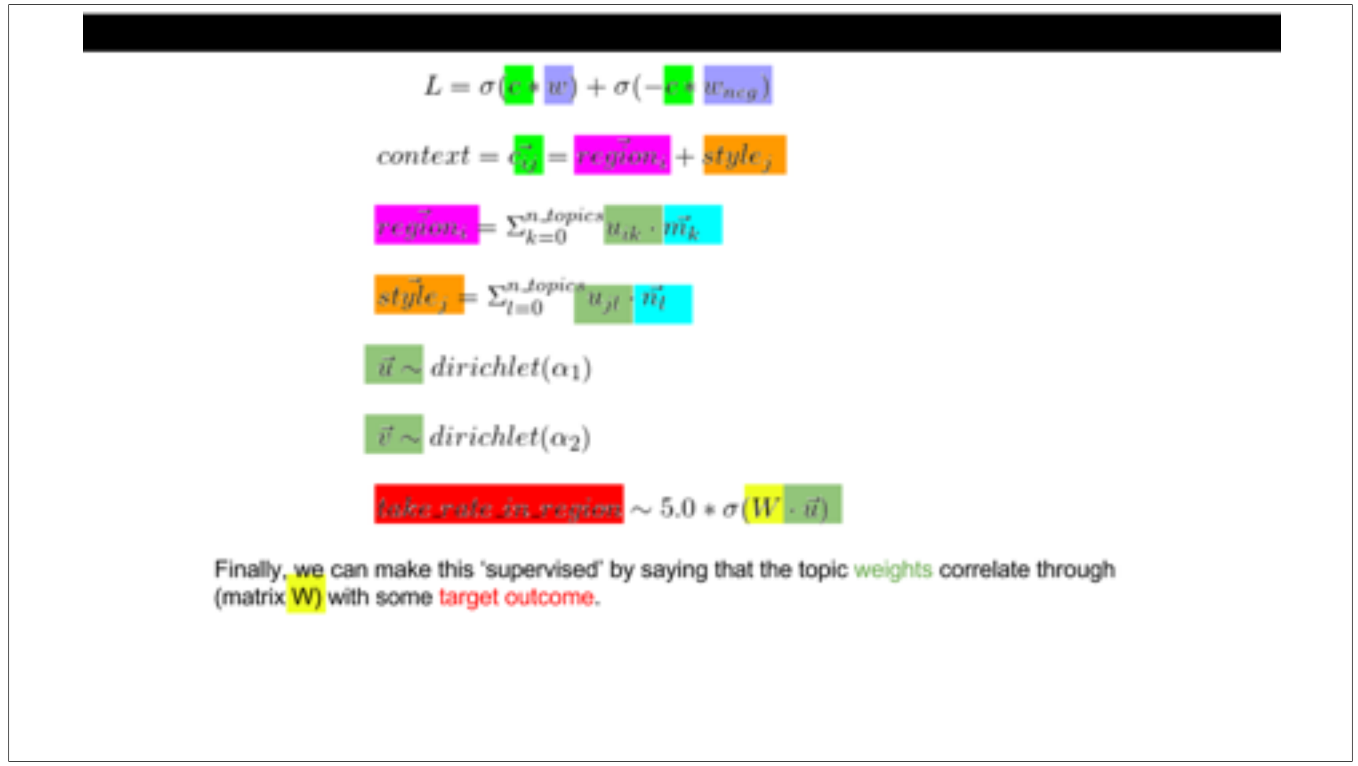
$$\vec{style_j} = \Sigma_{l=0}^{n\_topics} u_{jl} \cdot \vec{n_l}$$

