Word embeddings

Victor Kitov

v.v.kitov@yandex.ru

Interpretable word embeddings

- $x \in \mathbb{R}^K$, where x^i is some i-th interpretable feature, e.g.
  - x^1: part of speech
  - x^2: gender (for nouns)
  - x^3: tense (for verbs)
  - x^4: starts from capital letter
  - x^5: #[letters]
  - x^6: category: machine learning, physics, biology, ...
  - x^7: subcategory: supervised, unsupervised, semi-supervised learning
  - ...
- Need to invent features for each task and extract them.
- Want this to be done automatically!

Uninterpretable word embeddings

- Clustering words with similar meaning to similar representations.
- **Distributional hypothesis:**
  words have similar meaning <=> they co-occur together frequently.
- "accuracy of SVM", "SVM gave accuracy", "lower accuracy, compared to SVM"
  - SVM and accuracy are connected!
- Typical dimensionality of embedding $\in [300, 500]$.

Phrase embeddings

We can treat collocations as separate units.

- e.g. fast food, post office, happily married, proud smile.
- may can extract collocations with

$$(w_i, w_j)\text{-collocation} \Longleftrightarrow \frac{p(w_i w_j) - \delta}{p(w_i)p(w_j)} > \textit{threshold}$$

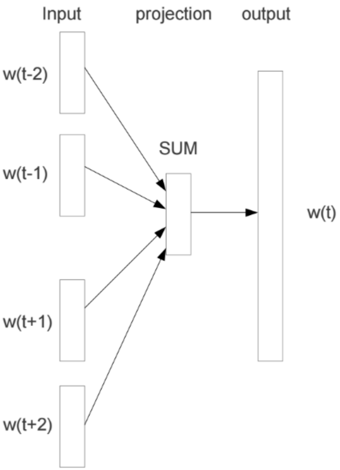
δ - parameter, discouraging rarely co-occurring words as collocations.

Table of Contents

Word2vec

- For each w models evaluate:
  - target word embedding v_w
  - context word embedding $\tilde{v}_w$
- Target&context embeddings may be averaged or concatenated later.

Continious bag of words (CBOW)



Continuous bag of words (CBOW)

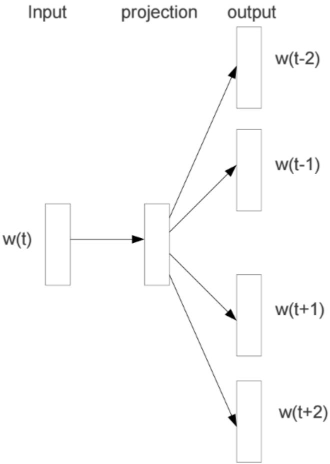
CBOW: predict current word given context.

$$\frac{1}{T}\sum_{t=1}^{T}\ln p(w_t|w_{t-c},..w_{t-1},w_{t+1},...w_{t+c})\to\max_{\theta}$$

where $v_{context}=\sum_{-c\le i\le c,\,i\ne 0}v_{w_{t+i}}$ and

$$p(w_t|w_{t-c},..w_{t-1},w_{t+1},...w_{t+c})=\frac{\exp\left(v_{context}^T\tilde{v}_{w_t}\right)}{\sum_{w=1}^{V}\exp\left(v_{context}^T\tilde{v}_w\right)}$$

Skip-gram model



Skip-gram model

Skip-gram: predict context, given current word:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\le i\le c,\, i\ne 0}\ln p(w_{t+i}|w_t)\to\max_{\theta}$$

$$p(w_{t+i}|w_t)=\frac{\exp\left(v_{w_t}^T\tilde{v}_{w_{t+i}}\right)}{\sum_{w=1}^{V}\exp\left(v_{w_t}^T\tilde{v}_w\right)}$$

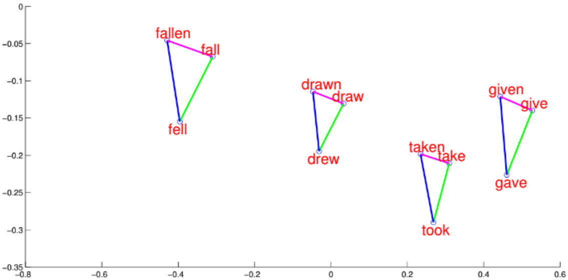
Comments

- May extract embeddings for other objects, appearing in a sequence.
  - symbols, trigrams of symbols (see *FastText*), sentences
  - genes in DNA sequence
  - services ordered by a customer
- May use ensemble of embeddings from different methods.

