Spherical Text Embedding

Yu Meng[1], Jiaxin Huang[1], Guangyuan Wang[1], Chao Zhang[2],
Honglei Zhuang[3], Lance Kaplan[4] and Jiawei Han[1]

University of Illinois at Urbana-Champaign[1], Georgia Institute of Technology[2],
Google Research[3], U.S. Army Research Laboratory[4]

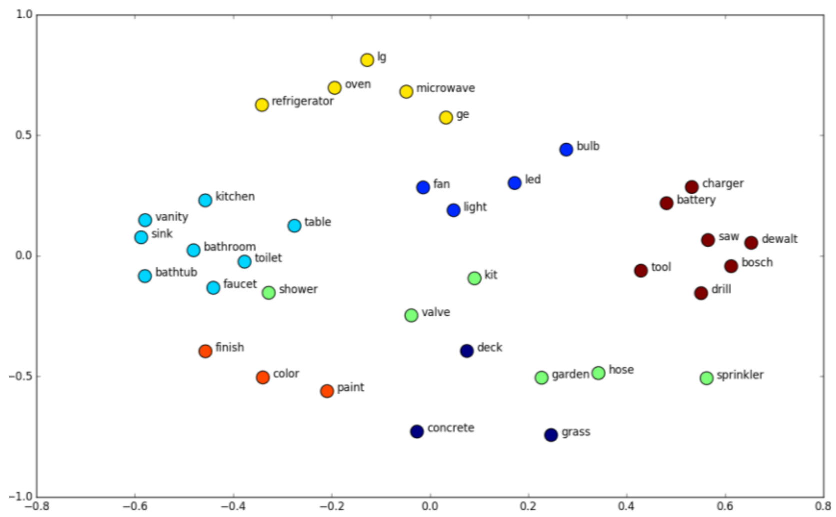
Code: https://github.com/yumeng5/Spherical-Text-Embedding

Outline

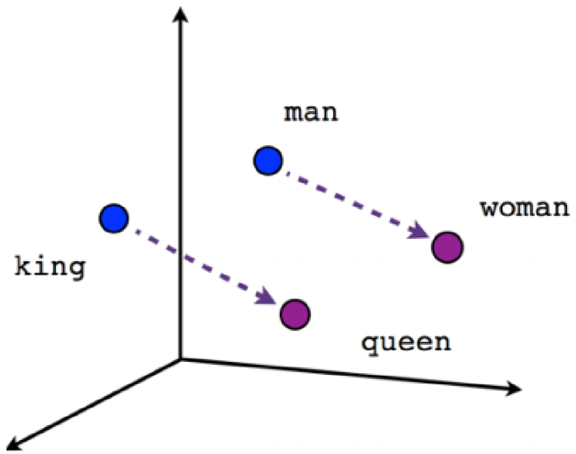
- Preliminaries
- Motivations & Introduction
- Model: Spherical Text Embedding
- Method: Optimization on the Sphere
- Experiments
- Conclusions

Preliminaries: Text Embedding

- A milestone in NLP and ML:
  - Unsupervised learning of text representations—No supervision needed
  - Embed one-hot vectors into lower-dimensional space—Address "curse of dimensionality"
  - Word embedding captures useful properties of word semantics
    - Word similarity: Words with similar meanings are embedded closer
    - Word analogy: Linear relationships between words (e.g. king - queen = man - woman)