$$\vec{u} \sim dirichlet(\alpha_1)$$

$$\vec{v} \sim dirichlet(\alpha_2)$$

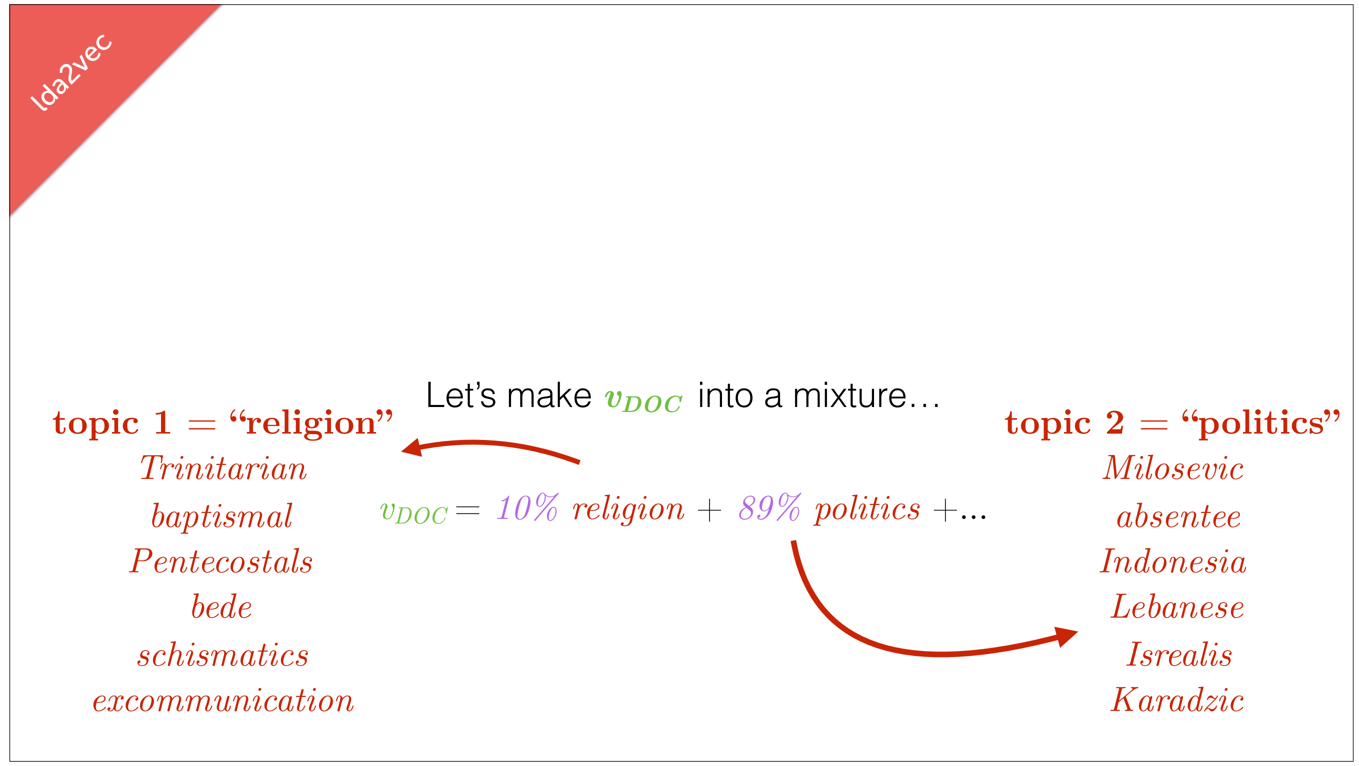
But the weights can still end up being very dense -- which meant everyone of my documents was a mixture of almost every component. This made it difficult to interpret what the document was, because it had membership in many groups.

So next we enforce a simplex with dirichlet & enforce sparsity with the concentration on the weights. The dirichlet is also nice but not critical, we could've had a non-negative decomposition or just stuck with all reals. But since Dirichlet components sum to 100%, it is easier to explain to analysts that a document is "10% of some_topic + 90% some_other_topic" rather than saying "-2.3 * some_topic and +0.5 of some_other_topic".



Can measure similarity between topic vectors m and n, and word vectors w

This gets you the 'top' words in a topic, can figure out what that topic is



This is now on the 20 newsgroups dataset…

Doc is now 10% religion 89% politics

mixture models are powerful for interpretability