Table of Contents

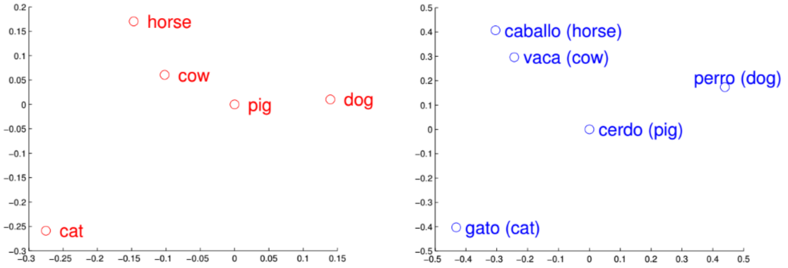
Word forms

Similar in meaning and in spelling words cluster together.



So word embeddings may help in obtaining forms of rare words.

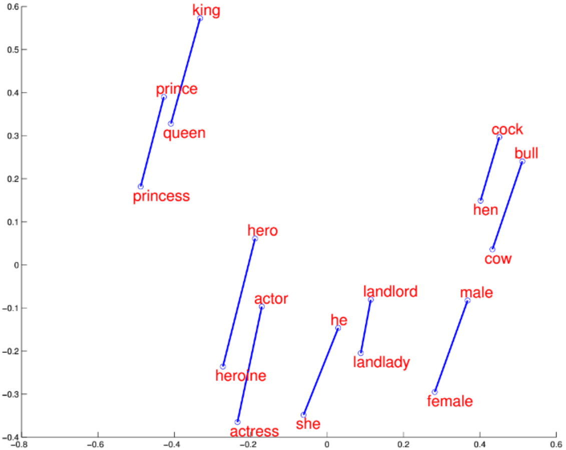
Word embeddings for different languages are similar[1]



So word embeddings may help in language translation.

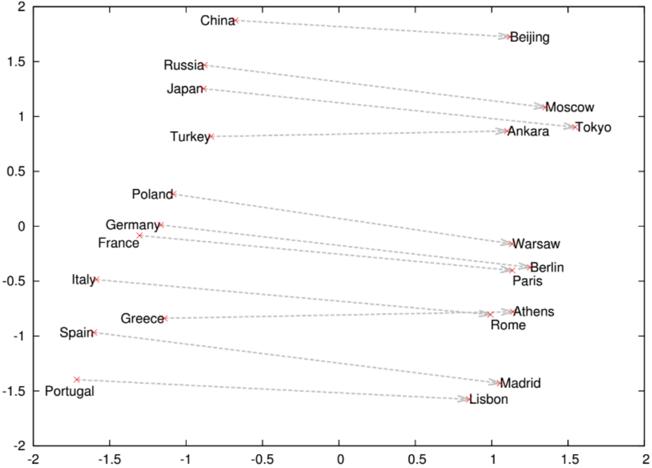
[1] Images were manually rotated and scaled.

Regularities in vector space



So (prince-princess)+queen≈king! Helps in question answering.

Regularities in vector space



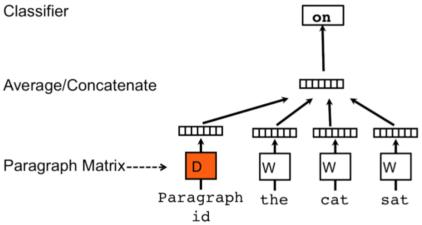
So (Beijing-China)+Russia≈Moscow! Helps in question answering.

Table of Contents

Paragraph to vector - motivation

- Now need to encode the paragraph (document) into fixed size vector.
- Simple approach: paragraph vector - average of word vectors it contains.
    - or weighted average, considering side information about a word (e.g. stop word/specific term, etc.)
- Alternative: learn paragraph representation.

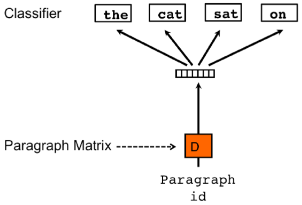
Paragraph vector - PV-DM model



Distributed Memory Model of Paragraph Vectors (PV-DM)

- On training documents are divided into paragraphs. Each paragraph is associated its column in paragraph2vec matrix D.
- The same task is solved as in CBOW, using vectors and general paragraph information (context).
- Called *Distributed Memory Model of Paragraph Vectors (PV-DM)*.

Model



model

- Similar to skip-gram: predicts random sequence of words from the paragraph, using paragraph embedding only.
- Compared to previous, model is simpler: need to estimate only D & softmax parameters; word embedding are not used.
- Called *Distributed Bag of Words version of Paragraph Vector (PV-DBOW)*

StarSpace[2]

- Facebook library to convert general objects into vector representations.
- Available at github.
- Learns representations of 2 kinds of objects (x and y, may coincide) into the same space embedding$\in \mathbb{R}^D$.
    - $x \to a \in \mathbb{R}^D$; $y \to b \in \mathbb{R}^D$ - can compare different objects!
- Representations are found by sampling randomly matching pair (a, b) and K mismatched pairs

$$\sum_{(a,b)\in E+;\ b^-\in E^-} L^{batch}\left(sim(a,b), sim(a,b_1^-), ...sim(a,b_K^-)\right) \to \min_{\{a\},\{b\}}$$

where $sim\left(a,b\right) = a^T b$ or $sim\left(a,b\right) = \frac{a^T b}{\|a\|\|b\|}$; L - hinge (worked better) or log-loss.

[2]Wu et al. StarSpace: Embed All The Things!

StarSpace: Applications

- Multiclass classification: *a*-feature vector, *b*-class.
- Multiclass classification: *a*-feature vector, *b*-one of matching classes.
- Collaborative filtering: *a*-user, *b*-item, he likes.
  - does not extend to new users and items
- Collaborative filtering: *a*-"user"=avg. of items user likes, with excluded item *b*.
  - extends to new users, by averaging their item embeddings
  - does not extend to new items
- Collaborative filtering: *a*-"user"=avg. of items user likes, with excluded item *b*. Inside *a*=avg(liked items), *b*=avg(contained features) - e.g. document consists of words.
  - extends both to new users and new items, as all are featurized.

StarSpace: Applications

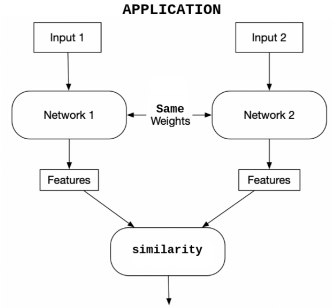
- Link prediction in graphs. Consider graph of relations (h, r, t) where h-head concept, r-relationship, t-tail concept, e.g. (Beyonce, born in, Houston). Then 2 options possible for sampled (h, r, t):
- $a = (h + r)/2$, $b = t$ or $a = h$, $b = (r + t)/2$
  - thus, can do question answering (Beyonce, born in, ?)
- Information retrieval:
  - supervised data given: a-query, b-relevant document
  - no supervision: a-random sentence from document b
    - both methods produce document embeddings!
- Word embeddings: a-surrounding words, b-central word.
- Sentence embeddings: a, b-random sentences from the same document (or close enough to each other)

May find embeddings for 2 tasks simultaneously (sum losses). E.g. jointly learn sentence embeddings and sentence classification according to sentiment (semi-supervised learning).

Table of Contents

Siamese network

- Siamese network uses 2 or more duplicate networks, producing embeddings.
- Then these embeddings are compared.

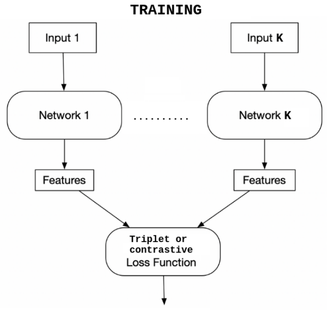


Application: finding similarity or a relationship between two comparable things.

