A second look at word embeddings

W4705: Natural Language Processing

Fei-Tzin Lee

October 23, 2019

Last time...

- Distributional representations (SVD)
- word2vec
- Analogy performance

This time

- Homework-related topics
- Non-homework topics

Outline

1. GloVe
2. How good are our embeddings?
3. The math behind the models?
4. Word embeddings, new and improved

Outline

A recap of GloVe

Our motivation:

- SVD places too much emphasis on unimportant matrix entries
- word2vec never gets to look at global co-occurrence statistics

Can we create a new model that balances the strengths of both of these to better express linear structure?

Setting

We'll use more or less the same setting as in previous models:

- A corpus D
- A word vocabulary V from which both target and context words are drawn
- A co-occurrence matrix M_{ij}, from which we can calculate conditional probabilities $P_{ij} = M_{ij}/M_i$

The GloVe objective, in overview

Idea: we want to capture not individual probabilities of co-occurrence, but *ratios* of co-occurrence probability between pairs (w_i, w_k) and (w_j, w_k).

Why? This essentially lets us probe for the specific areas in which w_i and w_j differ.

For example, $P(house|cat)$ may be approximately equal to $P(house|dog)$, but the ratio $P(catnip|cat)/P(catnip|dog)$ will be very high.

Deriving the GloVe objective function

We start with the most general possible model, $F(w_i, w_j, c_k) = \frac{P_{ik}}{P_{jk}}$, and then specify additional desired properties:

- Differences between words should be encoded as vector differences
- We want a simple linear structure
- Since M is symmetric, we should be able to swap word-context roles within a pair

We end up with the objective

$$J = \sum_{i,j \in V} f(M_{ij}) \left(\vec{w}_i \cdot \vec{c}_j + b_i + b'_j - \log M_{ij} \right)^2$$

GloVe and skipgram

Recall the skipgram objective is to maximize corpus probability (equivalently, minimize log corpus probability).

Outline

1. GloVe
2. How good are our embeddings?
3. The math behind the models?
4. Word embeddings, new and improved

How can we tell whether our embeddings are good or not?

Two main flavors of evaluation: intrinsic and extrinsic.

- Intrinsic - direct evaluation of the contents of the embedding space
- Extrinsic - evaluation of impact of embeddings on an external task

These can be orthogonal.

Intrinsic evaluation measures

Designed to test what information is contained in the space

- Word similarity
- Analogy solving
- Word relatedness

Extrinsic evaluation measures

Essentially, how well do our embeddings do as input to downstream models?

- Typical scenario: if I plug in these embeddings as input to my classifier as opposed to those other ones, which do better?

Open problems with intrinsic evaluation

- Datasets are often overly homogeneous (more recent datasets address this)
- Intrinsic performance may not correlate well with downstream performance, and downstream performance on different tasks may not correlate either [Schnabel et al., 2015]

Developing more insightful intrinsic evaluation measures is a promising avenue of research.

Outline

1. GloVe
2. How good are our embeddings?
3. The math behind the models?
4. Word embeddings, new and improved

Theoretical foundations

We can evaluate and measure performance of word embeddings in different ways, and empirically analyze their properties.

But *why* do they work the way they do?

Theoretical foundations

There still hasn't been that much work on theoretical models for word embeddings!

- Arora et al. [2016] proposed a random walk model for text
- Hashimoto et al. [2016] suggested a metric embedding perspective

We'll focus on the former in this lecture, as more subsequent work has been based on it.

In overview

At a high level, the approach goes something like this:

- Specify a generative model for text
- Show that under this model word embedding methods will recover the true distribution
- Analyze the properties of the model to yield other nice results

An expanded setting

Here our familiar old setting has some new twists. We have roughly the same components, but some in a slightly different form:

- A text corpus D
- A vocabulary V of words
- A *true latent vector* $\vec{w} \in \mathbb{R}^d$ for each word $w \in V$
- A *latent context vector* $\vec{c}_t \in \mathbb{R}^d$ that moves slowly over time

But in addition to these, we also have an entirely new apparatus to relate them

A generative model for text

The key to our understanding here lies in specifying a model of how our data was generated in the first place:

- We assume the word vectors are drawn i.i.d. from the d-dimensional spherical Gaussian, then randomly scaled
- At each timestep t, a word is drawn with probability

$$Pr[w|\vec{c}_t] \propto \exp(\vec{w}, \vec{c}_t)$$

- The hidden context vector $\vec{c}_t$ undergoes a slow random walk over time

Conditions

In order to proceed, we need to specify a few conditions.

- For almost any choice of c, the normalization factor $Z_c = \sum_w \exp(\vec{w} \cdot \vec{c})$ is close to a constant Z (within $(1 \pm \epsilon)Z$ for small ϵ)
- The matrix of word vectors behaves approximately like a random matrix

These follow naturally from the specified generative model.

PMI and compositionality

Under these assumptions, it is possible to show that

- PMI of two words is approximately proportional to their inner product
- Additive compositionality should approximately hold when d is much smaller than the vocabulary size.

But let's go back to those assumptions.

The most important assumption is that the word vectors are roughly uniformly distributed in space. But is this true? How might we test it?

Verification

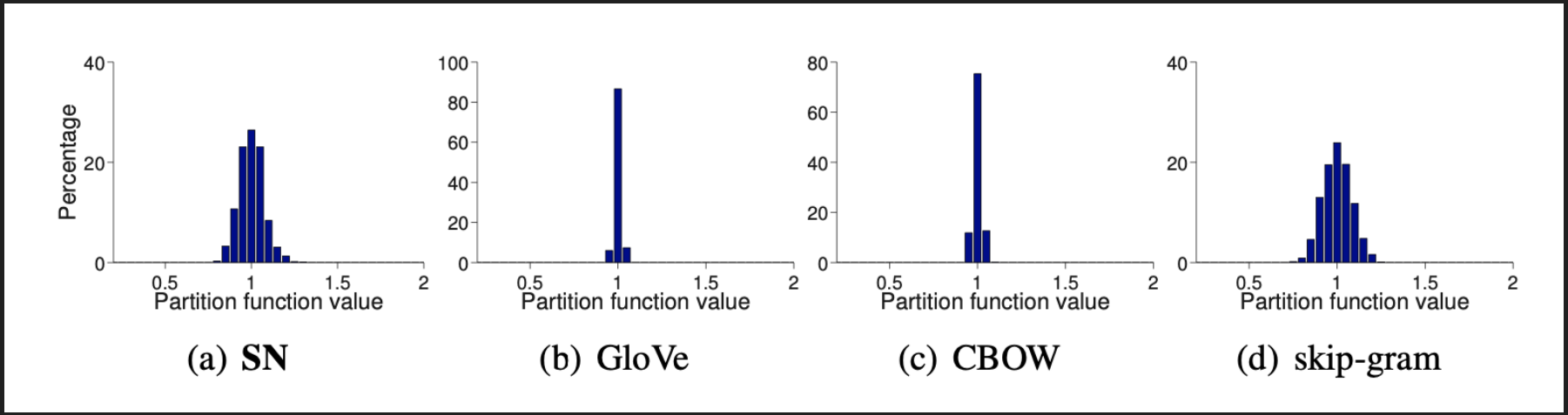

Figure: Sampled partition functions.

This seems convincing enough. But...

A counterexample

Mimno and Thompson [2017] show that the distribution of embeddings produced by skip-gram with negative sampling lies approximately within a narrow cone!

But SGNS has similar performance on analogy tasks to other models. So clearly that's not all that's going on here.

In practice?

- Theoretical models can offer important insight, but they necessarily rest on foundational assumptions
- Empirical verification of the assumptions is also important!

In conclusion?

Some promising theoretical work, but with definite avenues open for expansion

Outline

1. GloVe
2. How good are our embeddings?
3. The math behind the models?
4. Word embeddings, new and improved

The picture up to now...

Static word embeddings

- One vector per word
- *One point in space* per word
- Vectors are learned using (order-independent) co-occurrence

This is a pretty limited view!

Context

The word embedding methods we have looked at so far are what one might term 'static' word embeddings: methods that assign one embedding to each word in your vocabulary.

But in practice, words do not have a single fixed meaning in all contexts.

- Polysemous words: 'my *train* leaves at seven', 'I'll *train* thirty-six models in the next homework assignment'

Context

The word embedding methods we have looked at so far are what one might term 'static' word embeddings: methods that assign one embedding to each word in your vocabulary.

But in practice, words do not have a single fixed meaning in all contexts.

- Polysemous words: 'my *train* leaves at seven', 'I'll *train* thirty-six models in the next homework assignment'
- Even monosemous words: 'I *love* word embeddings' vs 'I *love* going to the dentist...'

Contextual embeddings

Thus, a new class of word representation has become popular over the last couple years: vectors generated by *contextual* models.

- Instead of one vector per unique word in vocabulary (word type), one vector per *instance* of each word (word token)
- The vector for a word *depends on context*

How?

Generally speaking, through some variant of language modeling.

