ADVANCED WORD2VEC

Bogdan Borzdov,
Anna Kuzmina

Ivan Dziubenko,
Anna Tatarnikova

March 21, 2019

Outline

- Introduction
- Word2Vec
- PENN+DIEM
- Word2Gaussian
- Results
- Conclusion

Background

Word embedding is the collective name for a set of language modeling and feature learning techniques in natural language processing (NLP) where words or phrases from the vocabulary are mapped to vectors of real numbers.

Word embeddings are a way to represent natural language in computers, they capture semantics of the words.

Known approaches

- TF-IDF
- **Word2Vec**:
  - Skip-Gram
  - CBOW
- GloVe
- Many experimental methods based on Word2Vec and GloVe

Problem Formulation

Word2vec does not capture a lot of useful data and can be improved.

Idea 1: Order

These approaches do not explicitly preserve word order in their word embeddings. They ignore all order completely in their modeling and model only co-occurrence based probability in their embeddings.

Idea:

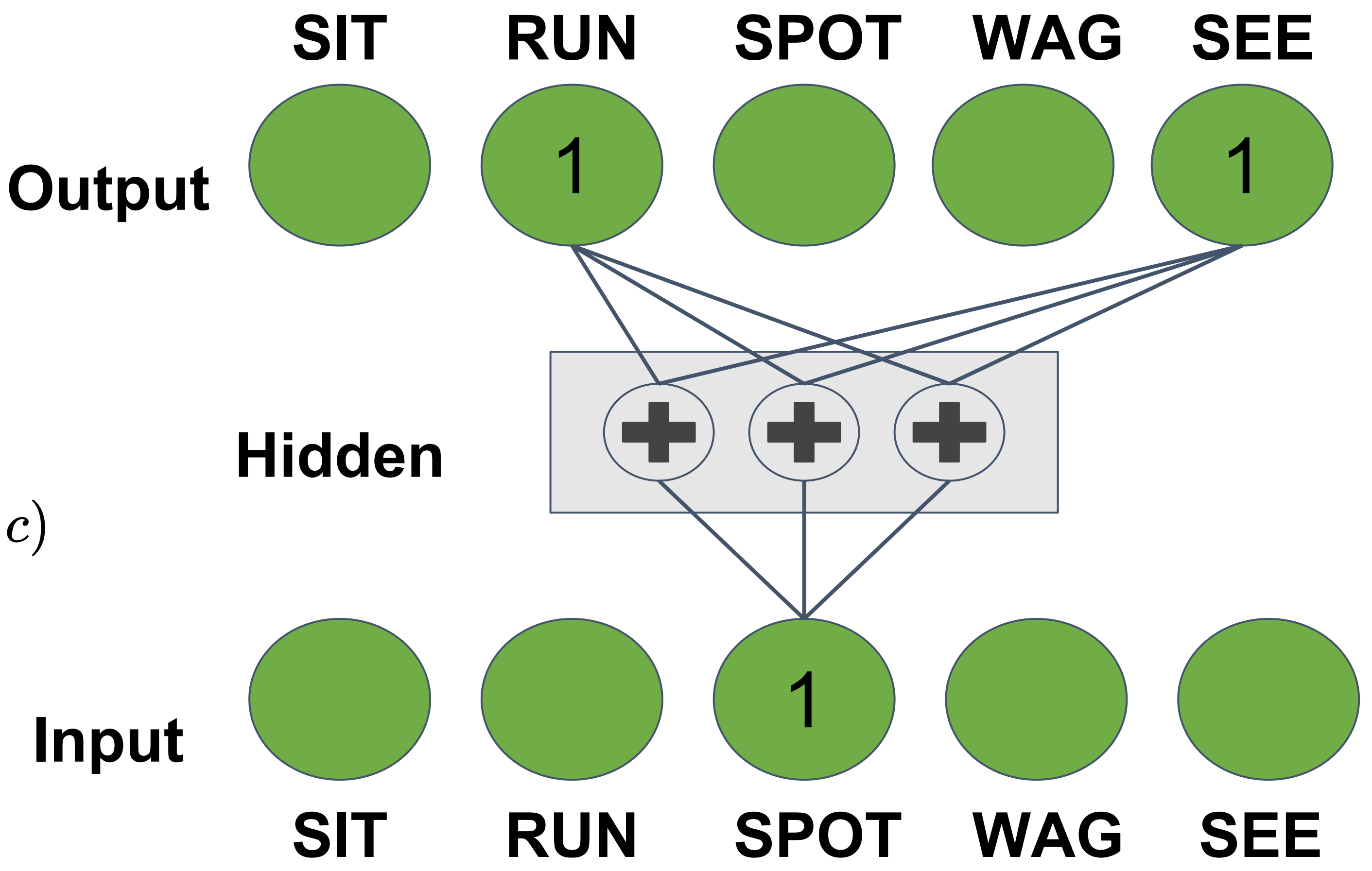
Modeling with the order in which words occur.

