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

Content

-Background Knowledge Introductions

-Structure of our Approach

-Original Corpus Division

-Merging Approaches

  -Concatenation+PCA

  -Manifold Alignment + Vector Averaging

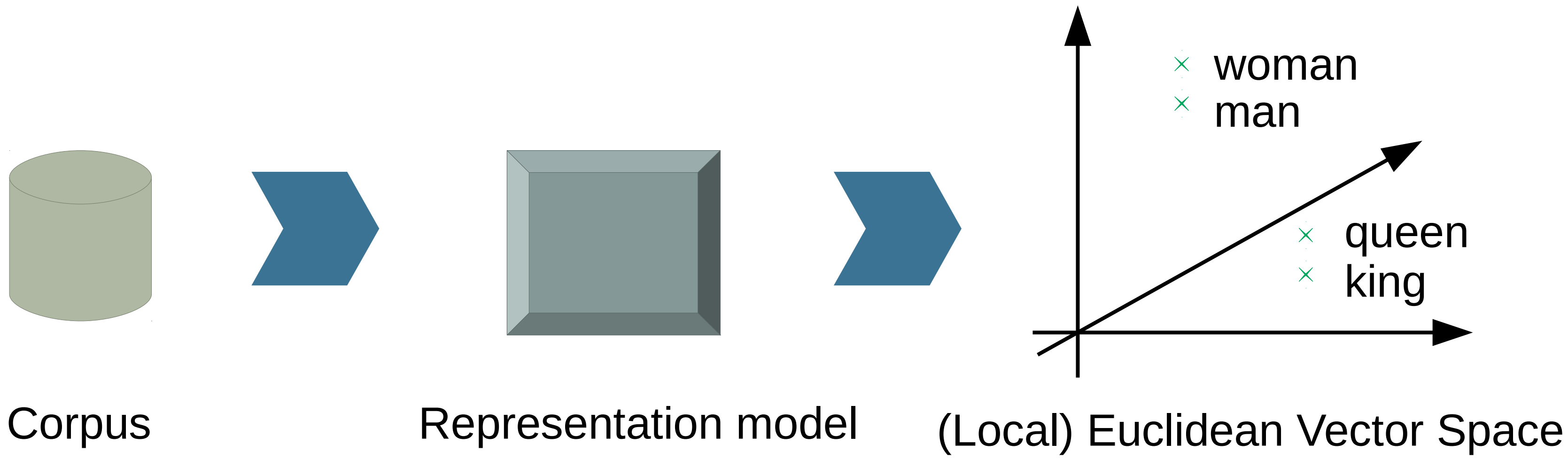
-Experiments Setup

-Performance and Timing

-Conclusion and Future Works

Background Knowledge Introduction

Word Vector Representation



Background Knowledge Introduction

Why Word2Vec:

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 is more scalable
popular in downstream applications

Background Knowledge Introduction

Challenge of Word2Vec:

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

On 14GB English Wikidump: ≈ **36 hours** !

(dim: 500 Vocabulary: 300k)

Not **scalable** w.r.t. large corpus

Background Knowledge Introduction

Skip Gram Negative Sampling

window

quick brown fox jumps over a lazy dog

central word *w*

context word *c*

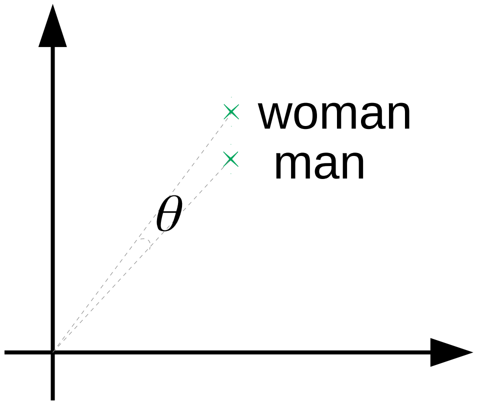
fox → predict

quick
blue
paper

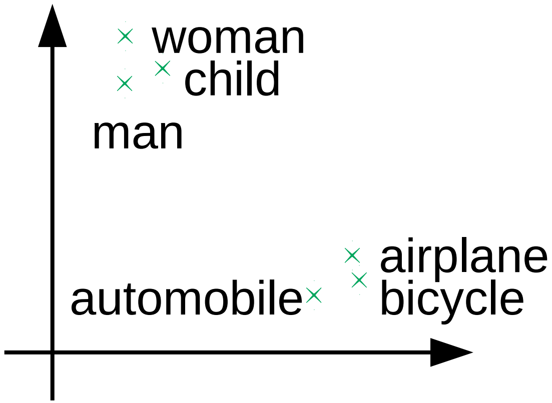
$$J(\theta)=\sum_{w\in D}\sum_{c\in l(w)}\left\{\ln\underbrace{\left[p\left((w,c)|\theta\right)\right]}_{\text{Skip Gram}}+E_{\bar{c}\in V}\ln\left[1-\underbrace{p\left((w,\bar{c});\theta\right)}_{\text{Negative Sampling}}\right]\right\}$$

Background Knowledge Introduction

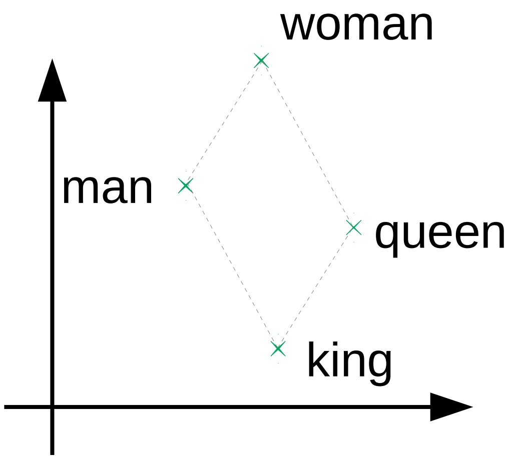
Properties of Word2Vec [1]



$\cos(\theta)$ ~ similarity



categories ⇒ clusters



analogy ⇒ parallelization

[1]: 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.

Background Knowledge Introduction

Recently work:

Optimizing SGD schema (online, synchronized) :
- HogWild!, BLAS-3, optimizing cache usage, column-wise distribution

Adaptive SGD :
- SGD-Momentum, AdaGrad, RMSProp

Heterogeneous combination:
- Combine vectors from Word2Vec, WordNet, GloVe, ConceptNet etc.

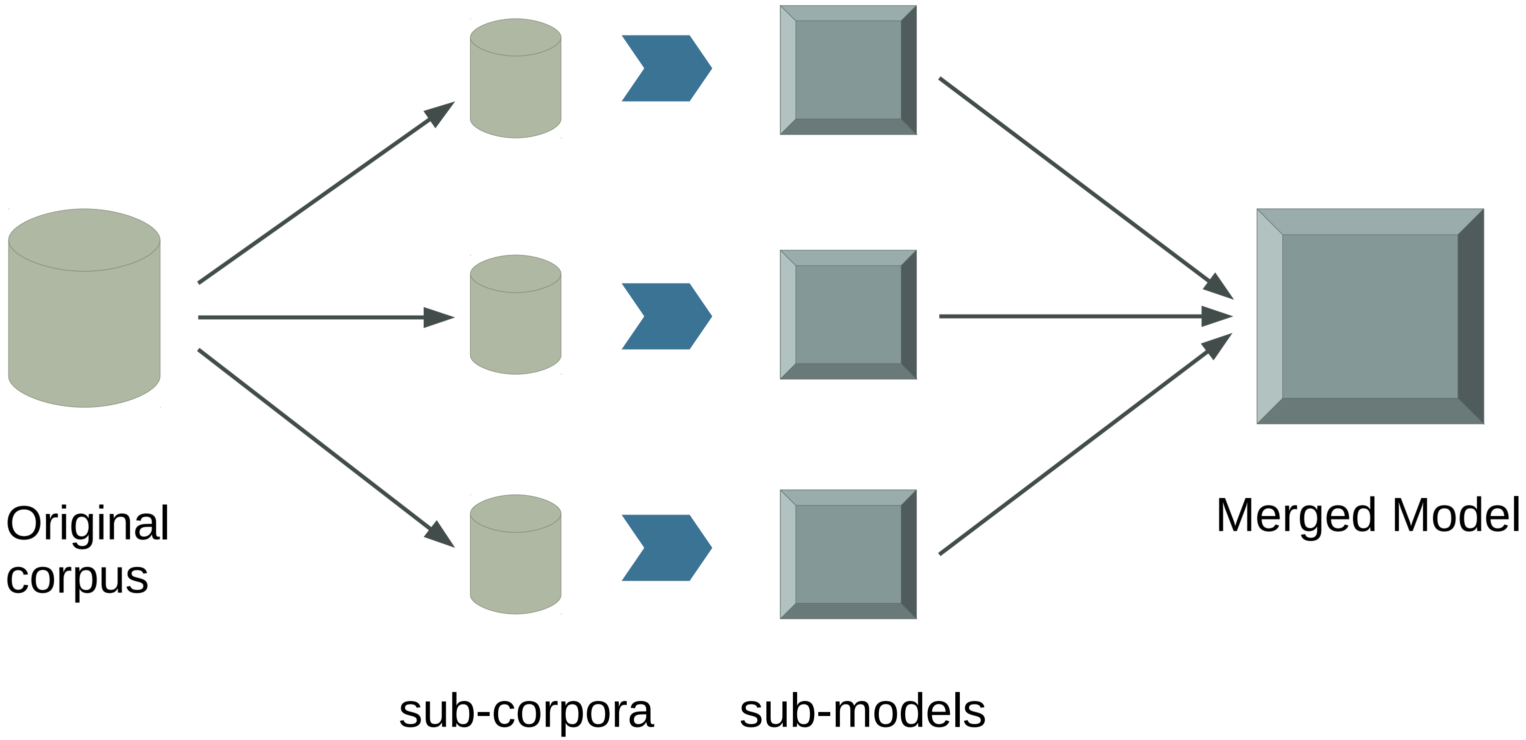
Incremental Word2Vec training

Background Knowledge Introduction

Our contribution:

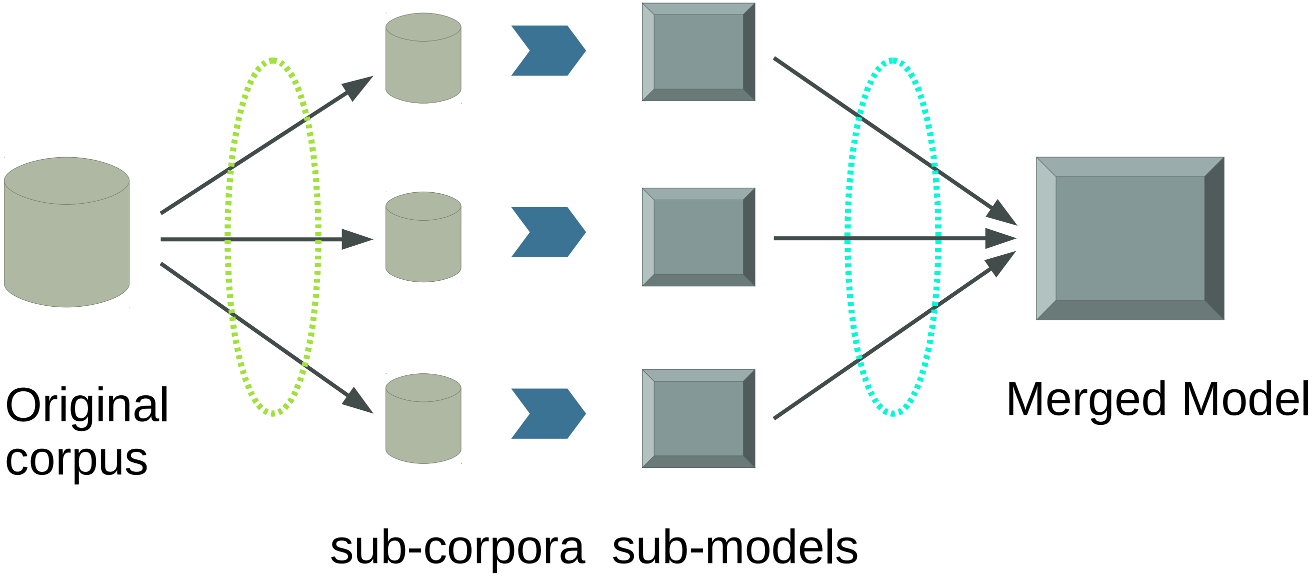
TBF

Structure of our Approach



Structure of our Approach

Research Questions:



1) How do we sample from the original corpus?

2) How do we combine sub-models?