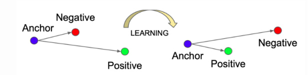
Application examples

- Classification:
  - input: two objects
  - output: how semantically similar they are (a proxy for class similarity)
- Information retrieval. inputs: are a document and a query,
  - input: a document and a query (may be an image and a visual query-find by image)
  - application: ranking by relevance to a query
- Paraphrase detection:
  - inputs: two sentences
  - output: a score of how similar they are.
- Signature verification
  - inputs: scans of two signatures
  - output: a score that they belong to the same person

Training



- Loss function:
  - similar objects should have similar embeddings
  - different objects should have distant embeddings



Losses[4]

Contrastive loss[3]:

$$\mathbb{I}[y_i = y_j] \left\| f_\theta(x_i) - f_\theta(x_j)) \right\|^2 + \mathbb{I}[y_i \neq y_j] \max \left\{0, \alpha - \| f_\theta(x_i) - f_\theta(x_j)) \| \right\}^2$$

Triplet loss:

- x - random object,
- x, x^+ are similar (belong to the same class)
- x, x^- are dissimilar (belong to different classes)
- $\alpha > 0$ - hyperparameter (desired margin between positive and negative pairs)

$$\mathcal{L}(x, x^+, x^-) = \max \left\{ \left\| f_\theta(x) - f_\theta(x^+)) \right\|^2 - \left\| f_\theta(x) - f_\theta(x^-)) \right\|^2 - \alpha; 0 \right\}$$

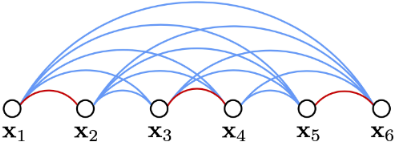
[3] Chopra et al. Learning a Similarity Metric Discriminatively, with Application to Face Verification.

[4] Good overview of losses.

Lifted structural loss

(a) Contrastive embedding

(b) Triplet embedding



(c) Lifted structured embedding

Lifted structural loss[5] utilizes all pairwise similarities.

[5] Deep Metric Learning via Lifted Structured Feature Embedding.

Lifted structural loss

- Define $D_{ij} = \|f(x_i) - f(x_j)\|_2$, P, N-the sets of positive and negative pairs.

$$\mathcal{L} = \frac{1}{2|P|} \sum_{(i,j)\in P} \max\left\{0; \mathcal{L}_{struct}^{ij}\right\}^2$$

$$\mathcal{L}_{struct}^{ij} = D_{ij} + \max\left\{ \max_{(i,k)\in N} \varepsilon - D_{ik}, \max_{(j,k)\in N} \varepsilon - D_{jk} \right\}$$

- Red parts represent *hard negative mining*.
    - concentrated on hardest negative object, slow training
- Solution: smoothed version:

$$\mathcal{L}_{struct}^{ij} = D_{ij} + \log\left(\sum_{(i,k)\in N} e^{\varepsilon - D_{ik}} + \sum_{(j,k)\in N} e^{\varepsilon - D_{jk}} \right)$$

Soft-Nearest Neighbors Loss[7]

- Soft-Nearest Neighbors Loss uses many positive objects from the minibatch.

$$\mathcal{L} = -\frac{1}{B}\sum_{i=1}^{B}\log\left(\frac{\sum_{j=1\dots B; i\neq j; y_i=y_j} e^{-\left\|f(x_i), f(x_j)\right\|^2/\tau}}{\sum_{k=1,\dots B; i\neq k} e^{-\|f(x_i), f(x_k)\|^2/\tau}}\right)$$

- τ - large => distance dominated by very similar embeddings, distant embeddings do not contribute.
- See also self-supervised contrastive loss[6].

Triplet and contrasive losses may be used for metric learning:

$\rho_\theta(x, x')$ small for x, x' belonging to the same class

[6]Khosla et al. Supervised Contrastive Learning

[7]Frosst et al. Analyzing and Improving Representations with the Soft Nearest Neighbor Loss.

Siamese architecture vs. classification

- Classification learns "what represents each class".
- Siamese network learns "what distinguishes each class from other classes".
- Classification outputs class scores.
- Siamese network outputs distances to each class in embedding space.
- Siamese network
    - is more robust to class imbalance
        - since during training consider instance of each class in turn evenly
        - model learns what makes classes the same/different from other pairs, so few examples of rare class may be enough (*one shot learning*).
    - works well in ensemble with classifier
        - use completely different logic, so much diversity in ensemble
    - requires more training
        - instance based learning=>pairwise learning.

Embeddings for classifier and Siamese network

Embeddings for classifier (last layer of MLP) and Siamese network for MNIST: