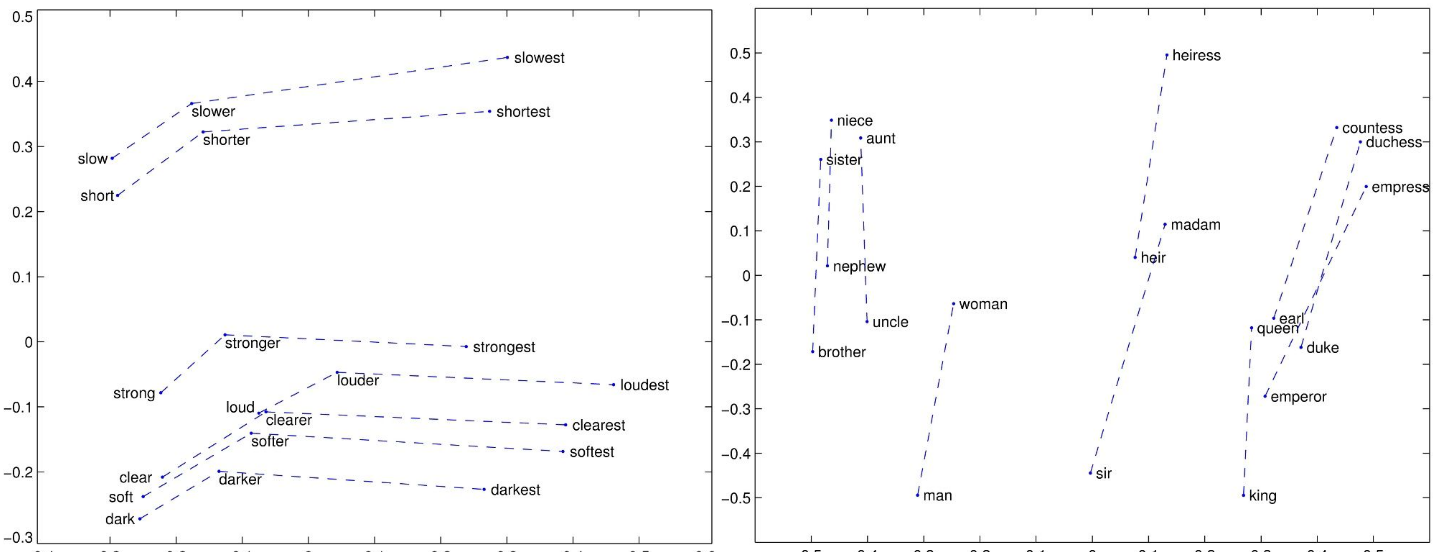
WORD EMBEDDINGS

On word embeddings - Part 3: The secret ingredients of word2vec

Word2vec is a pervasive tool for learning word embeddings. Its success, however, is mostly due to particular architecture choices. Transferring these choices to traditional distributional methods makes them competitive with popular word embedding methods.

SEBASTIAN RUDER
24 SEP 2016 • 9 MIN READ



word2vec and its connection to more traditional models.

Table of Contents:

Excuse the rather clickbait-y title. This is a blog post that I meant to write for a while. In this post, I want to highlight the factors, i.e. the secret ingredients that account for the success of word2vec.
In particular, I want to focus on the connection between word embeddings trained via neural models and those produced by traditional distributional semantics models (DSMs). By showing how these ingredients can be transferred to DSMs, I will demonstrate that distributional methods are in no way inferior to the popular word embedding methods.
Even though this is no new insight, I feel that traditional methods are frequently overshadowed amid the deep learning craze and their relevancy consequently deserves to be mentioned more often.

To this effect, the paper on which this blog post is based is *Improving Distributional Similarity with Lessons Learned from Word Embeddings* [1] by Levy et al. (2015). If you haven't read it, I recommend you to check it out.

Over the course of this blog post, I will first introduce GloVe, a popular word embedding model. I will then highlight the connection between word embedding models and distributional semantic methods. Subsequently, I will introduce the four models that will be used to measure the impact of the different factors. I will then give an overview present the results by Levy et al., their takeaways and recommendations.

