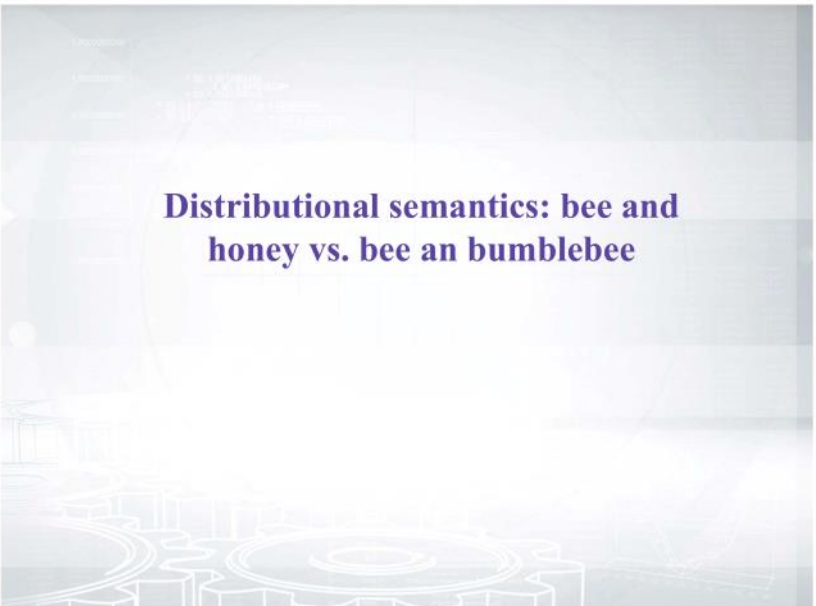
Word and sentence embedding

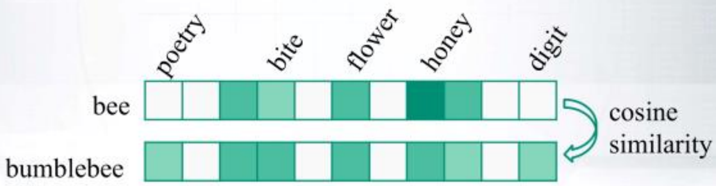
Friday, June 15, 2018 10:22 PM

Distributional semantics



Word similarities

- First order co-occurrences
  syntagmatic associates / relatedness (bee and honey)
- Second order co-occurences
  paradigmatic parallels / similarity (bee and bumblebee)

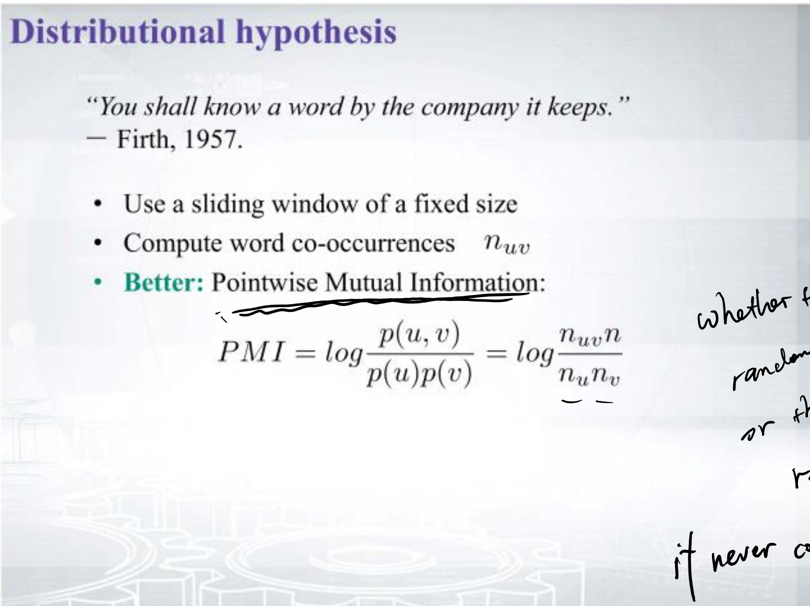


Schutze, H., & Pedersen, J. (1993). A vector model for syntagmatic and paradigmatic relatedness. In Making Sense of Words: Proceedings of the Conference, pp. 104-113, Oxford, England.

Distributional hypothesis

"You shall know a word by the company it keeps."
— Firth, 1957.

- Use a sliding window of a fixed size
- Compute word co-occurrences n_{uv}



Compute a second-order co-occurrence between the words *'bees'* and *'honey'* (the cosine similarity between their first-order co-occurrence vectors). Use the toy corpus:

These are the wrong sort of bees. Quite the wrong sort. So I should think they would make the wrong sort of honey.

- Let's define a context of a word as `three words to the left and three words to the right from the target word, occurred within the same sentence` (if there are any).
- For the first-order co-occurrence, let's consider pPMI values (the formula was given on slide 5 of the first video).

<u>Hint:</u> in this question you actually do not need to *compute* anything... And the answer would be the same for any type of first-order co-occurrence.

From <https://www.coursera.org/learn/language-processing/exam/bK2iE/word-and-sentence-embeddings>

whether the words randomly occur or they are related.

	wrong	sort	of
bee	1	1	1
honey	1	1	1

if never co-occur — $\frac{p(u,v)}{p(u)p(v)} \approx 0$

$\Rightarrow PMI \to -\infty$

So, ~~the~~ take $\max\left(\log\frac{n_{uv}n}{n_u n_v}, 0\right)$

Distributional hypothesis

"You shall know a word by the company it keeps."
— Firth, 1957.

- Use a sliding window of a fixed size
- Compute word co-occurrences n_{uv}
- **Better:** Pointwise Mutual Information:

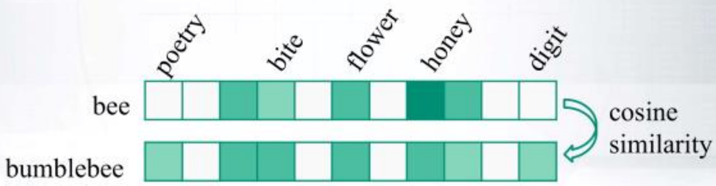
$$PMI = log\frac{p(u,v)}{p(u)p(v)} = log\frac{n_{uv}n}{n_u n_v}$$

- **Even better:** positive Pointwise Mutual Information:

$$pPMI = \max(0, PMI)$$

Any problems here?