Original Corpus Division

Criterion of Division: “Mimicking” the Original Term Distribution

objective function:

$$J(\theta)=\sum_{w\in D}\sum_{c\in l(w)}\{\ln[p((w,c)|\theta)]+E_{\bar{c}\in V}\ln[1-p((w,\bar{c});\theta)]\}$$

local optimum:

$$\frac{\partial J(\theta)}{\partial \theta}=0$$

final result [1]:

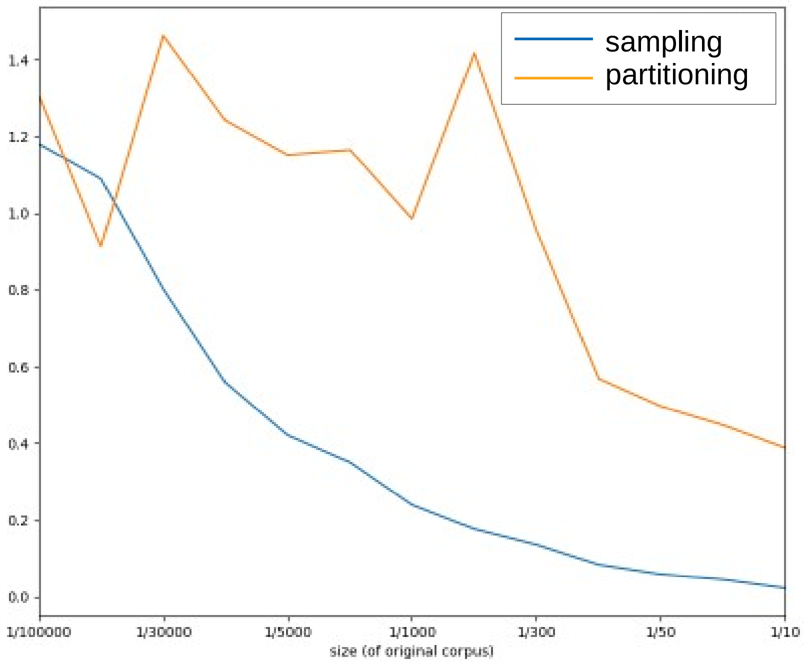
$$\boldsymbol{w}\,\boldsymbol{c}=\log\frac{N\#(w,c)}{\#(w)\#(c)}-\log k$$

Depends on distribution of words/central-contextual pairs only

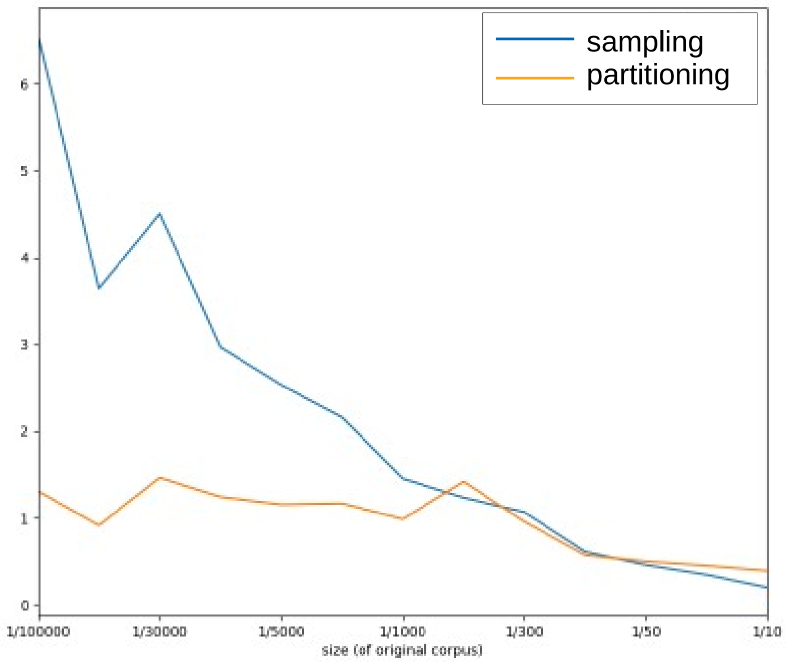
[1]: Levy O, Goldberg Y.
Neural word embedding as implicit matrix factorization
Advances in neural information processing systems. 2014: 2177-2185.

