TTIC 31210:
Advanced Natural Language Processing

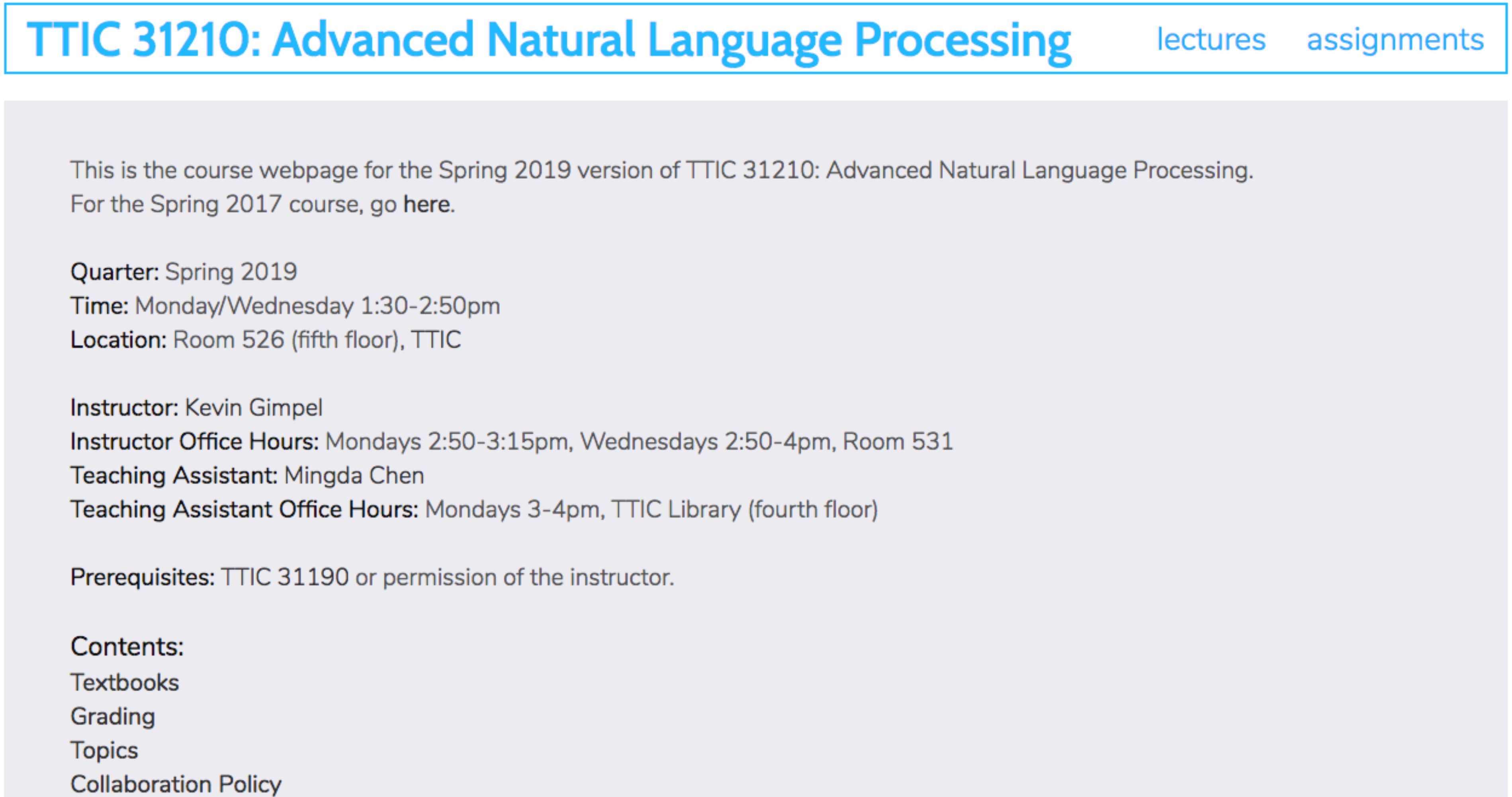
Kevin Gimpel

Spring 2019

Lecture 3:
Loss Functions and Word Embeddings

Course Web Page

https://ttic.uchicago.edu/~kgimpel/teaching/31210-s19/index.html



TTIC 31210: Advanced Natural Language Processing lectures assignments

This is the course webpage for the Spring 2019 version of TTIC 31210: Advanced Natural Language Processing.
For the Spring 2017 course, go **here**.

Quarter: Spring 2019
Time: Monday/Wednesday 1:30-2:50pm
Location: Room 526 (fifth floor), TTIC

Instructor: Kevin Gimpel
Instructor Office Hours: Mondays 2:50-3:15pm, Wednesdays 2:50-4pm, Room 531
Teaching Assistant: Mingda Chen
Teaching Assistant Office Hours: Mondays 3-4pm, TTIC Library (fourth floor)

Prerequisites: TTIC 31190 or permission of the instructor.