- Mid-2018: ELMo (BiLSTM language model)
- Late 2018: **BERT** (Transformer-based 'masked language model')
- 2019: GPT-2, XLNet, etc.

The Transformer architecture

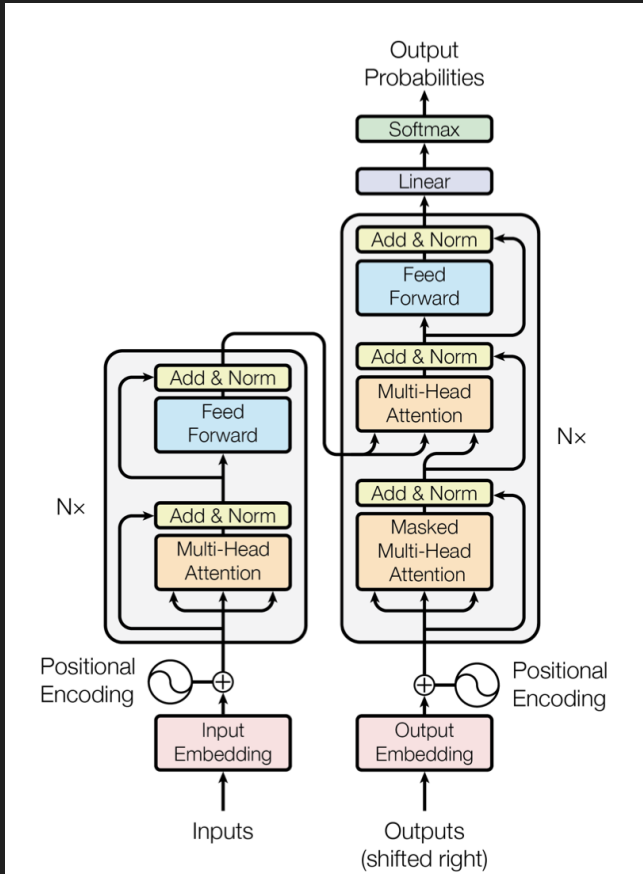

Figure 1: The Transformer - model architecture.

Figure: From Vaswani et al. [2017]

Multi-head attention

- Multi-head: just several bits of attention stacked together
- Scaled dot product attention: compute a query, key and value for each word in the sequence; use $query \cdot key$ to weight a sum of values

Objective?

Two training tasks:

- Masked word prediction
- Next sentence prediction

So what actually happens?

- Hewitt and Manning [2019] show that *entire syntax trees* can be recovered from BERT via simple linear projection
- Coenen et al. [2019] provide a visualization of BERT embeddings and elaborate on Hewitt and Manning's results
- Tenney et al. [2019] analyze the layers of BERT and find that it approximately follows the steps of a classical end-to-end NLP system

In conclusion

Lots of work still to be done here! Analysis of contextual embedding space is an exciting and still relatively new field

Thanks!

Questions?

References I

Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. A latent variable model approach to PMI-based word embeddings. Transactions of the Association for Computational Linguistics, 4: 385–399, 2016. doi: 10.1162/tacl_a_00106. URL https://www.aclweb.org/anthology/Q16-1028.

Andy Coenen, Emily Reif, Ann Yuan, Been Kim, Adam Pearce, Fernanda B. Viégas, and Martin Wattenberg. Visualizing and measuring the geometry of BERT. CoRR, abs/1906.02715, 2019. URL http://arxiv.org/abs/1906.02715.

Tatsunori B. Hashimoto, David Alvarez-Melis, and Tommi S. Jaakkola. Word embeddings as metric recovery in semantic spaces. Transactions of the Association for Computational Linguistics, 4:273–286, 2016. doi: 10.1162/tacl_a_00098. URL https://www.aclweb.org/anthology/Q16-1020.

References II

John Hewitt and Christopher D. Manning. A structural probe for finding syntax in word representations. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 4129–4138, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1419. URL https://www.aclweb.org/anthology/N19-1419.

David Mimno and Laure Thompson. The strange geometry of skip-gram with negative sampling. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2873–2878, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1308. URL https://www.aclweb.org/anthology/D17-1308.

References III

Tobias Schnabel, Igor Labutov, David Mimno, and Thorsten Joachims. Evaluation methods for unsupervised word embeddings. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 298–307, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1036. URL https://www.aclweb.org/anthology/D15-1036.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. BERT rediscovers the classical NLP pipeline. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 4593–4601, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1452. URL https://www.aclweb.org/anthology/P19-1452.

References IV

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, Advances in Neural Information Processing Systems 30, pages 5998–6008. Curran Associates, Inc., 2017. URL http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf.