Word2Vec

Lexical Resources

October 21st, 2020

Outline

Outline

Overview

“You shall know a word by the company it keeps”

- Word embeddings correspond to the linguistic theory of “distributional semantics” (DS, or distributional semantics models, DSM)
- The general idea of DSM is that the meaning of a word can be known by the context in which it occurs, hence the quote from Firth (1957): “You shall know a word by the company it keeps”
- Word embeddings are context-based vector representation of words used in machine learning, whereas DS is a semantic theory of meaning, which generally employs vectors to represent meanings.
- Word2vec is a word embedding algorithm that was presented in three papers: Mikolov, Yih, and Zweig (2013) and Mikolov et al. (2013b,a)

Overview

Why does word2vec matter?

Word embeddings in general, and word2vec in particular are widely used in descriptive & theoretical linguistics

- from social biases study (Bolukbasi et al., 2016) ...
- ... to theoretical morphology (Bonami and Paperno, 2018)

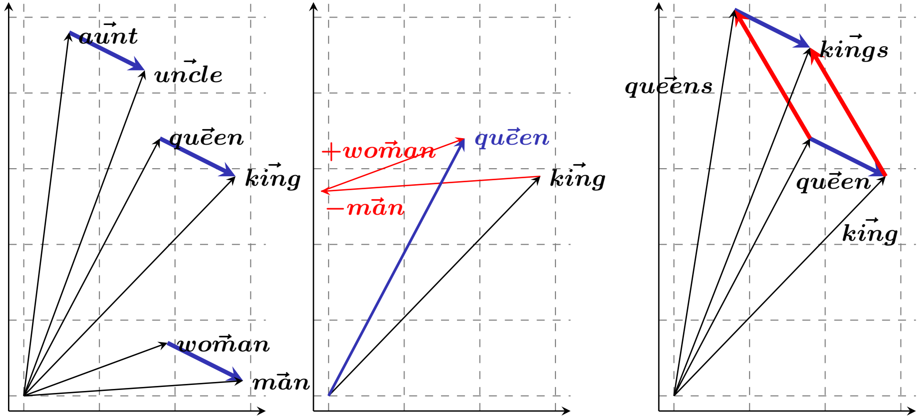
The success of word2vec comes from a multiplicity of factors:

- spearheaded the transition to neural-networks in NLP and DS in linguistics
- an efficient algorithm, shown to be equivalent to count-based vectors (Goldberg and Levy, 2014)
- useful for initializing other neural networks (Artetxe et al., 2017).
- has been shown to describe a latent code
- highlights how to combine various nifty tricks from the machine learning community

Overview

Latent code

- Latent code: vector addition encodes meaningful semantics.
- Other networks have been shown to do similar things (GAN: Shen et al., 2020)
- Mikolov, Yih, and Zweig (2013) used it to operationalize formal analogy

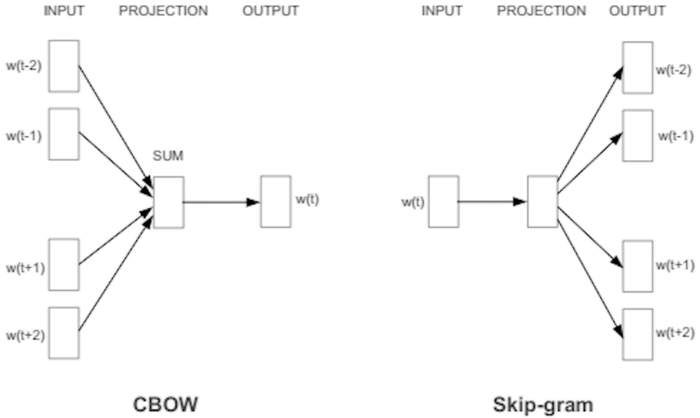


Outline

Technical Definition

word2vec comprises 2 architectures

- CBOW uses the context of a word as the input, and tries to predict the word
- Skip-gram uses a word as an input, and tries to predict each word in its context.



Technical Definition

CBOW architecture

- CBOW is comprised of one linear projection $\boldsymbol{W_P} = [V \times D]$ and a log-linear classifier $\boldsymbol{W_C} = [D \times V]$
  NB: V is the size of the vocabulary and D is the number of dimensions.

- All context words are first transformed as one-hot vectors, then down-projected in a vector space R^D using the projection $\boldsymbol{W_P}$. The average of all projected vectors is then used as input for the log-linear classifier $\boldsymbol{W_C}$ itself.

$$\vec{h_i} = \frac{1}{2t}\left(\sum_{j=i-1-t}^{i-1} \boldsymbol{W_P} \cdot \vec{w_j} + \sum_{j=i+1}^{i+1+t} \boldsymbol{W_P} \cdot \vec{w_j}\right)$$

$$\hat{y_i} = \mathrm{softmax}(\boldsymbol{W_C} \cdot \vec{h_i})$$

NB: The above corresponds to a context *window* of size t around the target word

Technical Definition

CBOW word vectors

- The use of one-hot vectors allows us to transform a vocabulary index in a vector.
- Given a word w_i, and its index i in the vocabulary, we define

$$\vec{w_i} = \left(c_1, \ \ldots, \ c_d\right)$$

$$c_j = \begin{cases} 1 & \text{if } j = i \\ 0 & \text{otherwise} \end{cases}$$

- Therefore the down-projection using $\boldsymbol{W_P}$ corresponds to selecting the i$^{\text{th}}$ row of $\boldsymbol{W_P}$.
- The row-vectors of the $\boldsymbol{W_P}$ matrix therefore are the actual word2vec embeddings.
- The classifier $\boldsymbol{W_C}$ only serves for training, and is to be discarded afterwards.

Technical Definition

Cbow training

- As it's a log-classifier, the training objective is to maximize the log-likelihood of the probability of predicting the correct word based on its context.
- The probability distribution is implicitly given by the softmax function:

$$\begin{aligned}\hat{y}_j &= \text{softmax}(\boldsymbol{W_C} \cdot \vec{h}) \\ &= \frac{\exp(\boldsymbol{W_C}^j \cdot \vec{h})}{\sum_{j'} \exp(\boldsymbol{W_C}^{j'} \cdot \vec{h})}\end{aligned}$$

  where $\boldsymbol{W_C}^j$ is the j^{th} column vector of the matrix $\boldsymbol{W_C}$. The components of $\hat{y}$ sum to 1, and therefore define a probability distribution for each element of our vocabulary ($\hat{y}$ is a vector of dimension V).
- maximizing the probability of predicting the current word knowing the context is equivalent to minimizing the negative log-likelihood for that word.

$$\mathcal{L}(\hat{y}, w_i) = -\log \hat{y}_i$$

Technical Definition

Skip-gram architecture

In broad terms, skip-gram can be thought of as a “reversed” CBOW architecture

- In skip-gram, we aim to predict the context based on the current word.
- We first project the current word using a linear projection $\boldsymbol{W_P} = [V \times D]$, and use a classifier to predict each word in the context $\boldsymbol{W_C} = [D \times V]$.
- Like with CBOW, we derive vectors from the $\boldsymbol{W_P}$ matrix
- A probability distribution is inferred by applying a softmax after the classifier's output.

$$\vec{h_i} = \boldsymbol{W_P} \cdot \vec{w_i}$$
$$\hat{y_i} = \mathrm{softmax}(\boldsymbol{W_C} \cdot \vec{h_i})$$

where $\vec{w_i}$ is the one-hot vector for word w_i.

Technical Definition

Skip-gram Training

- As all context words are to be predicted using the same input word, we aim to maximize the joint probability of all context words knowing the current word.

$$p(w_{i-t},\ \ldots,\ w_{i+t}|w_i)$$

- We can estimate this probability using the chain rule:

$$\prod_{j=i-t}^{i-1} p(w_j|w_i) \times \prod_{j=i+1}^{i+t} p(w_j|w_i)$$

- We can transform the product into a sum by maximizing the log-likelihood instead
- So the model is trained by minimizing the joint negative log-likelihood of each context word.

$$\mathcal{L}(\hat{y}, \langle w_{i-t}, \ldots, w_{i+t} \rangle) = -\left(\sum_{j=i-t}^{i-1} \log \hat{y}_j + \sum_{j=i+1}^{i+t} \log \hat{y}_j \right)$$

NB: In practice, the loss is averaged over the full input sentence.

Technical Definition

Negative sampling—new objective

- Obtaining the multinomial distribution of the skip-gram model is computationally inefficient.
- We may instead consider to train the the classifier to distinguish whether a given context is attested for a given word.
- Let D^+ the set of all pairs of words w and contexts c that occurs in our dataset, and let D^- a set of *negative examples* (also pairs of words and contexts), such that $D^+ \cap D^- = \emptyset$. Let $p(X = 1|w, c)$ the probability that $\langle w,\ c\rangle \in D^+$.
- We can redefine the classifier's objective as maximizing $p(X = 1|w, c)$ for all $\langle w,\ c\rangle \in D^+$, and minimizing $p(X = 1|w, c)$ for all $\langle w,\ c\rangle \in D^-$.
- Minimizing $p(X = 1|w, c)$ is equivalent to maximizing $1 - p(X = 1|w, c)$.
- The objective is therefore to maximize

$$\prod_{\langle w,\ c\rangle \in D^+} p(X = 1|w, c) \prod_{\langle w,\ c\rangle \in D^-} \big(1 - p(X = 1|w, c)\big)$$

Technical Definition

Negative sampling—adapting the architecture

- We have to amend the network's architecture. We don't need a full distribution over the vocabulary, so we can replace the softmax function with a sigmoid: $\sigma(y) = \frac{1}{1+\exp(-y)}$. Vector representations for words and contexts still have to be drawn from two different matrices.

- We therefore compute the score for $\langle w_j,\ w_i \rangle$ simply using $\sigma\big(\boldsymbol{W_C}^j \cdot (\boldsymbol{W_P} w_i)\big)$ for pairs drawn from D^+, and $\sigma\big(-\boldsymbol{W_C}^j \cdot (\boldsymbol{W_P} w_i)\big)$ for pairs drawn from D^-, as $1 - \sigma(y) = \sigma(-y)$

- To limit computation complexity, we estimate the second term using only k negative examples.

- To obtain the loss function, we replace the negative likelihood of predicting word w_j knowing word w_i from previous loss functions.

$$-\log p(w_j|w_i) = -\log \sigma\big(\boldsymbol{W_C}^j \cdot (\boldsymbol{W_P} w_i)\big) + \sum_{w_n \in N} \sigma\big(-\boldsymbol{W_C}^n \cdot (\boldsymbol{W_P} w_i)\big)$$

  where N is a set of k negative examples sampled for w_i.

Technical Definition

Hierarchical softmax & subsampling

- *Alternatively* to negative sampling, we can use a a **hierarchical softmax** and encode probabilities using a binary tree structure. Leaves correspond to words in the vocabulary, and each node n stores the relative probabilities of its children using a dedicated weight vector $\vec{v_n}$.
- Let $\mathcal{P}(w_i) = \{n_0, \ \ldots \ , n_{w_i}\}$ the path from the root node n_0 to the leaf node n_{w_i} for word w_i. We can redefine the output probability as

$$p(w_j|\vec{h_i}) = \prod_{n \in \mathcal{P}(w_j)} \sigma\left(\vec{v_n} \cdot \vec{h_i}\right)$$

- Mikolov & al also proposed to avoid issues arising with class imbalance ("Zipf's law") by dropping words from the training set based on their frequency. They define the "**subsampling**" rate:

$$P(w_i) = 1 - \sqrt{\frac{t}{f(w_i)}}$$

where t is an hyperparameter (typically 10^{-5}) and $f(w)$ is the frequency of word w.

Outline

Word2vec in linguistics

Main remarks

- The latent code of word2vec gives a principled way of modeling semantic representations
- Word2vec (and DSM in general) are therefore invaluable to data-driven (computational) linguistic studies
  NB: Some studies avoid fastText on the grounds that it is not *solely* a representation of meaning
- It comes at the cost of **assuming** the distributional hypothesis
- There are technical limitations: e.g., rare words have unreliable vectors, rare phenomena may not be consistently modeled
- Harris (1954), who put forward the idea of distributional structures, did not equate them with **meaning**:

> *To the extent that formal (distributional) structure can be discovered in discourse, it correlates in some way with the substance of what is being said [...] However, this is not the same thing as saying that the distributional structure of language (phonology, morphology, and at most a small amount of discourse structure) conforms in some one-to-one way with some independently discoverable structure of meaning.*

Word2vec in linguistics

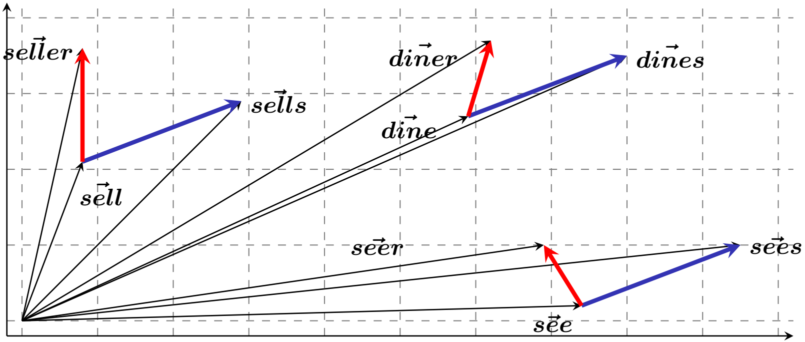
Example study: Bonami and Paperno (2018), I

- In morphology, inflection has been claimed to be more semantically regular than derivation (Stump, 1998; Štekauer, 2014, e.g.)
  - *know–knows*, *dance—dances*:
    knowing the meaning of the former entails knowing the meaning of the latter
  - *sell–seller*, but *dine–diner* or *see–seer*:
    knowing the meaning of the former does **not** entail knowing the meaning of the latter
- Bonami and Paperno (2018) test whether this assumption is consistent with distributional semantics
- Assuming it is, we would expect linear offsets for inflectional relations (e.g., $b\vec{ar}e - 3^{r\vec{d}}sg$) to be more consistent than those for derivational relations (e.g., $v\vec{e}rb - ag\vec{e}nt$)

Word2vec in linguistics

Example study: Bonami and Paperno (2018), II

- Many factors to control: frequency, but also the inherent semantics of the words under consideration
- The solution of Bonami and Paperno (2018) is to use word triples



- They find that derivational relations yield significantly more variation than inflectional ones: derivational pairs stray more from the average value than inflectional pairs.

Word2vec in linguistics

Exercise

Let's imitate the experiment of Bonami and Paperno (2018)

1. Retrieve embeddings from `http://vectors.nlpl.eu/repository/20/6.zip`
2. Retrieve the CSV document containing agent–verb–3$^{\text{rd}}$ sg. triples for this exercise from `https://github.com/TimotheeMickus/lexres-2020/blob/main/lecture-4/triples.csv`.
3. Load the embeddings for all words in the CSV.
4. For each row, compute the offsets (i.e., the vector difference):
   - 4.1 compute the offsets between agent noun and bare verb
   - 4.2 compute the offsets between bare verb and 3$^{\text{rd}}$ sg. form
5. Compute the **average** offset between agent and verb (using the offsets from step 4.1), and the **average** offset between verb and 3$^{\text{rd}}$ sg. (using the offsets from step 4.2)
6. For each offset, compute its Euclidean distance to the average offset.
   NB: You now have two series of measurements, representing how your sample varies with respect to the average value.
7. Take the two comparable series of measurements you got from step 6, and compute a paired t-test (e.g. using the function `scipy.stats.ttest_rel`). What can you conclude?
8. Repeat steps 6 and 7, this time using cosine distance (i.e., $1 - \cos(\vec{u}, \vec{v})$)

Outline

Conclusion

There has been and there still is an important body of research using word2vec. Here are some things to look into:

- papers introducing word2vec: Mikolov et al. (2013b), Mikolov, Yih, and Zweig (2013), and Mikolov et al. (2013a)
- papers explaining word2vec: Goldberg and Levy (2014), Rong (2014), and Levy and Goldberg (2014) ...
- original word2vec repository : `https://code.google.com/archive/p/word2vec/`, or on Mikolov's github : `https://github.com/tmikolov/word2vec`
- NLPL's repository containing many pre-trained word2vec models (as well as other embeddings) for multiple languages: `http://vectors.nlpl.eu/repository/`

References I

Artetxe, Mikel et al. (2017). “Unsupervised Neural Machine Translation”. In: *arXiv preprint arXiv:1710.11041*.

Bolukbasi, Tolga et al. (2016). “Man is to Computer Programmer as Woman is to Homemaker? Debiasing Word Embeddings”. In: *Advances in Neural Information Processing Systems 29*. Ed. by D. D. Lee et al. Curran Associates, Inc., pp. 4349–4357. URL: `http://papers.nips.cc/paper/6228-man-is-to-computer-programmer-as-woman-is-to-homemaker-debiasing-word-embeddings.pdf`.

Bonami, Olivier and Denis Paperno (2018). “A characterisation of the inflection-derivation opposition in a distributional vector space”. In: *Lingua e Langaggio*.

Firth, J. R. (1957). “A synopsis of linguistic theory 1930-55.”. In: *Studies in Linguistic Analysis (special volume of the Philological Society)* 1952-59, pp. 1–32.

References II

Goldberg, Yoav and Omer Levy (2014). "word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method". In: *CoRR* abs/1402.3722. arXiv: 1402.3722. URL: http://arxiv.org/abs/1402.3722.

Harris, Zellig (1954). "Distributional structure". In: *Word* 10.23, pp. 146–162.

Levy, Omer and Yoav Goldberg (2014). "Neural Word Embedding as Implicit Matrix Factorization". In: *Advances in Neural Information Processing Systems 27*. Ed. by Z. Ghahramani et al. Curran Associates, Inc., pp. 2177–2185. URL: http://papers.nips.cc/paper/5477-neural-word-embedding-as-implicit-matrix-factorization.pdf.

Mikolov, Tomas, Wen-tau Yih, and Geoffrey Zweig (2013). "Linguistic Regularities in Continuous Space Word Representations.". In: *HLT-NAACL*, pp. 746–751.

References III

Mikolov, Tomas et al. (2013a). “Distributed Representations of Words and Phrases and Their Compositionality”. In: *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*. NIPS’13. Lake Tahoe, Nevada: Curran Associates Inc., pp. 3111–3119. URL: http://dl.acm.org/citation.cfm?id=2999792.2999959.

Mikolov, Tomas et al. (2013b). “Efficient Estimation of Word Representations in Vector Space”. In: *CoRR* abs/1301.3781. arXiv: 1301.3781. URL: http://arxiv.org/abs/1301.3781.

Rong, Xin (2014). “word2vec Parameter Learning Explained”. In: *CoRR* abs/1411.2738. arXiv: 1411.2738. URL: http://arxiv.org/abs/1411.2738.

Shen, Yujun et al. (2020). “Interpreting the Latent Space of GANs for Semantic Face Editing”. In: *CVPR*.

References IV

Štekauer, P. (2014). "The Oxford Handbook of Derivational Morphology". In: *The Oxford Handbook of Derivational Morphology . Oxford: Oxford University Press*. Ed. by R. Lieber and P. tekauer. Oxford University Press. Chap. Derivational Paradigms, pp. 354–369.

Stump, Gregory (1998). "The handbook of morphology". In: ed. by A. Spencer & A. M. Zwicky. Oxford: Blackwell. Chap. Inflection. Pp. 13–43.