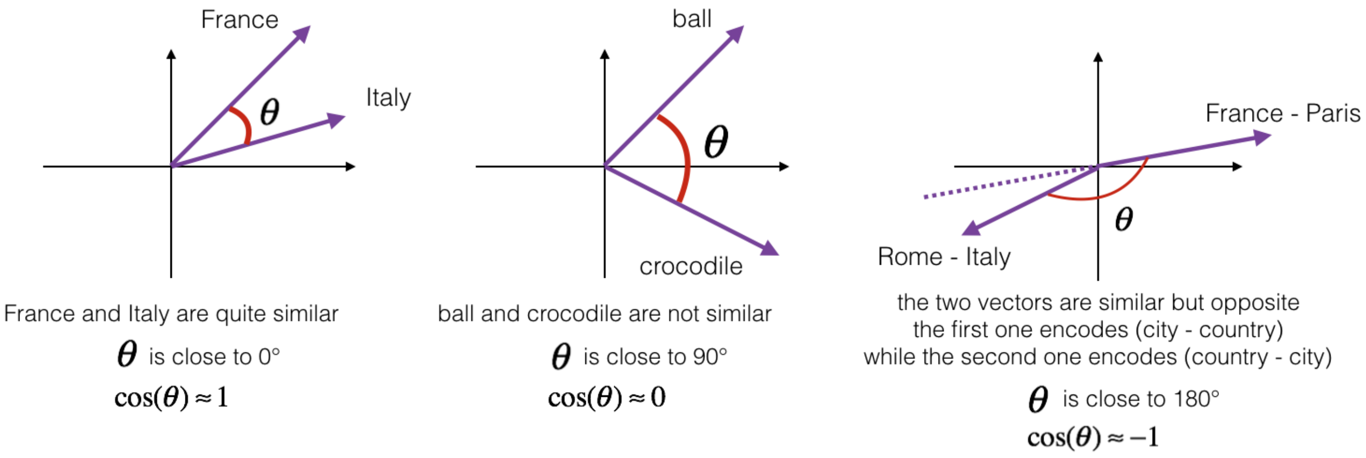
Word Similarity

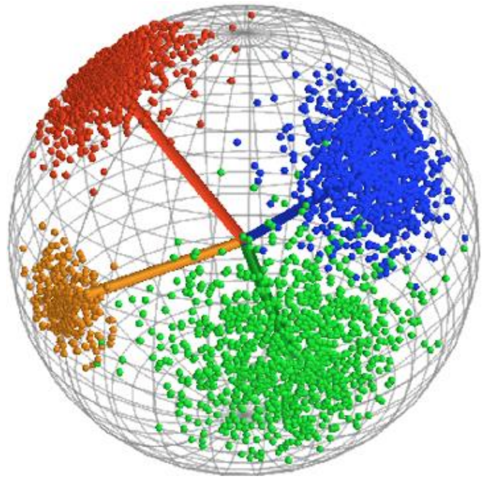


Word Analogy

Preliminaries: Text Embedding

- How to use text embeddings? Mostly directional similarity (i.e., cosine similarity)
  - Word similarity is derived using cosine similarity





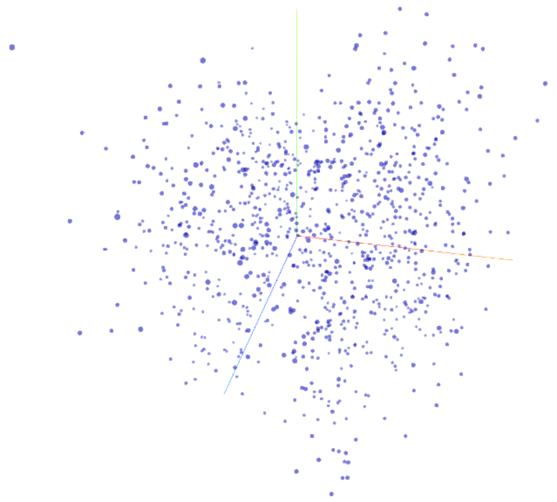
  - Word clustering (e.g. TaxoGen) is performed on a sphere
  - Better document clustering performances when embeddings are normalized and spherical clustering algorithms are used

Outline

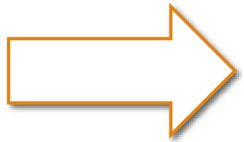
- Preliminaries
- Motivations & Introduction
- Model: Spherical Text Embedding
- Method: Optimization on the Sphere
- Experiments
- Conclusions

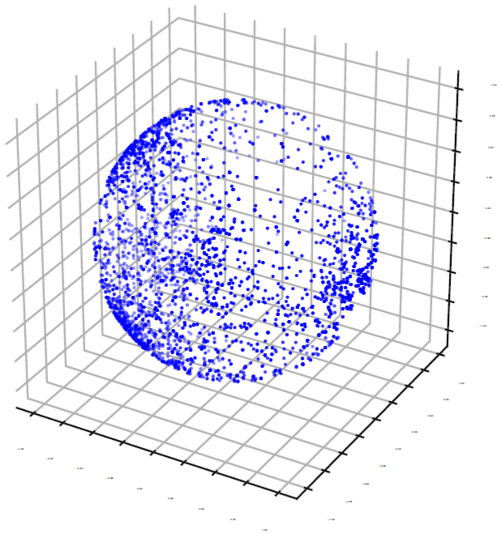
Motivations

- Issues with previous word embedding frameworks:
  - Although directional similarity has shown effective for various applications, previous embeddings (e.g. Word2Vec, GloVe, fastText) are trained in the Euclidean space
  - A gap between training space and usage space: Trained in Euclidean space but used on sphere



Embedding Training in Euclidean Space





Embedding Usage on the Sphere
(Similarity, Clustering, etc.)

Motivations

- What is the consequence of the inconsistency between word embedding training and usage space?
  - The objective we optimize during training is not really the one we use
  - Regardless of the different training objective, Word2Vec, GloVe and fastText all optimize the embedding **dot product** during training, but **cosine similarity** is what actually used in applications

Embedding dot product is optimized during training

$$p(w_O|w_I) = \frac{\exp\left({v'_{w_O}}^\top v_{w_I}\right)}{\sum_{w=1}^{W} \exp\left({v'_w}^\top v_{w_I}\right)}$$

Word2Vec

$$J = \sum_{i,j=1}^{V} f\left(X_{ij}\right)\left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij}\right)^2$$

GloVe

$$s(w,c) = \sum_{g \in \mathcal{G}_w} \mathbf{z}_g^\top \mathbf{v}_c$$

fastText

Motivations

- What is the consequence of the inconsistency between word embedding training and usage space?
  - The objective we optimize during training is not really the one we use
  - E.g. Consider two pairs of words, A: lover-quarrel; B: rock-jazz. Pair B has higher ground truth similarity than pair A in WordSim353 (a benchmark testset)
  - Word2Vec assigns higher dot product to pair B, but its cosine similarity is still smaller than pair A

Training

Usage

Metrics	A: *lover-quarrel*		B: *rock-jazz*
Dot Product	5.284	$<$	6.287
Cosine Similarity	0.637	$>$	0.628

Inconsistency

Motivations

- Apart from the training/usage space inconsistency issue, previous embedding frameworks only leverage **local contexts** to learn word representations
  - Local contexts can only partly define word semantics in unsupervised word embedding learning

If I hear someone screwing with my car (ie, setting off the **alarm**) and **taunting** me to come out, you can be very sure that my Colt Delta Elite will also be coming with me. It is not the screwing with the car that would get them **shot**, it is the potential physical **danger**. If they are **taunting** like that, it's very possible that they also intend to **rob** me and or do other physically *harmful* things. Here in Houston last year a woman heard the sound of someone …

Local contexts of "harmful"

Introduction

- In this work, we propose "Spherical Text Embedding"
  - "**Spherical**": Embeddings are trained on the unit sphere, where vector norms are ignored and directional similarity is directly optimized
  - "**Text Embedding**": Instead of training word embeddings only, we jointly train paragraph (document) embeddings with word embeddings to capture the local and global contexts in text embedding
- Contributions:
  - A spherical generative model that jointly exploits word-word (local) and word-paragraph (global) co-occurrence statistics
  - An efficient optimization algorithm in the spherical space with convergence guarantee
  - State-of-the-art performances on various text embedding applications

Outline

- Preliminaries
- Motivations & Introduction
- Model: Spherical Text Embedding
- Method: Optimization on the Sphere
- Experiments
- Conclusions

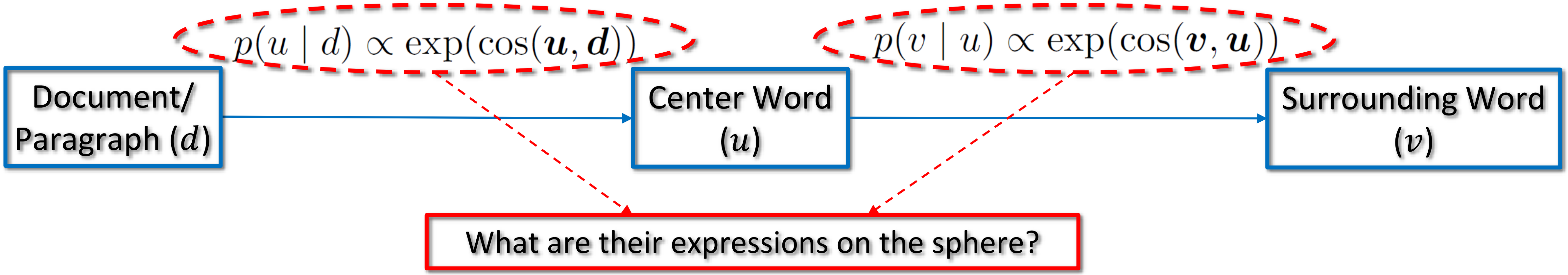
Model: Spherical Text Embedding

- We design a generative model on the sphere that follows how humans write articles:
  - We first have a general idea of the paragraph/document, and then start to write down each word in consistent with not only the paragraph/document, but also the surrounding words
  - Assume a two-step generation process:

$$\text{Document/ Paragraph } (d) \xrightarrow{p(u \mid d) \propto \exp(\cos(\boldsymbol{u}, \boldsymbol{d}))} \text{Center Word } (u) \xrightarrow{p(v \mid u) \propto \exp(\cos(\boldsymbol{v}, \boldsymbol{u}))} \text{Surrounding Word } (v)$$

Model: Spherical Text Embedding

- How to define the generative model in the spherical space?



