Master Thesis Defense

Scalable approach for Learning Word Representations

Name: Zhang, Zijian
Matrikelnummer: 3184680

First Examiner: Prof.Dr. Avishek Anand
Second Examiner: Prof. Dr. techn. Dipl.-Ing. Wolfgang Nejdl
Advisor: Prof. Dr. Avishek Anand

Agenda

Motivation

Problem Statement

Related Work

Our Contribution

Divide, Train and Merge

- Distribution Preserving Sampling
- Merging Approaches

-Experiments Setup

-Performance and Timing

-Conclusion and Works on Going

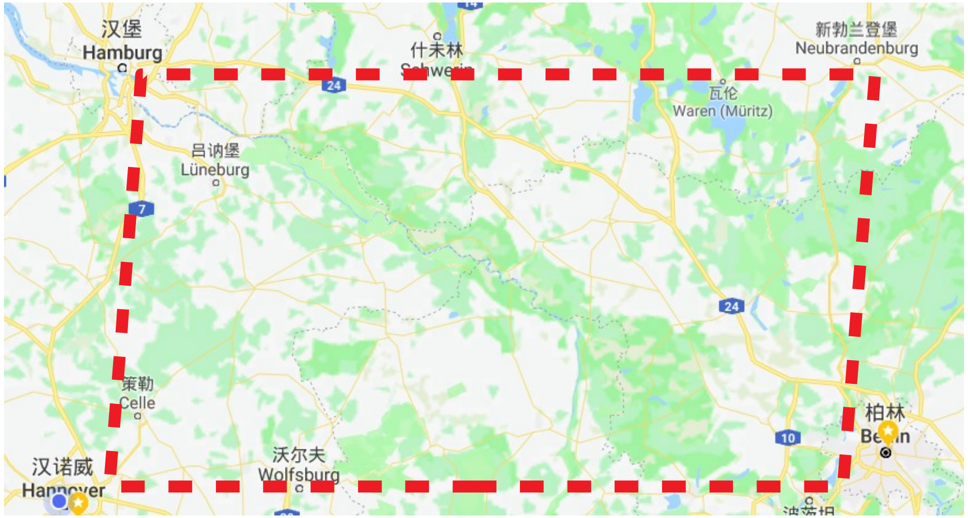
Motivation

1. Hannover – Berlin + Hamburg = ?

Motivation

Hannover – Berlin + Hamburg = Neubrandenburg!

According to the map:



Then what is

King – Man + Woman?

Motivation

A map (the one on a paper or in an App):

maps (mathematical jargon) the city in an area of the surface of the Earth to a point in E^2.

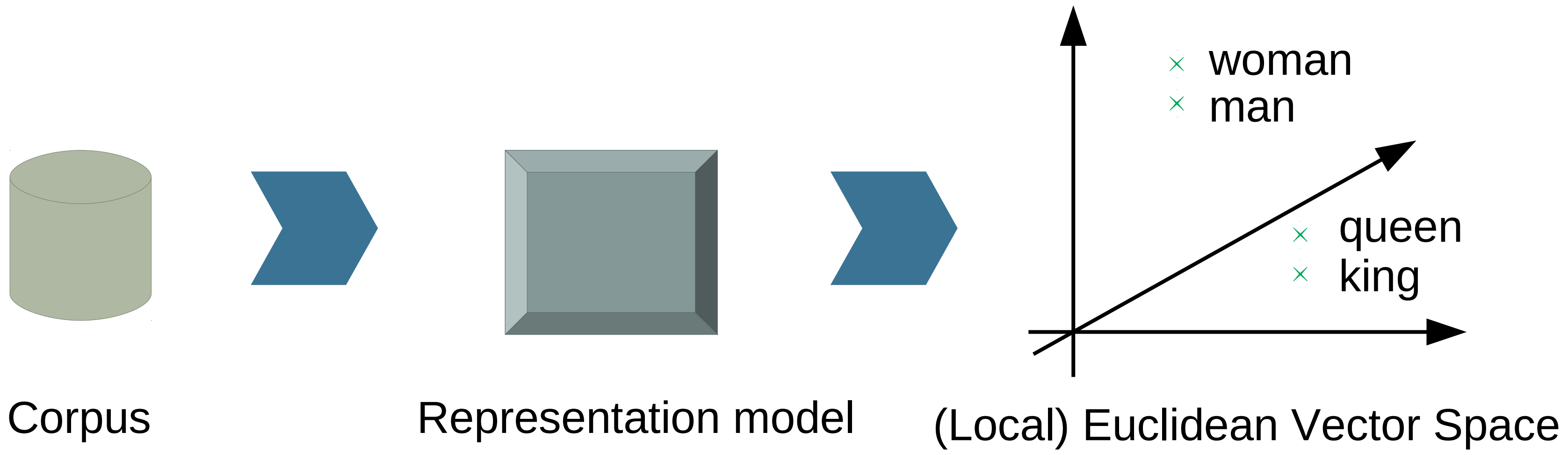
If we change “cities” to “words” and the “surface of Earth” to “Language”

King – Man + Woman

seems solvable!

