CS 4803 / 7643: Deep Learning
Guest Lecture:
Embeddings and world2vec

Feb. 18th 2020

Ledell Wu

Research Engineer, Facebook AI

ledell@fb.com

Outline

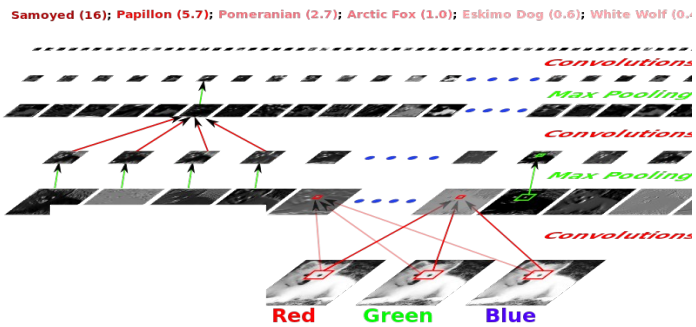
- Word Embeddings
- Graph Embeddings
- Applications
- Discussions

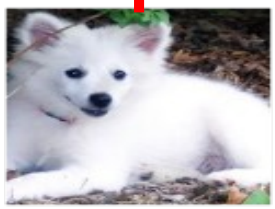
word2vec

world2vec

Mapping objects to Vectors through a trainable function

[0.4, -1.3, 2.5, -0.7,…..]



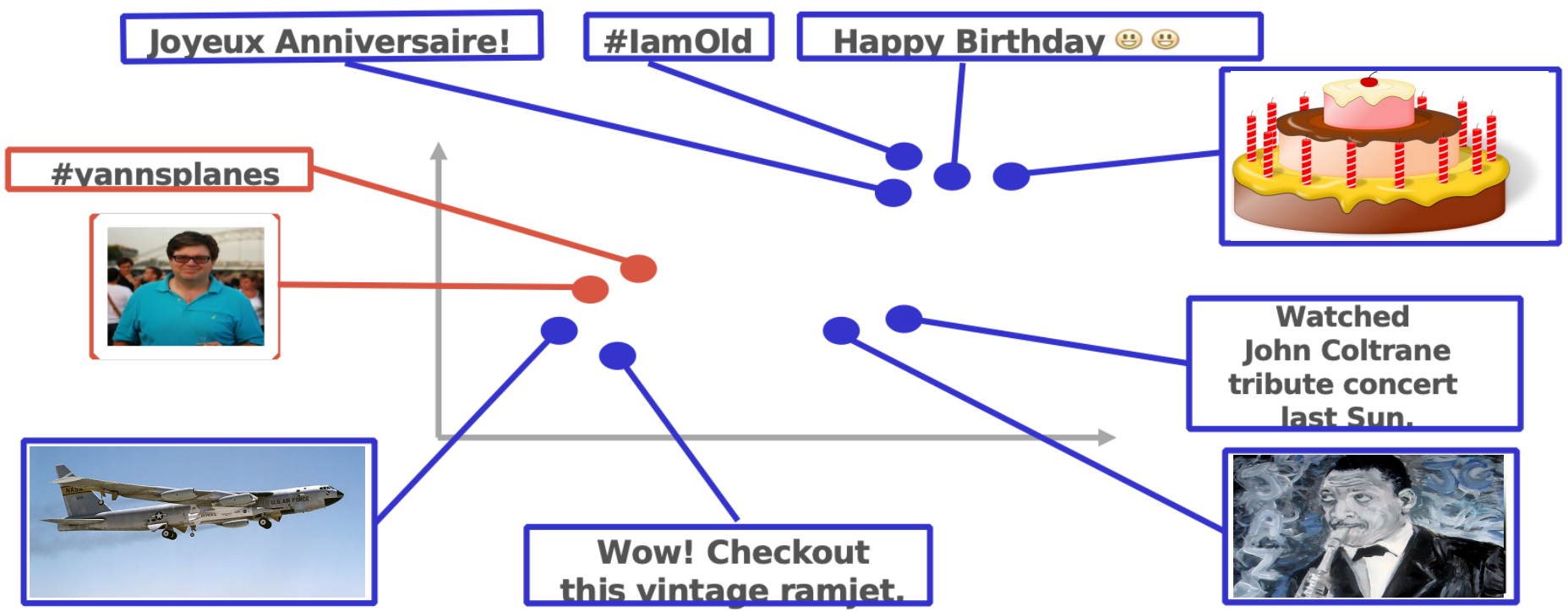


[0.2, -2.1, 0.4, -0.5,…..]

Neural Net

"The neighbors' dog was a samoyed, which looks a lot like a Siberian husky"

(Credit: Yann LeCun)



(Credit: Yann LeCun)

Outline

- Word Embeddings
- Graph Embeddings
- Applications
- Discussions

Representing words as discrete symbols

In traditional NLP, we regard words as discrete symbols:
hotel, conference, motel – a localist representation

Means one 1, the rest 0s

Words can be represented by one-hot vectors:

motel = [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]
hotel = [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0]

Vector dimension = number of words in vocabulary (e.g., 500,000)

(Credit: Richard Socher, Christopher Manning)

Problem with words as discrete symbols

Example: in web search, if user searches for "Seattle motel", we would like to match documents containing "Seattle hotel".

But:

motel = [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]
hotel = [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0]

These two vectors are orthogonal.

There is no natural notion of **similarity** for one-hot vectors!

Solution:

- Could try to rely on WordNet's list of synonyms to get similarity?
  - But it is well-known to fail badly: incompleteness, etc.
- **Instead: learn to encode similarity in the vectors themselves**

(Credit: Richard Socher, Christopher Manning)

Representing words by their context

- Distributional semantics: **A word's meaning is given by the words that frequently appear close-by**
  - *"You shall know a word by the company it keeps"* (J. R. Firth 1957: 11)
  - One of the most successful ideas of modern statistical NLP!
- When a word *w* appears in a text, its **context** is the set of words that appear nearby (within a fixed-size window).
- Use the many contexts of *w* to build up a representation of *w*

…government debt problems turning into	***banking***	*crises as happened in 2009…*
…saying that Europe needs unified	***banking***	*regulation to replace the hodgepodge…*
…India has just given its	***banking***	*system a shot in the arm…*

These context words will represent ***banking***

(Credit: Richard Socher, Christopher Manning)

Word vectors

We will build a dense vector for each word, chosen so that it is similar to vectors of words that appear in similar contexts

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

Note: word vectors are sometimes called word embeddings or word representations. They are a distributed representation.

(Credit: Richard Socher, Christopher Manning)

3. Word2vec: Overview

Word2vec (Mikolov et al. 2013) is a framework for learning word vectors

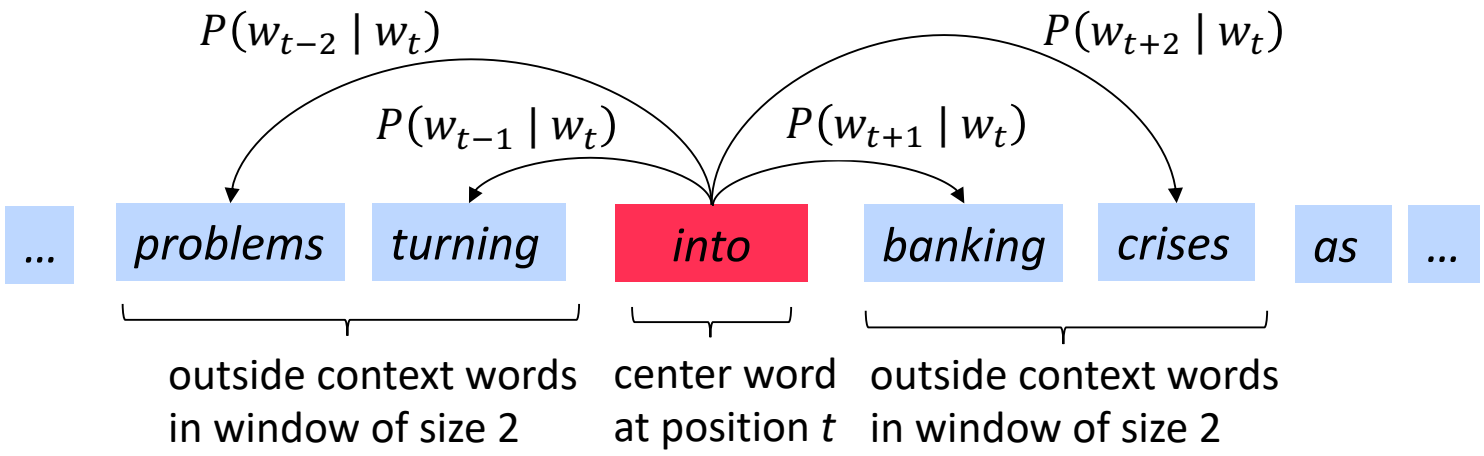
Idea:

- We have a large corpus of text
- Every word in a fixed vocabulary is represented by a vector
- Go through each position *t* in the text, which has a center word *c* and context ("outside") words *o*
- Use the similarity of the word vectors for *c* and *o* to calculate the probability of *o* given *c* (or vice versa)
- Keep adjusting the word vectors to maximize this probability

(Credit: Richard Socher, Christopher Manning)

Word2Vec Overview

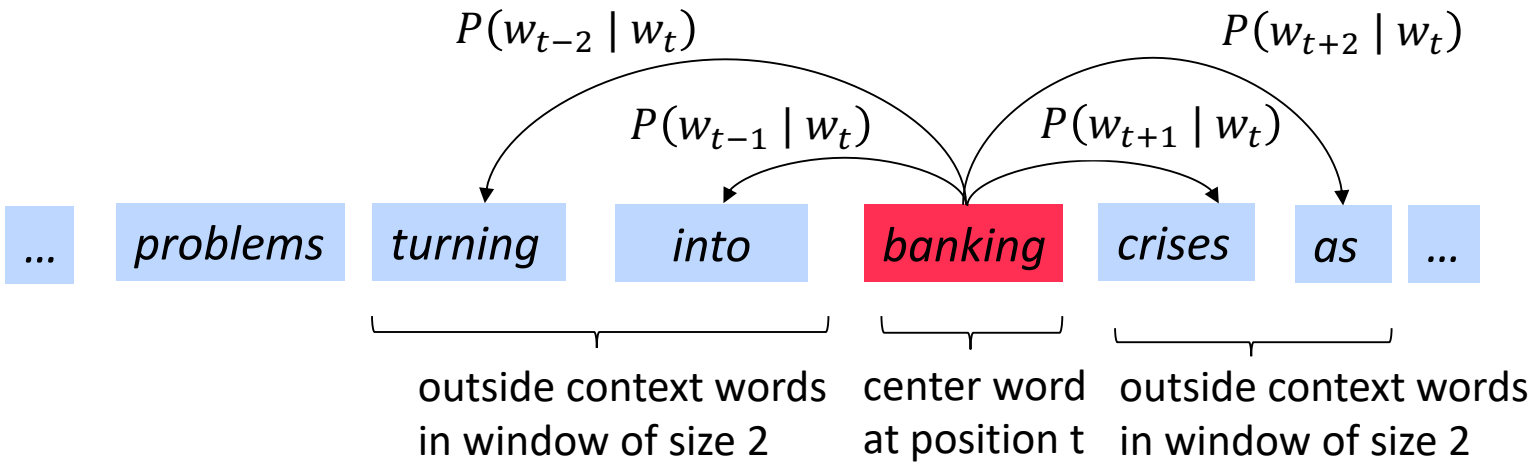
- Example windows and process for computing $P(w_{t+j} \mid w_t)$



(Credit: Richard Socher, Christopher Manning)

Word2Vec Overview

- Example windows and process for computing $P(w_{t+j} \mid w_t)$



(Credit: Richard Socher, Christopher Manning)

Word2vec: objective function

For each position $t = 1, \ldots, T$, predict context words within a window of fixed size m, given center word w_j.

Likelihood = $$L(\theta) = \prod_{t=1}^{T} \prod_{\substack{-m \le j \le m \\ j \neq 0}} P(w_{t+j} \mid w_t; \theta)$$

θ is all variables to be optimized

sometimes called *cost* or *loss* function

The objective function $J(\theta)$ is the (average) negative log likelihood:

$$J(\theta) = -\frac{1}{T} \log L(\theta) = -\frac{1}{T} \sum_{t=1}^{T} \sum_{\substack{-m \le j \le m \\ j \neq 0}} \log P(w_{t+j} \mid w_t; \theta)$$

Minimizing objective function $\Leftrightarrow$ Maximizing predictive accuracy

(Credit: Richard Socher, Christopher Manning)

Word2vec: objective function

- We want to minimize the objective function:

$$J(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-m\leq j\leq m \\ j\neq 0}} \log P\left(w_{t+j} \mid w_t; \theta\right)$$

- Question: How to calculate $P\left(w_{t+j} \mid w_t; \theta\right)$?
- Answer: We will *use two* vectors per word *w*:
  - v_w when *w* is a center word
  - u_w when *w* is a context word
- Then for a center word *c* and a context word *o*:

$$P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w\in V}\exp(u_w^T v_c)}$$

(Credit: Richard Socher, Christopher Manning)

Word2vec: prediction function

② Exponentiation makes anything positive

① Dot product compares similarity of *o* and *c*.
$u^T v = u.v = \sum_{i=1}^{n} u_i v_i$
Larger dot product = larger probability

$$P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}$$

③ Normalize over entire vocabulary to give probability distribution

- This is an example of the **softmax function** $\mathbb{R}^n \rightarrow (0,1)^n$

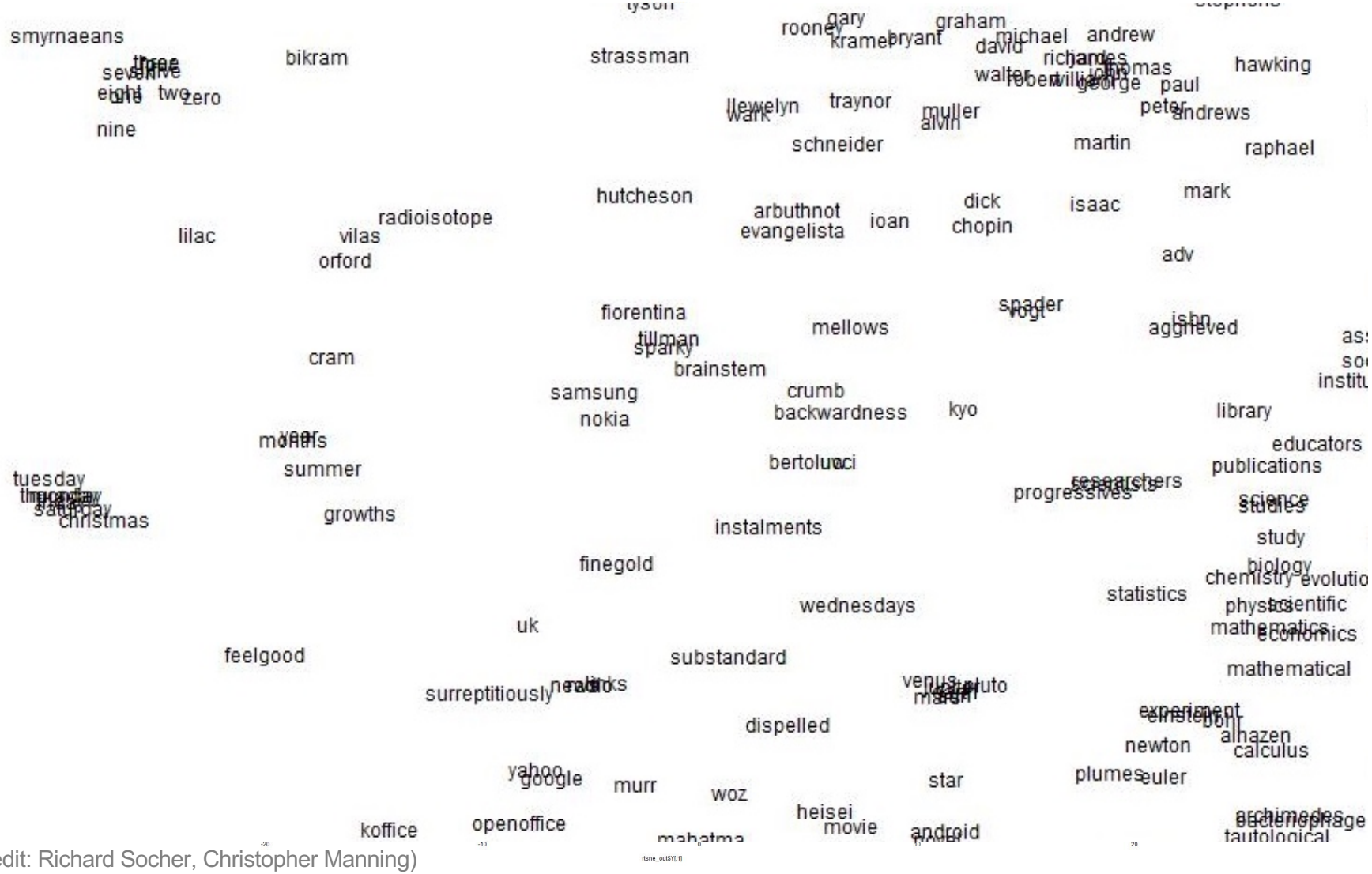
$$\text{softmax}(x_i) = \frac{\exp(x_i)}{\sum_{j=1}^{n} \exp(x_j)} = p_i$$

Open region

- The softmax function maps arbitrary values x_i to a probability distribution p_i
  - "max" because amplifies probability of largest x_i
  - "soft" because still assigns some probability to smaller x_i
  - Frequently used in Deep Learning

(Credit: Richard Socher, Christopher Manning)

Word2vec maximizes objective function by putting similar words nearby in space



(Credit: Richard Socher, Christopher Manning)

Word2vec: More details

Why two vectors? → Easier optimization. Average both at the end.

Two model variants:

1. Skip-grams (SG)

   Predict context ("outside") words (position independent) given center word

2. Continuous Bag of Words (CBOW)

   Predict center word from (bag of) context words

This lecture so far: **Skip-gram model**

Additional efficiency in training:

1. Negative sampling

A subset of words:

Practical: 5 - 10

So far: Focus on **naïve softmax** (simpler training method)

5. How to evaluate word vectors?

- Related to general evaluation in NLP: Intrinsic vs. extrinsic
- Intrinsic:
  - Evaluation on a specific/intermediate subtask
  - Fast to compute
  - Helps to understand that system
  - Not clear if really helpful unless correlation to real task is established
- Extrinsic:
  - Evaluation on a real task
  - Can take a long time to compute accuracy
  - Unclear if the subsystem is the problem or its interaction or other subsystems
  - If replacing exactly one subsystem with another improves accuracy → Winning!

(Credit: Richard Socher, Christopher Manning)

Intrinsic word vector evaluation

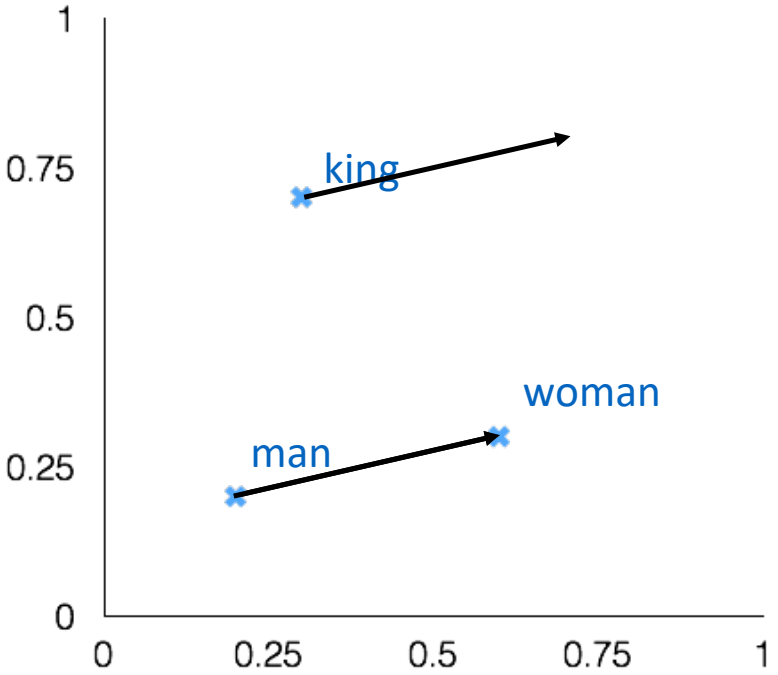
- Word Vector Analogies

a:b :: c:?

man:woman :: king:?

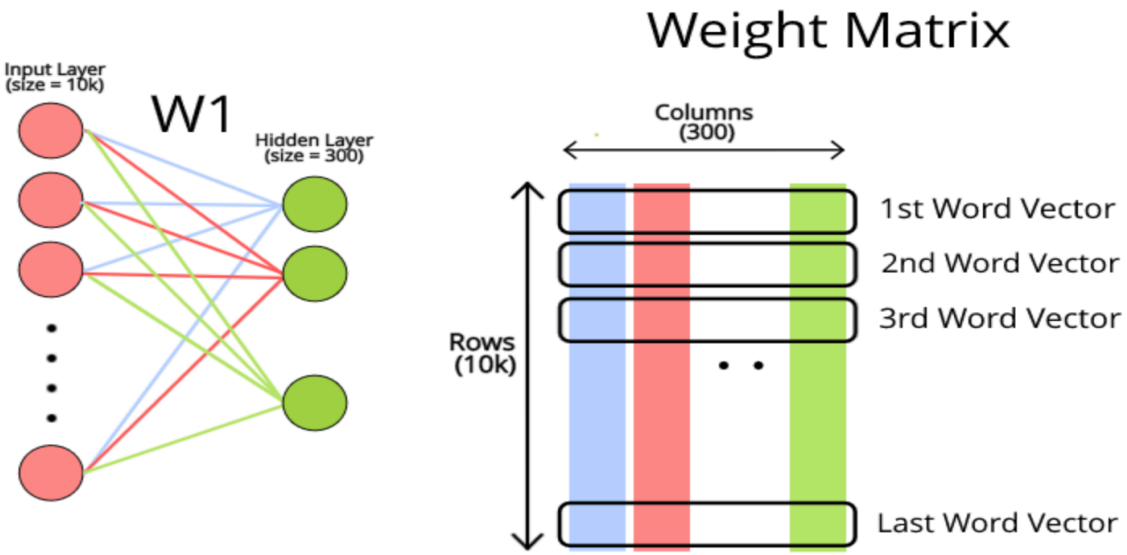
$$d = \arg\max_i \frac{(x_b - x_a + x_c)^T x_i}{||x_b - x_a + x_c||}$$

- Evaluate word vectors by how well their cosine distance after addition captures intuitive semantic and syntactic analogy questions
- Discarding the input words from the search!
- Problem: What if the information is there but not linear?



