Computational Linguistics

Continuous space representations for distributional semantics

Clayton Greenberg[1] and Stefan Thater[2]

[1]Graduate School of Computer Science
[2]Department of Computational Linguistics and Phonetics
[1,2]Saarland University

21 July 2016

Review: matrix factorization

Decomposing a term-document matrix improves performance for IR:

- Latent Semantic Analysis using Singular Value Decomposition

$$A = T \cdot S \cdot D^t$$

- Probabilistic Latent Semantic Analysis

$$W(t,k) \leftarrow W(t,k) \sum_{d=1}^{N} \frac{A(t,d)}{\sum_{k'=1}^{K} W(t,k')H(k',d)} H(k,d)$$

- Non-negative Matrix Factorization

$$A = W \cdot H$$

Review: other sparse representations

Distributional semantics:

- Word-Context Matrix: define a *word* by the company it keeps
- Pair-Pattern Matrix: define a *pair of words* by how they connect

Anatomy of a vector space

A term-document matrix from Landauer et al. (1998):

	c1	**c2**	**c3**	**c4**	**c5**	**m1**	**m2**	**m3**	**m4**
human	1	0	0	1	0	0	0	0	0
interface	1	0	1	0	0	0	0	0	0
computer	1	1	0	0	0	0	0	0	0
user	0	1	1	0	1	0	0	0	0
system	0	1	1	2	0	0	0	0	0
response	0	1	0	0	1	0	0	0	0
time	0	1	0	0	1	0	0	0	0
EPS	0	0	1	1	0	0	0	0	0
survey	0	1	0	0	0	0	0	0	1
trees	0	0	0	0	0	1	1	1	0
graph	0	0	0	0	0	0	1	1	1
minor	0	0	0	0	0	0	0	1	1

Anatomy of a vector space

A toy corpus from Landauer et al. (1998):

Example of text data: Titles of Some Technical Memos

c1:	*Human* machine *interface* for ABC *computer* applications
c2:	A *survey* of *user* opinion of *computer system response time*
c3:	The *EPS user interface* management *system*
c4:	*System* and *human system* engineering testing of *EPS*
c5:	Relation of *user* perceived *response time* to error measurement
m1:	The generation of random, binary, ordered *trees*
m2:	The intersection *graph* of paths in *trees*
m3:	*Graph minors* IV: Widths of *trees* and well-quasi-ordering
m4:	*Graph minors*: A *survey*

Anatomy of a vector space

The word-context matrix:

human	0	1	1	0	2	0	0	1	0	0	0	0
interface	1	0	1	1	1	0	0	1	0	0	0	0
computer	1	1	0	1	1	1	1	0	1	0	0	0
user	0	1	1	0	2	2	2	1	1	0	0	0
system	2	1	1	2	2	1	1	3	1	0	0	0
response	0	0	1	2	1	0	2	0	1	0	0	0
time	0	0	1	2	1	2	0	0	1	0	0	0
EPS	1	1	0	1	3	0	0	0	0	0	0	0
survey	0	0	1	1	1	1	1	0	0	0	1	1
trees	0	0	0	0	0	0	0	0	0	0	2	1
graph	0	0	0	0	0	0	0	0	1	2	0	2
minors	0	0	0	0	0	0	0	0	1	1	2	0

Enhancing the word-context matrix

Levy et al. (2015): let's make this thing better! (hyperparameters)

human	0	1	1	0	2	0	0	1	0	0	0	0
interface	1	0	1	1	1	0	0	1	0	0	0	0
computer	1	1	0	1	1	1	1	0	1	0	0	0
user	0	1	1	0	2	2	2	1	1	0	0	0
system	2	1	1	2	2	1	1	3	1	0	0	0
response	0	0	1	2	1	0	2	0	1	0	0	0
time	0	0	1	2	1	2	0	0	1	0	0	0
EPS	1	1	0	1	3	0	0	0	0	0	0	0
survey	0	0	1	1	1	1	1	0	0	0	1	1
trees	0	0	0	0	0	0	0	0	0	0	2	1
graph	0	0	0	0	0	0	0	0	1	2	0	2
minors	0	0	0	0	0	0	0	0	1	1	2	0

Dynamic context window (`dyn`)

Pennington et al. (2014): Weigh contexts with harmonic function: $\frac{5}{5}, \frac{4}{5}, \frac{3}{5}, \frac{2}{5}, \frac{1}{5}$

0.0	**1.0**	**0.8**	0.0	2.0	0.0	0.0	0.8	0.0	0.0	0.0	0.0
1.0	0.0	1.0	1.0	1.0	0.0	0.0	0.8	0.0	0.0	0.0	0.0
0.8	1.0	0.0	1.0	1.0	0.8	0.6	0.0	0.8	0.0	0.0	0.0
0.0	1.0	1.0	0.0	1.6	1.6	1.2	1.0	1.0	0.0	0.0	0.0
2.0	1.0	1.0	1.6	1.6	1.0	0.8	2.2	0.6	0.0	0.0	0.0
0.0	0.0	0.8	1.6	1.0	0.0	2.0	0.0	0.4	0.0	0.0	0.0
0.0	0.0	0.6	1.2	0.8	2.0	0.0	0.0	0.2	0.0	0.0	0.0
0.8	0.8	0.0	1.0	2.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.8	1.0	**0.6**	0.4	0.2	0.0	0.0	0.0	**0.8**	1.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.8	1.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.8	1.8	0.0	2.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	2.0	0.0

Subsampling (`sub`)

Remove very frequent (stop) words:
before counting → *dirty*, after counting → *clean*

human	0.0	1.0	0.8	0.0	2.0	0.0	0.0	0.8	0.0	0.0	0.0
interface	1.0	0.0	1.0	1.0	1.0	0.0	0.0	0.8	0.0	0.0	0.0
computer	0.8	1.0	0.0	1.0	1.0	0.8	0.6	0.0	0.0	0.0	0.0
user	0.0	1.0	1.0	0.0	1.6	1.6	1.2	1.0	0.0	0.0	0.0
system	2.0	1.0	1.0	1.6	1.6	1.0	0.8	2.2	0.0	0.0	0.0
response	0.0	0.0	0.8	1.6	1.0	0.0	2.0	0.0	0.0	0.0	0.0
time	0.0	0.0	0.6	1.2	0.8	2.0	0.0	0.0	0.0	0.0	0.0
EPS	0.8	0.8	0.0	1.0	2.2	0.0	0.0	0.0	0.0	0.0	0.0
trees	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**1.8**	**1.0**
graph	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.8	0.0	2.0
minors	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	2.0	0.0

Deleting rare words (`del`)

Remove very rare words: minimum number of occurrences in training corpus

human	0.0	1.0	0.8	0.0	2.0	0.0	0.0	0.8	0.0	0.0
interface	1.0	0.0	1.0	1.0	1.0	0.0	0.0	0.8	0.0	0.0
computer	0.8	1.0	0.0	1.0	1.0	0.8	0.6	0.0	0.0	0.0
user	0.0	1.0	1.0	0.0	1.6	1.6	1.2	1.0	0.0	0.0
system	2.0	1.0	1.0	1.6	1.6	1.0	0.8	2.2	0.0	0.0
response	0.0	0.0	0.8	1.6	1.0	0.0	2.0	0.0	0.0	0.0
time	0.0	0.0	0.6	1.2	0.8	2.0	0.0	0.0	0.0	0.0
EPS	0.8	0.8	0.0	1.0	2.2	0.0	0.0	0.0	0.0	0.0
graph	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0
minors	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	0.0

Shifted PMI (`neg`)

Right now: we are counting $\#(w,c)$

Some of these may happen by chance, especially if *w* and *c* are frequent.

Instead, use

$$PMI(w,c) = \log \frac{\#(w,c)|D|}{\#(w)\#(c)}$$

Better version:

$$SPPMI(w,c) = \max(PMI(w,c) - \log(k), 0)$$

Shifted PMI (`neg`)

Using *SPPMI* with $k = 1$ (not optimal)

human	0.0	0.4	0.8	0.0	0.1	0.0	0.0	0.1	0.0	0.0
interface	0.4	0.0	**0.0**	**1.0**	1.0	0.0	0.0	0.0	0.0	0.0
computer	0.8	0.0	0.0	1.0	1.0	0.8	0.6	0.0	0.0	0.0
user	0.0	1.0	1.0	0.0	1.6	1.6	1.2	1.0	0.0	0.0
system	0.1	1.0	1.0	1.6	1.6	1.0	0.8	0.2	0.0	0.0
response	0.0	0.0	0.8	1.6	1.0	0.0	1.1	0.0	0.0	0.0
time	0.0	0.0	0.6	1.2	0.8	1.1	0.0	0.0	0.0	0.0
EPS	0.1	0.0	0.0	1.0	0.2	0.0	0.0	0.0	0.0	0.0
graph	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.7
minors	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.7	0.0

Context distribution smoothing (`cds`)

All context counts are raised to the power of $\alpha = 0.75$
lowers *PMI* of *w* co-occurring with rare context *c*

human	0.0	1.0	0.8	0.0	2.0	0.0	0.0	0.8	0.0	0.0
interface	1.0	0.0	1.0	1.0	1.0	0.0	0.0	0.8	0.0	0.0
computer	0.8	1.0	0.0	1.0	1.0	0.8	0.6	0.0	0.0	0.0
user	0.0	1.0	1.0	0.0	1.6	1.6	1.2	1.0	0.0	0.0
system	2.0	1.0	1.0	1.6	1.6	1.0	0.8	2.2	0.0	0.0
response	0.0	0.0	0.8	1.6	1.0	0.0	**1.6**	0.0	0.0	0.0
time	0.0	0.0	0.6	1.2	0.8	**1.7**	0.0	0.0	0.0	0.0
EPS	0.8	0.8	0.0	1.0	2.2	0.0	0.0	0.0	0.0	0.0
graph	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.2
minors	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.2	0.0

Adding context vectors (w+c)

1st order similarity ($w_* \cdot c_*$):
the tendency of one word to co-occur with another (term-term matrix)

2nd order similarity ($w_x \cdot w_y, c_x \cdot c_y$):
the extent to which two words are replaceable

dense methods (e.g. SVD) capture 1st, sparse methods (e.g. SPPMI) do not

1st is less important, but if you have it, you can use it: $\vec{v_{cat}} = \vec{w_{cat}} + \vec{c_{cat}}$

Eigenvalue weighting (`eig`)

Suppose we decompose the word-context matrix using SVD.

First pass assignment: $W = T \cdot S, C = D^t$

A better one: $W = T \cdot \sqrt{S}, C = D^t \cdot \sqrt{S}$

More generally: $W = T \cdot S^p, C = D^t \cdot S^{1-p}$
with $p = 0.0, 0.5, 1.0$

Vector normalization (nrm)

We can normalize across rows, columns, both, or neither.
This is rows:

human	0.	0.4	0.32	0.	0.8	0.	0.	0.32	0.	0.
interface	0.46	0.	0.46	0.46	0.46	0.	0.	0.37	0.	0.
computer	0.37	0.46	0.	0.46	0.46	0.37	0.28	0.	0.	0.
user	0.	0.32	0.32	0.	0.52	0.52	0.39	0.32	0.	0.
system	0.48	0.24	0.24	0.38	0.38	0.24	0.19	0.52	0.	0.
response	0.	0.	0.31	0.62	0.38	0.	0.62	0.	0.	0.
time	0.	0.	0.26	0.52	0.35	0.74	0.	0.	0.	0.
EPS	0.3	0.3	0.	0.37	0.82	0.	0.	0.	0.	0.
graph	0.	0.	0.	0.	0.	0.	0.	0.	0.	1.
minors	0.	0.	0.	0.	0.	0.	0.	0.	1.	0.

The predictable idea

Let's decompose this matrix, too!

- We have already looked at three ways to do it!
- And all three are really slow!

The objective function for matrix factorization

The computer's answer to: how trained is my model?

- Kullback-Leibler divergence: $D(A||WH)$
- Frobenius norm: $\frac{1}{2}|A - WH|^2$

Often, this really means: how likely is my training data?

A different objective function

maximize: $p(\mathit{word}|\mathit{context})$
for each word in my training corpus

This is a language model.

But, we're not really trying to predict new words (yet).

We want to predict the $(\mathit{word}|\mathit{context})$ pairs in the training data.

A different objective function

From the analysis in Goldberg and Levy (2014):

Let $p(D = 1|w, c)$ be the probability that the given pair is present in the training data (D).

$$\ell = \operatorname*{argmax}_{\theta} \prod_{(w,c)\in D} p(D = 1|w, c; \theta)$$

And with soft-max for some $\vec{w}$ and $\vec{c}$, this becomes

$$\ell = \operatorname*{argmax}_{\theta} \sum_{(w,c)\in D} \log \sigma(\vec{w} \cdot \vec{c})$$

But to get this, we could just set all of the vectors equal to each other!