Original Corpus Division

Distribution preservation through uniform sampling



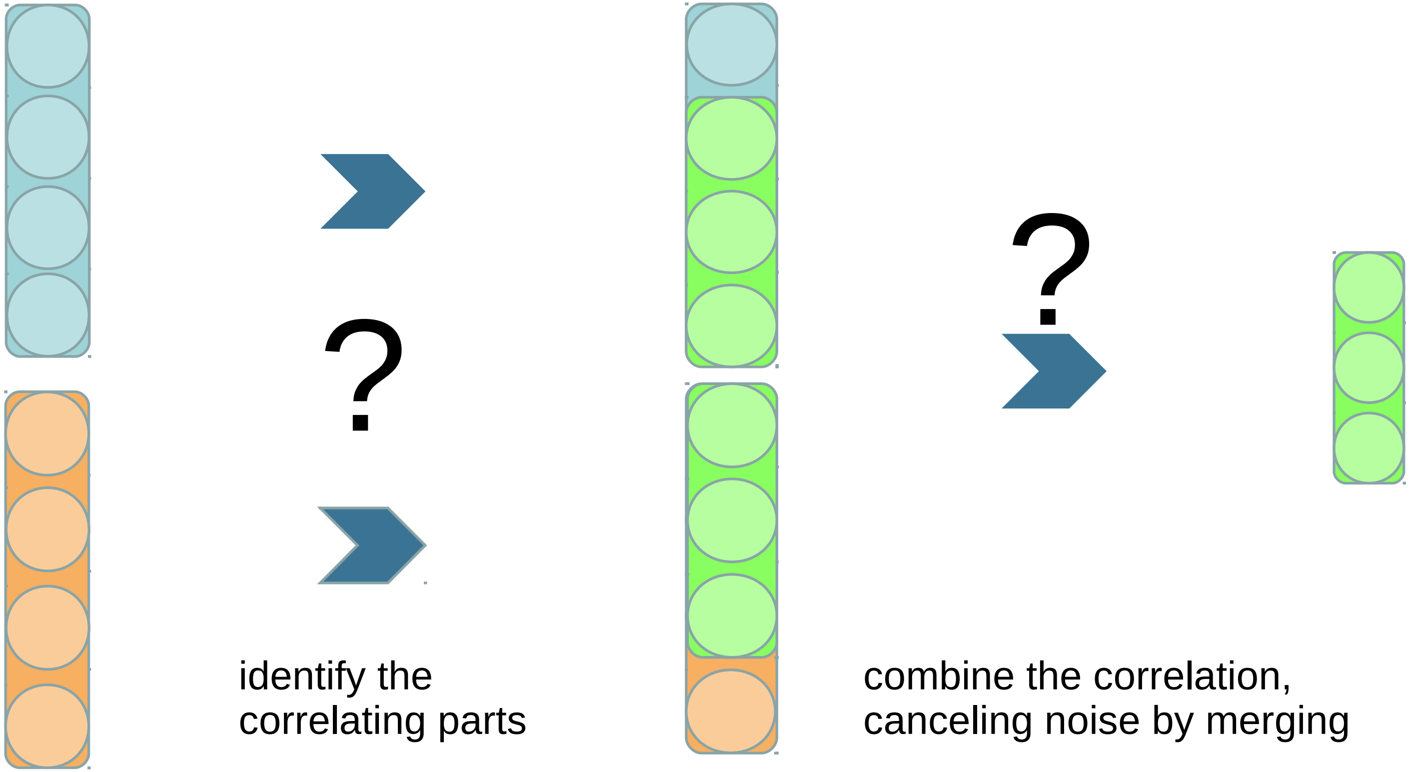
Word distribution



Central contextual words pair distribution

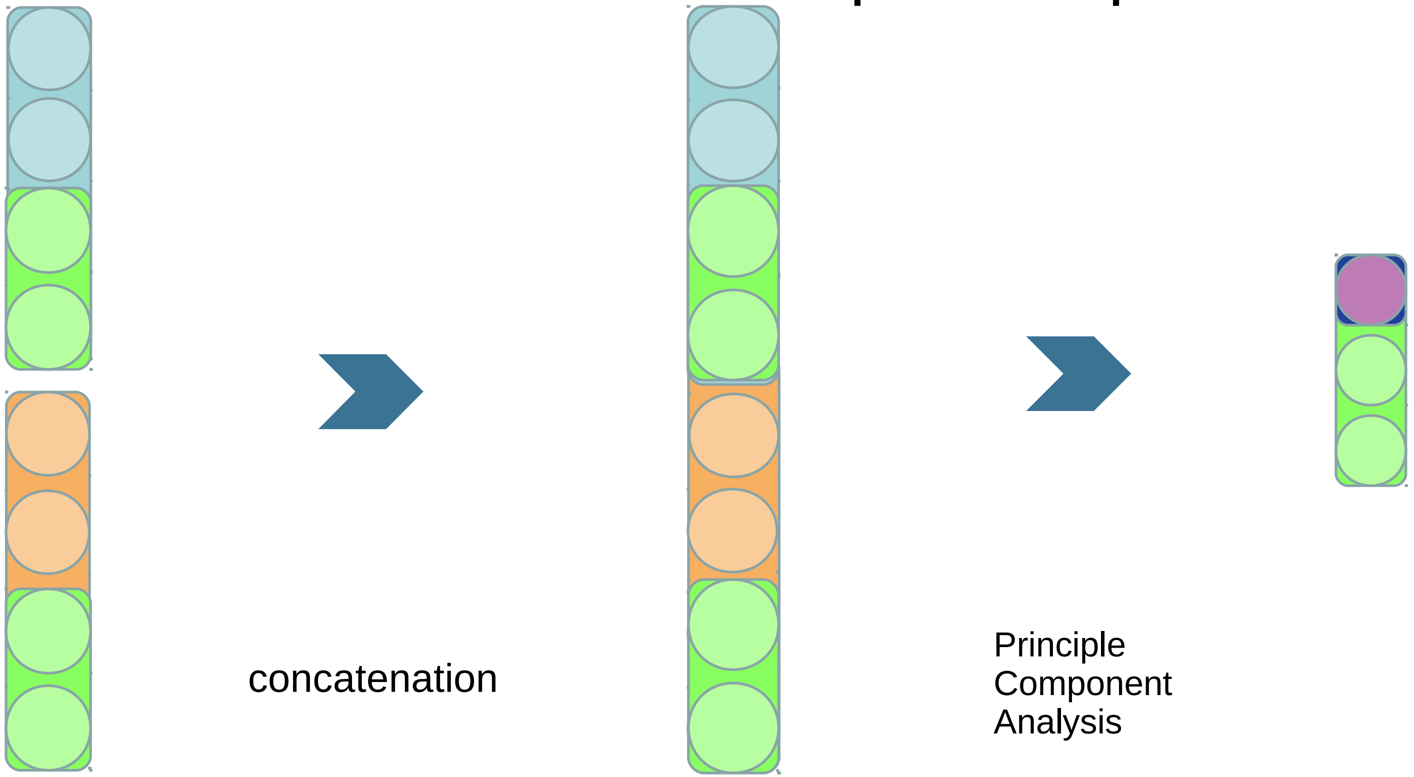
Merging Approaches