(Credit: Richard Socher, Christopher Manning)

Word Embeddings Continued



- GloVe [Pennington et al. 2014]
- FastText [Bojanowski et al 2017]
  - subword unit
  - text classification

https://fasttext.cc/

(Picture from: https://mc.ai/deep-nlp-word-vectors-with-word2vec/)

More on NLP

Future Lectures will cover:

- Recurrent Neural Networks (RNNs)
- Self-Attention, Transformers
- Language modeling, translation, etc.

Word embeddings can be used in other neural net models such as RNN.

Outline

- Word Embeddings
- Graph Embeddings
- Applications
- Discussions

(Big) Graph Data is Everywhere

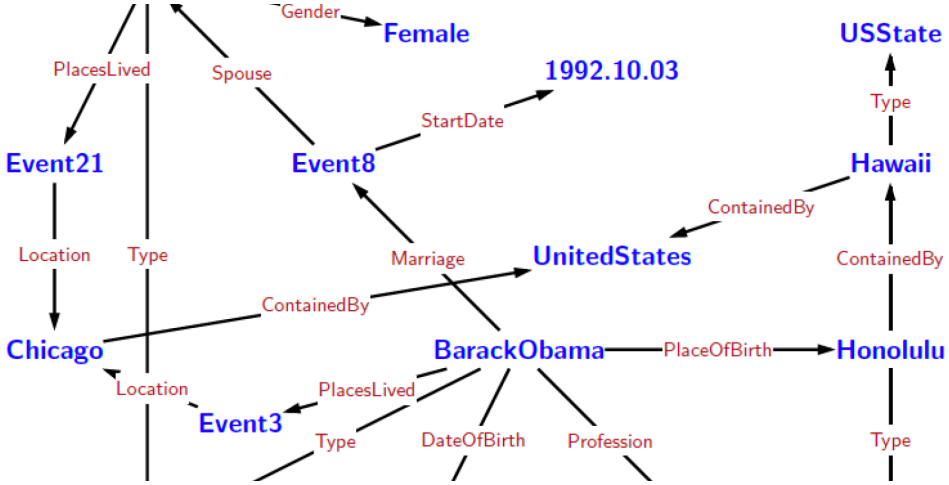
Recommender Systems

Deals with graph-like data, but supervised
(MovieLens, …)

	user_id	movie_id	rating
0	196	242	3
1	186	302	3

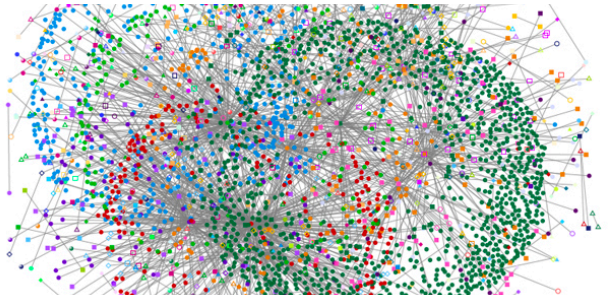
Knowledge Graphs

Standard domain for studying graph embeddings
(Freebase, …)



Social Graphs

Predict attributes based on homophily
or structural similarity
(Twitter, Yelp, …)



Wang, Zhenghao & Yan, Shengquan & Wang, Huaming & Huang, Xuedong. (2014). An Overview of Microsoft Deep QA System on Stanford WebQuestions Benchmark.

https://threatpost.com/researchers-graph-social-networks-spot-spammers-061711/75346/

(Credit: Adam Lerer)

Graph Embedding & Matrix Completion

- **Relations between items (and people)**
  - Items in {people, movies, pages, articles, products, word sequences....}
  - Predict if someone will like an item, if a word will follow a word sequence

	item1	item2	item3	item4	item5	item6	….	itemN
person1		+	-	+		+		
person2			+	+	+	-	+	
person3	+	?	-	+		+		
….								
personP			+	+	?		+	

(Credit: Yann LeCun)

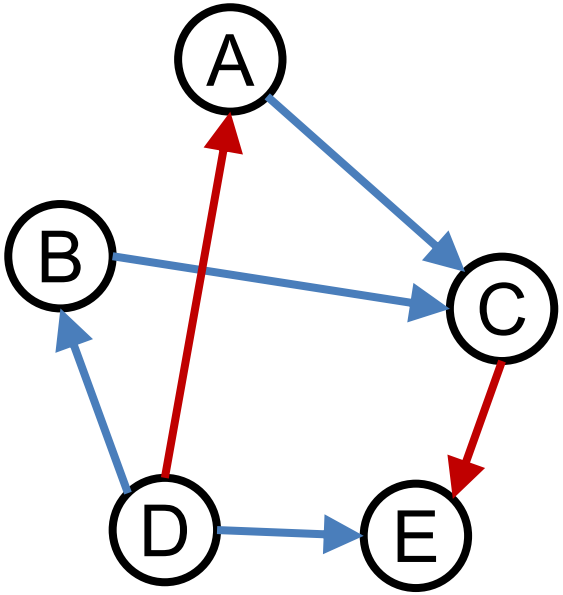
Graph Embedding & Matrix Completion

- **Find Xi and Yj such that F(Xi,Yj) = Mij**
  - F is a “simple” function, such as a dot product

	item1	item2	item3	item4	item5	item6	….	itemN
person1		+	-	+		+		
person2			+	+	+	-	+	
person3	+	?	-	+		+		
….								
personP			+	+	?		+	

(Credit: Yann LeCun)

Graph Embeddings

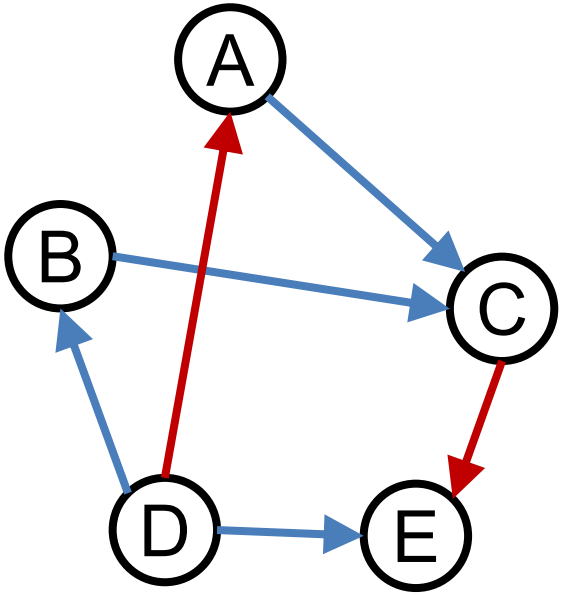


A multi-relation graph

- **Embedding:** A learned map from *entities* to *vectors of numbers* that encodes similarity
  - Word embeddings: word -> vector
  - Graph embeddings: node -> vector
- **Graph Embedding:** Train embeddings with the objective that **connected nodes have more similar embeddings** than unconnected nodes
  - Not the same as graph neural networks. GNNs are a parametric, supervised model over graphs.

(Credit: Adam Lerer)

Why Graph Embeddings?



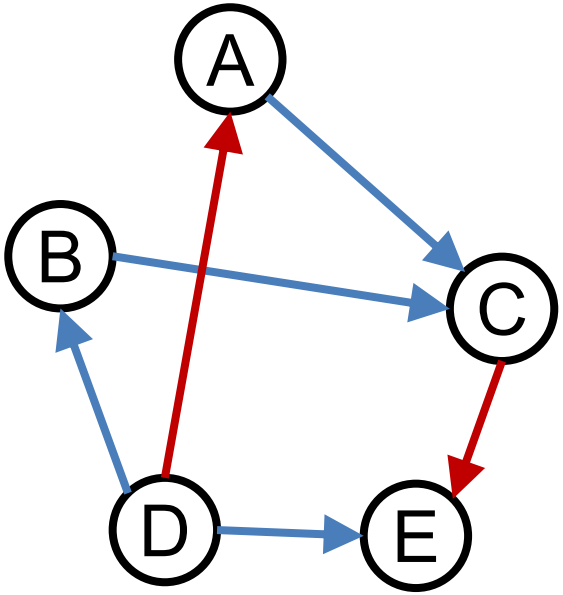
A multi-relation graph

Like word embeddings, graph embeddings are a form of **unsupervised learning** on graphs.

- **Task-agnostic** node representations
- Features are useful on downstream tasks without much data
- Nearest neighbors are semantically meaningful

(Credit: Adam Lerer)

Graph Embeddings



A multi-relation graph

Margin loss between the score for an edge $f(e)$ and a negative sampled edge $f(e')$

$$\mathcal{L} = \sum_{e \in G} \sum_{e' \in S'_e} \max(f(e) - f(e') + \lambda, 0))$$

The score for an edge is a similarity (e.g. dot product) between the source embedding and a transformed version of the destination embedding, e.g.

$$f(e) = \cos(\theta_s, \theta_r + \theta_d)$$

Negative samples are constructed by taking a real edge and replacing the source or destination with a random node.

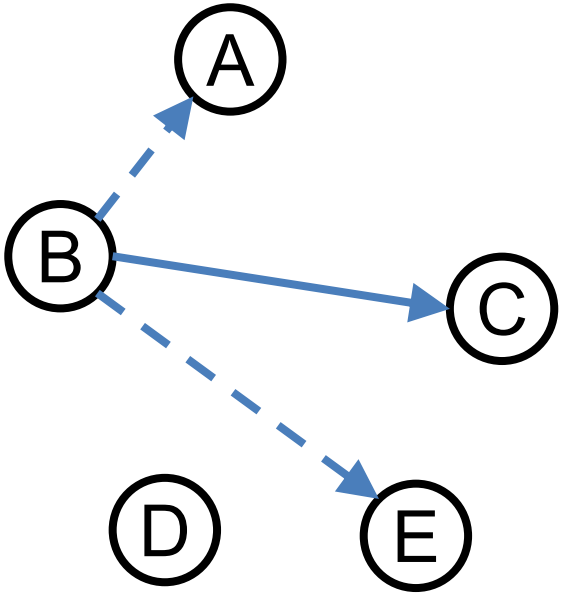
$$S'_e = \{(s', r, d) | s' \in V\} \cup \{(s, r, d' | d' \in V\}$$

Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., and Yakhnenko, O. **"Translating embeddings for modeling multi-relational data."** *NIPS, 2013*

Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., and Bouchard, G. "**Complex embeddings for simple link prediction."** *ICML, 2016*

(Credit: Adam Lerer)

Graph Embeddings



A multi-relation graph

Margin loss between the score for an edge $f(e)$ and a negative sampled edge $f(e')$

$$\mathcal{L} = \sum_{e \in G} \sum_{e' \in S'_e} \max(f(e) - f(e') + \lambda, 0))$$

The score for an edge is a similarity (e.g. dot product) between the source embedding and a transformed version of the destination embedding, e.g.

$$f(e) = \cos(\theta_s, \theta_r + \theta_d)$$

Negative samples are constructed by taking a real edge and replacing the source or destination with a random node.

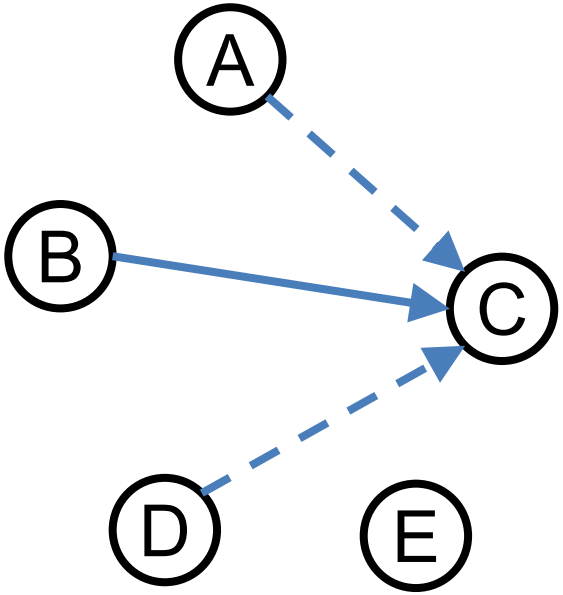
$$S'_e = \{(s', r, d) | s' \in V\} \cup \{(s, r, d' | d' \in V\}$$

Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., and Yakhnenko, O. **"Translating embeddings for modeling multi-relational data."** *NIPS, 2013*

Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., and Bouchard, G. "**Complex embeddings for simple link prediction."** *ICML, 2016*

(Credit: Adam Lerer)

Graph Embeddings



A multi-relation graph

Margin loss between the score for an edge $f(e)$ and a negative sampled edge $f(e')$

$$\mathcal{L} = \sum_{e \in G} \sum_{e' \in S'_e} \max(f(e) - f(e') + \lambda, 0))$$

The score for an edge is a similarity (e.g. dot product) between the source embedding and a transformed version of the destination embedding, e.g.

$$f(e) = \cos(\theta_s, \theta_r + \theta_d)$$

Negative samples are constructed by taking a real edge and replacing the source or destination with a random node.

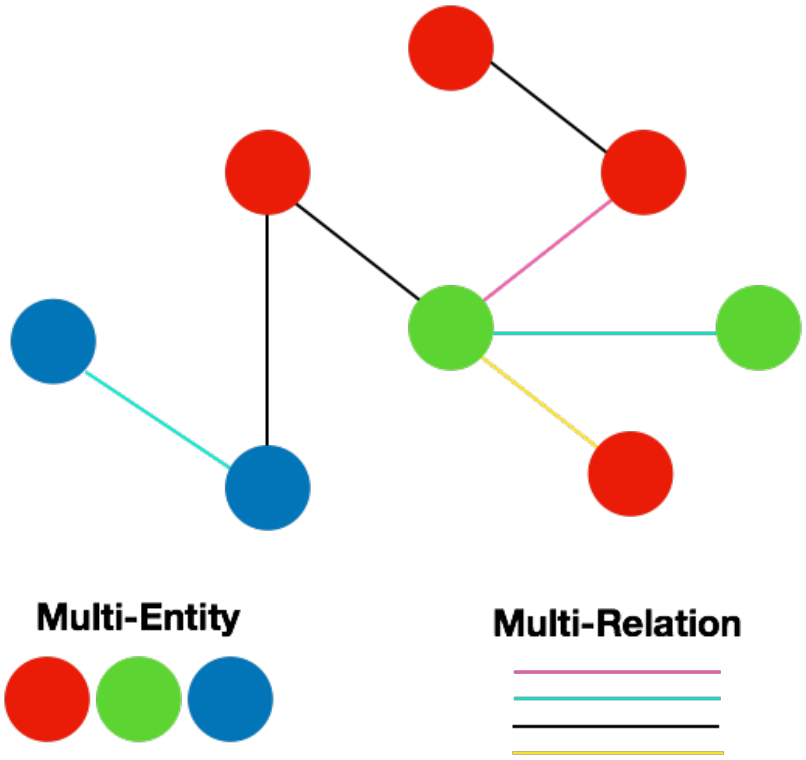
$$S'_e = \{(s', r, d) | s' \in V\} \cup \{(s, r, d' | d' \in V\}$$

Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., and Yakhnenko, O. **"Translating embeddings for modeling multi-relational data."** *NIPS, 2013*

Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., and Bouchard, G. "**Complex embeddings for simple link prediction."** *ICML, 2016*

(Credit: Adam Lerer)

Multiple Relations in Graphs



- Identity

$$g(x) = x$$

- Translation

$$g(x|\Delta) = x + \Delta$$

- Affine

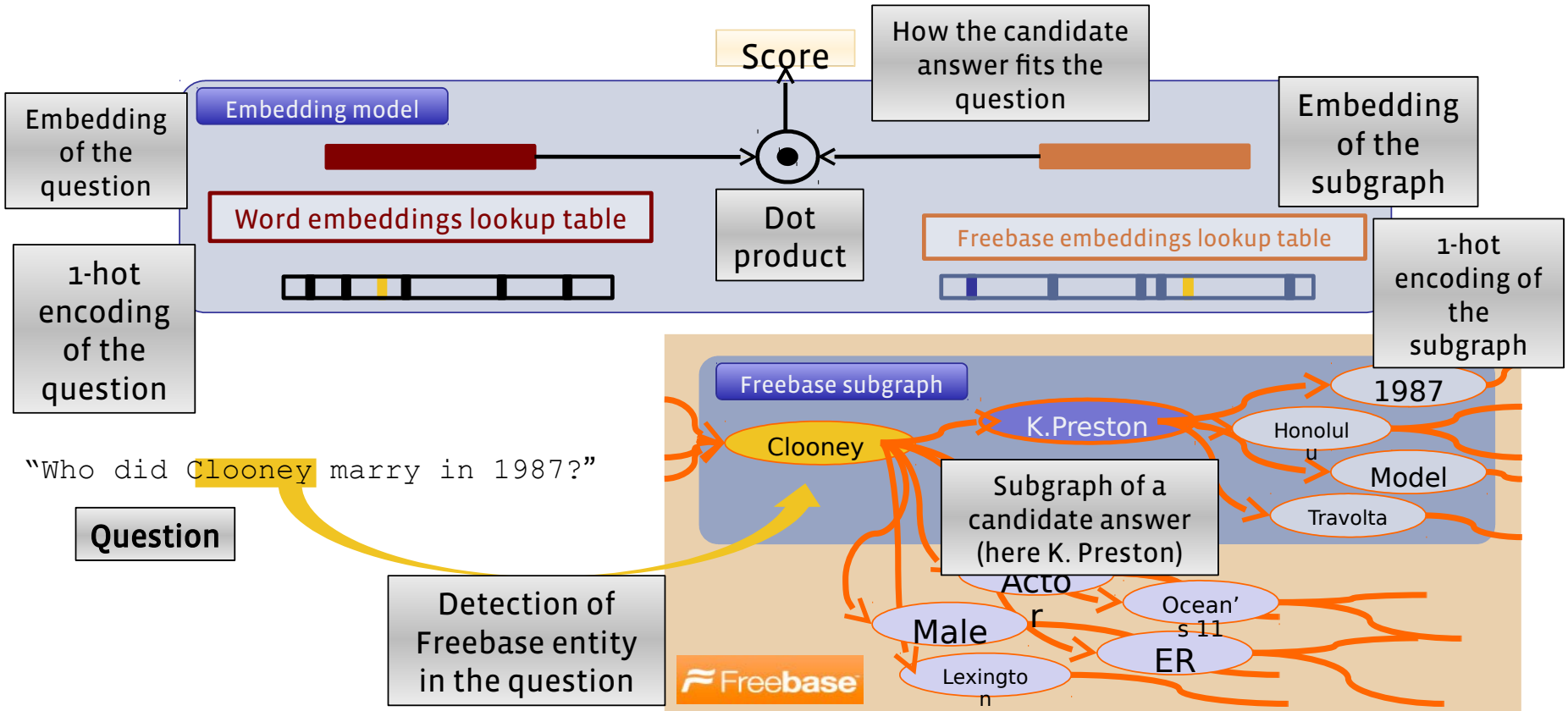
$$g(x|A, \Delta) = Ax + \Delta$$

- Diagonal

$$g(x|b) = b \odot x$$

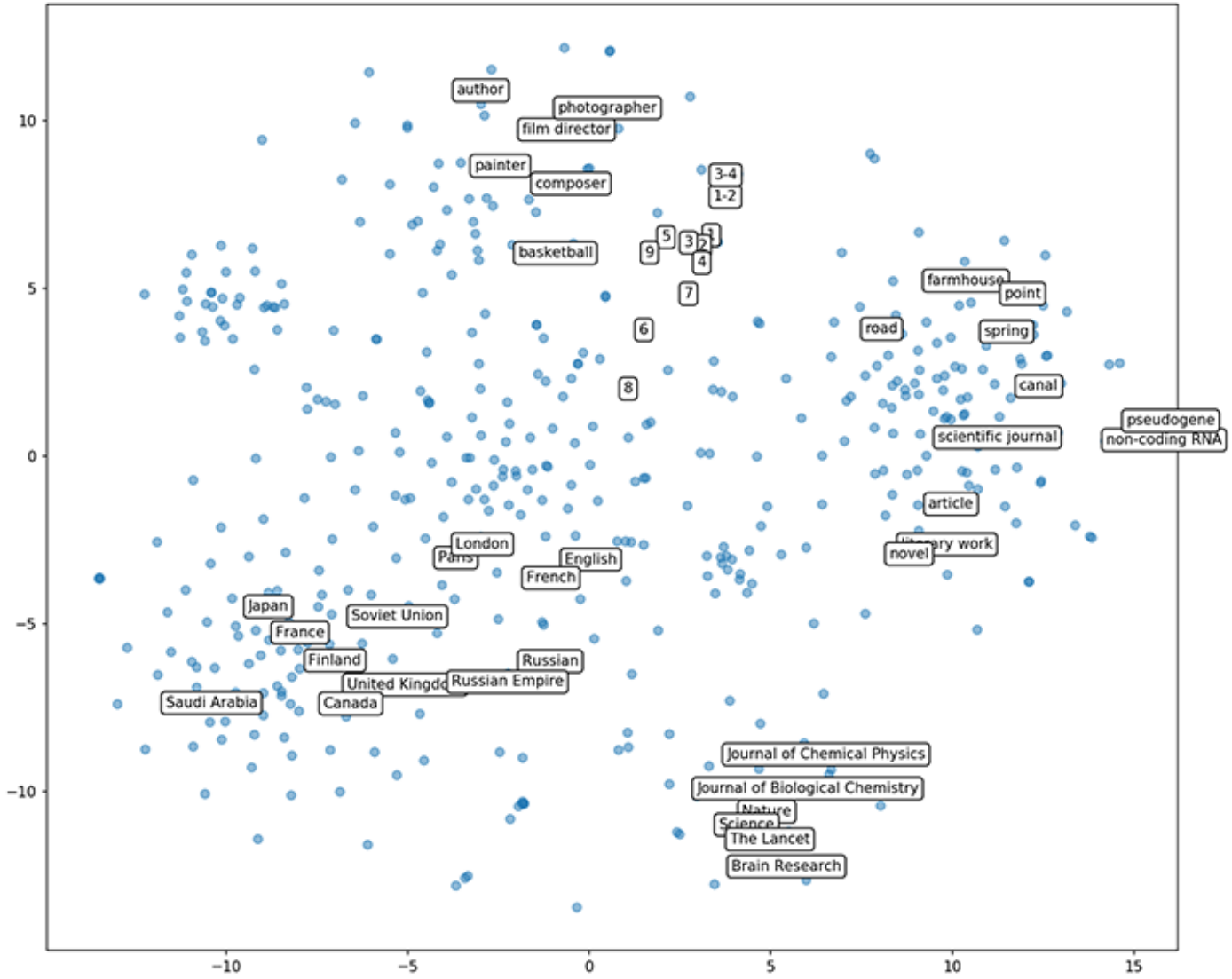
(Credit: Alex Peysakhovich , Ledell Wu)

Embedding a Knowledge Base [Bordes et al. 2013]



(Credit: Yann LeCun)

Embedding Wikidata Graph [Lerer et al. 2019]

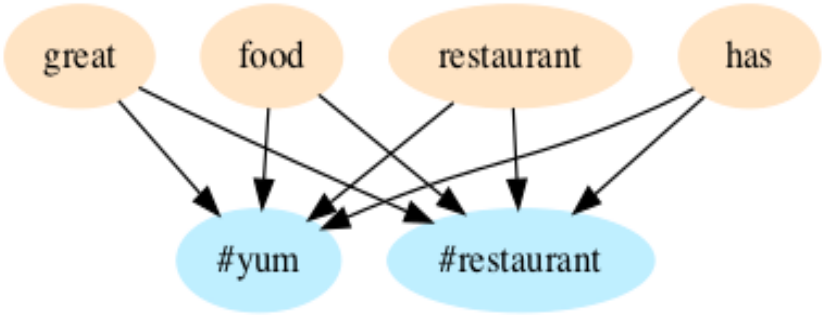


(Credit: Ledell Wu)

Outline

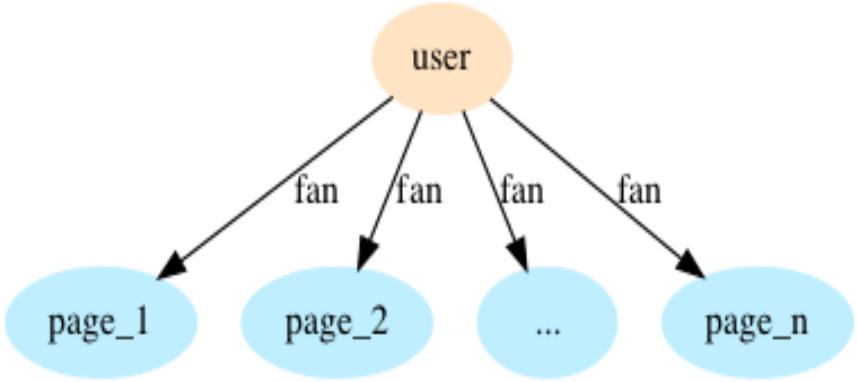
- Word Embeddings
- Graph Embeddings
- Applications
- Discussions

Application: TagSpace, PageSpace



Input: restaurant has great food
Label: #yum, #restaurant

Use-cases:
- Labeling posts
- Clustering of hashtags



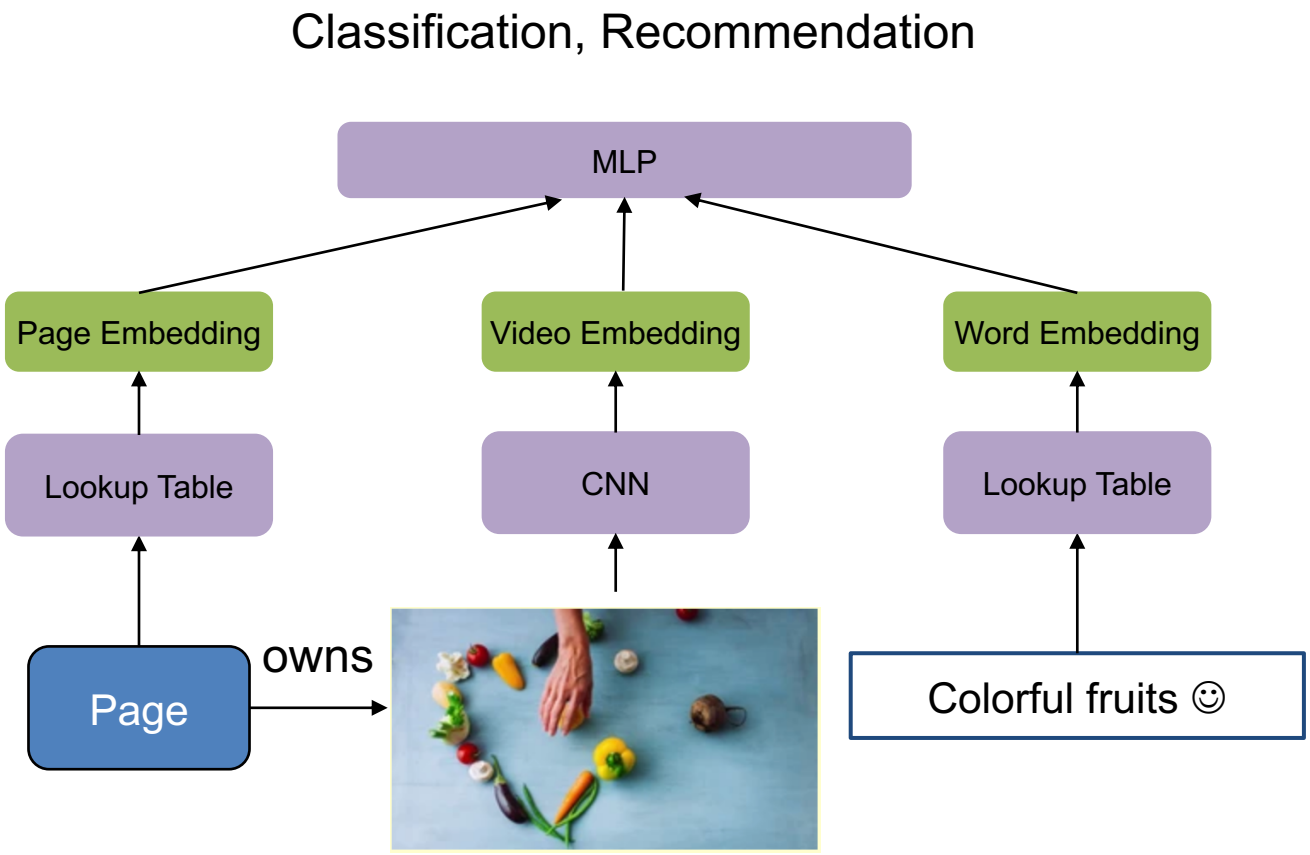
Input: (user, page) pairs

Use-cases:
- Clustering of pages
- Recommending pages to users

Reference: [Weston et al. 2014], [Wu et al. 2018]
https://github.com/facebookresearch/StarSpace

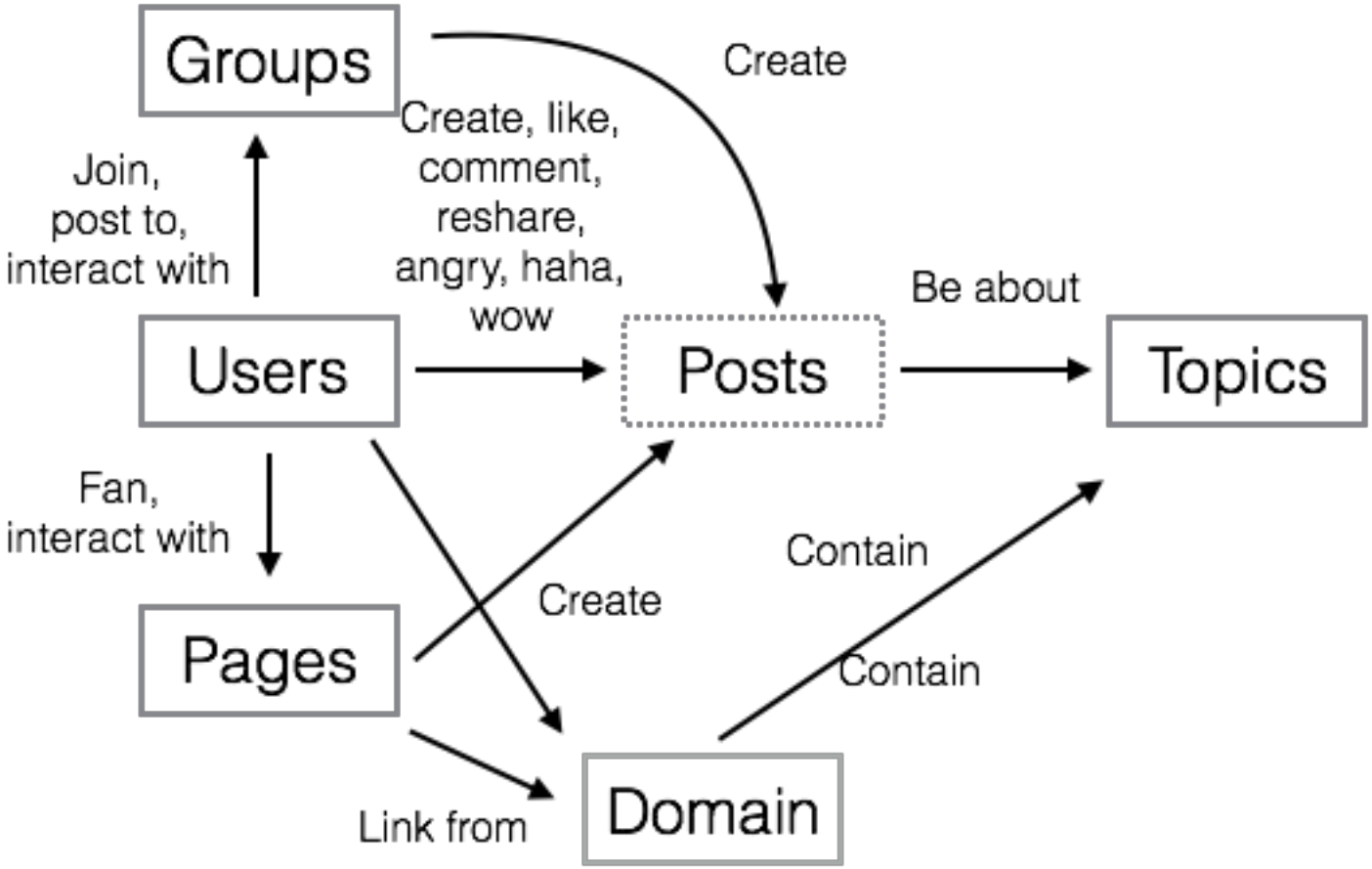
(Credit: Ledell Wu)

Application: VideoSpace



(Credit: Ledell Wu)

Application: world2vec

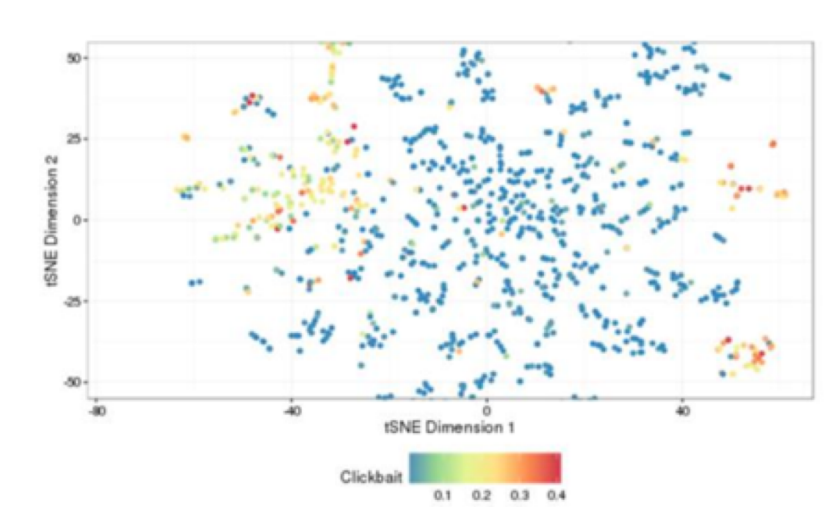


(Credit: Alex Peysakhovich)

General Purpose: Useful for other tasks

- Users
  - Bad Actor Cluster
- Groups
  - ‘For Sale’ Group prediction
- Pages
  - Recommendation
  - Page category prediction
  - Identify spam / hateful pages
- Domains
  - Domain type prediction
  - Identify mis-Information

Clickbait Model Prediction



(Credit: Alex Peysakhovich)

Outline

- Word Embeddings
- Graph Embeddings
- Applications
- Discussions

Take-away

- Word Embeddings: **word2vec**
- Graph Embeddings:
  - Map different entities (word, text, image, user, etc.) into vector space
  - Handle multiple relations
- Applications:
  - Similarity in vector space (search, recommendation)
  - Clustering (bad actor, etc.)
  - General feature representation

Discussions

- Word embeddings in NLP models
  - RNN, LSTM, Transformers
- Large-scale Graph embedding system
  - https://github.com/facebookresearch/PyTorch-BigGraph [Lerer et al. 19]
- Ethical Considerations
  - Amplification of existing discriminatory and unfair behaviors
  - “Man is to Computer Programmer as Woman is to Homemaker? Debiasing Word Embeddings” [Bolukbasi et al. 16]