Embeddings

Andrey Gusev 171

How do we represent the meaning of a word?

How do we represent the meaning of a word?

- Discrete representation

How do we represent the meaning of a word?

- Discrete representation
- One-Hot representation

How do we represent the meaning of a word?

- Discrete representation
- One-Hot representation

$$\begin{aligned}&\text{motel} \; [0\;0\;0\;0\;0\;0\;0\;0\;0\;0\;1\;0\;0\;0\;0]^T \\ &\text{hotel} \; [0\;0\;0\;0\;0\;0\;0\;1\;0\;0\;0\;0\;0\;0\;0] = 0\end{aligned}$$

How do we represent the meaning of a word?

- Discrete representation
- One-Hot representation

$$\text{motel}\ [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0]^T$$
$$\text{hotel}\ [0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0] = 0$$

- Distributional similarity based representations

How do we represent the meaning of a word?

- Discrete representation
- One-Hot representation

$$\text{motel}\ [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0]^T$$
$$\text{hotel}\ \ [0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0] = 0$$

- Distributional similarity based representations

$$\textit{linguistics} = \begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \end{pmatrix}$$

How do we represent the meaning of a word?

- Discrete representation
- One-Hot representation

$$\text{motel}\ [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0]^T$$
$$\text{hotel}\ [0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0] = 0$$

- Distributional similarity based representations

$$\textit{linguistics} = \begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \end{pmatrix}$$

government debt problems turning into banking crises as has happened in

saying that Europe needs unified banking regulation to replace the hodgepodge

↖ These words will represent *banking* ↗

word2vec

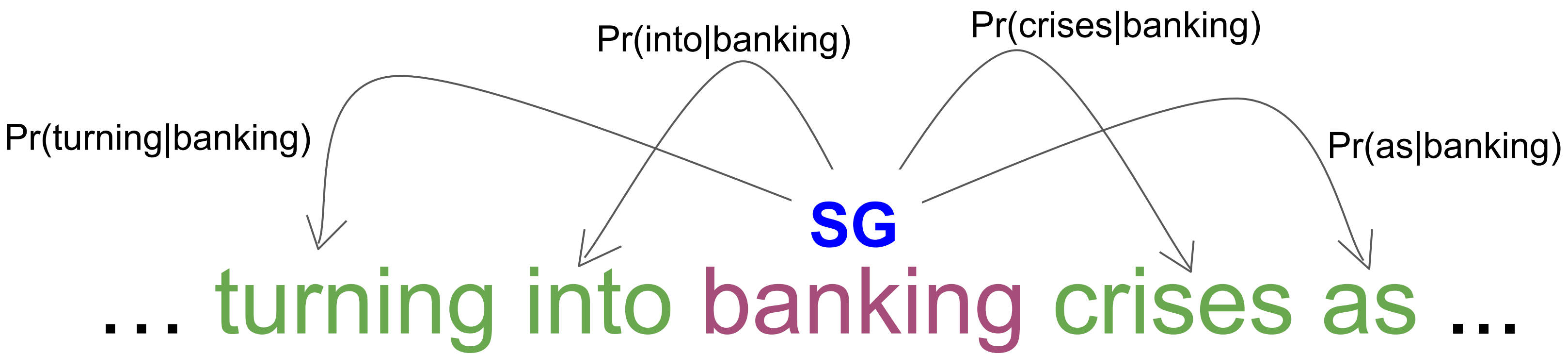
Two algorithms:

1. Skip-gram (SG)
2. Continuous Bag of Words (CDOW)

word2vec

Two algorithms:

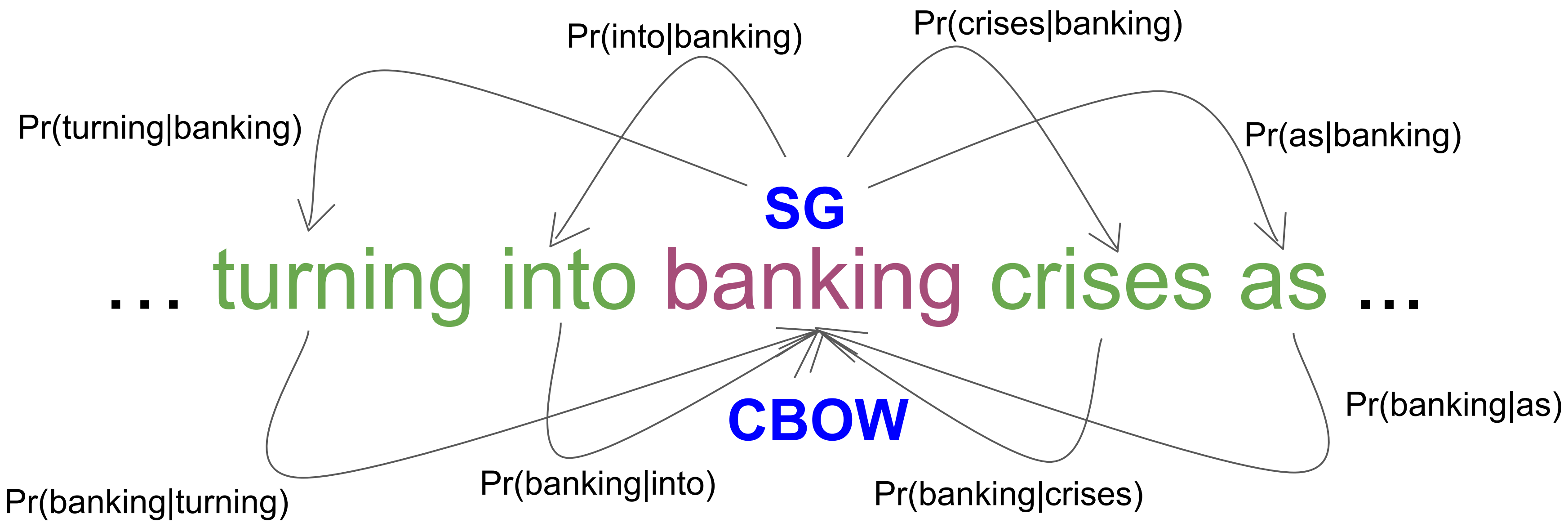
1. Skip-gram (SG)
2. Continuous Bag of Words (CDOW)



word2vec

Two algorithms:

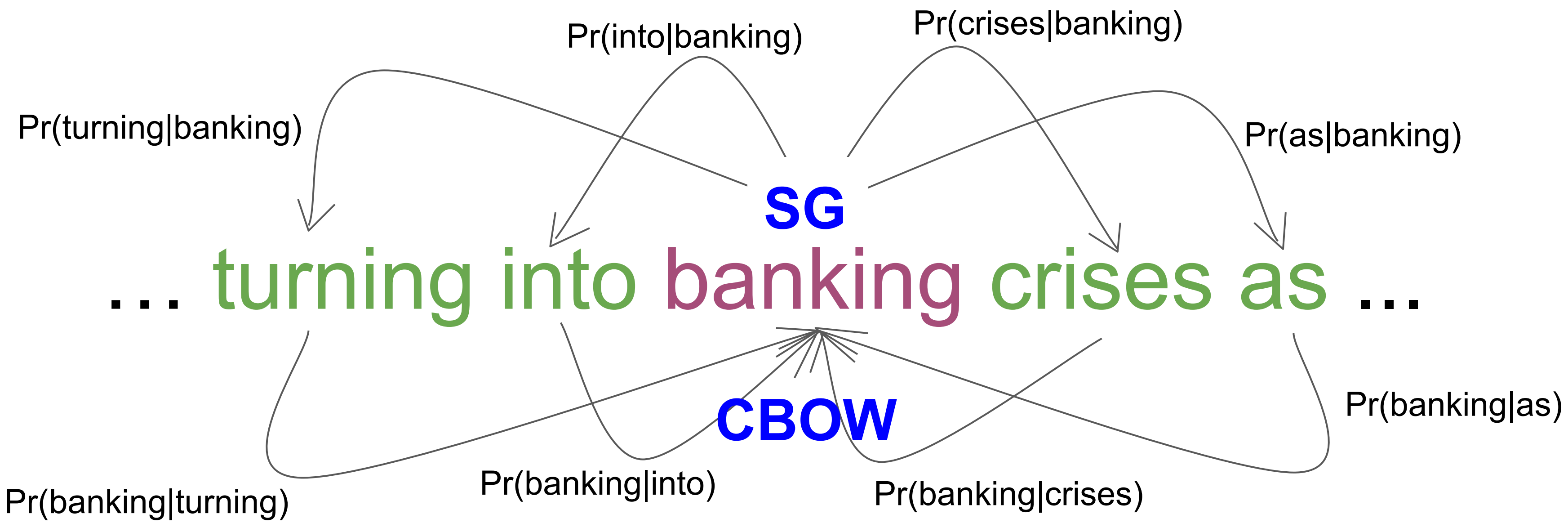
1. Skip-gram (SG)
2. Continuous Bag of Words (CBOW)



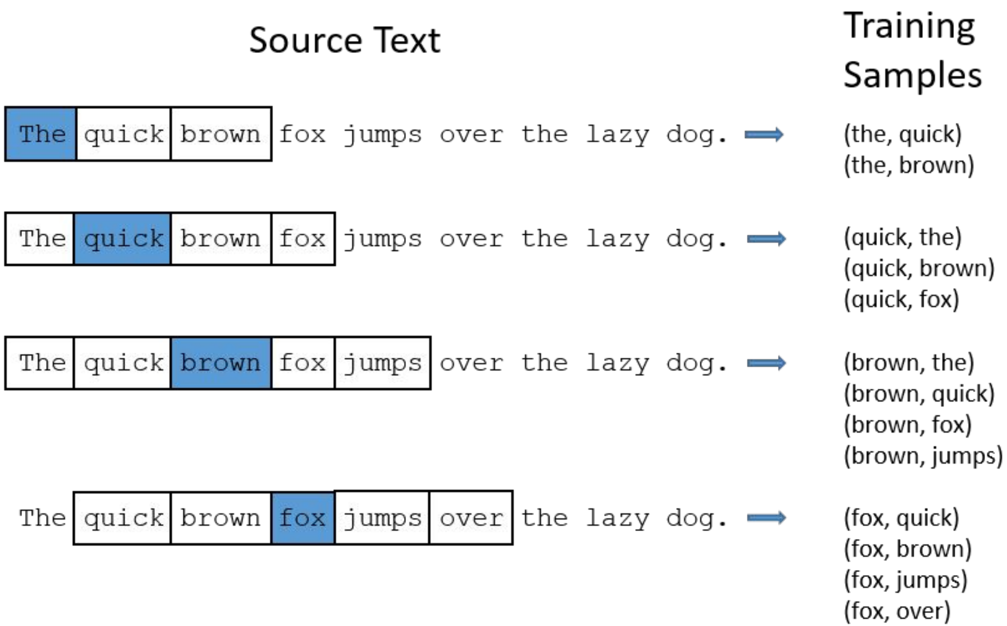
word2vec

Two algorithms:

1. **Skip-gram (SG)**
2. Continuous Bag of Words (CDOW)



Creating data for word2vec



word2vec

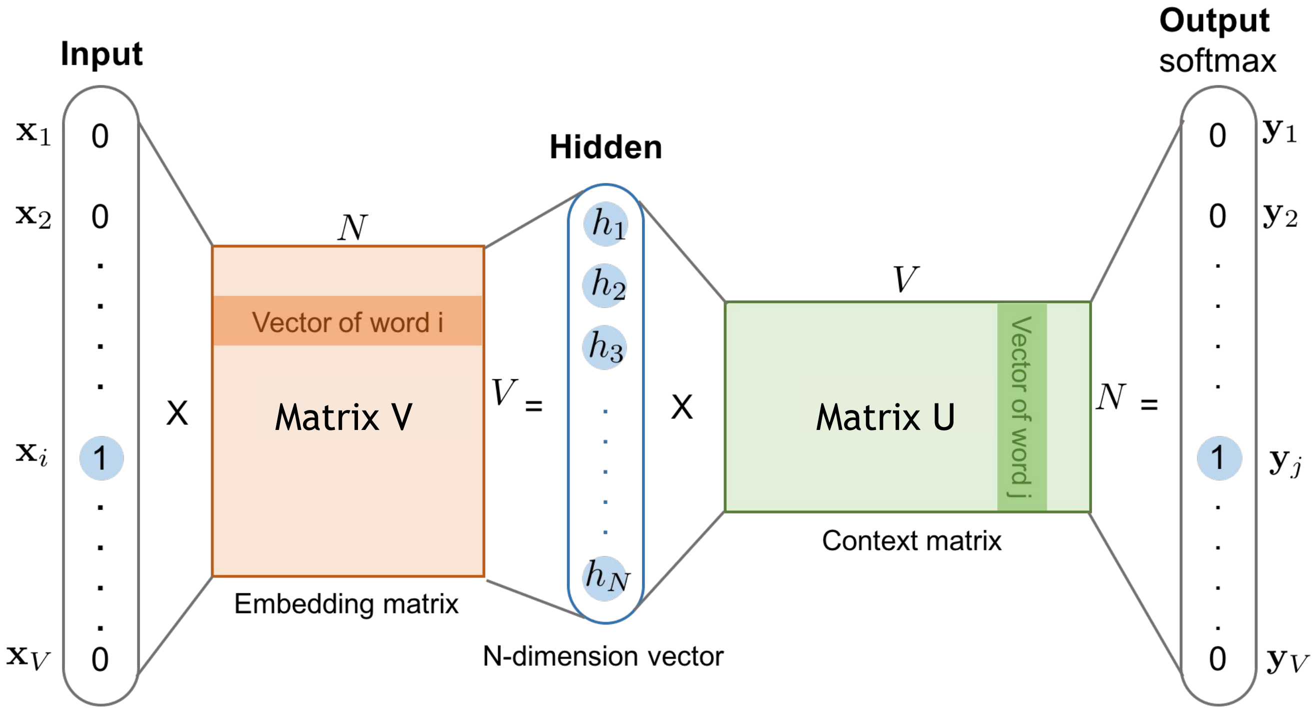
Objective function: Maximize the probability of any context word given the current center word.

$$J'(\theta) = \prod_{t=1}^{T} \prod_{-m \leqslant j \leqslant m} \Pr(w_{t+j} | w_t; \theta)$$

$$J(\theta) = -\frac{1}{T} \sum_{t=1}^{T} \sum_{-m \leqslant j \leqslant m} \log \Pr(w_{t+j} | w_t; \theta)$$

Where theta represents all variables we will optimize.