Contents:
Textbooks
Grading
Topics
Collaboration Policy

Assignment 1

- any questions?
- what if negative examples are actually positive examples?
  - you could check them and resample until they’re negative, or just ignore this
  - it shouldn’t make a big difference

- If you’re graduating this quarter, Assignment 5 is optional

Roadmap

- intro (1 lecture)
- **deep learning for NLP (5 lectures)**
- structured prediction: sequence labeling, syntactic and semantic parsing, dynamic programming (4 lectures)
- generative models, latent variables, unsupervised learning, variational autoencoders (2 lectures)
- Bayesian methods in NLP (2 lectures)
- Bayesian nonparametrics in NLP (2 lectures)
- review & other topics (1 lecture)

Today

- loss functions for similarity modeling and neural NLP in general
- review of word embeddings
- modeling subword structure in words

- last time, we talked about similarity functions and started talking about learning
- learning setting: we have pairs of structured objects that are assumed to be similar:

$$\langle \boldsymbol{x}_1, \boldsymbol{x}_2 \rangle$$

Learning for Similarity

- We want to learn input representation function $f_{\boldsymbol{\theta}}$ and all parameters of the similarity function (if any)
- We'll just write all these parameters as $\boldsymbol{\theta}$
- How about this loss?

$$\min_{\boldsymbol{\theta}} \sum_{\langle \boldsymbol{x}_1, \boldsymbol{x}_2 \rangle \in \mathcal{T}} -sim(f_{\boldsymbol{\theta}}(\boldsymbol{x}_1), f_{\boldsymbol{\theta}}(\boldsymbol{x}_2))$$

- Can lead to degenerate solutions for certain similarity functions

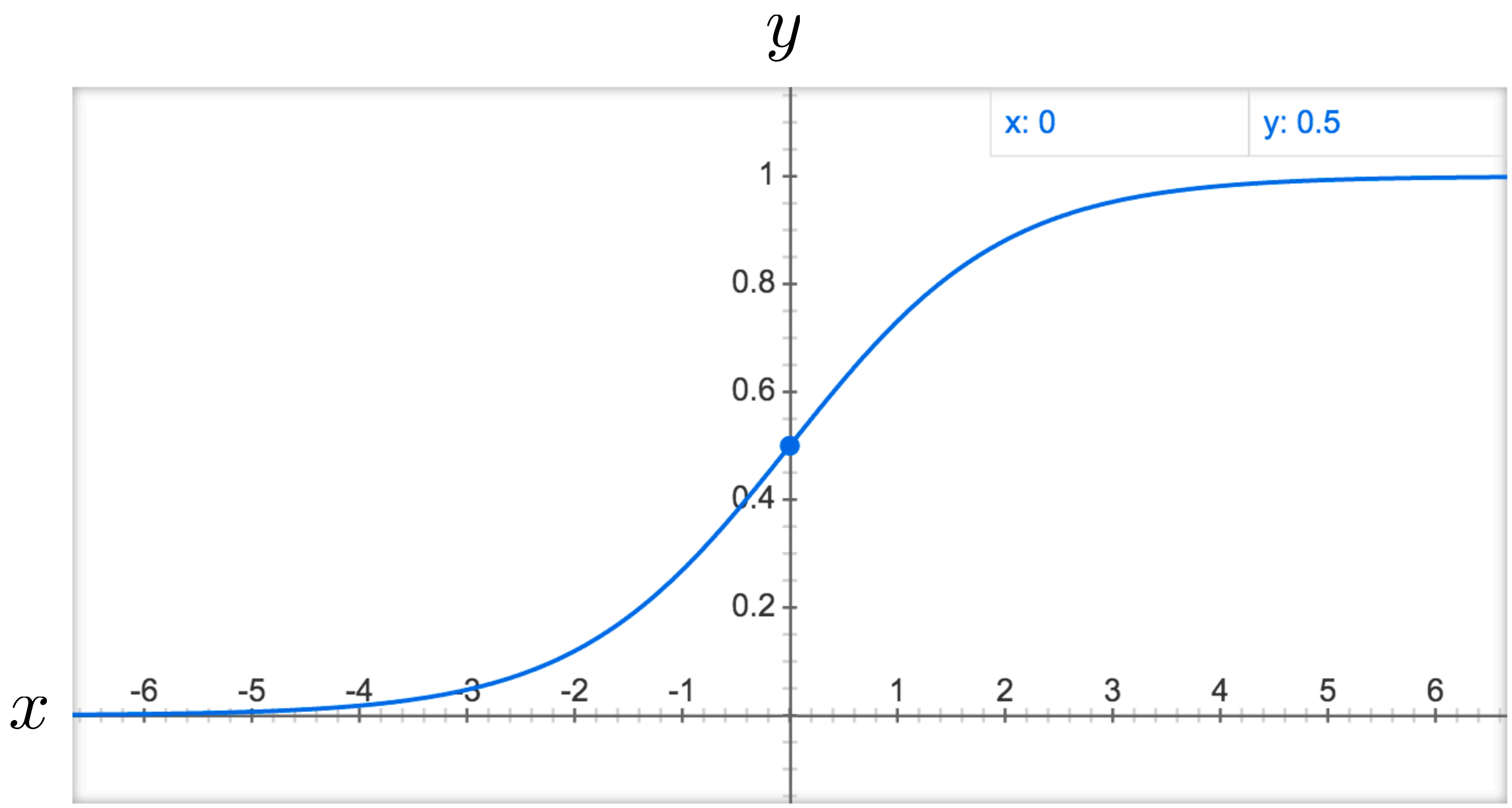
Binary Log Loss with Negative Sampling

- use a binary classifier based solely on similarity function:

$$\sigma(sim(f_{\boldsymbol{\theta}}(\boldsymbol{x}_1), f_{\boldsymbol{\theta}}(\boldsymbol{x}_2))$$