Motivation

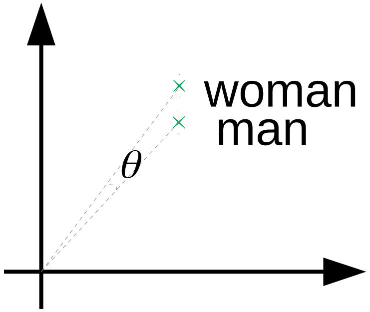
Word Vector Representation



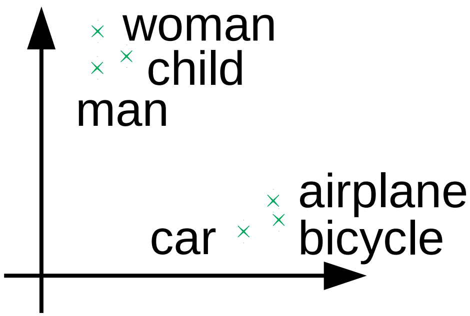
Motivation

Advantages of Word2Vec

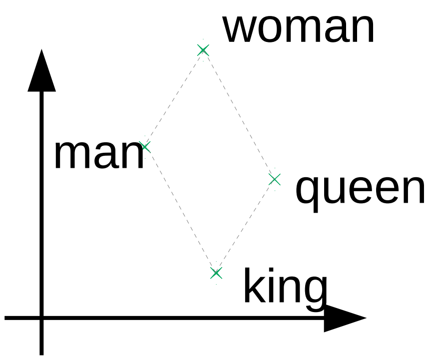
(Mikolov2013)



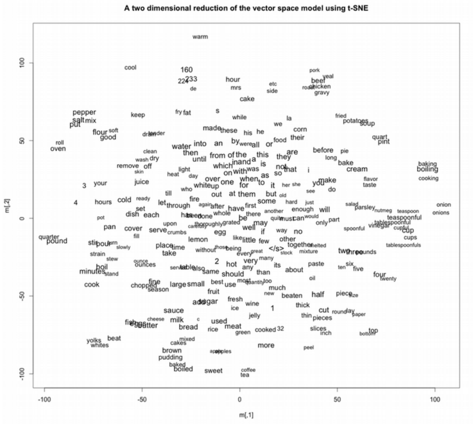
$\cos(\theta)$ ~ similarity



categories ⇒ clusters



analogy ⇒ parallelization



Dimension reduction

Problem Statement

Current Challenges:

larger corpus: more **expressive**, longer **time**

On 14GB English Wikidump: $\approx$ **36 hours** !

(dim: 500 Vocabulary: 300k)

Vanilla Word2Vec isn't **scalable**

Related Work

Word2Vec uses SGD to optimize its objective

One solution: make SGD faster:

- Optimizing SGD schema (online, synchronized) :
  - HogWild!(Recht2011), BLAS-3 SGNS(Ji2016), cache optimized Hierachical Softmax(Eickhoff2016), column-wise distribution (Ordentlich2016)
- Adaptive SGD learning rate:
  - SGD-Momentum(Rumelhart1986), AdaGrad, RMSProp

Heterogeneous combination:

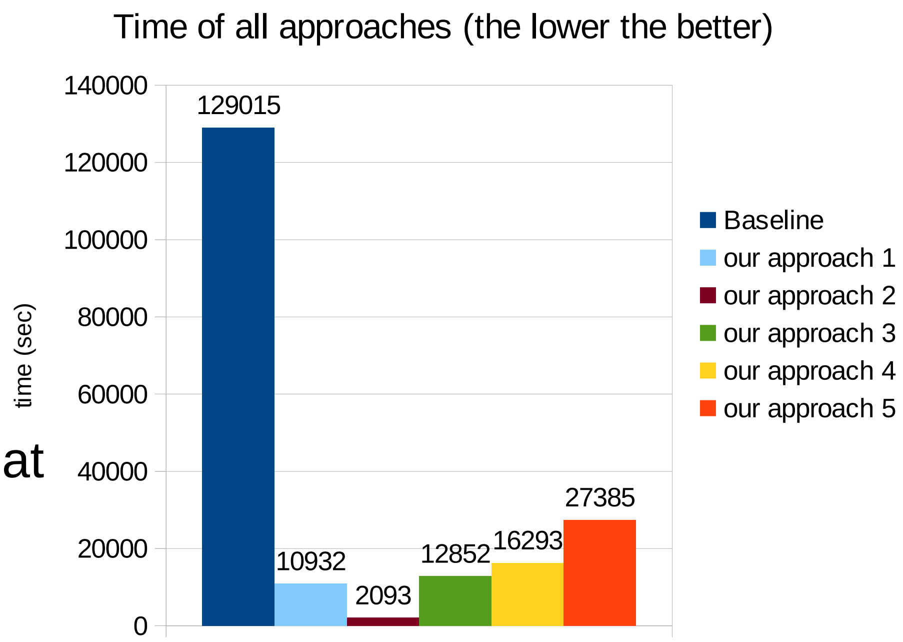
- Combine vectors from Word2Vec, WordNet, GloVe, ConceptNet etc.