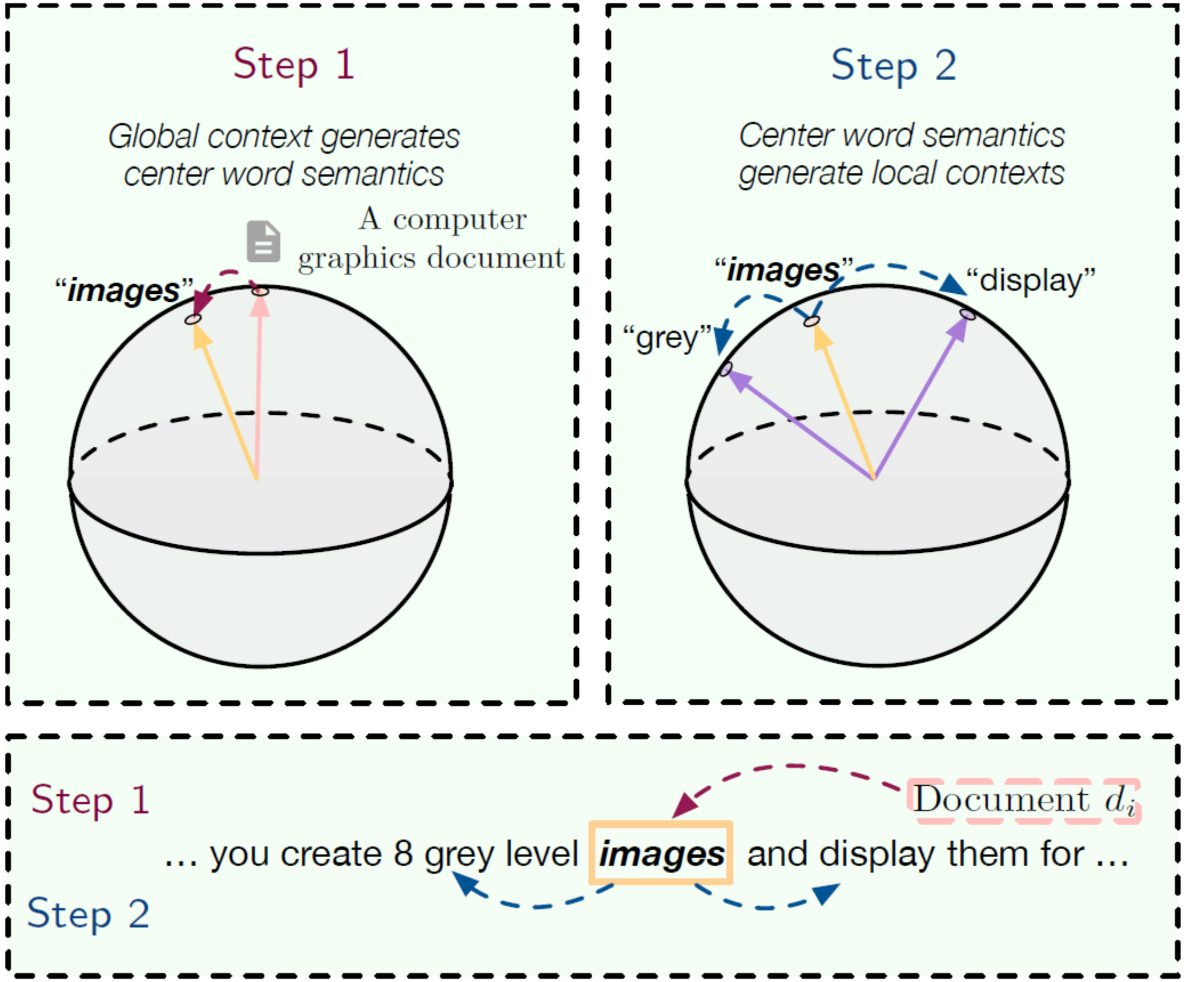
- We prove a theorem connecting the above generative model with a spherical probability distribution:

