Zero-Training Sentence Embedding via Orthogonal Basis

Ruslan Rakhimov Alexey Bokhovkin
Ivan Fursov Eugenia Cheskidova

Skolkovo Institute of Science and Technology

December 2018

Outline

Introduction

Approach

- Quantify new semantic meaning
- Novelty
- Significance
- Uniqueness
- Sentence vector

Experiments

- Algorithm
- STS benchmark
- Paragraph embeddings
- Supervised tasks

Improvements

- n-grams

Summary

Introduction

A **word embedding** is a learned representation for text where words that have the same meaning have a similar representation. Word embeddings are in fact a class of techniques where individual words are represented as real-valued vectors in a predefined vector space.

- Parameterized sentence embeddings
  - SkipThought (Kiros et al., 2015)
  - Sent2Vec (Pagliardini et al., 2018)
  - InferSent (Conneau et al. 2017)
  - Universal Sentence Encoder (Yang et al., 2018; Cer et al., 2018)
- Non-parameterized sentence embedding
  - SIF (Arora et al., 2017)
  - Concatenated p-mean (Ruckle et al., 2018)

Problem statement

We study a new simple and robust non-parameterized approach for building sentence representations.

Inspired by the Gram-Schmidt Process in geometric theory, authors build an orthogonal basis of the subspace spanned by a word and its surrounding context in a sentence.

Quantify new semantic meaning

Consider a word w_i in sequence and its m-neighbourhood window inside sentence. Then **Contextual Window Matrix**:

$$\boldsymbol{S}^i = [\boldsymbol{v}_{w_{i-m}}, \ldots, \boldsymbol{v}_{w_{i-1}}, \boldsymbol{v}_{w_{i+1}}, \ldots, \boldsymbol{v}_{w_{i+m}}, \boldsymbol{v}_{w_i}] \in \mathbb{R}^{d \times (2m+1)}$$

Then compute novel semantic information compared with its context: $\boldsymbol{S}^i = \boldsymbol{Q}^i \boldsymbol{R}^i$. In this way we generate new orthogonal word embedding vector:

$$\boldsymbol{Q}^i_{:,2m+1} = \boldsymbol{q}_i \rightarrow \{\boldsymbol{q}_1, \ldots, \boldsymbol{q}_{i-1}\}$$

In order to generate the embedding for a sentence, weights of *novelty*, *significance* and *uniqueness* will be assigned to each of its words.

Novelty

A word $\boldsymbol{w_i}$ is more important to a sentence if its novel orthogonal basis vector $\boldsymbol{q}_i$ is a large component in $\boldsymbol{v}_{w_i}$.

$$\alpha_n = \exp\left(\frac{r_{-1}}{\|v_{w_i}\|_2}\right) = \exp\left(\frac{r_{-1}}{\|r\|_2}\right)$$

where r is the last column of $\boldsymbol{R}_i$, and r_{-1} is the last element of r.

α_n is the exponential of the normalized distance between $\boldsymbol{v}_{w_i}$ and the subspace spanned by its context.

Significance

Intuition: The significance of a word is related to how semantically aligned it is to the meaning of its context. SVD is used in to identify principal meanings of the context.

$$\boldsymbol{S}^i = \boldsymbol{U}^i \boldsymbol{\Sigma}^i (\boldsymbol{V}^i)^T$$

A word is more important if its novel semantic meaning has a better alignment with more principal meanings.

$$\alpha_s = \frac{\|\sigma(\boldsymbol{S}^i) \odot (\boldsymbol{q}_i^\top \boldsymbol{U}^i)\|_2}{2m+1} = \frac{r_{-1}}{2m+1}$$

α_s is essentially the distance between w_i and the context hyper-plane, normalized by the context size.

Uniqueness

A corpus-wise uniqueness of *stop words* (commonly present in the corpus) is small. We compute the principal directions of the corpus and then measure their alignment with the novel orthogonal basis vector $\boldsymbol{q}_i$. A high alignment means a relatively low corpus-wise uniqueness score, and vice versa.

We want to obtain an intermediate coarse-grained sentence embedding matrix $\boldsymbol{X}^c = [\boldsymbol{g}_1, \ldots, \boldsymbol{g}_N] \in \mathbb{R}^{d \times N}$

Suppose $\boldsymbol{S} = [\boldsymbol{v}_{w_1}, \ldots, \boldsymbol{v}_{w_n}] = \boldsymbol{U\Sigma V^T}$. Then the coarse-grained embedding for the i-th sentence is defined as:

$$\boldsymbol{g}_i = \sum_{j=1}^{n} f(\sigma_j) \boldsymbol{U}_{:,j}$$

where $f(\sigma_j)$ is a monotonically increasing function.