- σ is logistic sigmoid function

(logistic) sigmoid: $y = \frac{1}{1 + \exp\{-x\}}$



Binary Log Loss with Negative Sampling

- use a binary classifier based solely on similarity function:

$$\sigma(sim(f_{\boldsymbol{\theta}}(\boldsymbol{x}_1), f_{\boldsymbol{\theta}}(\boldsymbol{x}_2))$$

- logistic sigmoid outputs a number between 0 and 1
- we can interpret it as the probability that the pair is a true pair under a binary classifier
- this is a very common technique

Binary Log Loss with Negative Sampling

- generate negative samples and train using binary log loss:

$$\min_{\boldsymbol{\theta}} \sum_{\langle \boldsymbol{x}_1, \boldsymbol{x}_2 \rangle \in \mathcal{T}} \underbrace{-\log \sigma(sim(f_{\boldsymbol{\theta}}(\boldsymbol{x}_1), f_{\boldsymbol{\theta}}(\boldsymbol{x}_2))}_{\text{binary classification log loss for example with label “true”}} - \sum_{\boldsymbol{x} \in NEG} \underbrace{\log(1 - \sigma(sim(f_{\boldsymbol{\theta}}(\boldsymbol{x}_1), f_{\boldsymbol{\theta}}(\boldsymbol{x})))}_{\text{binary classification log loss for example with label “false”}}$$

- *NEG* is a set chosen by the practitioner and may be task-dependent

Binary Log Loss with Negative Sampling

- generate negative samples and train using binary log loss:

$$\min_{\boldsymbol{\theta}} \sum_{\langle \boldsymbol{x}_1, \boldsymbol{x}_2 \rangle \in \mathcal{T}} -\log \sigma(sim(f_{\boldsymbol{\theta}}(\boldsymbol{x}_1), f_{\boldsymbol{\theta}}(\boldsymbol{x}_2)) - \sum_{\boldsymbol{x} \in NEG} \log\left(1 - \sigma(sim(f_{\boldsymbol{\theta}}(\boldsymbol{x}_1), f_{\boldsymbol{\theta}}(\boldsymbol{x}))\right)$$

- this loss shows sampling negatives to replace $\boldsymbol{x}_2$, but we could include another sum for negatives that replace $\boldsymbol{x}_1$

Binary Log Loss with Negative Sampling

- generate negative samples and train using binary log loss:

$$\min_{\boldsymbol{\theta}} \sum_{\langle \boldsymbol{x}_1, \boldsymbol{x}_2 \rangle \in \mathcal{T}} -\log \sigma(sim(f_{\boldsymbol{\theta}}(\boldsymbol{x}_1), f_{\boldsymbol{\theta}}(\boldsymbol{x}_2)) - \sum_{\boldsymbol{x} \in NEG} \log\left(1 - \sigma(sim(f_{\boldsymbol{\theta}}(\boldsymbol{x}_1), f_{\boldsymbol{\theta}}(\boldsymbol{x})))\right)$$

- when implementing a similar loss in Assignment 1, you should divide the second term by the size of *NEG* so that it doesn't overwhelm the first term

Losses for Similarity Learning

- Contrastive hinge loss:

$$\min_{\boldsymbol{\theta}} \sum_{\langle \boldsymbol{x}_1, \boldsymbol{x}_2 \rangle \in \mathcal{T}} [-sim(f_{\boldsymbol{\theta}}(\boldsymbol{x}_1), f_{\boldsymbol{\theta}}(\boldsymbol{x}_2)) + sim(f_{\boldsymbol{\theta}}(\boldsymbol{x}_1), f_{\boldsymbol{\theta}}(\boldsymbol{v}))]_+$$

$$[a]_+ = \max(0, a)$$

- $\boldsymbol{v}$ is a negative example
- potential problems with this? **(Q1 on handout)**
  – global optimum achieved by making all embeddings the same (for certain similarity functions)

Losses for Similarity Learning

- **Large-margin** contrastive hinge loss:

$$\min_{\boldsymbol{\theta}} \sum_{\langle \boldsymbol{x}_1, \boldsymbol{x}_2 \rangle \in \mathcal{T}} [\Delta - sim(f_{\boldsymbol{\theta}}(\boldsymbol{x}_1), f_{\boldsymbol{\theta}}(\boldsymbol{x}_2)) + sim(f_{\boldsymbol{\theta}}(\boldsymbol{x}_1), f_{\boldsymbol{\theta}}(\boldsymbol{v}))]_+$$

$$[a]_+ = \max(0, a)$$

- Δ is the "margin"
- avoids degenerate solution
- contrastive hinge losses are widely used

Losses for Similarity Learning

- Large-margin contrastive hinge loss:

$$\min_{\boldsymbol{\theta}} \sum_{\langle \boldsymbol{x}_1, \boldsymbol{x}_2 \rangle \in \mathcal{T}} [\Delta - sim(f_{\boldsymbol{\theta}}(\boldsymbol{x}_1), f_{\boldsymbol{\theta}}(\boldsymbol{x}_2)) + sim(f_{\boldsymbol{\theta}}(\boldsymbol{x}_1), f_{\boldsymbol{\theta}}(\boldsymbol{v}))]_+$$

- How should we choose negative examples?