Theorem 1. When the corpus has infinite vocabulary, *i.e.*, $|V| \to \infty$, the analytic forms of $p(u \mid d) \propto \exp(\cos(\boldsymbol{u}, \boldsymbol{d}))$ and $p(v \mid u) \propto \exp(\cos(\boldsymbol{v}, \boldsymbol{u}))$ are given by the von Mises-Fisher (vMF) distribution with the prior embedding as the mean direction and constant 1 as the concentration parameter, *i.e.*,

$$\lim_{|V| \to \infty} p(v \mid u) = \text{vMF}_p(\boldsymbol{v}; \boldsymbol{u}, 1), \quad \lim_{|V| \to \infty} p(u \mid d) = \text{vMF}_p(\boldsymbol{u}; \boldsymbol{d}, 1).$$

Model: Spherical Text Embedding

- Understanding the spherical generative model



Model: Spherical Text Embedding

- Training objective:
  - The final generation probability:

$$p(v, u \mid d) = p(v \mid u) \cdot p(u \mid d) = \text{vMF}_p(\boldsymbol{v}; \boldsymbol{u}, 1) \cdot \text{vMF}_p(\boldsymbol{u}; \boldsymbol{d}, 1)$$

  - Maximize the log-probability of a real co-occurred tuple (v, u, d), while minimize that of a negative sample (v, u', d), with a max-margin loss:

$$\begin{aligned}\mathcal{L}_{\text{joint}}(\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{d}) = \max\Big(0, m - &\boxed{\log\left(c_p(1)\exp(\cos(\boldsymbol{v}, \boldsymbol{u})) \cdot c_p(1)\exp(\cos(\boldsymbol{u}, \boldsymbol{d}))\right)} \quad \text{Positive Sample} \\ &+ \boxed{\log\left(c_p(1)\exp(\cos(\boldsymbol{v}, \boldsymbol{u}')) \cdot c_p(1)\exp(\cos(\boldsymbol{u}', \boldsymbol{d}))\right)}\Big) \quad \text{Negative Sample} \\ = \max\left(0, m - \cos(\boldsymbol{v}, \boldsymbol{u}) - \cos(\boldsymbol{u}, \boldsymbol{d}) + \cos(\boldsymbol{v}, \boldsymbol{u}') + \cos(\boldsymbol{u}', \boldsymbol{d})\right),&\end{aligned}$$

Outline

- Preliminaries
- Motivations & Introduction
- Model: Spherical Text Embedding
- Method: Optimization on the Sphere
- Experiments
- Conclusions

Method: Optimization on the Sphere

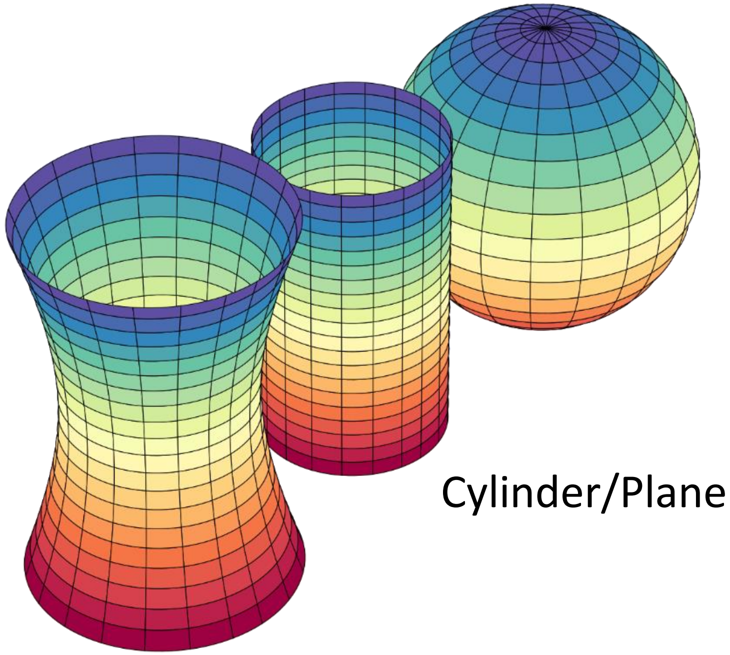
- The constrained optimization problem:

$$\min_{\boldsymbol{\Theta}} \mathcal{L}_{\text{joint}}(\boldsymbol{\Theta}) \quad \text{s.t.} \quad \forall \boldsymbol{\theta} \in \boldsymbol{\Theta} : \|\boldsymbol{\theta}\| = 1 \qquad\qquad \boldsymbol{\Theta} = \{\boldsymbol{u}_i\}\big|_{i=1}^{|V|} \cup \{\boldsymbol{v}_i\}\big|_{i=1}^{|V|} \cup \{\boldsymbol{d}_i\}\big|_{i=1}^{|\mathcal{D}|}$$

- Challenge: Parameters must be always updated on the sphere, but Euclidean optimization methods (e.g. SGD) are not constrained on a curvature space
- Need to consider the nature of the spherical space

Method: Optimization on the Sphere

- Riemannian manifold:
  - The sphere is a Riemannian manifold with constant positive curvature



Method: Optimization on the Sphere

- Riemannian optimization with Riemannian SGD:
  - Riemannian gradient:

$$\operatorname{grad} f(\boldsymbol{x}) := \left(I - \boldsymbol{x}\boldsymbol{x}^\top\right) \nabla f(\boldsymbol{x})$$

  - Exponential mapping (maps from the tangent plane to the sphere):

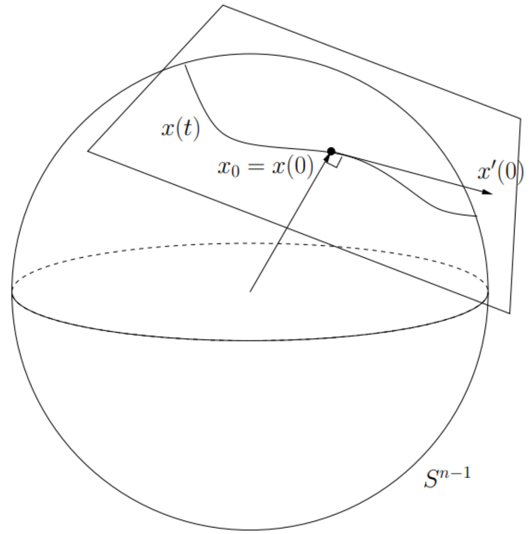
$$\exp_{\boldsymbol{x}}(\boldsymbol{z}) := \begin{cases} \cos(\|\boldsymbol{z}\|)\boldsymbol{x} + \sin(\|\boldsymbol{z}\|)\frac{\boldsymbol{z}}{\|\boldsymbol{z}\|}, & \boldsymbol{z} \in T_{\boldsymbol{x}}\mathbb{S}^{p-1} \backslash \{\mathbf{0}\}, \\ \boldsymbol{x}, & \boldsymbol{z} = \mathbf{0}. \end{cases}$$

  - Riemannian SGD:

$$\boldsymbol{x}_{t+1} = \exp_{\boldsymbol{x}_t}\left(-\eta_t \operatorname{grad} f(\boldsymbol{x}_t)\right)$$

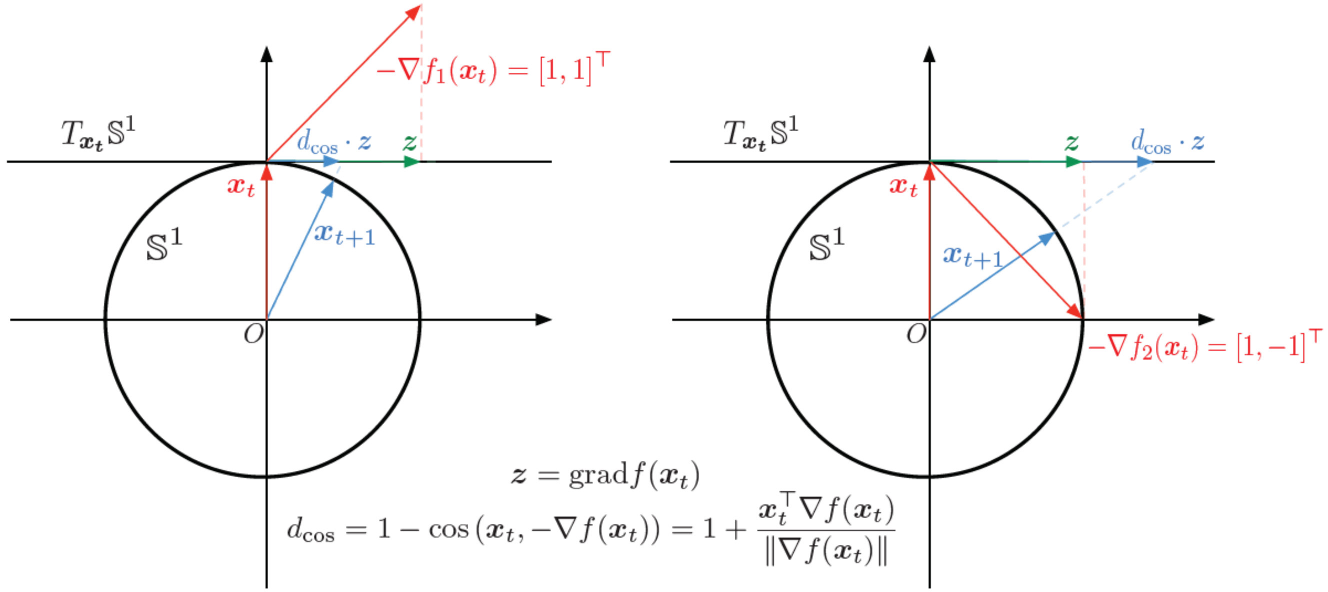
  - Retraction (first-order approximation of the exponential mapping):