Uniqueness

1. We then compute the top K principal vectors $\{\boldsymbol{d}_1, \dots, \boldsymbol{d}_K\}$ of $\boldsymbol{X}^c$, with singular values $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_K$
2. For each sentence, $\{\boldsymbol{d}_1, \dots, \boldsymbol{d}_K\}$ are re-ranked in descending order of their correlation with sentence matrix $\boldsymbol{S}$. The correlation is defined as $o_i = \sigma_i \|\boldsymbol{S}^T \boldsymbol{d}_i\|_2, 1 \leq i \leq K$.
3. Next, the top h principal vectors after re-ranking based on o_i are selected: $\boldsymbol{D} = \{\boldsymbol{d}_{t_1}, \dots, \boldsymbol{d}_{t_h}\}$, with $o_{t_1} \geq o_{t_2} \geq \dots o_{t_h}$ and their singular values in $\boldsymbol{X}^c$ are $\boldsymbol{\sigma}_d = [\sigma_{t_1}, \dots, \sigma_{t_h}] \in \mathbb{R}^h$.

Finally, a word w_i with new semantic meaning vector q_i in this sentence will be assigned a corpus-wise uniqueness score:

$$\alpha_u = \exp\left(-\|\boldsymbol{\sigma}_d \odot (\boldsymbol{q}_i^T \boldsymbol{D})\|_2 / h\right)$$

This ensures that common stop words will have their effect diminished since their embeddings are closely aligned with the corpus' principal directions.

Sentence vector

A sentence vector $\boldsymbol{c}_s$ is computed as a weighted sum of its word embeddings, where the weights come from three scores: a novelty score (α_n), a significance score (α_s) and a corpus-wise uniqueness score (α_u).

$$\alpha_i = \alpha_n + \alpha_s + \alpha_u$$

$$\boldsymbol{c}_s = \sum_i \alpha_i \boldsymbol{v}_{w_i}$$

Given a set of sentence vectors, removing projections onto the principal components of the spanned subspace can significantly enhance the performance on semantic similarity task. However, as each sentence may have a different semantic meaning, it could be sub-optimal to remove the same set of principal components from all sentences.

$$\boldsymbol{c}_s \leftarrow \boldsymbol{c}_s - \sum_{j=1}^{K} (\boldsymbol{d}_{t_j}^T \boldsymbol{c}_s) \boldsymbol{d}_{t_j}$$

Algorithm

Algorithm 1 Geometric Embedding (GEM)

Inputs:
  A set of sentences $\mathcal{S}$, vocabulary $\mathcal{V}$, word embeddings $\{\boldsymbol{v}_w \in \mathbb{R}^d \,|\, w \in \mathcal{V}\}$
Outputs:
  Sentence embeddings $\{\boldsymbol{c}_s \in \mathbb{R}^d \,|\, s \in \mathcal{S}\}$
for ith sentence s in $\mathcal{S}$ **do**
  Form matrix $\boldsymbol{S} \in \mathbb{R}^{d \times n}$, $\boldsymbol{S}_{:,j} = \boldsymbol{v}_{w_j}$ and w_j is the jth word in s
  The SVD is $\boldsymbol{S} = \boldsymbol{U\Sigma V}^T$
  The ith column of the coarse-grained sentence embedding matrix $\boldsymbol{X}^c_{:,i}$ is $\boldsymbol{U}(\sigma(\boldsymbol{S}))^3$
end for
Take first K singular vectors $\{\boldsymbol{d}_1, ..., \boldsymbol{d}_K\}$ and singular values $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_K$ of $\boldsymbol{X}^c$
for sentence s in $\mathcal{S}$ **do**
  Re-rank $\{\boldsymbol{d}_1, ..., \boldsymbol{d}_K\}$ in descending order by $o_i = \sigma_i \|\boldsymbol{S}^T \boldsymbol{d}_i\|_2, 1 \leq i \leq K$.
  Select top h principal vectors as $\boldsymbol{D} = [\boldsymbol{d}_{t_1}, ..., \boldsymbol{d}_{t_h}]$, with singular values $\boldsymbol{\sigma}_d = [\sigma_{t_1}, ..., \sigma_{t_h}]$.
  for word w_i in s **do**
    $\boldsymbol{S}^i = [\boldsymbol{v}_{w_{i-m}}, ..., \boldsymbol{v}_{w_{i-1}}, \boldsymbol{v}_{w_{i+1}}, ..., \boldsymbol{v}_{w_{i+m}}, \boldsymbol{v}_{w_i}]$ is the contextual window matrix of w_i.
    Do QR decomposition $\boldsymbol{S}^i = \boldsymbol{Q}^i \boldsymbol{R}^i$, let $\boldsymbol{q}_i$ and $\boldsymbol{r}$ denote the last column of $\boldsymbol{Q}^i$ and $\boldsymbol{R}^i$
    $\alpha_n = \exp(\boldsymbol{r}_{-1}/\|\boldsymbol{r}\|_2), \alpha_s = \boldsymbol{r}_{-1}/(2m+1), \alpha_u = \exp\left(-\|\boldsymbol{\sigma}_d \odot (\boldsymbol{q}_i^T \boldsymbol{D})\|_2/h\right)$
    $\alpha_i = \alpha_n + \alpha_s + \alpha_u$
  end for
  $\boldsymbol{c}_s = \sum_{\boldsymbol{v}_i \in \boldsymbol{s}} \alpha_i \boldsymbol{v}_{w_i}$
  Principal vectors removal: $\boldsymbol{c}_s \leftarrow \boldsymbol{c}_s - \boldsymbol{D}\boldsymbol{D}^T \boldsymbol{c}_s$
end for