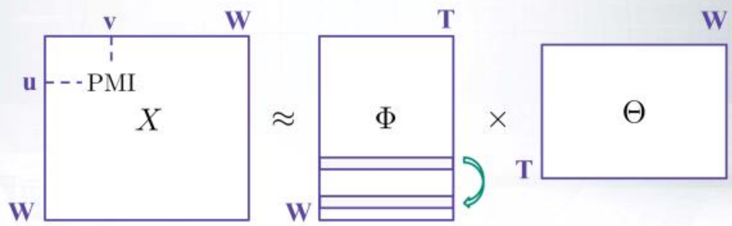
- First order co-occurrences
  syntagmatic associates / relatedness (bee and honey)
- Second order co-occurences
  paradigmatic parallels / similarity (bee and bumblebee)



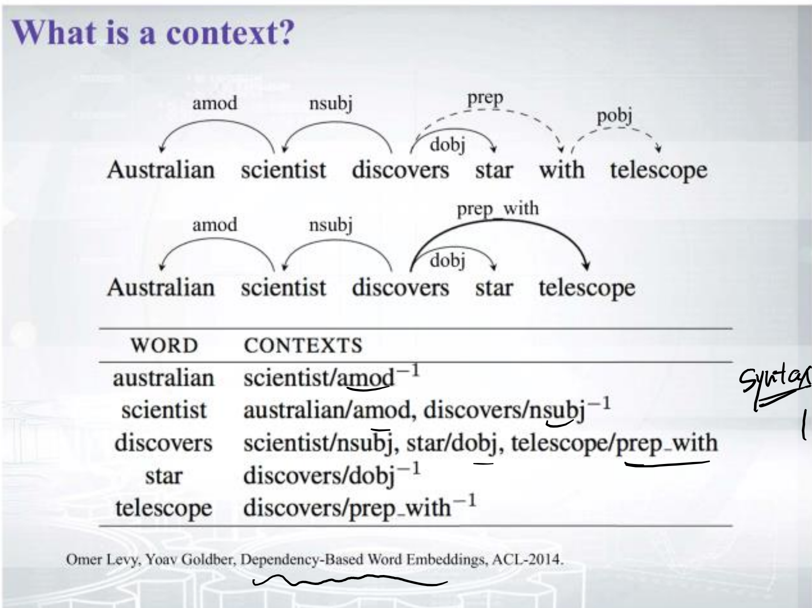
Schutze, H., & Pedersen, J. (1993). A vector model for syntagmatic and paradigmatic relatedness. In Making Sense of Words: Proceedings of the Conference, pp. 104-113, Oxford, England.

Vector Space Models of Semantics

- **Input:** word-word co-occurrences (counts, PMI, …)
- **Method:** dimensionality reduction (SVD, …)
- **Output:** similarity between vector representations of words



Turnay, P.D., Pantel, P.: from Frequency to Meaning: Vector Space Models of Semantics, 2010.



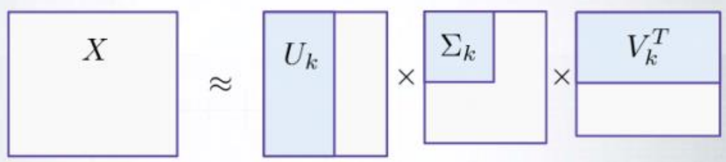
Syntax helps to understand what is local context and what is not matrix factorization

2. Choose correct statements about Singular Value Decomposition (SVD), an important notion from the linear algebra. Feel free to consult any additional resource like wiki if needed.

- [x] Truncated SVD is the best rank k approximation of the original matrix in terms of Frobenius norm.
- [x] Any rectangular matrix with real entries has a singular value decomposition.
- [ ] Singular values can be negative.
- [x] Squares of singular values of a matrix X are eigenvalues of $X^T X$ (or $X X^T$).
- [ ] Singular values of a rectangular matrix are its eigenvalues.
- [x] Singular values decomposition is not unique (for example, the zero matrix can be decomposed in infinitely many ways).

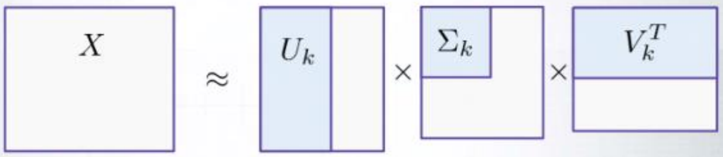
Truncated SVD

Keep only first k components: $\hat{X}_k = U_k \Sigma_k V_k^T$



Truncated SVD

Keep only first k components: $\hat{X}_k = U_k \Sigma_k V_k^T$



It's the best approximation of rank k in terms of Frobenius norm:

$$||X - \hat{X}||_F = \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{m} (x_{ij} - \hat{x}_{ij})^2}$$

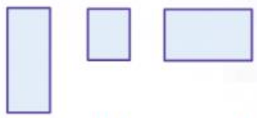
How do we use it?

Option 1:

$$\underbrace{\Phi = U_k \Sigma_k}\quad \underbrace{\Theta = V_k^T}$$

makes two matrices

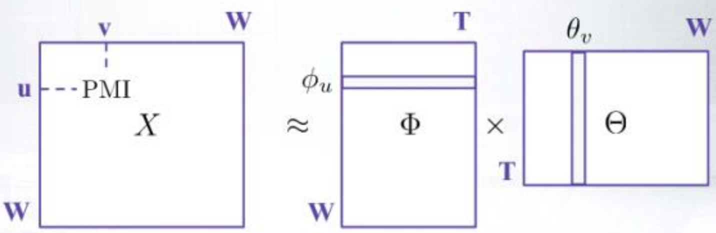
Option 2:



$$\underbrace{\Phi = U_k \sqrt{\Sigma_k}}\quad \underbrace{\Theta = \sqrt{\Sigma_k} V_k^T}$$

Vector Space Models of Semantics

- **Input:** word-word co-occurrences (counts, PMI, …)
- **Method:** dimensionality reduction (SVD, …)
- **Output:** similarity between vector representations of words



Turnay, P.D., Pantel, P.: from Frequency to Meaning: Vector Space Models of Semantics, 2010.

matrix factorization

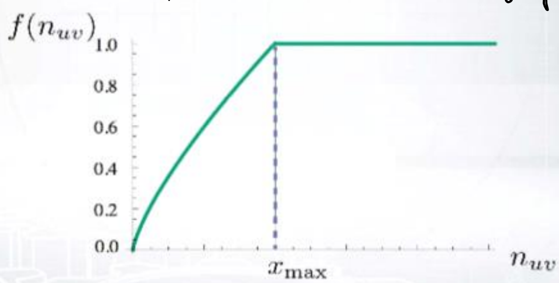
Weighted squared loss: GloVe

original matrices. log
weighted square loss.

Fill X with $\log n_{uv}$ and try another objective:

$$\sum_{u \in W} \sum_{v \in W} f(n_{uv}) \left(\langle \phi_u, \theta_v \rangle + b_u + b'_v - \boxed{\log n_{uv}} \right)^2 \to \min_{\phi_u, \theta_v, b_u, b'_v}$$

higher weights for important pairs of words [higher count]



caps at 1, otherwise too frequent words get too much weights.

Pennington et. al. GloVe: Global Vectors for Word Representation, 2014.

Word prediction: skip-gram model

Training this model.

SGD.

Predict *context words* given a focus word:

$$p(w_{i-h}, \dots w_{i+h} | w_i) = \prod_{-h \leq k \leq h,\, k \neq 0} p(w_{i+k} | w_i)$$

Model each probability with a $\boxed{\textit{softmax}:}$

$$p(u|v) = \frac{\exp \langle \phi_u, \theta_v \rangle}{\sum_{u' \in W} \exp \langle \phi_{u'}, \theta_v \rangle}$$

Still two matrices of parameters.

How do we train the model?

Log-likelihood maximization:

$$\mathcal{L} = \sum_{u \in W} \sum_{v \in W} n_{uv} \log p(u|v)$$

word co-occurrence

Method:
- SGD, online by word pairs in the corpus

Problem:
- *softmax* over vocabulary is slow!

Skip-gram Negative Sampling (SGNS)

Instead of predicting a word for another word, predict "yes" or "no" for word pairs:

$$\sum_{u \in W} \sum_{v \in W} n_{uv} \log \sigma \left(\langle \phi_u, \theta_v \rangle \right) +$$

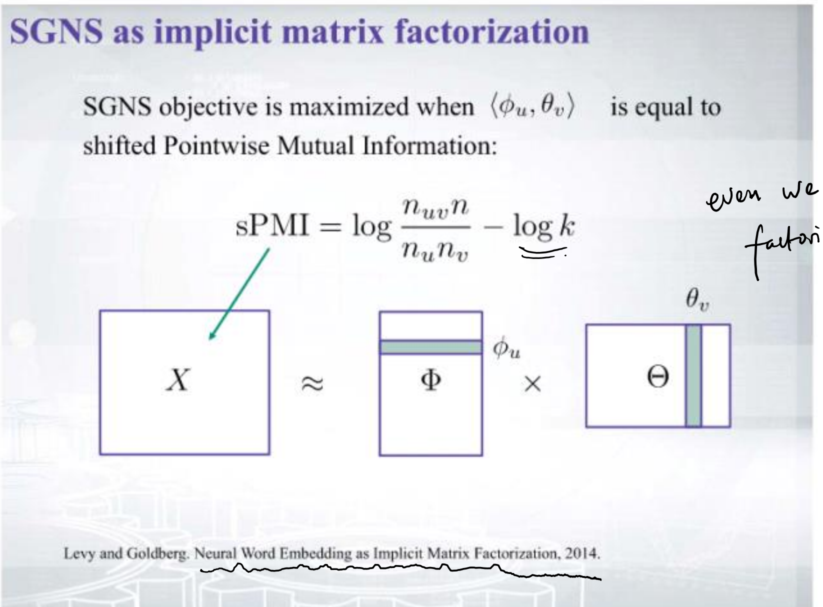
$$k \, \mathbb{E}_{\bar{v}} \log \sigma \left(- \langle \phi_u, \theta_{\bar{v}} \rangle \right) \to \max_{\phi_u, \theta_v}$$

sample for every u word

- Use positive examples from data: v co-occurred with u
- Sample negative examples: k random $\bar{v}$ from the vocabulary

sigmoid function

Train with SGD to find two matrices of parameters (as usual).



even we don't think about matrix factorization, it's still implicitly matrix factorization.

word2vec and doc2vec

Word2vec and doc2vec
(and how to evaluate them)

Word2vec

Two architectures:

- CBOW (Continuous Bag-of-words): context → word of interest.

$$p(w_i|w_{i-h}, \dots w_{i+h})$$

- Continuous Skip-gram:

$$p(w_{i-h}, \dots w_{i+h}|w_i)$$

word of interest → context

Two ways to avoid softmax:

- Negative sampling
- Hierarchical softmax

Open-source and fast: code.google.com/archive/p/word2vec/

Evaluation: word similarities

How do we test that *similar words* have *similar vectors*?

not straightforward task

- Linguists know a lot about what is "similar".
- We can use *human judgements* for word pairs.

Not the best way.

- Compare *Spearman's correlation* between two lists:

linguists' table.

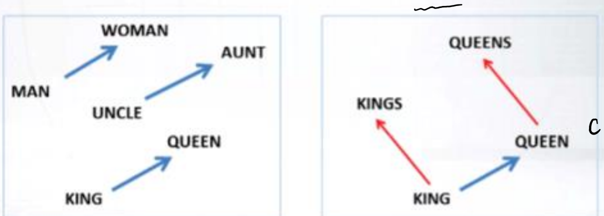
tiger	tiger	10.00
media	radio	7.42
tiger	cat	7.37
train	car	6.31
...	…	…

tiger	tiger	$\cos(\phi_u, \phi_v)$
media	radio	…
tiger	cat	…
train	car	…
...	…	…

Evaluation: word analogies

- In cognitive science well known as *relational similarity* (vs. *attributional similarity*).
- a : a' is as b : b' (man : woman is as king : ?)

$$\cos(b - a + a',\ x) \to \max_x$$



Gentner, D. Structure-mapping: A theoretical framework for analogy. Cognitive Science, 1983.
Mikolov et. al. Linguistic Regularities in Continuous Space Word Representations, 2013.

relational similarity.

before. we talk about
attributional similarity.

Word similarity task performance

- For word similarity task, count-based methods (PPMI, SVD) perform on par with predictive methods (GloVe, SGNS).

`win`	Method	WordSim Similarity	WordSim Relatedness	Bruni et al. MEN	Radinsky et al. M. Turk
2	PPMI	.732	**.699**	.744	.654
	SVD	.772	.671	**.777**	.647
	SGNS	**.789**	.675	.773	**.661**
	GloVe	.720	.605	.728	.606
5	PPMI	.732	**.706**	.738	**.668**
	SVD	.764	.679	**.776**	.639
	SGNS	**.772**	.690	.772	.663
	GloVe	.745	.617	.746	.631

`win` is the width of the window for co-occurrences collection.

Levy et. al. Improving distributional similarity with lessons learned from word embeddings, 2015.

what's GloVe.

Picture on page "Word and sentence embedding"

matrix factorization of log-counts
wrt weighted squared loss

Word analogy task performance

- Word analogy task is solved with 70% average accuracy.

win	Method	Google Add / Mul	MSR Add / Mul
2	PPMI	.552 / .677	.306 / .535
	SVD	.554 / .591	.408 / .468
	SGNS	.676 / **.689**	.617 / **.644**
	GloVe	.649 / .666	.540 / .591
5	PPMI	.518 / .649	.277 / .467
	SVD	.532 / .569	.369 / .424
	SGNS	.692 / **.714**	.605 / **.645**
	GloVe	.700 / .712	.541 / .599

`Add` is the way of analogy solving that we discussed. `Mull` is a modification.

Levy et. al. Improving distributional similarity with lessons learned from word embeddings, 2015.

Paragraph2vec aka doc2vec

$$\underbrace{\text{And the only reason for being a}}_{\text{contexts}}\ \underset{\substack{\uparrow\\ \text{focus}\\ \text{word}}}{\text{bee}}\ \underbrace{\text{that I know of is making honey.}}_{\text{contexts}}$$

Condition not on word, but on document

DM (Distributed Memory):

$$p(w_i|w_{i-h}, \dots w_{i+h}, d)$$

DBOW (Distributed Bag Of Words):

$$p(w_{i-h}, \dots w_{i+h}|d)$$

Evaluation: document similarities

How do we test that *similar documents* have *similar vectors*?

- ArXiv triplets: paper A, similar paper B, dissimilar paper C
- Measure the accuracy of guessing the dissimilar paper

http://arxiv.org/pdf/1206.5743	http://arxiv.org/pdf/cond-mat/0403258	http://arxiv.org/pdf/1408.0189
http://arxiv.org/pdf/1209.0268	http://arxiv.org/pdf/1307.7598	http://arxiv.org/pdf/math/0504051
http://arxiv.org/pdf/hep-ph/9908436	http://arxiv.org/pdf/nucl-th/9707019	http://arxiv.org/pdf/1112.3014
http://arxiv.org/pdf/1111.2905	http://arxiv.org/pdf/1303.2538	http://arxiv.org/pdf/1109.1922
http://arxiv.org/pdf/nucl-ex/0112013	http://arxiv.org/pdf/physics/9704013	http://arxiv.org/pdf/1408.4595
http://arxiv.org/pdf/0709.3419	http://arxiv.org/pdf/quant-ph/0611134	http://arxiv.org/pdf/0902.0616
http://arxiv.org/pdf/hep-th/9609148	http://arxiv.org/pdf/solv-int/9710009	http://arxiv.org/pdf/astro-ph/0508060

Andrew Dai, Cristopher Olah, Quoc Le. Document Embedding with Paragraph Vectors, CoRR, 2015.

Evaluation: document similarities

Integral formula of Minkowski type and new characterization of the Wulff shape

Yijun He [*] Haizhong Li [†]