GloVe

In a previous blog post, we have given an overview of popular word embedding models. One model that we have omitted so far is GloVe [2].

Briefly, GloVe seeks to make explicit what SGNS does implicitly: Encoding meaning as vector offsets in an embedding space -- seemingly only a serendipitous by-product of word2vec -- is the specified goal of GloVe.
Specifically, the authors of Glove show that the ratio of the co-occurrence probabilities of two words (rather than their co-occurrence probabilities themselves) is what contains information and aim to encode this information as vector differences.
To achieve this, they propose a weighted least squares objective J that directly aims to minimise the difference between the dot product of the vectors of two words and the logarithm of their number of co-occurrences:

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

where w_i and b_i are the word vector and bias respectively of word i, $\tilde{w}_j$ and b_j are the context word vector and bias respectively of word j, X_{ij} is the number of times word i occurs in the context of word j, and f is a weighting function that assigns relatively lower weight to rare and frequent co-occurrences.

As co-occurrence counts can be directly encoded in a word-context co-occurrence matrix, GloVe takes such a matrix rather than the entire corpus as input.

If you want to know more about GloVe, the best reference is likely the author of gensim here, in this Quora thread, and in this blog post.

Word embeddings vs. distributional semantics models

The reason why word embedding models, particularly word2vec and GloVe, became so popular is that they seemed to continuously and significantly outperform DSMs. Many attributed this to the neural architecture of word2vec or the fact that it predicts words, which seemed to have a natural edge over solely relying on co-occurrence counts.

We can view DSMs as *count* models as they "count" co-occurrences among words by operating on co-occurrence matrices. In contrast, neural word embedding models can be seen as *predict* models, as they try to predict surrounding words.

In 2014, Baroni et al. [3] showed that predict models consistently outperform count models in almost all tasks, thus providing a clear verification for the apparent superiority of word embedding models. Is this the end? No.

Already with GloVe we've seen that the differences are not as clear-cut: While GloVe is considered a predict model by Levy et al. (2015), it is clearly factorizing a word-context co-occurrence matrix, which brings it close to traditional methods such as PCA and LSA. Even more, Levy et al. [4] demonstrate that word2vec implicitly factorizes a word-context PMI matrix.

Consequently, while on the surface DSMs and word embedding models use different algorithms to learn word representations -- the first count, the latter predict -- fundamentally, both types of models act on the same

of this blog post to answering is the following:
Why do word embedding models still perform better than DSM with almost the same information?

Models

Following Levy et al. (2015), we will isolate and identify the factors that account for the success of neural word embedding models and show how these can be transferred to traditional methods by comparing the following four models:

- **Positive Pointwise Mutual Information (PPMI)**: PMI is a common measure for the strength of association between two words. It is defined as the log ratio between the joint probability of two words w and c and the product of their marginal probabilities: $PMI(w, c) = \log \frac{P(w,c)}{P(w)P(c)}$. As $PMI(w, c) = \log 0 = -\infty$ for pairs (w, c) that were never observed, PMI is in practice often replaced with *positive* PMI (PPMI), which replaces negative values with 0, yielding $PPMI(w, c) = \max(PMI(w, c), 0)$.
- **Singular Value Decomposition (SVD):** SVD is one of the most popular methods for dimensionality reduction and found its into NLP originally via latent semantic analysis (LSA). SVD factories the word-context co-occurrence matrix into the product of three matrices $U \cdot \Sigma \times V^T$ where U and V are orthonormal matrices (i.e. square matrices whose rows and columns are orthogonal unit vectors) and Σ is a diagonal matrix of eigenvalues in
decreasing order. In practice, SVD is often used to factorize the matrix produced by PPMI. Generally, only the top d elements of Σ are kept, yielding $W^{SVD} = U_d \cdot \Sigma_d$ and $C^{SVD} = V_d$, which are typically used as the word and context representations respectively.

sampling refer to my previous blog posts here and here respectively.

- **Global Vectors (GloVe)** as presented in the previous section.

Hyperparameters

We will look at the following hyper-parameters:

- Pre-processing
  - Dynamic context window
  - Subsampling frequent words
  - Deleting rare words
- Association metric
  - Shifted PMI
  - Context distribution smoothing
- Post-processing
  - Adding context vectors
  - Eigenvalue weighting
  - Vector normalisation

Pre-processing

Word2vec introduces three ways of pre-processing a corpus, which can be easily applied to DSMs.