Complexity: $$O(\underbrace{Nd(max\ length\ of\ sentence)^2}_{\text{1st cycle}} + \underbrace{dN^2}_{\text{SVD of } X^c} + \underbrace{Nndm^2 + 3NdK}_{\text{2nd cycle (QR + matvec of } q_i \text{ and } D \text{ + matvec } DD^\top c_s)})$$

Skoltech
Skolkovo Institute of Science and Technology

STS benchmark

- To predict a cosine similarity score of two sentences given a sentence pair
- To evaluate the Pearson's coefficient r between human-labeled similarity (0 - 5 points) and predictions.

model	dev (article)	test (article)
Gem + LexVec	77.1 (81.9)	**63.9** (76.5)
Gem + Glove	**78.9** (—)	63.7 (—)
Mean + LexVec	66.6 (58.78)	28.8 (50.43)
Mean + Glove	51.8 (52.4)	23.8 (40.6)

Table: STS benchmark results

One can see that our results differ from the ones obtained in the article. Likely because of a bit different precomputed embeddings.

Paragraph embeddings

Here we developed ourselves and compared three different techniques of the whole review embedding computation:

1. Treat paragraph as one sentence (without any punctuation)
2. Compute paragraph embedding as average of sentence embeddings
3. Compute sentence embeddings, then treat them as words and paragraph as a sentence

First (full)	Second	Third
65.30 (73.28)	61.75	56.50

Table: Accuracy of predicted sentiment with logistic regression; 2000 out of 25000 paragraphs (GEM embeddings on IMDb sentiments) are trained

Supervised tasks

We tested GEM on text classification, sentiment analysis and textual semantic similarity tasks. We evaluate performance using cosine similarity and 0.5 threshold.

Task	GEM[1]	DIIN[2]	ULMFiT[3]
Quora Question Pairs	66.1	89.06	—
IMDB	73.28	—	95.4

Table: Comparison of GEM and SOTA models (accuracy)

Supervised models are still superior to unsupervised methods.

[1]Here we treat paragraphs as one sentence

[2]Gong et al., 2018

[3]Howard and Ruder, 2018

n-grams

Motivation

The idea is to implement *n*-grams and to average embeddings of nearest words that stay close to each other. Then see how the performance changes.

e.g. bigrams: $\underbrace{\text{A girl}}\,\underbrace{\text{is Arya}}\,\underbrace{\text{Stark of}}\,\underbrace{\text{Winterfell NULL}}$

e.g. trigrams: $\underbrace{\text{A girl is}}\,\underbrace{\text{Arya Stark of}}\,\underbrace{\text{Winterfell NULL NULL}}$

n-grams

Performance Dependence

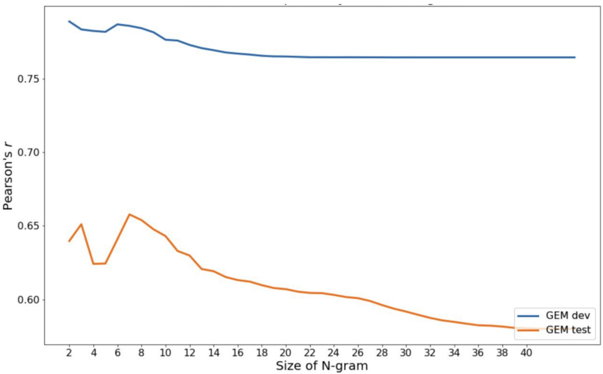

Figure: Performance dependency on *n* for *n*-grams (STS dataset)

n-grams

Speed Dependence

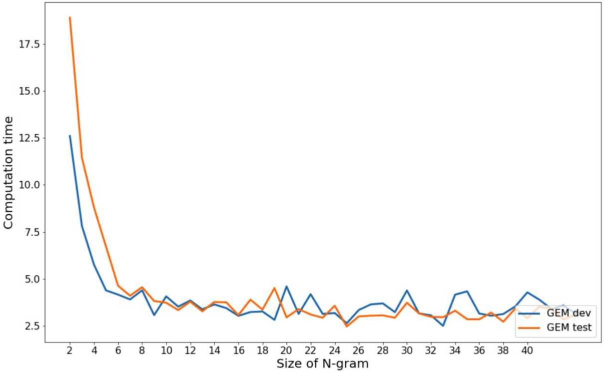

Figure: Speed dependency on n for n-grams (STS dataset)

Skoltech
Skolkovo Institute of Science and Technology

Summary

GEM has a potential use in model ensembling and fast prototyping.

Advantages

1. Fast compared to supervised methods
2. No parameters
3. Unsupervised
4. No training needed

Disadvantages

1. Time complexity grows as $O(n^2)$ where $n \doteq$ sentence length
2. Supervised methods are superior to GEM algorithm

References

Yang, Z., Zhu, C., Chen, W. (2018). Zero-training Sentence Embedding via Orthogonal Basis.

Skoltech
Skolkovo Institute of Science and Technology