Word2Vec

Optimization problem:

$$\arg\max_\theta \prod_{(w,c)\in d} p(w=1|c;\theta)$$
$$\prod_{(w,c)\in d'} p(w=0|c;\theta)$$

d is a collection of context-word pairs (w, c)
c is a single word in the context;
d' is a set of random (w, c) pairs.

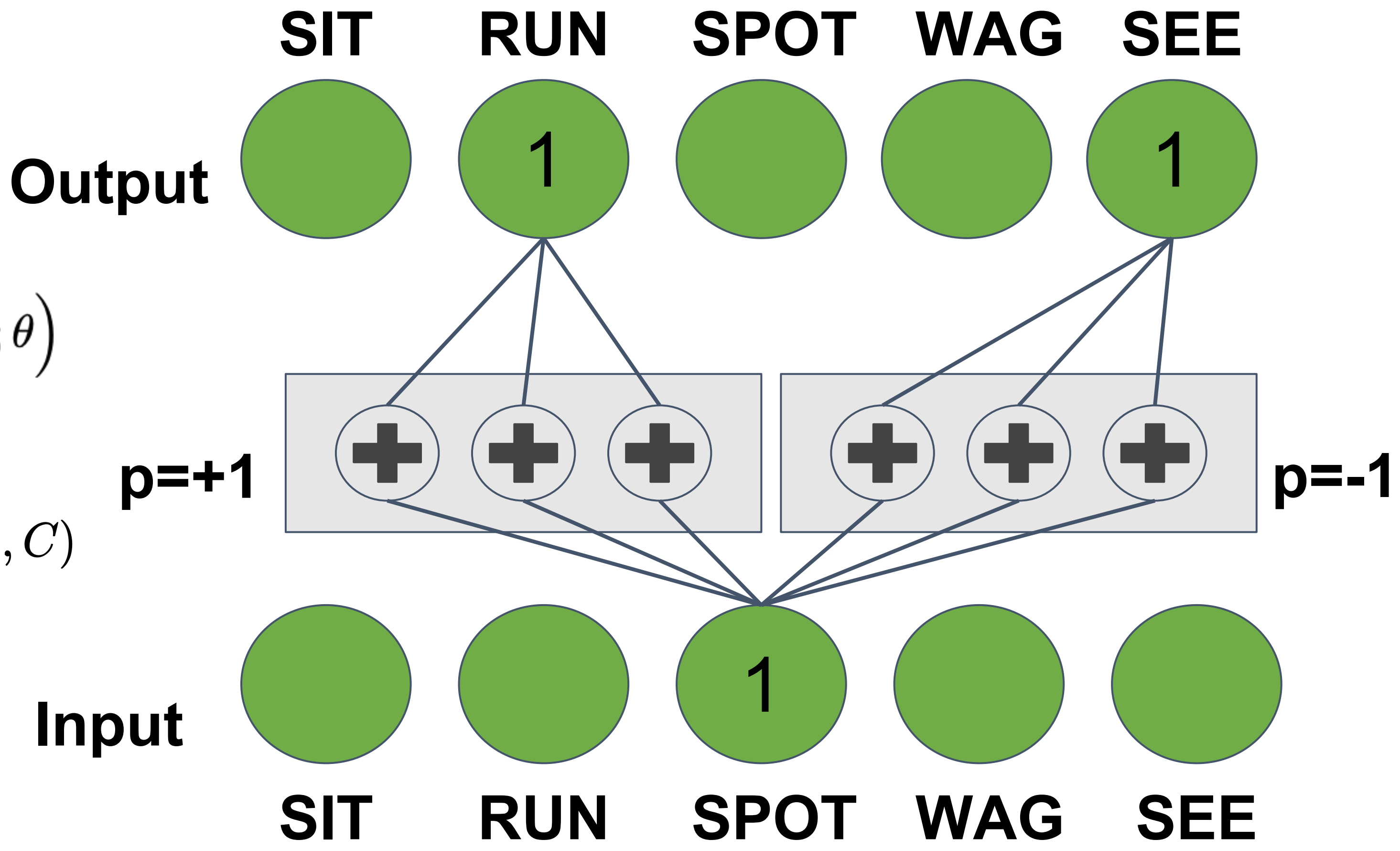


PENN

PENN (Partitioned Embedding Neural Network) optimization problem:

$$\arg\max_\theta \left(\prod_{(w,C)\in d} \sum_{-c\leq j\leq c, j\neq 0} p\left(w_j = 1 | c_j^j; \theta\right)\right.$$
$$\left.\prod_{(w,C)\in d'} \sum_{-c\leq j\leq c, j\neq 0} p\left(w_j = 0 | c_j^j; \theta\right)\right)$$

d is a collection of context-word pairs (w, C)
C is an unordered group of words in a context window;
d' is a set of random (w, C) pairs.



DIEM

DIEM (Dense Interpolated Embedding Model) generates syntactic embeddings begins by generating character embeddings using vanilla word2vec by predicting a focus character given its context.

DIEM Algorithm

Input: wordlength I, list char embeddings (e.g. the word) $char_i$, multiple M, char dim C, vector v_m

for $i = 0$ **to** $I - 1$ **do**

$s =$M $* \; i/1$

for $m = 0$ **to** $M - 1$ **do**

$d = pow(1 - (abs(s - m)) \; / \; M{,}2)$

$v_m = v_m + d * char_i$

end for

end for

Implementation

- Python with Pytorch
- Word2Vec, PENN, DIEM
- Negative sampling used for faster learning
- Adam with initial learning rate of 0.001 and 5 epoch.