Skip-gram



Input
$\mathbf{x}_1$
0
$\mathbf{x}_2$
0
$\mathbf{x}_i$
1
$\mathbf{x}_V$
0
X
N
Vector of word i
Matrix V
Embedding matrix
V =
Hidden
h_1
h_2
h_3
h_N
N-dimension vector
X
V
Matrix U
Vector of word j
Context matrix
N =
Output
softmax
0
$\mathbf{y}_1$
0
$\mathbf{y}_2$
1
$\mathbf{y}_j$
0
$\mathbf{y}_V$

Skip-gram

We try to predict surrounding words in a window of radius *m* of every word.

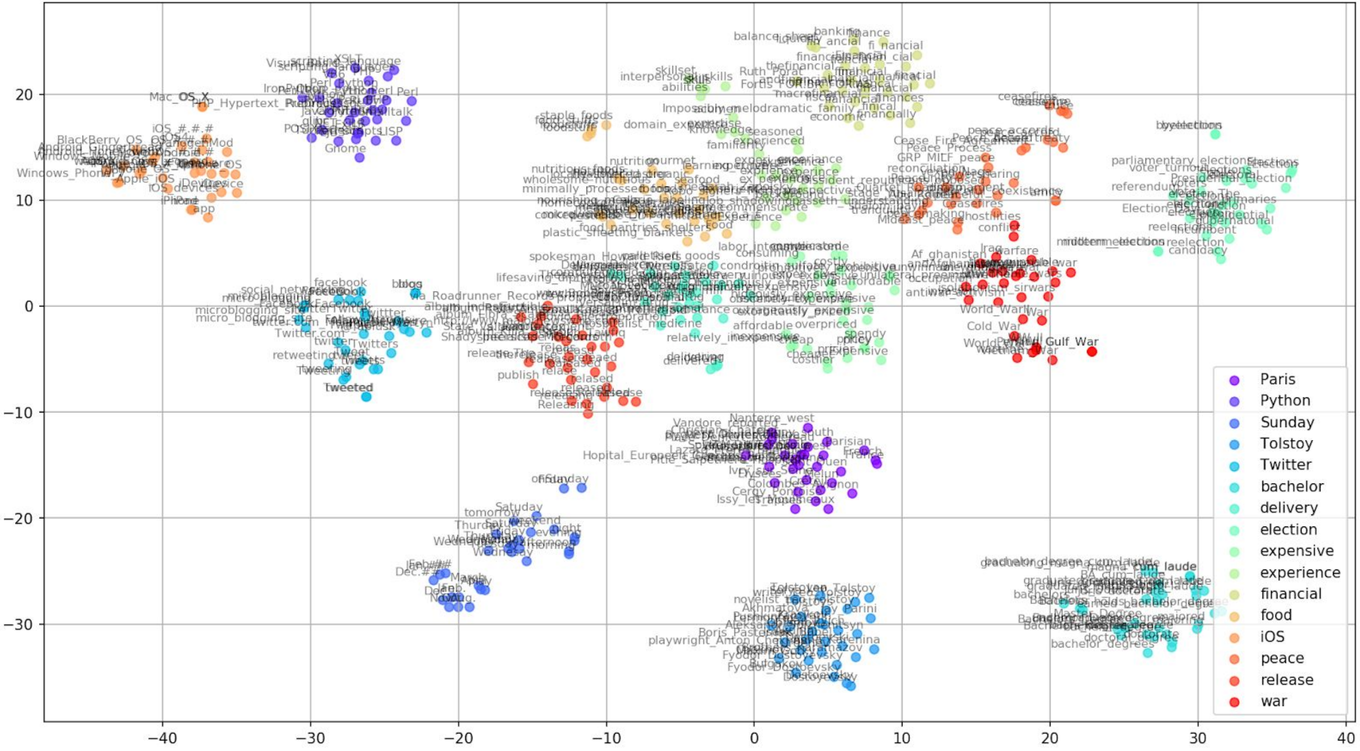
$$\Pr(w_o|w_c) = \frac{\exp(u_o^T v_c)}{\sum_{w=1}^{V} \exp(u_w^T v_c)}$$

$$\theta = \begin{bmatrix} v_{aardark} \\ v_a \\ \vdots \\ v_{zebra} \\ u_{aardark} \\ u_a \\ \vdots \\ u_{zebra} \end{bmatrix}$$

Where *o* is the outside word index, *c* is the center word index. Softmax using the outside word to obtain probability of the center word.

word2vec

word2vec improves objective function by putting similar words nearby in space.



Other technique

- 2 options: windows and full document

Other technique

- 2 options: windows and full document
- Window: Similar to word2vec, use window around each word to capture both syntactic and semantic information.

Other technique

- 2 options: windows and full document
- Window: Similar to word2vec, use window around each word to capture both syntactic and semantic information.
- Word-document co-occurrence matrix will give general topics leading to “Latent Semantic Analysis”.

Co-occurrence matrix example

Corpus: *I enjoy flying. I like NLP. I like deep learning.*

$$
X = \begin{array}{c} \\ I \\ like \\ enjoy \\ deep \\ learning \\ NLP \\ flying \\ . \end{array}
\begin{array}{c}
\begin{array}{cccccccc} I & like & enjoy & deep & learning & NLP & flying & . \end{array} \\
\begin{bmatrix}
0 & 2 & 1 & 0 & 0 & 0 & 0 & 0 \\
2 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 0
\end{bmatrix}
\end{array}
$$

Problems and solutions

- Idea: store “most” of the important information in a fixed, small number of dimensions: a dense vector.

Problems and solutions

- Idea: store “most” of the important information in a fixed, small number of dimensions: a dense vector.
- Usually 25-1000 dimensions, similar to word2vec.

Problems and solutions

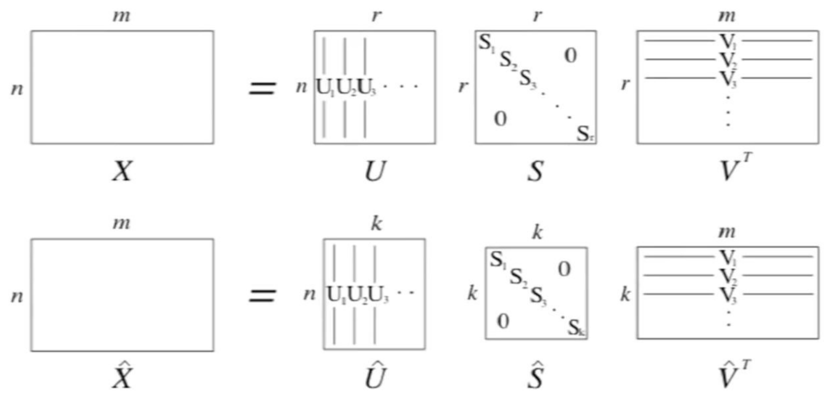
- Idea: store “most” of the important information in a fixed, small number of dimensions: a dense vector.
- Usually 25-1000 dimensions, similar to word2vec.
- How to reduce the dimensionality?

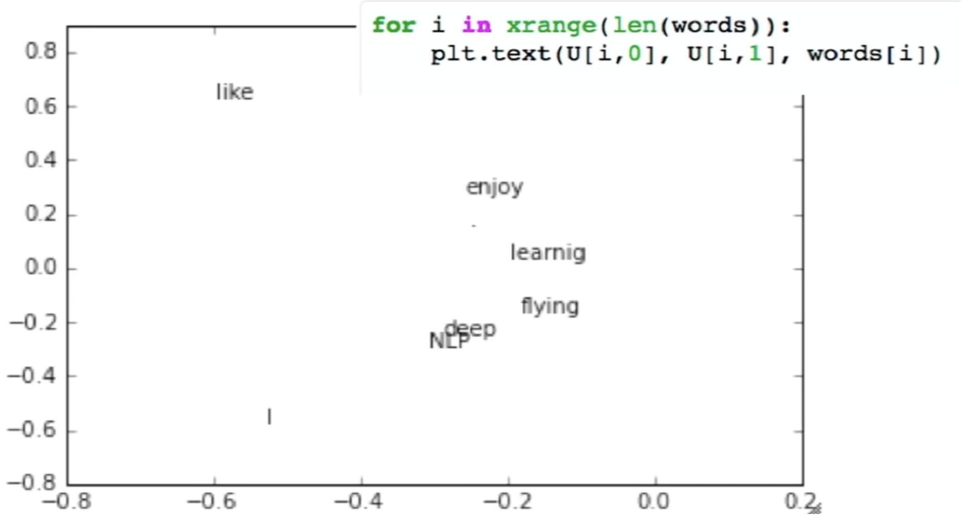
Problems and solutions

- Idea: store “most” of the important information in a fixed, small number of dimensions: a dense vector.
- Usually 25-1000 dimensions, similar to word2vec.
- How to reduce the dimensionality? **SVD!**

Dimensionality reduction of co-occurrence matrix

Singular value decomposition of co-occurrence matrix X:





Problems with SVD

- Computational cost scales quadratically for *n* x *m* matrix: $O(mn^2)$

Problems with SVD

- Computational cost scales quadratically for n x m matrix: $O(mn^2)$
- Bad for millions of words or documents.

Problems with SVD

- Computational cost scales quadratically for n x m matrix: $O(mn^2)$
- Bad for millions of words or documents.
- Hard to incorporate new words or documents.

GloVe

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

$$f(x) = \begin{cases} (x/\overset{100}{x_{\max}})^{\overset{3/4}{\alpha}} & \text{if } x < x_{\max} \\ 1 & \text{otherwise} \end{cases}$$

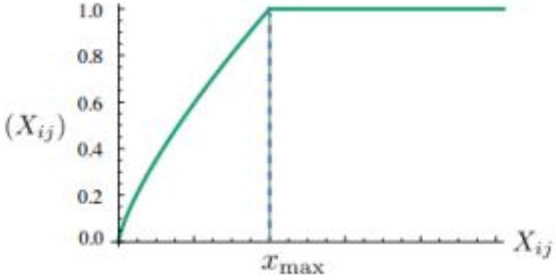
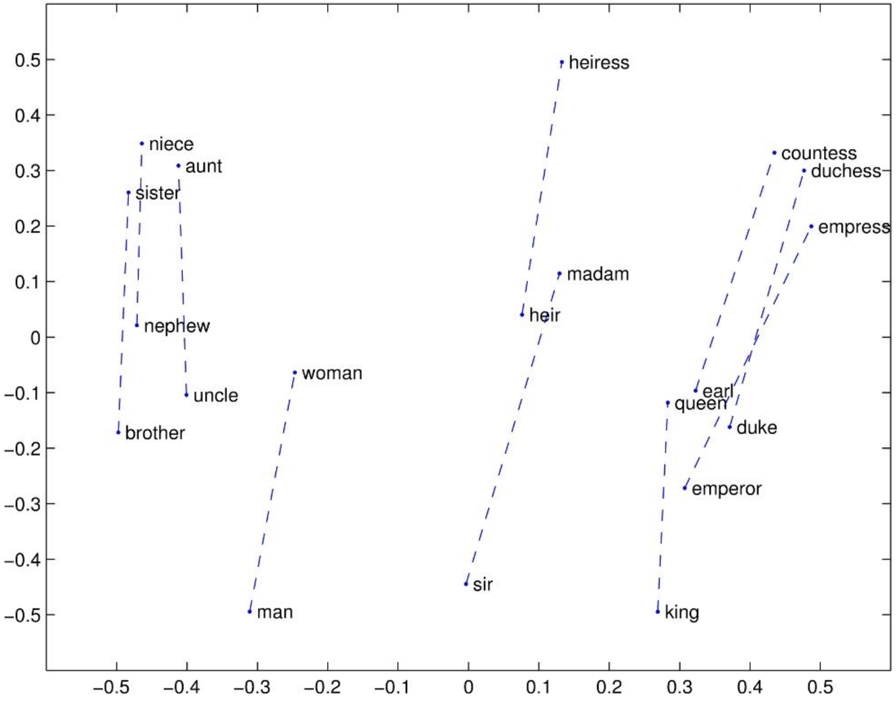


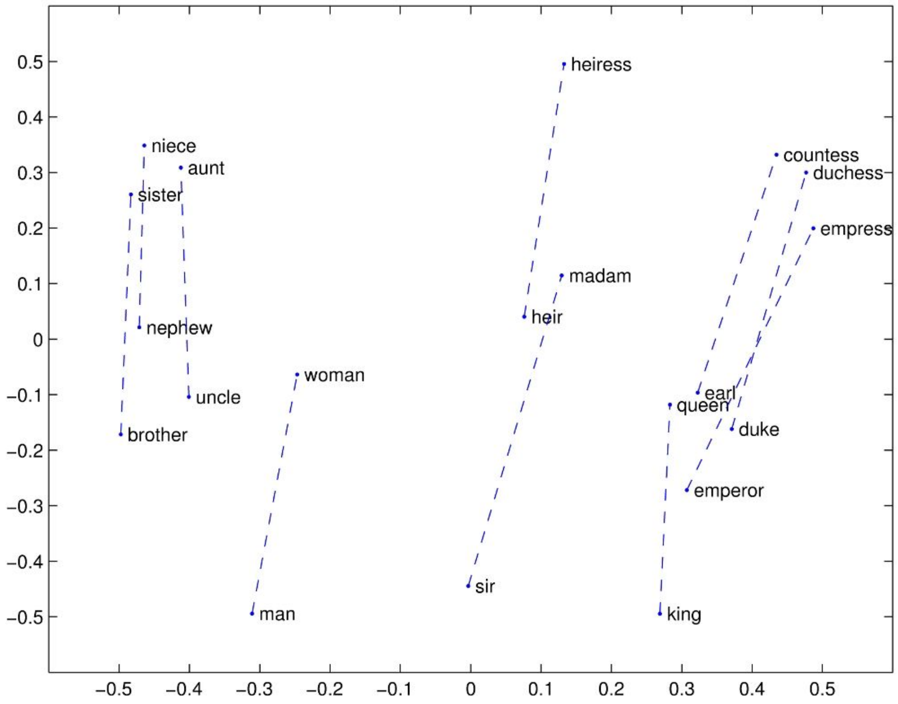
Figure 1: Weighting function f with $\alpha = 3/4$.

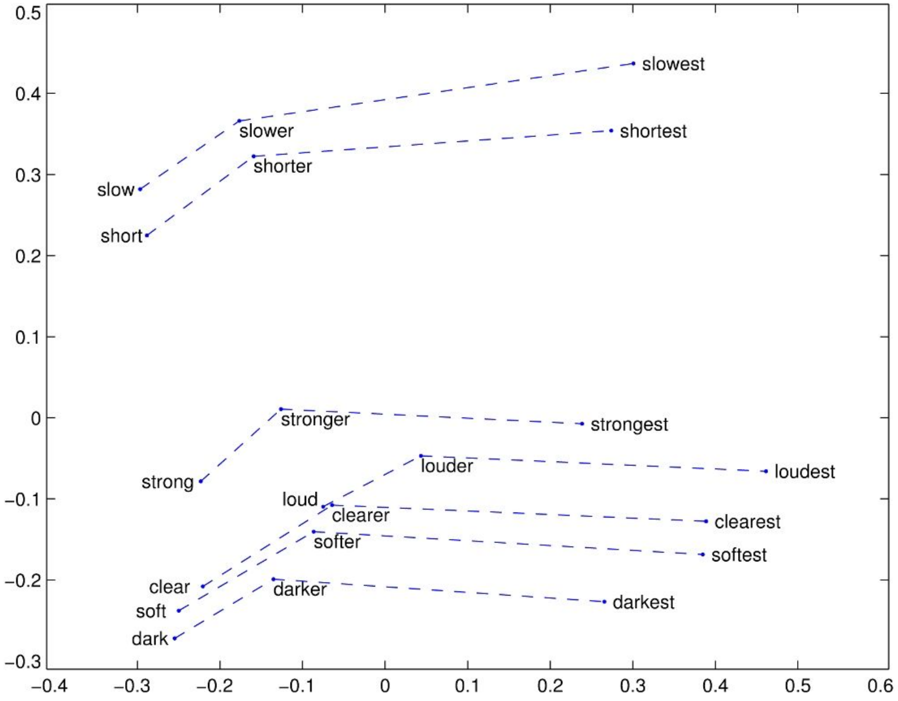
- Combining the best of both techniques.
- Fast training.
- Scalable to huge corpora.
- Good performance even with small corpus and small vectors.

GloVe visualisation



GloVe visualisation





Other fun embedding analogies

Expression	*Nearest token*
Paris - France + Italy	Rome
bigger - big + cold	colder
sushi - Japan + Germany	bratwurst
Cu - copper + gold	Au
Windows - Microsoft + Google	Android
Montreal Canadiens - Montreal + Toronto	Toronto Maple Leafs

fastText

- Starts with word representations that are averaged into text representation and feed them to a linear classifier (multinomial logistic regression).
- Text representation as a hidden state that can be shared among features and classes.
- Uses a bag of n-grams to maintain efficiency without losing accuracy. No explicit use of word order.
- Softmax layer to obtain a probability distribution over pre-defined classes.

fastText

- Hierarchial Softmax: Based on Huffman Coding Tree Used to reduce computational complexity $O(kh)$ to $O(h\log(k))$, where k is the number of classes and h is dimension of text representation.
- Uses hashing trick to maintain fast and memory efficient mapping of the n-grams.
- It is written in C++ and supports multiprocessing during training.

ELMo

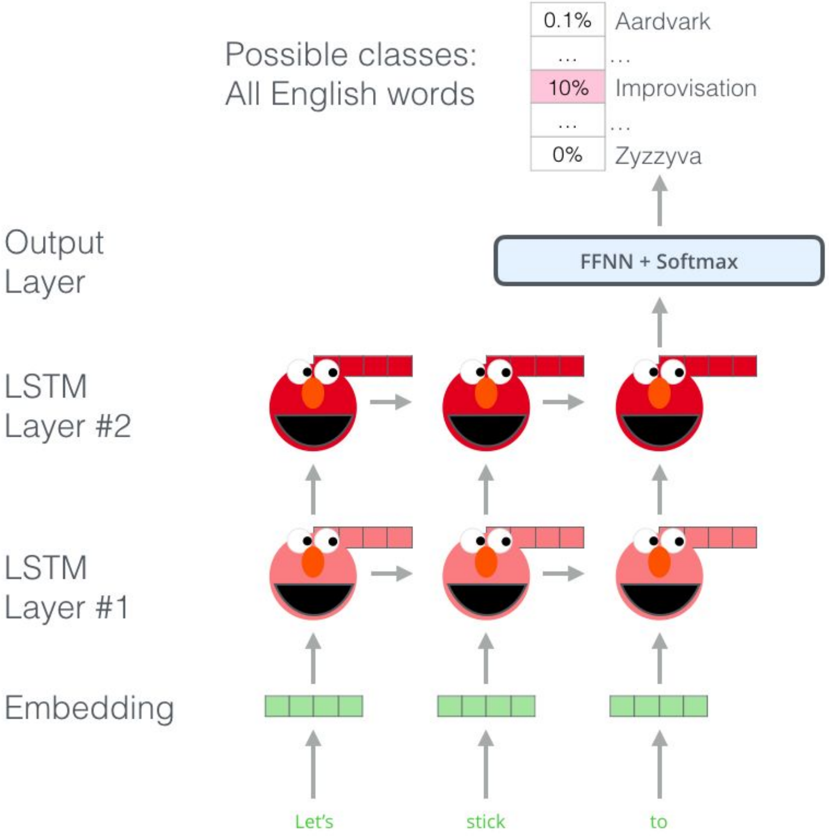

Hey ELMo, what's the embedding of the word "stick"?
There are multiple possible embeddings! Use it in a sentence.
Oh, okay. Here: "Let's stick to improvisation in this skit"
Oh in that case, the embedding is: -0.02, -0.16, 0.12, -0.1 ….etc

ELMo

ELMo representations are:

- *Contextual*: The representation for each word depends on the entire context in which it is used.
- *Deep*: The word representations combine all layers of a deep pre-trained neural network.
- *Character based*: ELMo representations are purely character based, allowing the network to use morphological clues to form robust representations for out-of-vocabulary tokens unseen in training.

ELMo


Possible classes:
All English words
0.1% Aardvark
... ...
10% Improvisation
... ...
0% Zyzzyva
Output
Layer
FFNN + Softmax
LSTM
Layer #2
LSTM
Layer #1
Embedding
Let's
stick
to

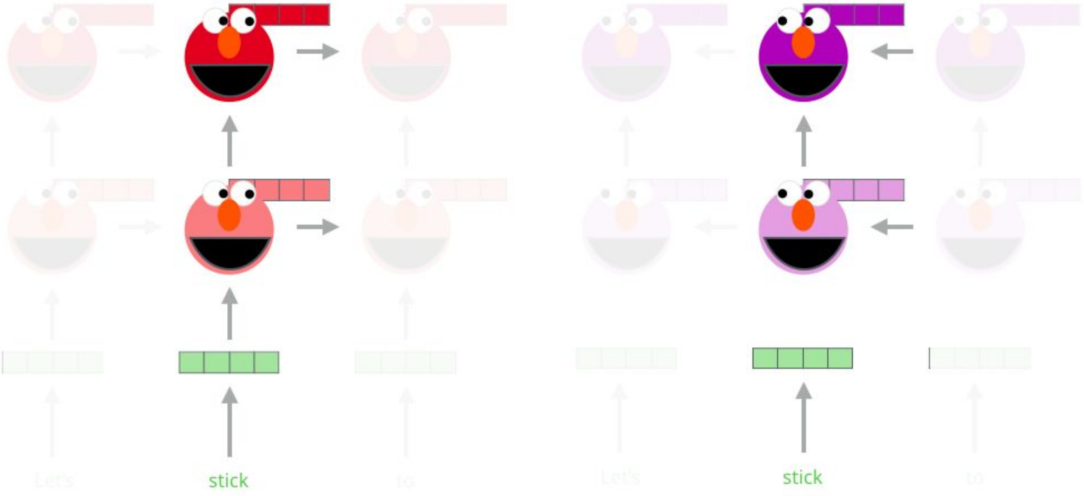
ELMo

Embedding of “stick” in “Let’s stick to” - Step #2

1- Concatenate hidden layers

Forward Language Model

Backward Language Model



2- Multiply each vector by a weight based on the task

3- Sum the (now weighted) vectors

ELMo embedding of “stick” for this task in this context

Вопросы

1. Опишите принцип обучения эмбеддингов Continuous Bag Of Words.
2. На каких данных обучается Skip-gram? Что подается модели на вход и что ожидается на выходе при обучении?
3. В чем заключается техника Latent Semantic Analysis? Какие проблемы есть у этой техники?