Losses for Similarity Learning

- Large-margin contrastive hinge loss:

$$\min_{\boldsymbol{\theta}} \sum_{\langle \boldsymbol{x}_1, \boldsymbol{x}_2 \rangle \in \mathcal{T}} [\Delta - sim(f_{\boldsymbol{\theta}}(\boldsymbol{x}_1), f_{\boldsymbol{\theta}}(\boldsymbol{x}_2)) + sim(f_{\boldsymbol{\theta}}(\boldsymbol{x}_1), f_{\boldsymbol{\theta}}(\boldsymbol{v}))]_+$$

- How should we choose negative examples?
  - random: just pick randomly from the data
  - max: $\boldsymbol{v} = \underset{s : \langle ., s \rangle \in \mathcal{T}, s \neq \boldsymbol{x}_2}{\operatorname{argmax}} sim(f_{\boldsymbol{\theta}}(\boldsymbol{x}_1), f_{\boldsymbol{\theta}}(\boldsymbol{s}))$
  - batch-max: perform max only over current mini-batch
  - many other ways depending on problem

Choosing Negative Examples

- max: $$\boldsymbol{v} = \operatorname*{argmax}_{s:\langle .,s\rangle \in \mathcal{T}, s \neq \boldsymbol{x}_2} sim(f_{\boldsymbol{\theta}}(\boldsymbol{x}_1), f_{\boldsymbol{\theta}}(\boldsymbol{s}))$$
- batch-max: perform max only over mini-batch

$$\boldsymbol{v}_b = \operatorname*{argmax}_{s:\langle .,s\rangle \in \text{batch}, s \neq \boldsymbol{x}_2} sim(f_{\boldsymbol{\theta}}(\boldsymbol{x}_1), f_{\boldsymbol{\theta}}(\boldsymbol{s}))$$

- intuition: choose something similar to $\boldsymbol{x}_1$ but not what it was paired with ($\boldsymbol{x}_2$)
- however, batch-max may work better than max, depending on the dataset. why? **(Q2 on handout)**

Summary so far

- since similarity modeling is not classification, log loss may not be appropriate
- we discussed two loss function families that are suitable for similarity modeling:
  - binary log loss
  - contrastive hinge loss
- both require choosing negative examples
  - but this does let us design the negative example selection scheme for the task

- binary log loss & contrastive hinge loss are not only useful for similarity modeling

- they're often used in classification tasks with large label spaces
  - when we want to avoid iterating over all labels during training

- for example, when learning word embeddings!