Our Contribution

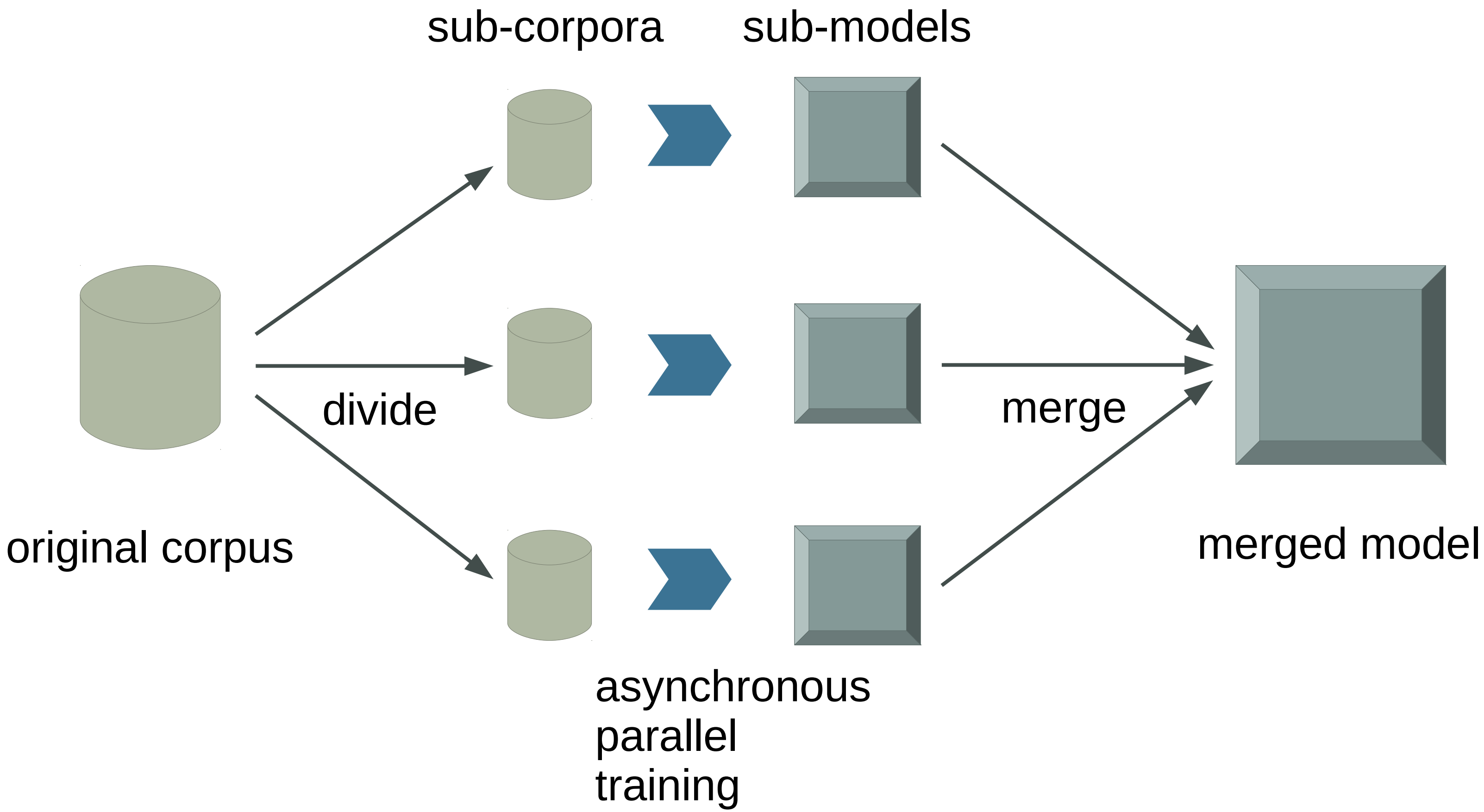
Find a

- scalable
- hardware independent
- asynchronous
- distributional paradigm,

  for training word vector representation that is comparable with the baseline

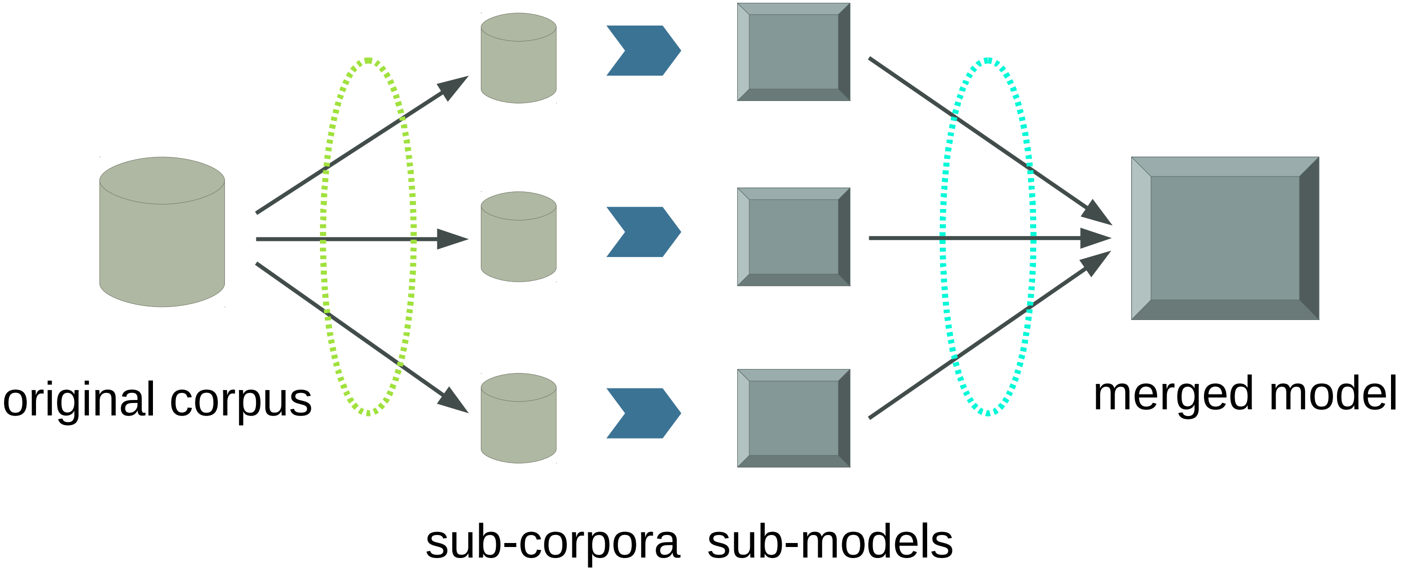


Divide, Train and Merge



Divide, Train and Merge

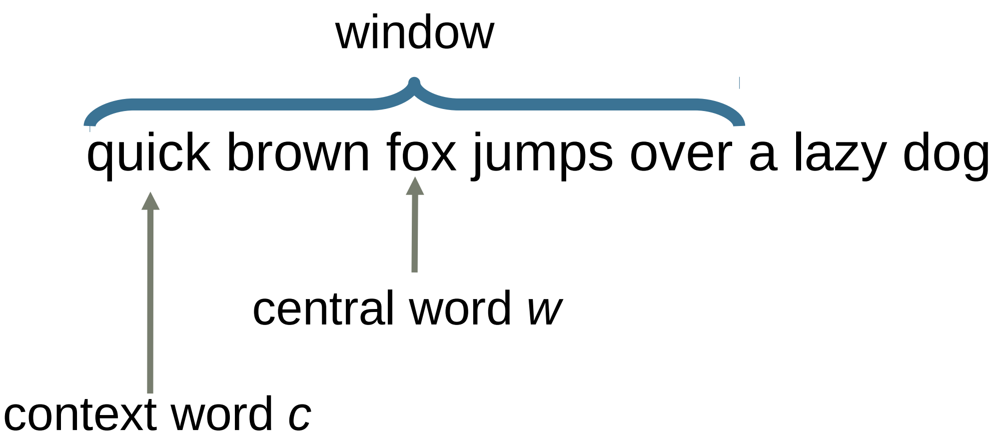
Research Questions:



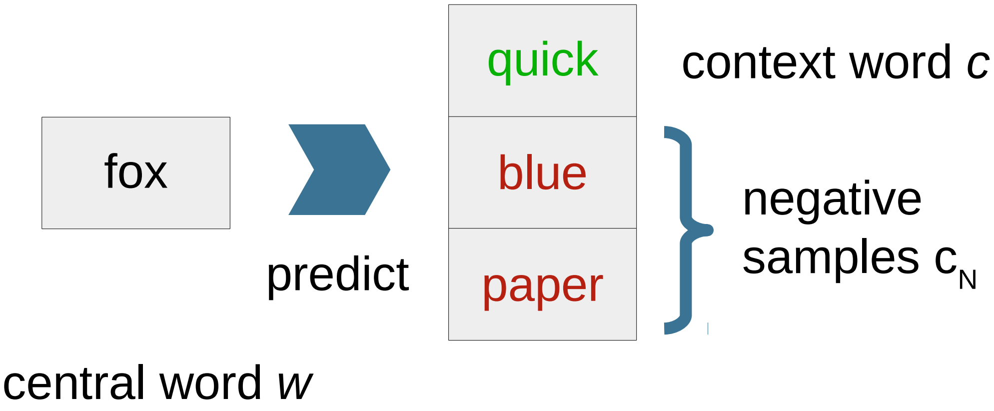
1) How do we sample from the original corpus?

2) How do we combine sub-models?

Distribution Preservation Sampling



Skip Gram



Negative Sampling

Distribution Preservation Sampling

Criterion of division: approach to the original term distribution

The local optimum[(Levy2014)]:

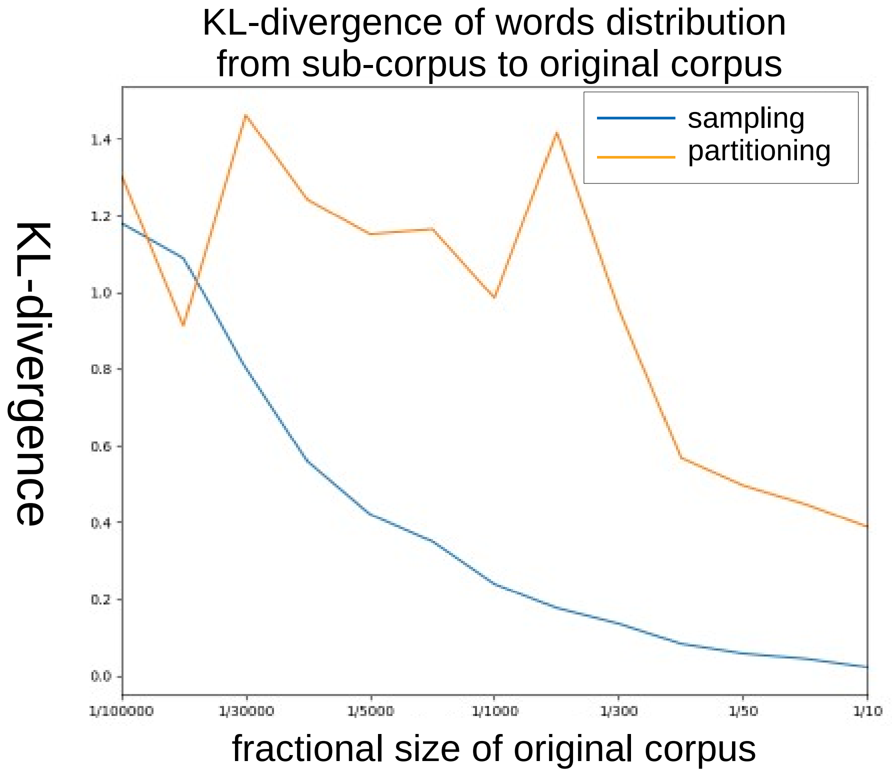
$$\vec{w} \cdot \vec{c} \propto \log \frac{p(w,c)}{p(w)\,p(c)}$$

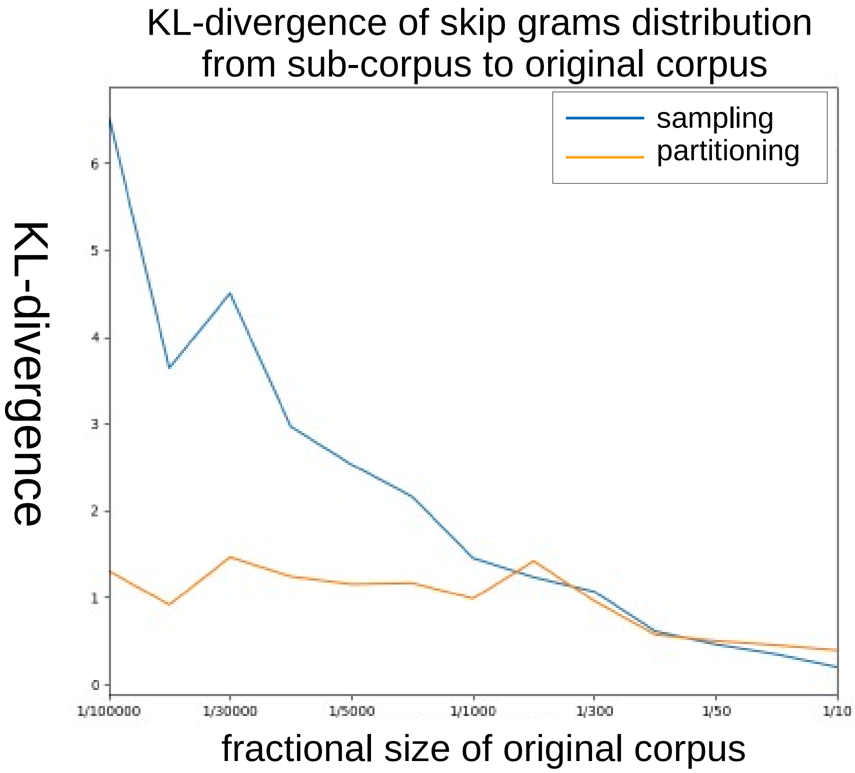
Point-wise
Mutual
Information

Depends on **distribution of Words/skip-grams** only!

Distribution Preserving Sampling

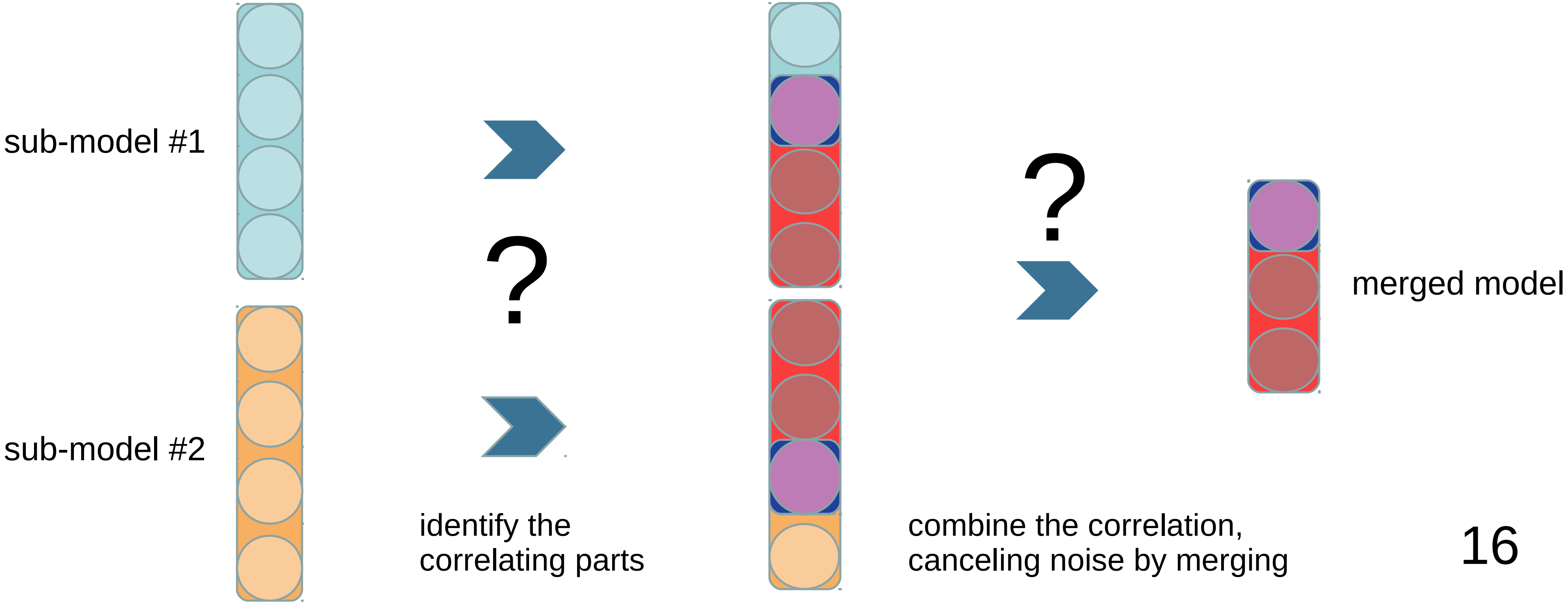
Distribution preservation through uniform reservoir sampling





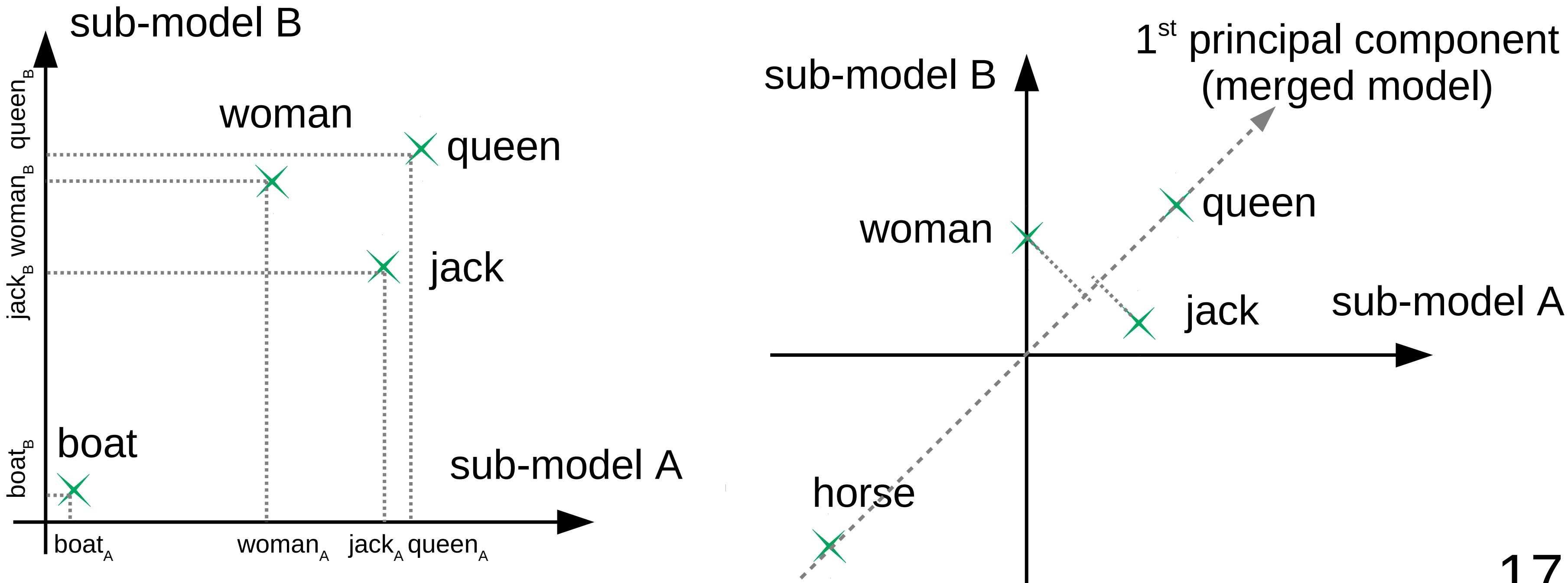
Merging Approaches

Problem so-far:



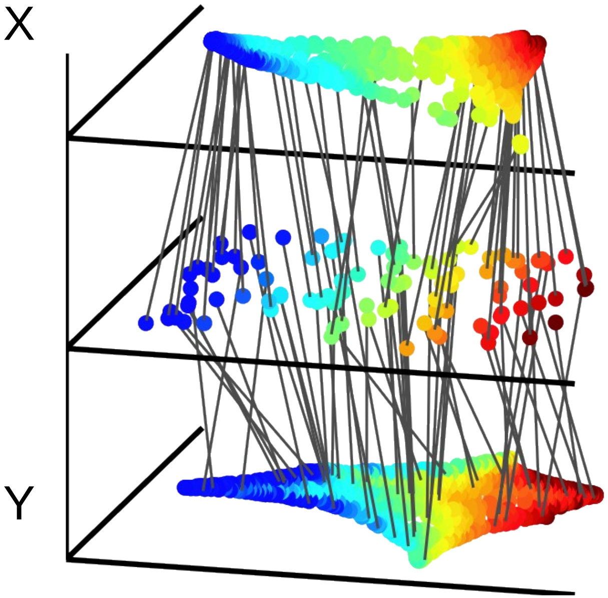
Merging Approaches

1st Approach: Concatenation + Principle Component Analysis



Merging Approaches

2nd Approach: Manifold Alignment (Boucher2015) + Vector Averaging



[2]

Low-rank reconstruction of X

X_{final}, Y_{final} = (1-μ)(reconstructions) +μ(correlation)

$M_{merged}=(X_{final}+Y_{final})/2$

Low-rank reconstruction of Y

[2]: Ham J, Lee D D, Saul L K.
Semisupervised alignment of manifolds
AISTATS. 2005: 120-127.

Experiments Setup

Data Set:

- *Original corpus*: English Wikipedia dump file 2016 (14G)
- Word2Vec implementation: gensim[1]
- *Cleaner*: Wikipedia LineSentences cleaner in Gensim
- *Sample/Partition size*: 1%, 10% of the original corpus

Baseline Model(Levy2015):

- *Vocabulary*: 300k words
- *Dimension (size of hidden layer)*: 500
- *Training time*: 36 hours

Benchmarks(Jastrzebski):

- *Word Similarity*: MEN, RW, RG65, WS353, SimLex999, MTurk
- *Analogy*: Google, SemEval 2012 #2, MSR
- *Categorization*: AP, BLESS, Battig

[1]: **https://radimrehurek.com/gensim/**

Performance and Timing

Effect of Dividing (sampling/partitioning):

Approach	AP	Battig	MEN	RG65	RW	WS353	Google	SemEval2012_2
`Partition(1/10,10)`	0.582	0.420	0.719	0.725	**0.232**	0.596	0.615	0.162
`Random(1/10,10)`	**0.604**	**0.435**	**0.740**	**0.757**	0.209	**0.637**	0.654	**0.188**
`Partition(1/100,100)`	0.505	0.358	0.622	0.723	0.203	0.516	0.467	0.129
`Random(1/100,100)`	0.517	0.381	0.644	0.746	0.257	0.542	0.487	0.138
`Baseline`	0.595	0.434	0.736	0.757	**0.299**	0.611	**0.661**	0.181

Table 2: Comparison between `Random(.,.)` and `Partition` sampling approaches at different granularities. PCA is used as the merging method.

Conclusions:

- Sampling performs overall better than partitioning
- Larger division size makes the performance better, but less scalability
- Surprisingly, some of our approaches even outperform the **baseline**!

Performance and Timing

Effect of Merging (concatenation/PCA/LRA):

Approach	AP	Battig	MEN	RG65	RW	WS353	Google	SemEval2012_2
Random(1/10,10) + Concat	0.595	0.429	**0.742**	0.765	0.268	0.611	0.645	0.187
Random(1/10,10) + LRA	0.560	0.327	0.685	**0.785**	0.264	0.606	**0.661**	0.158
Random(1/10,10) + PCA	**0.604**	**0.435**	0.740	0.757	0.209	**0.637**	0.654	**0.188**
Random(1/100,100) + Concat	**0.525**	0.377	0.643	**0.746**	**0.257**	**0.542**	**0.487**	**0.138**
Random(1/100,100) + LRA	–	–	–	–	–	–	–	–
Random(1/100,100) + PCA	0.517	**0.381**	**0.644**	**0.746**	**0.257**	**0.542**	**0.487**	**0.138**
Baseline	0.595	0.434	0.736	0.757	0.299	0.611	0.661	0.181

Table 3: Evaluation results for merging

Conclusions:

PCA performs overall the best

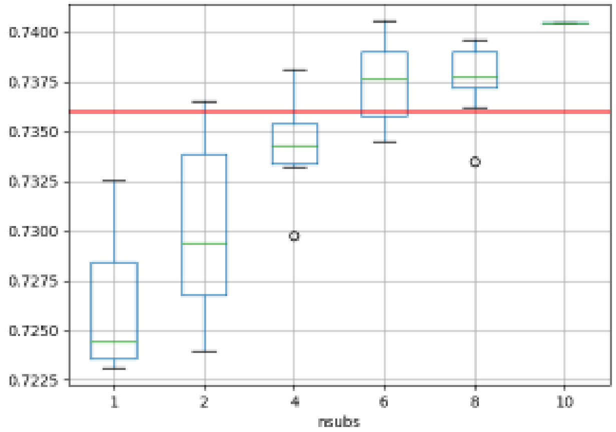
LRA and Vector average is the slowest

To see the information loss in the PCA, we added the concatenation as a single line, it performs surprisingly robust, despite its higher dimensionality

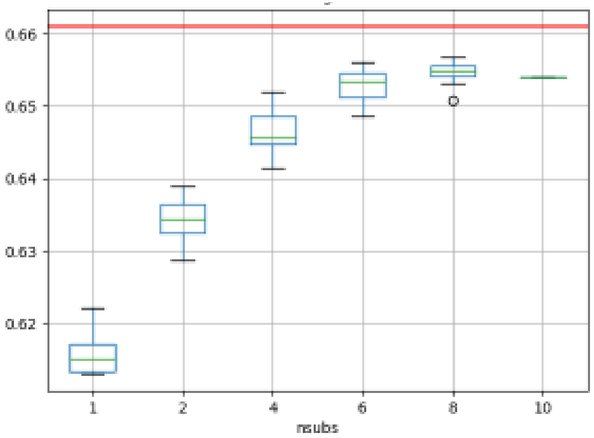
Performance and Timing

Effect of sample count (10% of original corpus):

correlation MEN (the higher the better)



correlation Google (the higher the better)



Conclusion:

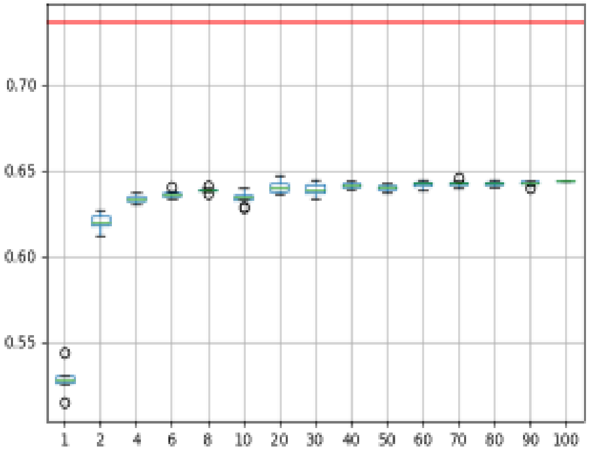
Larger count of sub-models doesn't help.
8~10 for 10% approach, 20~30 for 1% approach is enough.
After that it goes into flat road.

Performance and Timing

Effect of sample count (1% of original corpus):

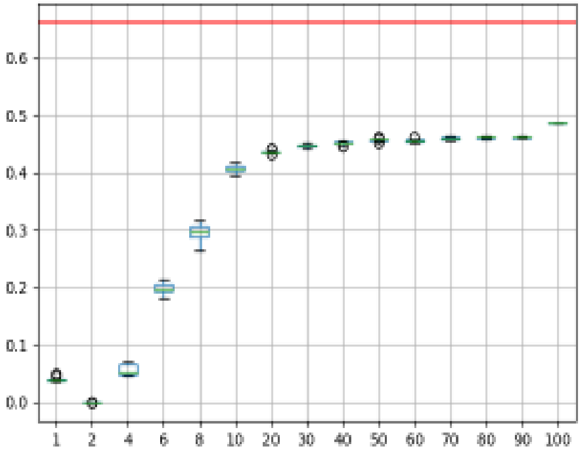
correlation

MEN (the higher the better)



correlation

Google (the higher the better)

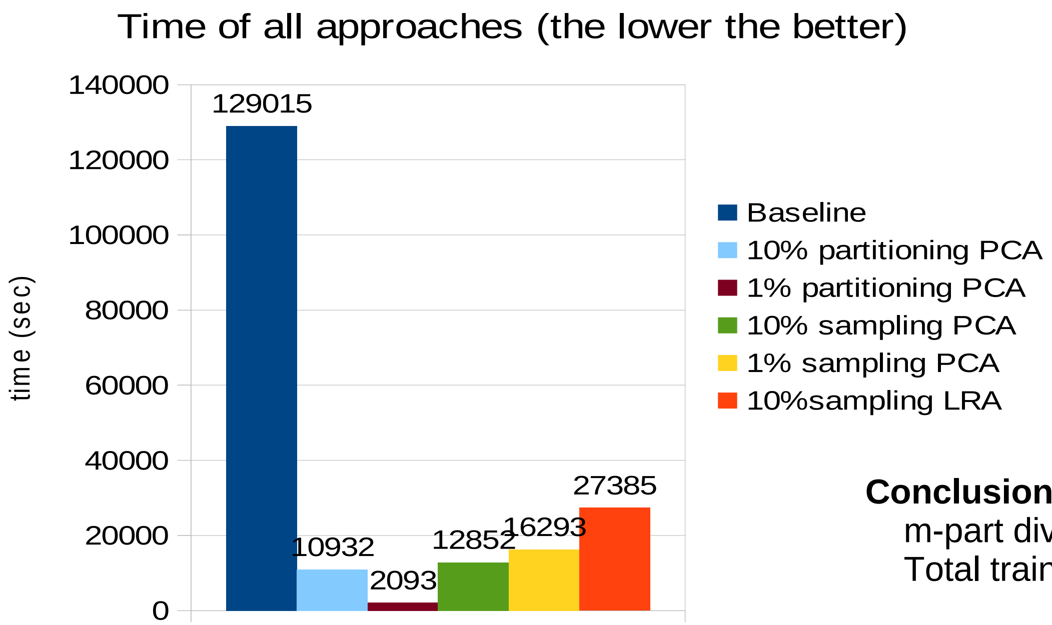


Conclusion:

Larger count of sub-models doesn't help.
8~10 for 10% approach, 20~30 for 1% approach is enough.
After that it goes into flat road.

Performance and Timing

Timing:



Conclusions:

m-part division leads to m-folds speed up

Total training time: from 36 hours @ baseline to
3.6 hours @ 10% and
< 1 hour @ 1% approach

Time of merging is in minutes

Conclusion and on Going Works

Conclusions:

- Sentence-wise sampling to divide, PCA to merge
- M-times speed up by M-fold dividing, sometimes even outperform the baseline.

Ongoing work:

- dealing with missing-word problem between sub-models

Thank you for your attention

Questions and comments are highly appreciated

Zhang, Zijian
Hannover
08. 03. 2018

References:

Mikolov2013: Mikolov T, Yih W, Zweig G.
Linguistic regularities in continuous space word representations
Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. 2013: 746-751.

Recht2011: Recht B, Re C, Wright S, et al.
Hogwild: A lock-free approach to parallelizing stochastic gradient descent
Advances in neural information processing systems. 2011: 693-701.

Ji2016: Ji, Shihao, et al.
Parallelizing word2vec in shared and distributed memory
arXiv preprint arXiv:1604.04661 (2016).

Eickhoff2016: Vuurens, Jeroen BP, Carsten Eickhoff, and Arjen P. de Vries
Efficient Parallel Learning of Word2Vec
arXiv preprint arXiv:1606.07822 (2016).

Ordentlich2016: Ordentlich, Erik, et al.
Network-efficient distributed Word2vec training system for large vocabularies
Proceedings of the 25th ACM International on Conference on Information and Knowledge Management. ACM, 2016.

Rumelhart1986: Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams
Learning representations by back-propagating errors
nature 323.6088 (1986): 533.

References:

Levy2014: Levy O, Goldberg Y.
Neural word embedding as implicit matrix factorization
Advances in neural information processing systems. 2014: 2177-2185.

Korenius2007: Korenius T, Laurikkala J, Juhola M.
On principal component analysis, cosine and Euclidean measures in information retrieval
Information Sciences, 2007, 177(22): 4893-4905.

Boucher2015: Boucher, Thomas et al.
Aligning Mixed Manifolds
AAAI. 2015

Levy2015: Levy, Omer, Yoav Goldberg, and Ido Dagan
Improving distributional similarity with lessons learned from word embeddings
Transactions of the Association for Computational Linguistics 3 (2015): 211-225.

Jastrzebski2017: Jastrzebski, Stanisław, Damian Leśniak, and Wojciech Marian Czarnecki
How to evaluate word embeddings? On importance of data efficiency and simple supervised tasks
arXiv preprint arXiv:1702.02170 (2017).

(Appendix) Other reasons why W2V is popular

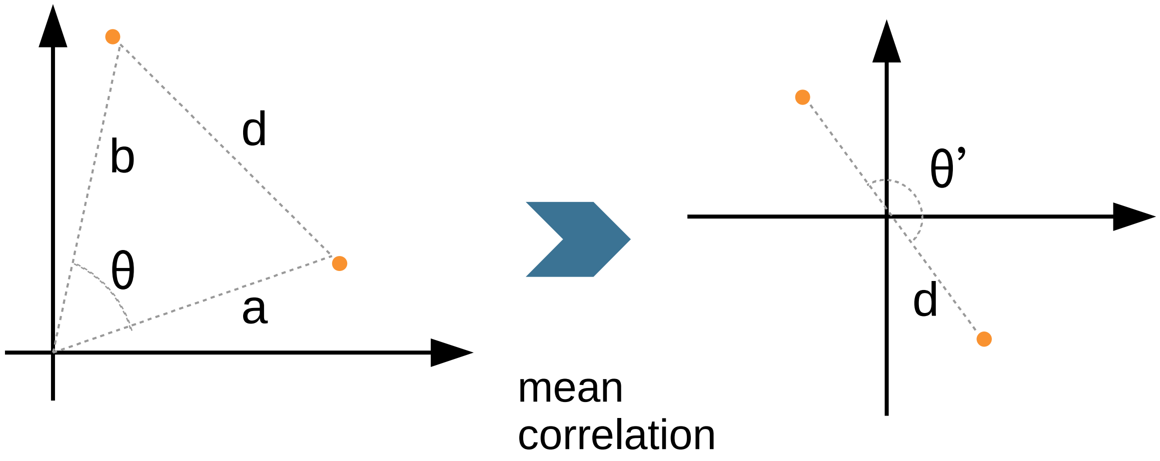
Other reasons why Word2Vec is popular:

Word representations:
used to be processed in computer:
One-hot / **hash tokenizing** (-lose of semantic information)
LSA: directly SVD on word-doc matrix (-linear model)
Bag-of-Word: meaning defined by set of labels (-low semantic accuracy, supervised)
Words co-occurrence matrix: count of word appears together (-sparse)

Therefore Word2Vec: a predictive model
preserves semantic information
dense representations for larger corpus
non-linear dimension reduction, relationship-preserving and robust
unsupervised way
a neural approach
popular in downstream applications

(Appendix) Some Comments on PCA
(Korenius2007)

Some Comments on PCA [1]



$$\|a\|^2 = \|b\|^2 = 1$$

$$\cos\theta \approx \frac{2-\|d\|^2}{2}$$, instead of $\cos\theta'$

[1]: Korenius T, Laurikkala J, Juhola M.
On principal component analysis, cosine and Euclidean measures in information retrieval
Information Sciences, 2007, 177(22): 4893-4905.