$$R_{\boldsymbol{x}}(\boldsymbol{z}) := \frac{\boldsymbol{x} + \boldsymbol{z}}{\|\boldsymbol{x} + \boldsymbol{z}\|}$$



Method: Optimization on the Sphere

- Training details:
  - Incorporate angular distances into Riemannian optimization



  - Multiply the Euclidean gradient with its angular distance from the current point

$$\boldsymbol{x}_{t+1} = R_{\boldsymbol{x}_t}\left(-\eta_t\left(1 + \frac{\boldsymbol{x}_t^\top \nabla f(\boldsymbol{x}_t)}{\|\nabla f(\boldsymbol{x}_t)\|}\right)\left(I - \boldsymbol{x}_t\boldsymbol{x}_t^\top\right)\nabla f(\boldsymbol{x}_t)\right).$$

Method: Optimization on the Sphere

- Convergence guarantee of the optimization procedure:

Theorem 2. When the update rule given by Equation (7) is applied to $\mathcal{L}(\boldsymbol{x})$, and the learning rate satisfies the usual condition in stochastic approximation, *i.e.*, $\sum_t \eta_t^2 < \infty$ and $\sum_t \eta_t = \infty$, $\boldsymbol{x}$ converges almost surely to a critical point $\boldsymbol{x}^*$ and grad $\mathcal{L}(\boldsymbol{x})$ converges almost surely to 0, *i.e.*,

$$\Pr\left(\lim_{t\to\infty} \mathcal{L}(\boldsymbol{x}_t) = \mathcal{L}(\boldsymbol{x}^*)\right) = 1, \quad \Pr\left(\lim_{t\to\infty} \operatorname{grad} \mathcal{L}(\boldsymbol{x}_t) = 0\right) = 1.$$

Outline

- Preliminaries
- Motivations & Introduction
- Model: Spherical Text Embedding
- Method: Optimization on the Sphere
- Experiments
- Conclusions

Experiments

- Word similarity:
  - Train 100-d word embedding on the latest Wikipedia dump (~13G)
  - Compute embedding **cosine similarity** between word pairs to obtain a ranking of similarity
  - Benchmark datasets contain human rated similarity scores
  - The more similar the two rankings are, the better embedding reflects human thoughts
  - **Spearman's rank correlation** is used to measure the ranking similarity

Experiments

- Word similarity baselines:
  - Euclidian Space:
    - Word2Vec: Distributed representations of words and phrases and their compositionality. In NIPS, 2013
    - GloVe: Glove: Global vectors for word representation. In EMNLP, 2014
    - fastText: Enriching word vectors with subword information. TACL, 2017
    - BERT: Pre-training of deep bidirectional transformers for language understanding. In NAACL, 2019
  - Poincare Space:
    - Poincaré glove: Hyperbolic word embeddings. In ICLR, 2019
  - Spherical Space:
    - JoSE (our full model)

Experiments

- Word similarity results:

Table 1: Spearman rank correlation on word similarity evaluation.

Embedding Space	Model	WordSim353	MEN	SimLex999
Euclidean	Word2Vec	0.711	0.726	0.311
	GloVe	0.598	0.690	0.321
	fastText	0.697	0.722	0.303
	BERT	0.477	0.594	0.287
Poincaré	Poincaré GloVe	0.623	0.652	0.321
Spherical	**JoSE**	**0.739**	**0.748**	**0.339**

- Why does BERT fall behind on this task?
  - BERT learns contextualized representations, but word similarity is conducted in a context-free manner
  - BERT is optimized on specific downstream tasks like predicting masked words and sentence relationships, which have no direct relation to word similarity

Experiments

- Document Clustering:
  - Train document embedding on 20News dataset (20 classes)
  - Perform K-means and Spherical K-means (SK-means)
  - Metrics: Mutual Information (MI), Normalized Mutual Information (NMI), Adjusted Rand Index (ARI), and Purity
  - Run clustering 10 times and report the above metrics with mean and standard deviation

Experiments

- Document clustering baselines:
  - Euclidian Space:
    - Avg. Word2Vec: Use the averaged word embedding of Word2Vec as document embedding
    - SIF: Simple but tough-to-beat baseline for sentence embeddings. In ICLR, 2017
    - BERT: Pre-training of deep bidirectional transformers for language understanding. In NAACL, 2019
    - Doc2Vec: Distributed representations of sentences and documents. In ICML, 2014
  - Spherical Space:
    - JoSE (our full model)

Experiments

- Document clustering results:

Table 2: Document clustering evaluation on the 20 Newsgroup dataset.