- Window size: 5
- Word vector size: 500

Idea 2: Word2Vec via Gaussian Embedding

Method for learning representations in the space of Gaussian distributions.

Lexical distributed representations maps each word to a point vector in low-dimensional space.

Mapping to a density provides better:

- capturing uncertainty about a representation and its relationships
- expressing asymmetries more naturally than dot product or cosine similarity
- enabling more expressive parameterization of decision boundaries

Word2Vec via Gaussian Embedding

Optimization problem

Max-margin ranking $$L_m(w, c_p, c_n) = \max(0, m - E(w, c_p) + E(w, c_n))$$

1. For Gaussians, the inner product is defined as

$$E(w, c_p) = E\left(P_i, P_j\right) = \int_{x \in \mathbb{R}^n} \mathcal{N}\left(x; \mu_i, \Sigma_i\right) \mathcal{N}\left(x; \mu_j, \Sigma_j\right) dx = \mathcal{N}\left(0; \mu_i - \mu_j, \Sigma_i + \Sigma_j\right)$$

2. KL-divergence

$$E(w, c_n) = -E(P_i, P_j) = D_{KL}(\mathcal{N}_j || \mathcal{N}_i) = \int_{x \in \mathbb{R}^n} \mathcal{N}(x; \mu_i, \Sigma_i) \log \frac{\mathcal{N}(x; \mu_j, \Sigma_j)}{\mathcal{N}(x; \mu_i, \Sigma_i)} dx$$

Implementation

- Python with Pytorch
- Word2Vec, Word embedding via Gaussian
- AdaGrad with initial learning rate of 0.05 and 1 epoch.
- Window size: 5
- Dataset of 58772 words
- Regularization C: 2.0
- Embedding size: 50

Dataset

- We trained our models using enwik8 compressed Wikipedia articles.
- The text was pre-processed to remove non-textual elements, stop words, and rare words.

```
anarchism originate term abuse early work class
radical include digger english revolution san
culotte french revolution .
term pejorative way describe act violent mean
destroy organization society take positive label
self define anarchist .
word anarchism derive greek archon ruler chief king .
anarchism political philosophy belief ruler
unnecessary abolish differ interpretation mean .
anarchism refer related social movement advocate
elimination authoritarian institution state .
word anarchy anarchist use imply chaos nihilism
anomie harmonious authoritarian society .
place regard authoritarian political structure
coercive economic institution anarchist advocate
social relation base voluntary association
autonomous individual mutual aid self governance .
anarchism define anarchist offer positive vision
believe free society .
```

Results

Word2Vec		PENN		DIEM	
“richard”	**similarity**	**“richard”**	**similarity**	**“richard”**	**similarity**
“herman”	0.809	“earl”	0.575	“hard”	0.724
“samuel”	0.808	“sir”	0.567	“rich”	0.723
“donald”	0.801	“ludwig”	0.565	“chart”	0.667
“fr”	0.800	“wilhelm”	0.552	“regard”	0.617
“mann”	0.798	“composer”	0.551	“record”	0.615

Results

Word2Vec	
"soviet" + "union"	**similarity**
"soviet"	0.89196
"guerrilla"	0.76682
"warsaw"	0.75891
"dissident"	0.74127
"veteran"	0.73465

PENN	
"soviet" + "union"	**similarity**
"union"	0.845
"confederate"	0.66566
"slave"	0.63696
"invasion"	0.63444
"warsaw"	0.63285

Results

Word2Vec	
“boy” + “girl”	**similarity**
“boy”	0.9074
“thirteen”	0.72938
“teenage”	0.72389
“beautiful”	0.70801
“rap”	0.70469

PENN	
“boy” + “girl”	**similarity**
“girl”	0.83915
“aisha”	0.70918
“catherine”	0.70796
“margaret”	0.70564
“wicked”	0.69833

Results

Word2Vec		PENN		Word2Gaussian	
“anarchism”	**similarity**	**“anarchism”**	**similarity**	**“anarchism”**	**similarity**
“capitalist”	0.93298	“individualist”	0.8593	“anarchist”	0.8471
“capitalism”	0.90261	“anarchist”	0.84715	“communist”	0.8331
“anarchist”	0.89451	“rothbard”	0.83162	“historical”	0.7984
“anarcho”	0.89449	“metaphysical”	0.82071	“political”	0.7734
“libertarian”	0.8798	“zionism”	0.81265	“power”	0.7684

Results

Word2Vec		PENN		Word2Gaussian	
“general”	**similarity**	**“general”**	**similarity**	**“general”**	**similarity**
“partisan”	0.50995	“leader”	0.5788	“despot”	0.453
“officer”	0.49944	“davi”	0.56246	“officer”	0.441
“naval”	0.49662	“senator”	0.56038	“structure”	0.427
“subordinate”	0.48742	“deputy”	0.55958	“chief”	0.357
“cauchy”	0.41986	“commission”	0.55916	“emperor”	0.342

Conclusion

- PENN mirrors semantic and syntactic relationships as Word2Vec does.
- PENN models the order in which words occur and in some cases it shows better results in semantics than Word2Vec.
- Both methods can learn much faster with negative sampling.
- Word2Vec and PENN capture syntactic relationships worse than special syntactic methods as DIEM.
- It appears that Word2Gaussian can provide better word similarity in comparison with Word2Vec and PENN.
- Word2Gaussian directly represents notions of uncertainty and enables a richer geometry in the embedded space as it considers densities over a latent space

References

- Andrew Trask, David Gilmore, Matthew Russell. Modeling Order in Neural Word Embeddings at Scale, 2015.
- Luke Vilnis, Andrew McCallum. Word Representations via Gaussian Embedding, 2015.
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. Proceedings of the International Conference on Learning Representations, 2013a.
- Le, Quoc V. and Mikolov, Tomas. Distributed representations of sentences and documents, 2014.