Problem so-far:



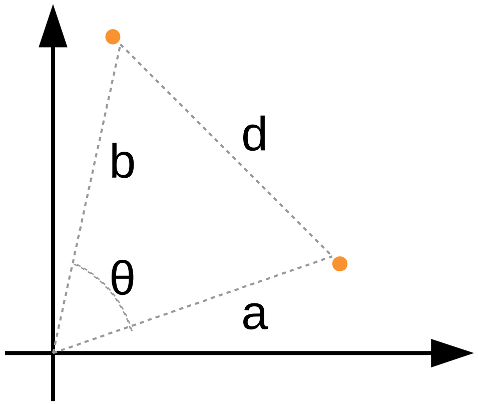
Merging Approaches

1st Approach: Concatenation + Principle Component Analysis

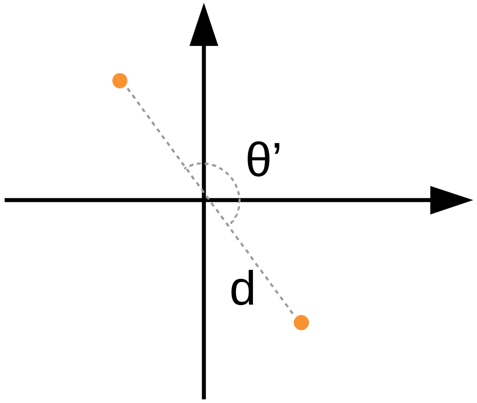


Merging Approaches

Some Comments on PCA [1]



mean
correlation



Law of cosines:

$$\|d\|^2 = \|a\|^2 + \|b\|^2 - 2\|a\|\|b\|\cos\theta$$

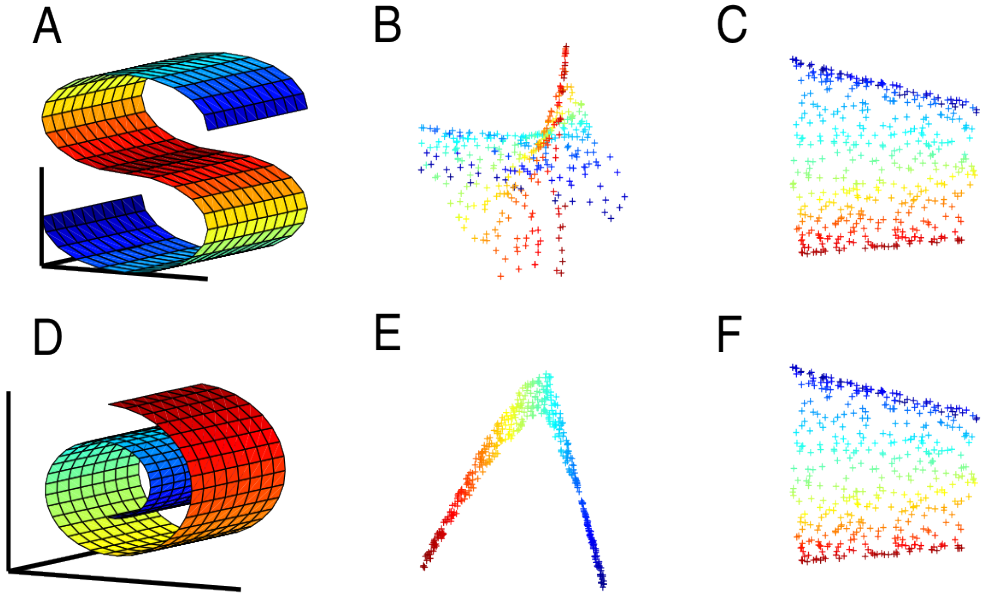
$$\|a\|^2 = \|b\|^2 = 1$$

$$\cos\theta = \frac{2 - \|d\|^2}{2}$$

[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.

Merging Approaches

2nd Approach: Manifold Alignment + Vector Averaging



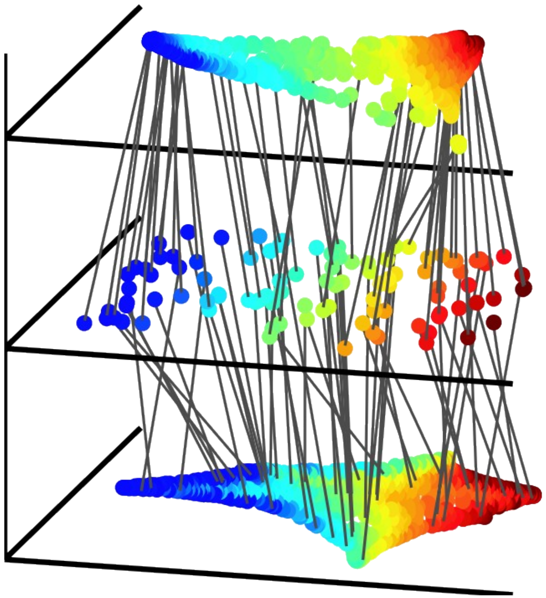
[1]

[1]: Ham, Ji Hun, Daniel D. Lee, and Lawrence K. Saul.
Learning high dimensional correspondences from low dimensional manifolds

Merging Approaches

2nd Approach: Manifold Alignment [1] + Vector Averaging

X



Y

[2]

Find a low-rank correlation of Y

X_{final}, Y_{final} = (1-μ)(low-rank reconstruction)
+μ(inter-manifold correlation)

Find a low-rank correlation of Y

[1]: Boucher, Thomas, et al. Aligning Mixed Manifolds AAAI. 2015.

[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)
- *Cleaner*: Wikipedia LineSentences cleaner in Gensim
- *Sample/Partition size*: 1%, 10% of the original corpus

Baseline Model[1]:

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

Benchmarks:

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

[1]: Jastrzebski, Stanisław, Damian Leśniak, and Wojciech Marian Czarnecki
How to evaluate word embeddings? On importance of data efficiency and simple supervised tasks

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

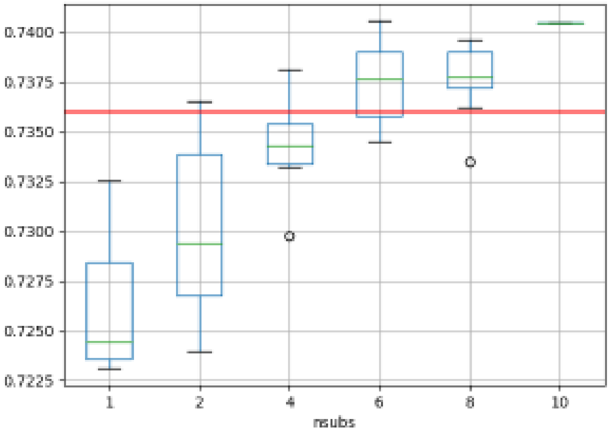
LRA and Vector average is the slowest

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

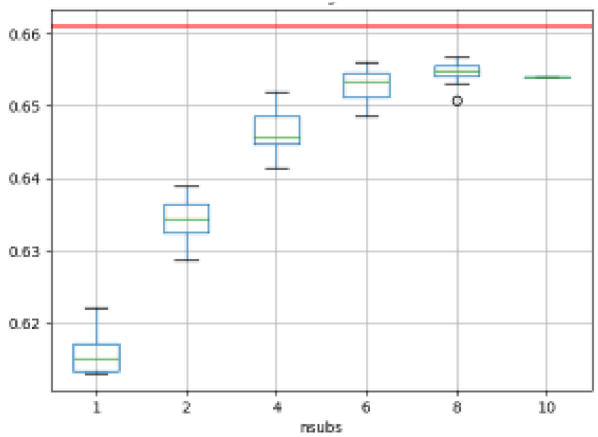
Performance and Timing

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

MEN



Google



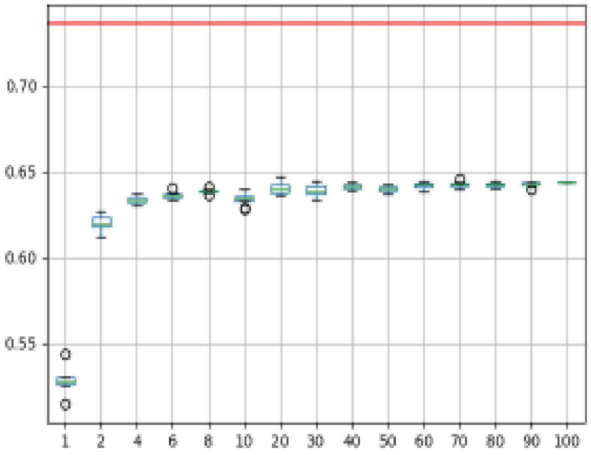
Conclusion:

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.

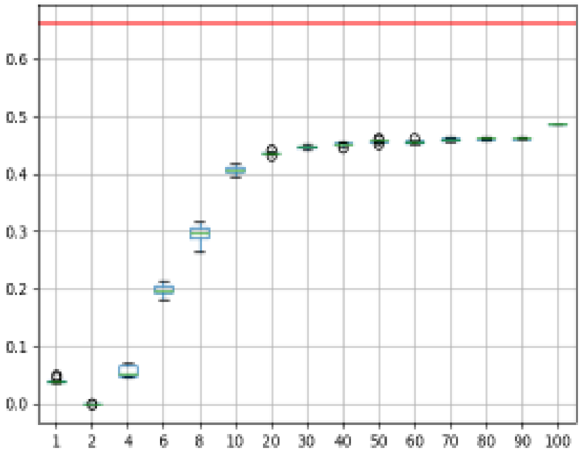
Performance and Timing

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

MEN



Google



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:

Subject name	Time Elapse (s)
10% partitioning PCA	10932
1% partitioning PCA	2093
10% sampling PCA	12852
1% sampling PCA	16293
10% sampling LRA	27385
Baseline	129015

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 Future Works

Conclusions:

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

Future works:

- Works on the go: 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