Dynamic context window

In DSMs traditionally, the context window is unweighted and of a constant size. Both SGNS and GloVe, however, use a scheme that weighting is implemented by having a dynamic window size that is sampled uniformly between 1 and the maximum window size during training.

Subsampling frequent words

SGNS dilutes very frequent words by randomly removing words whose frequency f is higher than some threshold t with a probability $p = 1 - \sqrt{\frac{t}{f}}$. As this subsampling is done *before* actually creating the windows, the context windows used by SGNS in practice are larger than indicated by the context window size.

Deleting rare words

In the pre-processing of SGNS, rare words are also deleted *before* creating the context windows, which increases the actual size of the context windows further. Levy et al. (2015) find this not to have a significant performance impact, though.

Association metric

PMI has been shown to be an effective metric for measuring the association between words. Since Levy and Goldberg (2014) have shown SGNS to implicitly factorize a PMI matrix, two variations stemming from this formulation can be introduced to regular PMI.

Shifted PMI

In SGNS, the higher the number of negative samples k, the more data is being used and the better should be the estimation of the parameters. k affects the shift of the PMI matrix that is implicitly factorized by word2vec, i.e. k k shifts the PMI values by log k.
If we transfer this to regular PMI, we obtain Shifted PPMI (SPPMI): $SPPMI(w, c) = \max(PMI(w, c) - \log k, 0)$.

unigram distribution, i.e. an unigram distribution raised to the power of α, which is empirically set to $\frac{3}{4}$. This leads to frequent words being sampled relatively less often than their frequency would indicate.
We can transfer this to PMI by equally raising the frequency of the context words $f(c)$ to the power of α:
$PMI(w, c) = \log \frac{p(w, c)}{p(w) p_\alpha(c)}$ where $p_\alpha(c) = \frac{f(c)^\alpha}{\sum_c f(c)^\alpha}$ and $f(x)$ is the frequency of word x.

Post-processing

Similar as in pre-processing, three methods can be used to modify the word vectors produced by an algorithm.

Adding context vectors

The authors of GloVe propose to add word vectors and context vectors to create the final output vectors, e.g. $\vec{v}_{\text{cat}} = \vec{w}_{\text{cat}} + \vec{c}_{\text{cat}}$. This adds first-order similarity terms, i.e $w \cdot v$. However, this method cannot be applied to PMI, as the vectors produced by PMI are sparse.

Eigenvalue weighting

SVD produces the following matrices: $W^{SVD} = U_d \cdot \Sigma_d$ and $C^{SVD} = V_d$. These matrices, however, have different properties: C^{SVD} is orthonormal, while W^{SVD} is not.
SGNS, in contrast, is more symmetric. We can thus weight the eigenvalue matrix Σ_d with an additional parameter p, which can be tuned, to yield the following:
$W^{SVD} = U_d \cdot \Sigma_d^p$.

Vector normalisation

Finally, we can also normalise all vectors to unit length.

Levy et al. (2015) train all models on a dump of the English wikipedia and evaluate them on the commonly used word similarity and analogy datasets. You can read more about the experimental setup and training details in their paper. We summarise the most important results and takeaways below.

Takeaways

Levy et al. find that SVD -- and not one of the word embedding algorithms -- performs best on similarity tasks, while SGNS performs best on analogy datasets. They furthermore shed light on the importance of hyperparameters compared to other choices:

1. Hyperparameters vs. algorithms:
   Hyperparameter settings are often more important than algorithm choice.
   No single algorithm consistently outperforms the other methods.
2. Hyperparameters vs. more data:
   Training on a larger corpus helps for some tasks.
   In 3 out of 6 cases, tuning hyperparameters is more beneficial.

Debunking prior claims

Equipped with these insights, we can now debunk some generally held claims:

1. Are embeddings superior to distributional methods?
   With the right hyperparameters, no approach has a consistent advantage over another.
2. Is GloVe superior to SGNS?
   SGNS outperforms GloVe on all tasks.

Recommendations

Finally -- and one of the things I like most about the paper -- we can give concrete practical recommendations:

- **DON'T** use shifted PPMI with SVD.
- **DON'T** use SVD "correctly", i.e. without eigenvector weighting (performance drops 15 points compared to with eigenvalue weighting with $p = 0.5$).
- **DO** use PPMI and SVD with short contexts (window size of 2).
- **DO** use many negative samples with SGNS.
- **DO** always use context distribution smoothing (raise unigram distribution to the power of $\alpha = 0.75$) for all methods.
- **DO** use SGNS as a baseline (robust, fast and cheap to train).
- **DO** try adding context vectors in SGNS and GloVe.

Conclusions

These results run counter to what is generally assumed, namely that word embeddings are superior to traditional methods and indicate that it generally makes *no difference whatsoever* whether you use word embeddings or distributional methods -- what matters is that you tune your hyperparameters and employ the appropriate pre-processing and post-processing steps.

Recent papers from Jurafsky's group [5] [6] echo these findings and show that SVD -- not SGNS -- is often the preferred choice when you care about accurate word representations.

I hope this blog post was useful in highlighting cool research that sheds light on the link between traditional distributional semantic and in-vogue embedding models. As we've seen, knowledge of distributional next time you train word embeddings, you will consider adding distributional methods to your toolbox or lean on them for inspiration.

As always, feel free to ask questions and point out the mistakes I made in this blog post in the comments below.

Other blog posts on word embeddings

If you want to learn more about word embeddings, these other blog posts on word embeddings are also available:

- On word embeddings - Part 1
- On word embeddings - Part 2: Approximating the softmax
- Unofficial Part 4: A survey of cross-lingual embedding models
- Unofficial Part 5: Word embeddings in 2017 - Trends and future directions

Cover images are courtesy of Stanford.

---

1. Levy, O., Goldberg, Y., & Dagan, I. (2015). Improving Distributional Similarity with Lessons Learned from Word Embeddings. Transactions of the Association for Computational Linguistics, 3, 211–225. Retrieved from https://tacl2013.cs.columbia.edu/ojs/index.php/tacl/article/view/570 ↩
2. Pennington, J., Socher, R., & Manning, C. D. (2014). Glove: Global Vectors for Word Representation. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, 1532–1543. http://doi.org/10.3115/v1/D14-1162 ↩
3. Baroni, M., Dinu, G., & Kruszewski, G. (2014). Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. ACL, 238–247. http://doi.org/10.3115/v1/P14-1023 ↩
4. Levy, O., & Goldberg, Y. (2014). Neural Word Embedding as Implicit Matrix Factorization. Advances in Neural Information Processing Systems (NIPS),

5. Hamilton, W. L., Clark, K., Leskovec, J., & Jurafsky, D. (2016). Inducing Domain-Specific Sentiment Lexicons from Unlabeled Corpora. Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics. Retrieved from http://arxiv.org/abs/1606.02820 ↩
6. Hamilton, W. L., Leskovec, J., & Jurafsky, D. (2016). Diachronic Word Embeddings Reveal Statistical Laws of Semantic Change. arXiv Preprint arXiv:1605.09096. ↩

MORE IN **WORD EMBEDDINGS**

AAAI 2019 Highlights: Dialogue, reproducibility, and more
7 Feb 2019 – 11 min read

EMNLP 2018 Highlights: Inductive bias, cross-lingual learning, and more
6 Nov 2018 – 11 min read

A Review of the Neural History of Natural Language Processing
1 Oct 2018 – 29 min read

See all 9 posts →

NATURAL LANGUAGE PROCESSING

Highlights of EMNLP 2016: Dialogue, deep learning, and more

This post discusses highlights of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP 2016). These include work on reinforcement learning, 14 NOV 2016 • 4 MIN READ

EVENTS

LxMLS 2016 Highlights

The Lisbon Machine Learning School (LxMLS) is an annual event that brings together researchers and graduate students in ML, NLP, and Computational Linguistics. This post discusses highlights, key insights, and takeaways from the 6th edition of the summer school.

SEBASTIAN RUDER
12 AUG 2016 • 14 MIN READ