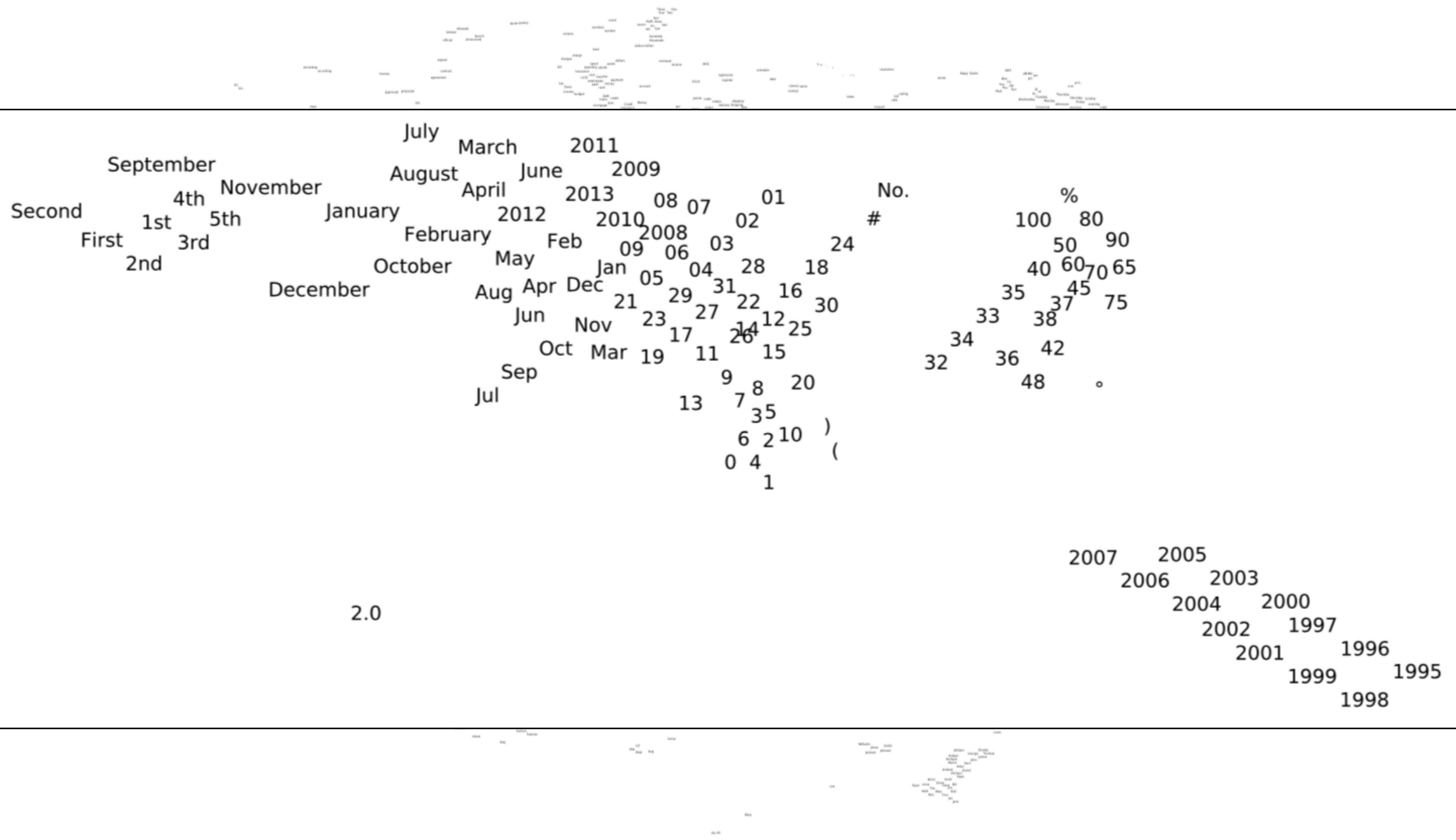
GloVe Word Embeddings

(Pennington et al., 2014)

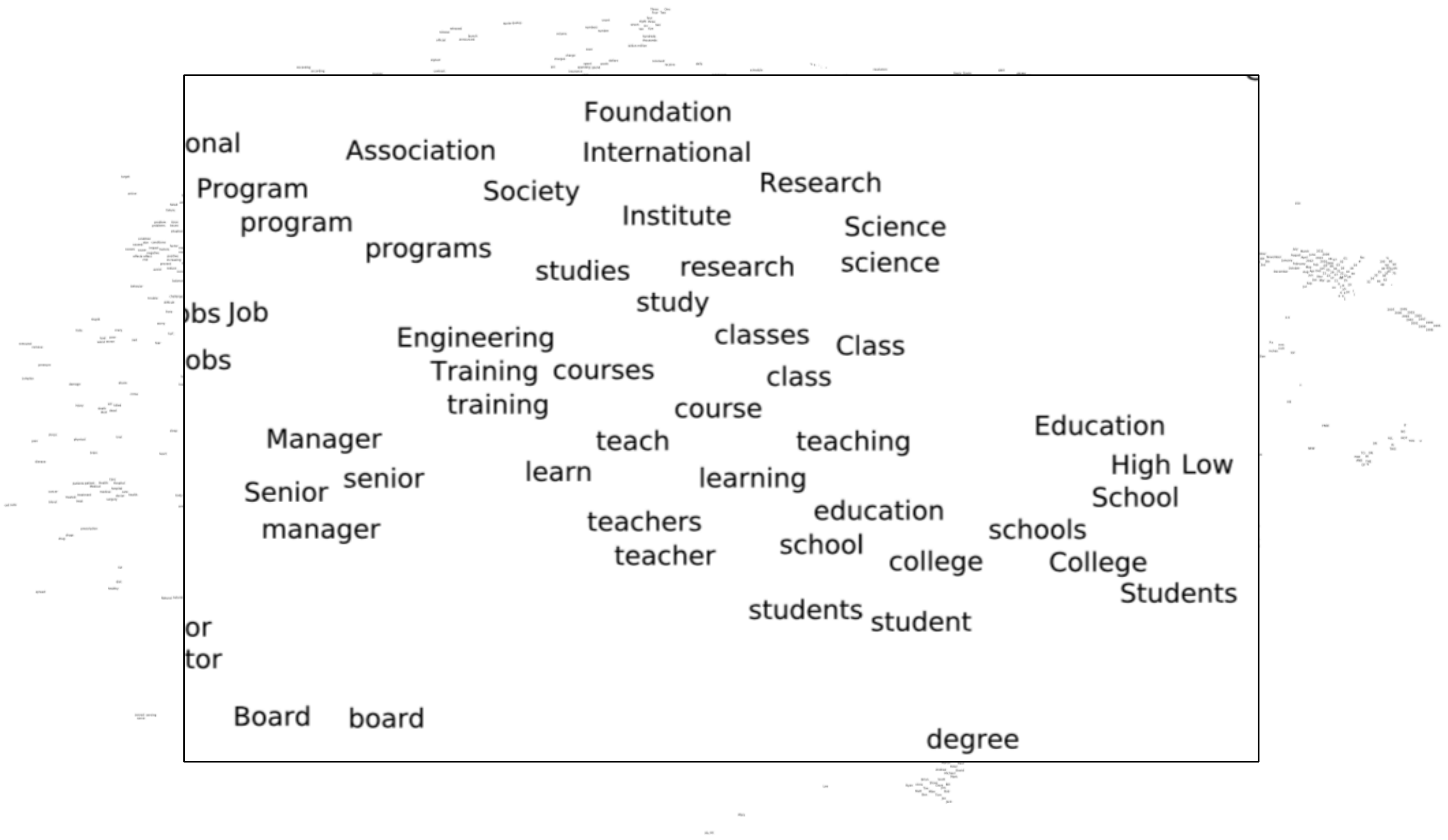
embeddings: 300-dim., pretrained on 840B tokens of Common Crawl

viz: ~3K words, t-SNE w/ cosine sim, adjustText (`github.com/Phlya/adjustText`)

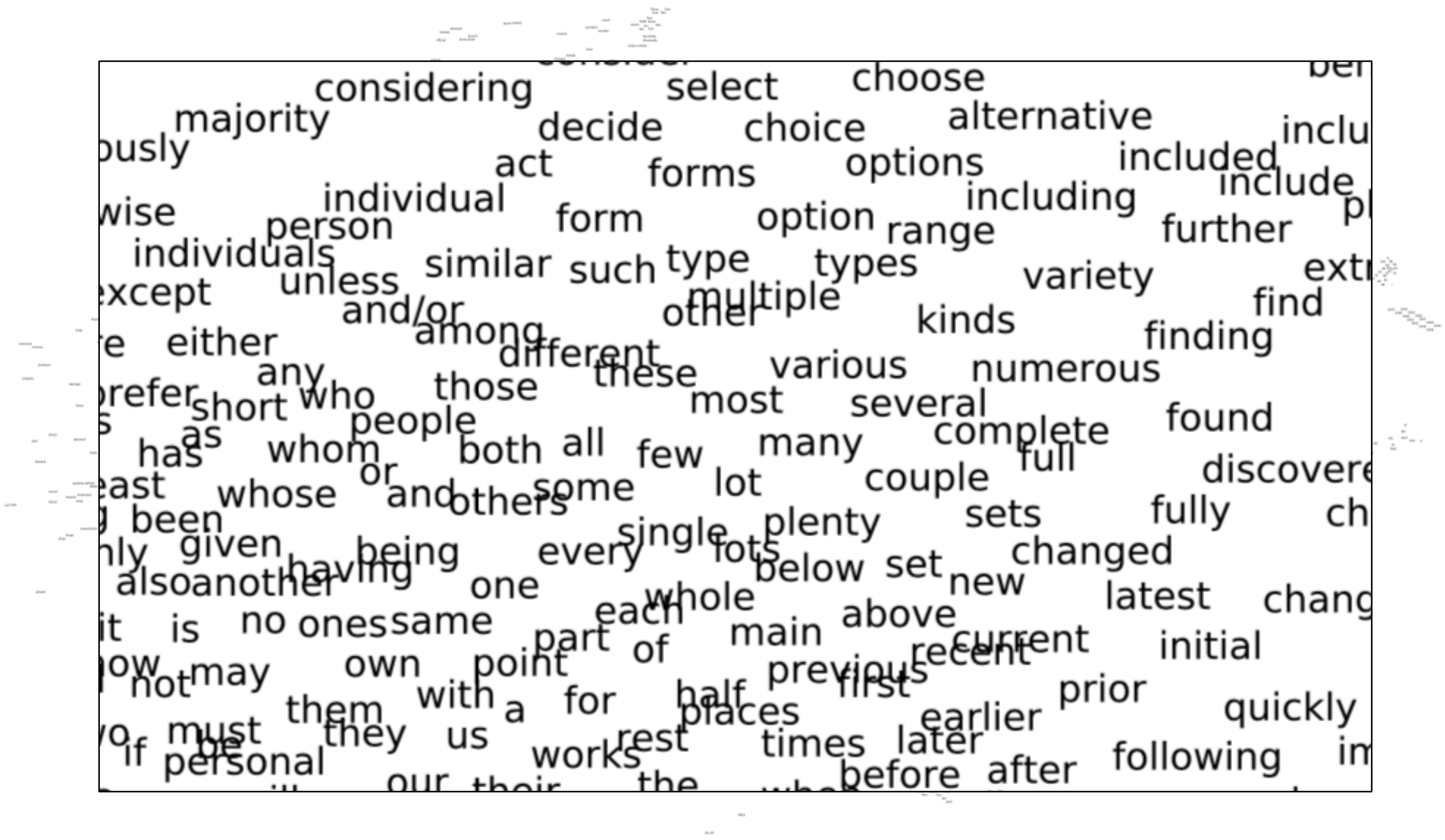
GloVe Word Embeddings



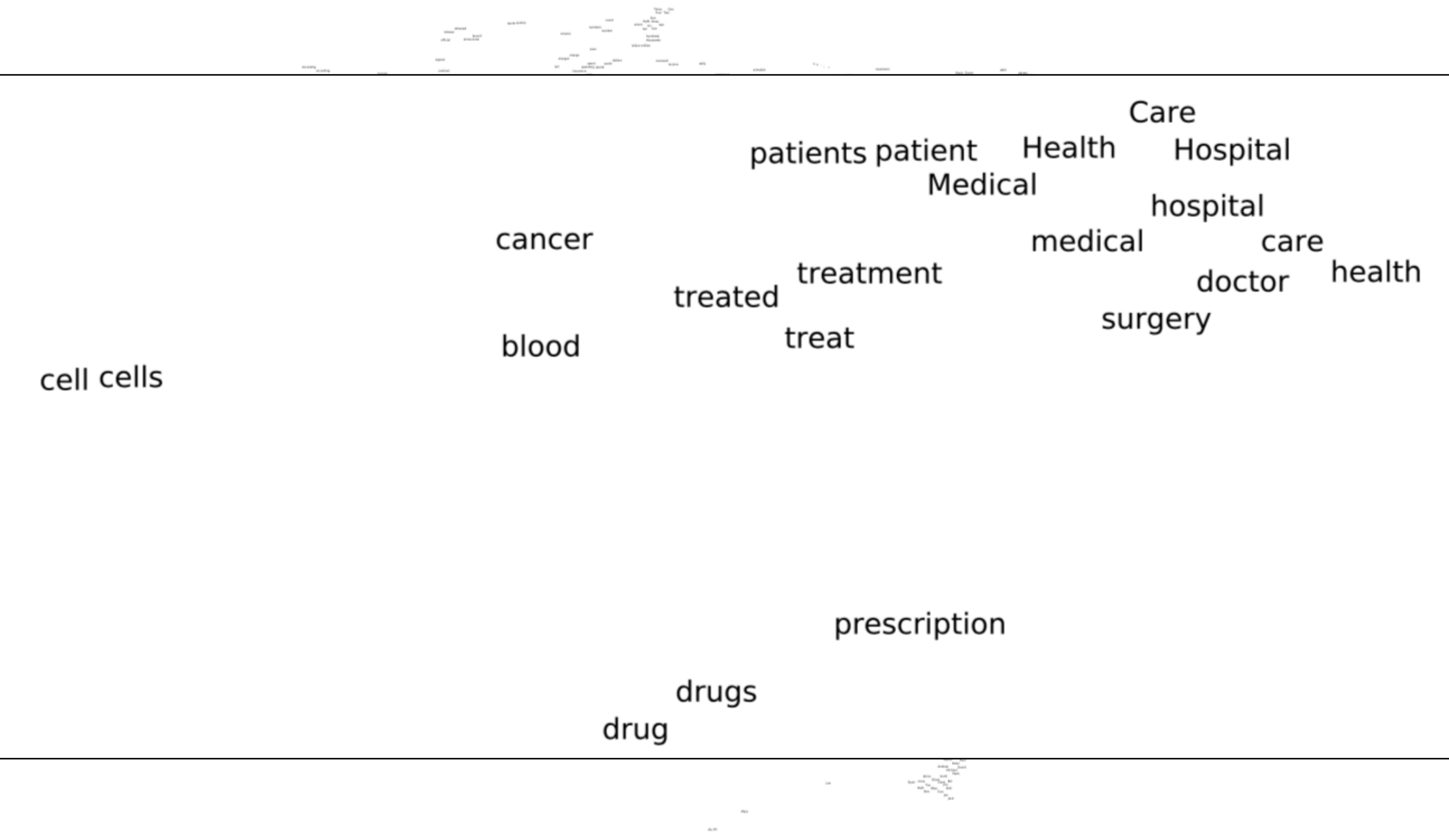
GloVe Word Embeddings



GloVe Word Embeddings



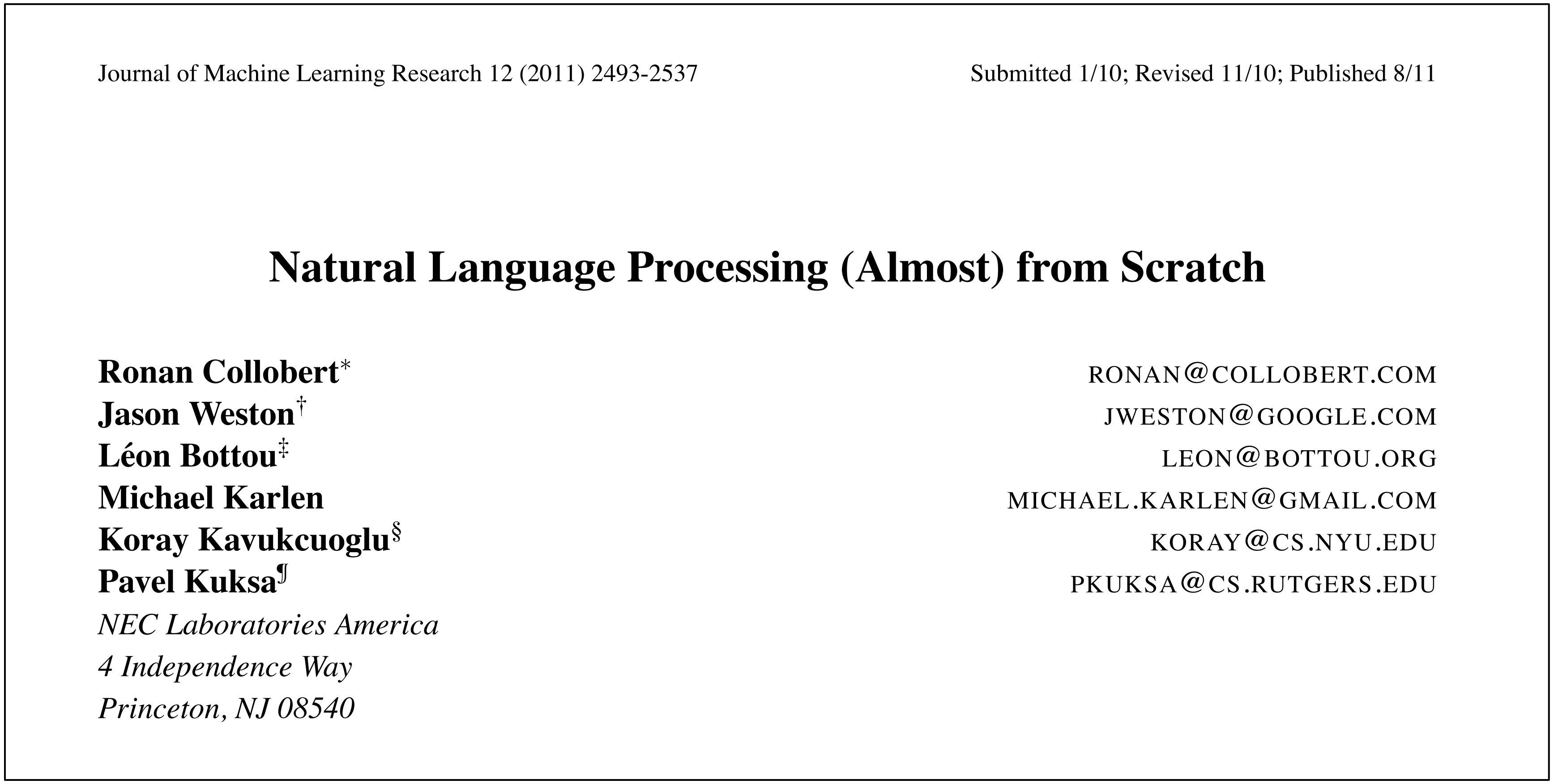
GloVe Word Embeddings



Word Embeddings

- many different kinds
- most use unsupervised or “self-supervised” learning (hold out some part of the data and predict it)
- learning signal typically based on context
  - words that appear in similar contexts should be similar in the embedding space
- since learning doesn’t use annotations, data is cheap, so training efficiency is important

Collobert et al. (2011)



Journal of Machine Learning Research 12 (2011) 2493-2537 Submitted 1/10; Revised 11/10; Published 8/11

Natural Language Processing (Almost) from Scratch

Ronan Collobert[*] RONAN@COLLOBERT.COM
Jason Weston[†] JWESTON@GOOGLE.COM
Léon Bottou[‡] LEON@BOTTOU.ORG
Michael Karlen MICHAEL.KARLEN@GMAIL.COM
Koray Kavukcuoglu[§] KORAY@CS.NYU.EDU