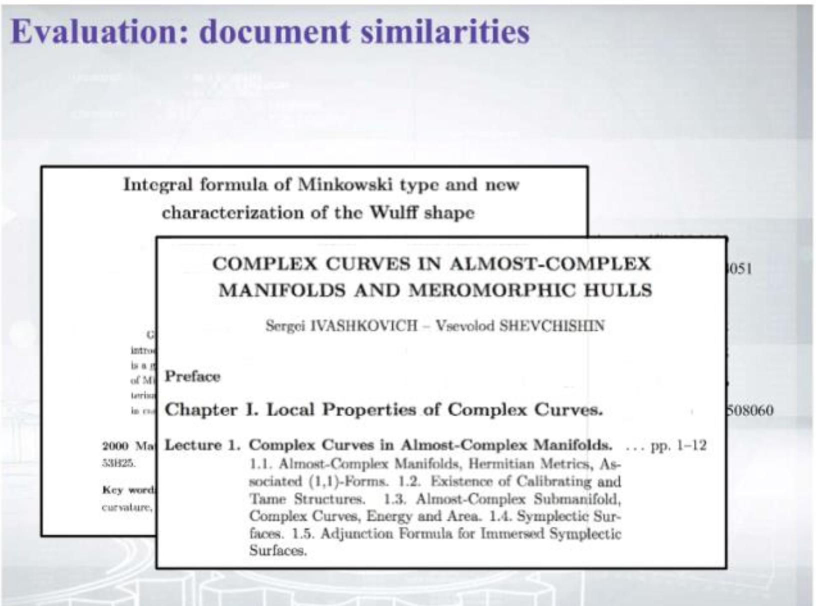
Abstract

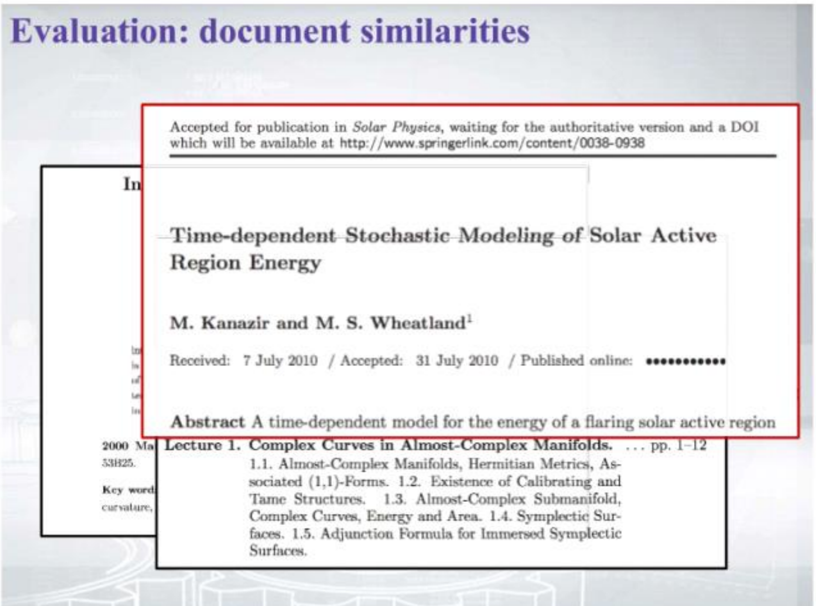
Given a positive function F on S^n which satisfies a convexity condition, we introduce the r-th anisotropic mean curvature M_r for hypersurfaces in $\mathbb{R}^{n+1}$ which is a generalization of the usual r-th mean curvature H_r. We get integral formulas of Minkowski type for compact hypersurfaces in R^{n+1}. We give some new characterizations of the Wulff shape by use of our integral formulas of Minkowski type, in case $F = 1$ which reduces to some well-known results.

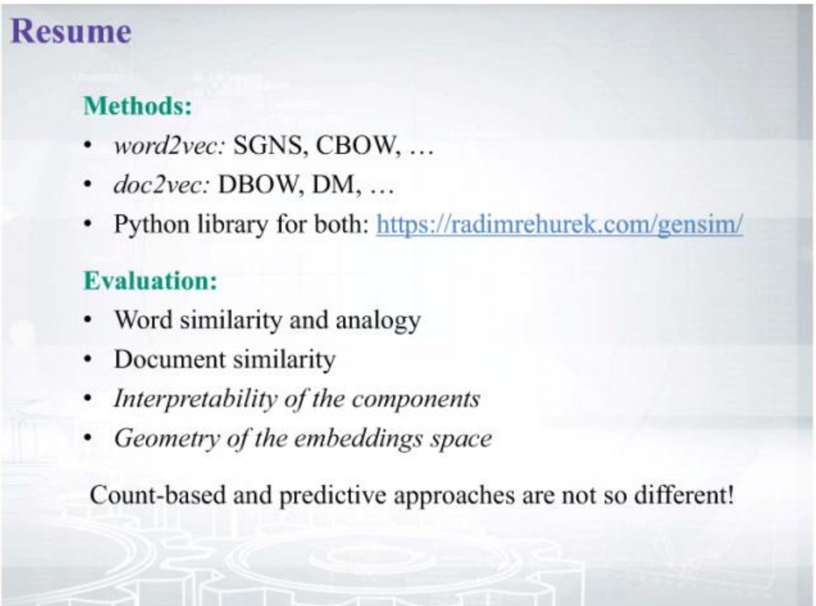
2000 Mathematics Subject Classification: Primary 53C42, 53A30; Secondary 53B25.

Key words and phrases: Wulff shape, F-Weingarten operator, anisotropic principal curvature, r-th anisotropic mean curvature, integral formula of Minkowski type.

xiv.org/pdf/1408.0189
xiv.org/pdf/math/0504051
xiv.org/pdf/1112.3014
xiv.org/pdf/1109.1922
xiv.org/pdf/1408.4595
xiv.org/pdf/0902.0616
xiv.org/pdf/astro-ph/0508060

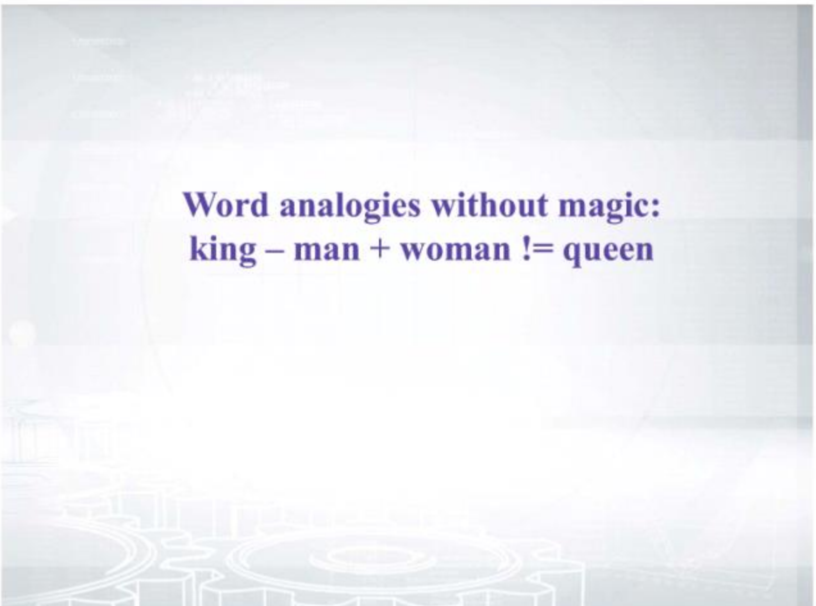

Evaluation: document similarities
Integral formula of Minkowski type and new
characterization of the Wulff shape
COMPLEX CURVES IN ALMOST-COMPLEX
MANIFOLDS AND MEROMORPHIC HULLS
Sergei IVASHKOVICH – Vsevolod SHEVCHISHIN
Preface
Chapter I. Local Properties of Complex Curves.
Lecture 1. Complex Curves in Almost-Complex Manifolds. ... pp. 1–12
1.1. Almost-Complex Manifolds, Hermitian Metrics, Associated (1,1)-Forms. 1.2. Existence of Calibrating and Tame Structures. 1.3. Almost-Complex Submanifold, Complex Curves, Energy and Area. 1.4. Symplectic Surfaces. 1.5. Adjunction Formula for Immersed Symplectic Surfaces.

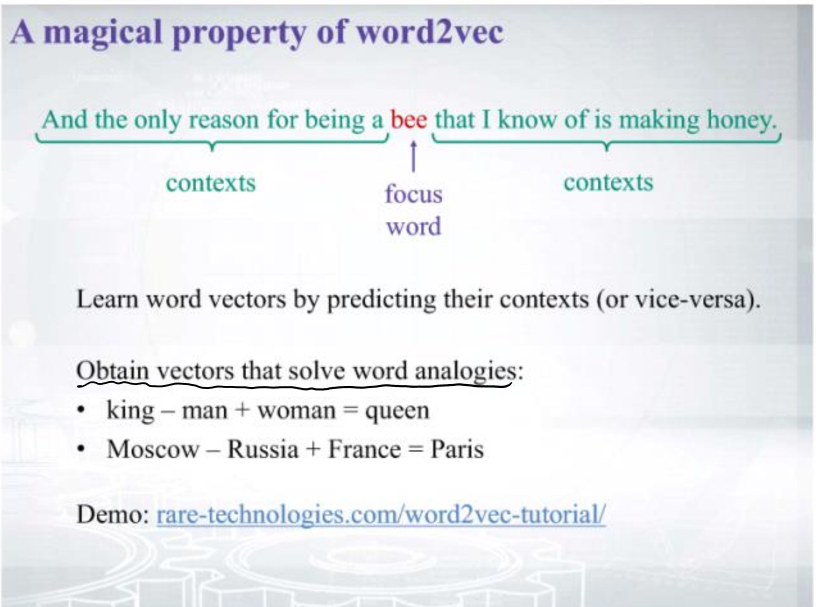

Evaluation: document similarities
Accepted for publication in Solar Physics, waiting for the authoritative version and a DOI which will be available at http://www.springerlink.com/content/0038-0938
Time-dependent Stochastic Modeling of Solar Active Region Energy
M. Kanazir and M. S. Wheatland1
Received: 7 July 2010 / Accepted: 31 July 2010 / Published online: •••••••••••
Abstract A time-dependent model for the energy of a flaring solar active region
Lecture 1. Complex Curves in Almost-Complex Manifolds. ... pp. 1–12
1.1. Almost-Complex Manifolds, Hermitian Metrics, Associated (1,1)-Forms. 1.2. Existence of Calibrating and Tame Structures. 1.3. Almost-Complex Submanifold, Complex Curves, Energy and Area. 1.4. Symplectic Surfaces. 1.5. Adjunction Formula for Immersed Symplectic Surfaces.
2000 Ma
53B25.
Key word
curvature,



count-based method
not much different
from
predictive-based method !

word analogy


Word analogies without magic:
king – man + woman != queen


A magical property of word2vec
And the only reason for being a bee that I know of is making honey.
contexts
focus word
contexts
Learn word vectors by predicting their contexts (or vice-versa).
Obtain vectors that solve word analogies:
• king – man + woman = queen
• Moscow – Russia + France = Paris
Demo: rare-technologies.com/word2vec-tutorial/

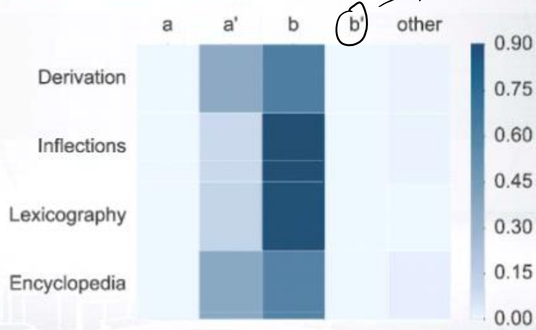
Closer look into analogy task

$$\cos(b - a + a',\, x) \rightarrow \max_{x \notin \{a, a', b\}}$$

king – man + woman = king:

never close to b' — queen

the target



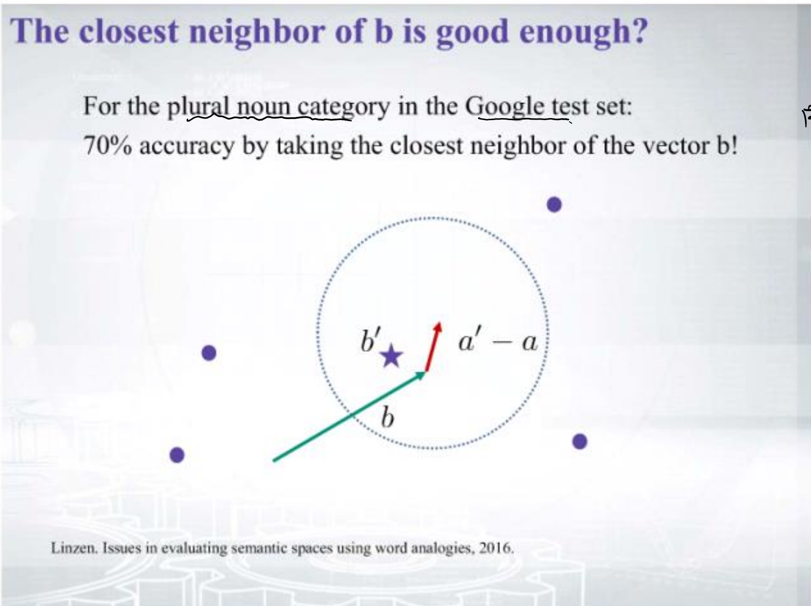
Different analogy types

Rogers et. al. The (Too Many) Problems of Analogical Reasoning with Word Vectors, 2017.

If we do not exclude b.

we end up with b.

不过为啥不 exclude b?

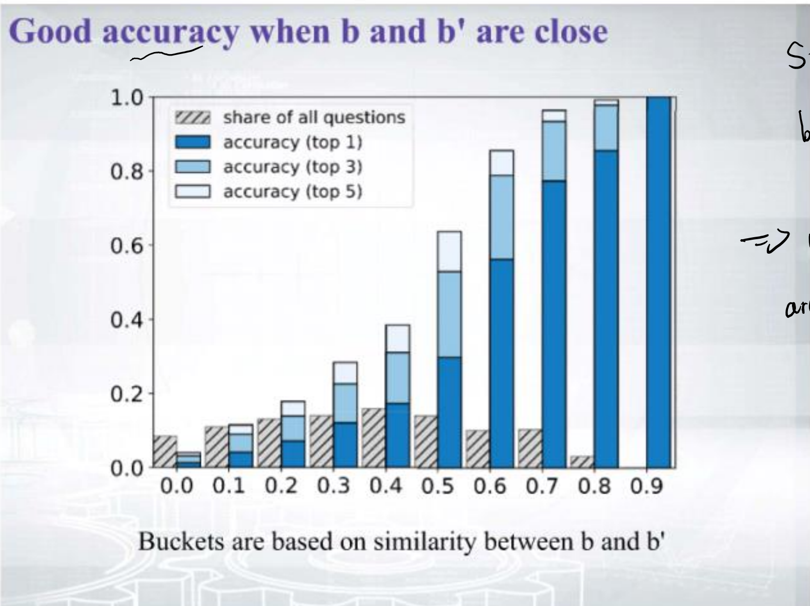


除 不管减掉了谁

找b的 ~~close~~ closest vector

就好了.

high accuracy



Split analogy example by the similarity of b & b'

=> works really nice when b and b' are similar.

works poorly when b & b' are different (complicated tasks.

BATS dataset

Inflectional morphology:

- *student:students, strong:stronger, follow:following, ...*

Derivational morphology:

- *bake:baker, edit:editable, home:homeless, ...*

Lexicographic semantics:

- Hypernyms, meronyms : peach:fruit, sea:water, player:team, …
- Antonyms, synonyms: up:down, clean:dirty, cry:scream, …

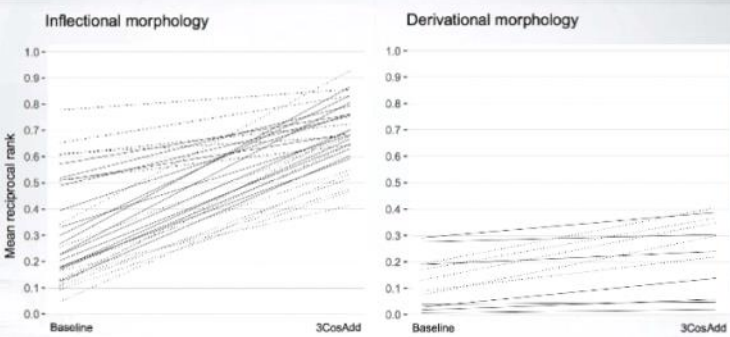
Encyclopedic semantics:

- Animals: cat:kitten, dog:bark, …
- Geography: Athens:Greece, Peru:Spanish, …
- People: Lincoln:president, Lincoln:American, …
- Other: blood:red, actor:actress, …

让AI做类似还是不给力吧.

Compare to baseline:
just use the closest

Performance by categories



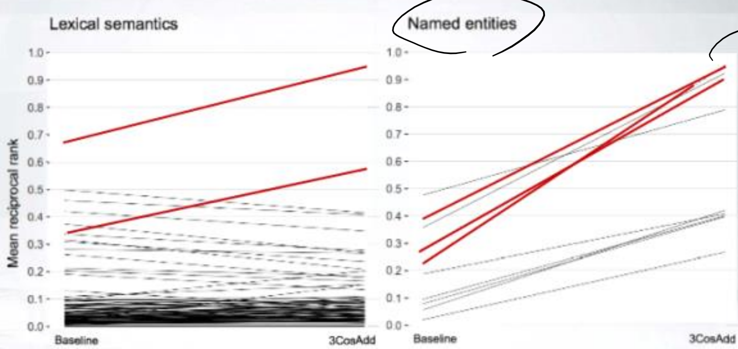
Finley et. al. What Analogies Reveal about Word Vectors and their Compositionality, 2017.

horizontal
no difference.
high difference.

Genders and counties are cherry-picks

looks fine ...

genders work well.



Finley et. al. What Analogies Reveal about Word Vectors and their Compositionality, 2017.

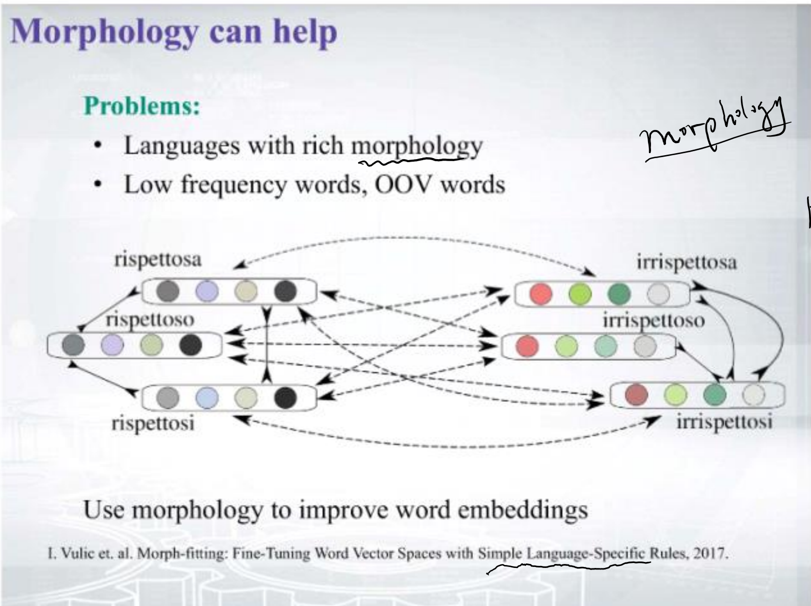
Resume

- *word2vec* works fine for word similarities
- But there are many questions with word analogies
- Be careful about hype!

Does not solve all analogy task.
Nice area for future research

why words

Why words? From character to sentence embeddings



morphology

Need linguist rules

FastText

FastText

Represent a word as a bag of character n-grams, e.g. for n = 3:

G_{where} : _wh, whe, her, ere, re_, _where_

Model a word vector as a sum of sub-word vectors:

word vector as sum of subword vectors

SGNS:

$$sim(u, v) = \langle \phi_u, \theta_v \rangle$$

FastText:

$$sim(u, v) = \sum_{g \in G_v} \langle \phi_u, \theta_g \rangle$$

Code and pre-trained embeddings: https://fasttext.cc/

P. Bojanowsky et al. Enriching Word Vectors with Subword Information, 2016.

Sent2vec

> not nice approach.

First ideas:

- Average pre-trained word vectors *(word2vec, GloVe, etc)*.
- Maybe use TF-IDF weights for averaging.

use TF-IDF as weights

Sent2vec:

- Learn sentence embedding as a sum of sub-sentence units:

Sub-sentence units.

$$sim(u, s) = \frac{1}{|G_s|} \sum_{g \in G_s} \langle \phi_u, \theta_g \rangle$$

where G_s is a set of word n-grams for the sentence *s*.

☆. Similar to FastText

Code and embeddings: https://github.com/epfml/sent2vec

☆. Just different units. word n-grams to represent sentence.

Pagliardini et. al. Unsupervised Learning of Sentence Embeddings using Compositional n-Gram Features, 2017.

Have you realized what's the crutial difference between averaging word2vec vectors and sent2vec?

- ✓ sent2vec represents a sentence not as an average of words, but as an average of n-grams (SUB-SENTENCE UNITS)

when n=1 then it's the same as averaging word-to-vec

- sent2vec represents sentences as an average of words DURING TRAINING, while averaging word2vec vectors is a postprocessing step

This should not be selected

Even though sent2vec is still based on word (or n-gram) vectors, it tunes them during traning to fit word-in-sentence co-occurrence data. This is not the same as fitting word-word co-occurrence data (as word2vec does) and can crutially improve the final representations of sentences.

StarSpace

General framework:

entities (e.g. sentences) and *features* (e.g. words)

Lot's of applications:

- Text classification, e.g. sentiment analysis
- Ranking, e.g. ranking web documents given a query
- Collaborative filtering-based recommendation
- Content-based recommendation
- Embedding graphs, e.g. Freebase

- Collaborative filtering-based recommendation
- Content-based recommendation
- Embedding graphs, e.g. Freebase
- Learning word, sentence or document embeddings

Code and tutorials: github.com/facebookresearch/Starspace

Wu et. al. StarSpace: Embed All The Things! 2017

Embed All the things!

StarSpace

Mode 3 (sentence embeddings):

Use case: learn pairwise similarity from collections of similar objects, e.g. sentence similarity.

Data format: each line is a collection of similar sentences:

```
sent1_word1 sent1_word2 … <tab> sent2_word1 sent2_word2 …
```

Training:

- Each sentence is represented as a bag of features (words or n-grams). They are embedded to predict sentence similarity
- Similar sentence pairs are taken from the collections
- Dissimilar sentence pairs are sampled at random

Deep learning?

The most popular options:

- Recurrent Neural Networks (sequence modelling)
- Convolutional Neural Networks (much faster)
- Recursive Neural Networks (use hierarchical structure)

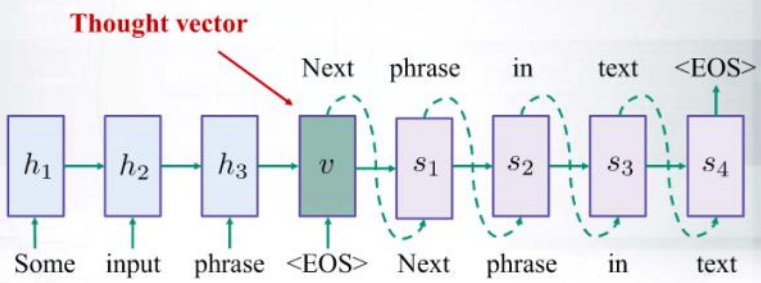
Linguistic structure is back:

- Morphology can help to build word embeddings
- Recursive Neural Networks, Tree-LSTMs, DAG-LSTMs, etc. use syntax, span annotations, co-reference links…

use language to build hierarchical representation

Skip-thought vectors

- Predict next and previous sentences in text
- RNN encoder-decoder architecture



Kiros et. al. Skip-Thought Vectors, 2015, https://github.com/ryankiros/skip-thoughts.

encoder-decoder architecture
不是用来翻译.
而是predict next sentence.
也是富有想象力的.