A different objective function

$$\ell = \underset{\theta}{\operatorname{argmax}} \sum_{(w,c)\in D} \log \sigma(\vec{w}\cdot\vec{c}) + \sum_{(w,c)\in D'} \log \sigma(-\vec{w}\cdot\vec{c})$$

where D' is the set of all combinations that did not occur in the training data.

D' is huge $\longrightarrow$ sample k combinations at a time.

Then, for a single $(w,c) \in D$,

Skip-gram with negative sampling objective function (Mikolov et al., 2013)

$$\ell(w,c) = \log \sigma(\vec{w}\cdot\vec{c}) + k \cdot \mathbb{E}[\log \sigma(-\vec{w}\cdot\vec{c})]$$

Not different after all

Levy and Goldberg (2014) show that this is equivalent to decomposing the PMI matrix!

Assuming that $\vec{w} \cdot \vec{c}$ terms are independent:

$$\ell(w,c) = \#(w,c) \log \sigma(\vec{w} \cdot \vec{c}) + k \cdot \#(w) \cdot \frac{\#(c)}{|D|} \log \sigma(-\vec{w} \cdot \vec{c})$$

Setting the derivative to zero, we obtain:

$$\vec{w} \cdot \vec{c} = \log \frac{\#(w,c) \cdot |D|}{\#(w) \cdot \#(c)} - \log k = PMI(w,c) - \log(k)$$

So why all the fuss about word embeddings?

Existing software packages have default values for the hyperparameters that are better than using the plain word-context matrix:

- *word2vec*: `http://code.google.com/p/word2vec/`
- *GloVe* : `http://nlp.stanford.edu/projects/glove/`

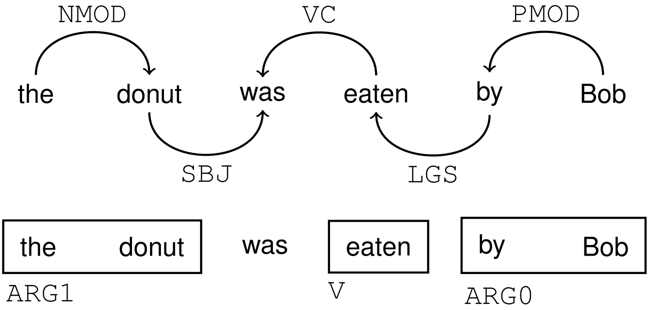
A different word-context matrix for every task?

It's doable, but expensive and not psycholinguistically motivated.

What if we could store all distributional information in one structure?

Distributional Memory (Baroni and Lenci, 2010): store values for word-link-word triples in a third order tensor.

Syntactic or semantic links (Sayeed and Demberg, 2014)



Applications of Distributional Memory

The $W_1 \times LW_2$ space:

1. Similarity Judgements
2. Synonym Detection
3. Noun Categorization
4. Selectional Preferences or Thematic Fit (Greenberg et al., 2015)

Applications of Distributional Memory

The $W_1 W_2 \times L$ space:

1. Solving Analogy Problems
2. Relation Classification
3. Qualia Extraction
4. Predicting Characteristic Properties

Summary

1. Motivation
2. Enhancing the word-context matrix
   - Pre-processing hyperparameters
   - Association metric hyperparameters
   - Post-processing hyperparameters
3. Word embeddings
4. Applications of continuous space representations

References I

Baroni, M. and Lenci, A. (2010). Distributional memory: A general framework for corpus-based semantics. *Computational Linguistics*, 36(4):673–721.

Goldberg, Y. and Levy, O. (2014). word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*.

Greenberg, C., Sayeed, A., and Demberg, V. (2015). Improving unsupervised vector-space thematic fit evaluation via role-filler prototype clustering. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 21–31, Denver, Colorado. Association for Computational Linguistics.

Landauer, T. K., Foltz, P. W., and Laham, D. (1998). An introduction to latent semantic analysis. *Discourse processes*, 25(2-3):259–284.

References II

Levy, O. and Goldberg, Y. (2014). Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems*, pages 2177–2185.

Levy, O., Goldberg, Y., and Dagan, I. (2015). Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Pennington, J., Socher, R., and Manning, C. (2014). GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

References III

Sayeed, A. and Demberg, V. (2014). Combining unsupervised syntactic and semantic models of thematic fit. In *Proceedings of the first Italian Conference on Computational Linguistics (CLiC-it 2014)*.