Embedding	Clus. Alg.	MI	NMI	ARI	Purity
Avg. W2V	K-Means	1.299 ± 0.031	0.445 ± 0.009	0.247 ± 0.008	0.408 ± 0.014
	SK-Means	1.328 ± 0.024	0.453 ± 0.009	0.250 ± 0.008	0.419 ± 0.012
SIF	K-Means	0.893 ± 0.028	0.308 ± 0.009	0.137 ± 0.006	0.285 ± 0.011
	SK-Means	0.958 ± 0.012	0.322 ± 0.004	0.164 ± 0.004	0.331 ± 0.005
BERT	K-Means	0.719 ± 0.013	0.248 ± 0.004	0.100 ± 0.003	0.233 ± 0.005
	SK-Means	0.854 ± 0.022	0.289 ± 0.008	0.127 ± 0.003	0.281 ± 0.010
Doc2Vec	K-Means	1.856 ± 0.020	0.626 ± 0.006	0.469 ± 0.015	0.640 ± 0.016
	SK-Means	1.876 ± 0.020	0.630 ± 0.007	0.494 ± 0.012	0.648 ± 0.017
JoSE	K-Means	1.975 ± 0.026	0.663 ± 0.008	0.556 ± 0.018	0.711 ± 0.020
	SK-Means	$\mathbf{1.982} \pm 0.034$	$\mathbf{0.664} \pm 0.010$	$\mathbf{0.568} \pm 0.020$	$\mathbf{0.721} \pm 0.029$

- Embedding quality is generally more important than clustering algorithms:
  - Using spherical K-Means only gives marginal performance boost over K-Means
  - JoSE embedding remains optimal regardless of clustering algorithms

Experiments

- Document Classification:
  - Train document embedding on 20News (20 classes) and Movie review (2 classes) dataset
  - Perform k-NN classification (k=3)
  - Metrics: Macro-F1 & Micro-F1
  - Baselines: Same with document clustering

Experiments

- Document classification results:

Table 3: Document classification evaluation using k-NN ($k = 3$).

Embedding	20 Newsgroup Macro-F1	20 Newsgroup Micro-F1	Movie Review Macro-F1	Movie Review Micro-F1
Avg. W2V	0.630	0.631	0.712	0.713
SIF	0.552	0.549	0.650	0.656
BERT	0.380	0.371	0.664	0.665
Doc2Vec	0.648	0.645	0.674	0.678
JoSE	**0.703**	**0.707**	**0.764**	**0.765**

  - We observe similar comparison results with k from [1, 10]
  - k-NN is non-parametric and directly reflect how well the topology of the embedding space captures document-level semantics

Experiments

- Training efficiency:

Table 4: Training time (per iteration) on the latest Wikipedia dump.

Word2Vec	GloVe	fastText	BERT	Poincaré GloVe	**JoSE**
0.81 hrs	0.85 hrs	2.11 hrs	> 5 days	1.25 hrs	**0.73 hrs**

- Why is JoSE training efficient?
  - Other models' objectives contain many non-linear operations (Word2Vec & fastText's objectives involve exponential functions; GloVe's objective involves logarithm functions), while JoSE only has linear terms in the objective

Outline

- Preliminaries
- Motivations & Introduction
- Model: Spherical Text Embedding
- Method: Optimization on the Sphere
- Experiments
- Conclusions

Conclusions

- In this work, we introduce a spherical text embedding framework
  - Address the discrepancy between the training procedure and the practical usage of text embedding
  - Introduce a spherical generative model to jointly learn word and paragraph embedding
  - Develop an efficient Riemannian optimization method to train text embedding on the unit hypersphere
  - Achieves state-of-the-art performances on common text embedding applications
- Future work:
  - Exploit spherical embedding space for other tasks like lexical entailment
  - Incorporate other signals such as subword information into spherical text embedding
  - Benefit other supervised tasks: Word embedding is commonly used as the first layer in DNN
    - Add norm constraints to word embedding layer
    - Apply Riemannian optimization when fine-tuning the word embedding layer

Thanks!