Pavel Kuksa[¶] PKUKSA@CS.RUTGERS.EDU
NEC Laboratories America
4 Independence Way
Princeton, NJ 08540

Collobert et al. Pairwise Ranking Loss

$$\min_{\boldsymbol{\theta}} \sum_{\langle x_1,\ldots,x_{11}\rangle \in \mathcal{T}} \sum_{w \in \mathcal{V}} [1 - f_{\boldsymbol{\theta}}(\langle x_1, \ldots, x_{11}\rangle) + f_{\boldsymbol{\theta}}(\langle x_1, \ldots, x_5, w, x_7, \ldots, x_{11}\rangle)]_+$$

$$[a]_+ = \max(0, a)$$

- $\mathcal{T}$ is training set of 11-word windows
- $\mathcal{V}$ is vocabulary
- what is this loss doing? **(Q3 on handout)**

Collobert et al. Pairwise Ranking Loss

$$\min_{\boldsymbol{\theta}} \sum_{\langle x_1,\ldots,x_{11}\rangle \in \mathcal{T}} \sum_{w \in \mathcal{V}} [1 - f_{\boldsymbol{\theta}}(\langle x_1, \ldots, x_{11}\rangle) + f_{\boldsymbol{\theta}}(\langle x_1, \ldots, x_5, w, x_7, \ldots, x_{11}\rangle)]_+$$

$$[a]_+ = \max(0, a)$$

- $\mathcal{T}$ is training set of 11-word windows
- $\mathcal{V}$ is vocabulary
- what is this loss doing?
  - make actual text window have higher score than all windows with center word replaced by *w*

Collobert et al. Pairwise Ranking Loss

$$\min_{\boldsymbol{\theta}} \sum_{\langle x_1, \ldots, x_{11} \rangle \in \mathcal{T}} \sum_{w \in \mathcal{V}} [1 - f_{\boldsymbol{\theta}}(\langle x_1, \ldots, x_{11} \rangle) + f_{\boldsymbol{\theta}}(\langle x_1, \ldots, x_5, w, x_7, \ldots, x_{11} \rangle)]_+$$

$$[a]_+ = \max(0, a)$$

- but this loss still has a sum over the entire vocabulary
- in practice, just sample a few words *w* from the vocabulary for each 11-word window

Collobert et al. (2011)

It is therefore desirable to define alternative training criteria. We propose here to use a *pairwise ranking* approach (Cohen et al., 1998). We seek a network that computes a higher score when given a legal phrase than when given an incorrect phrase.

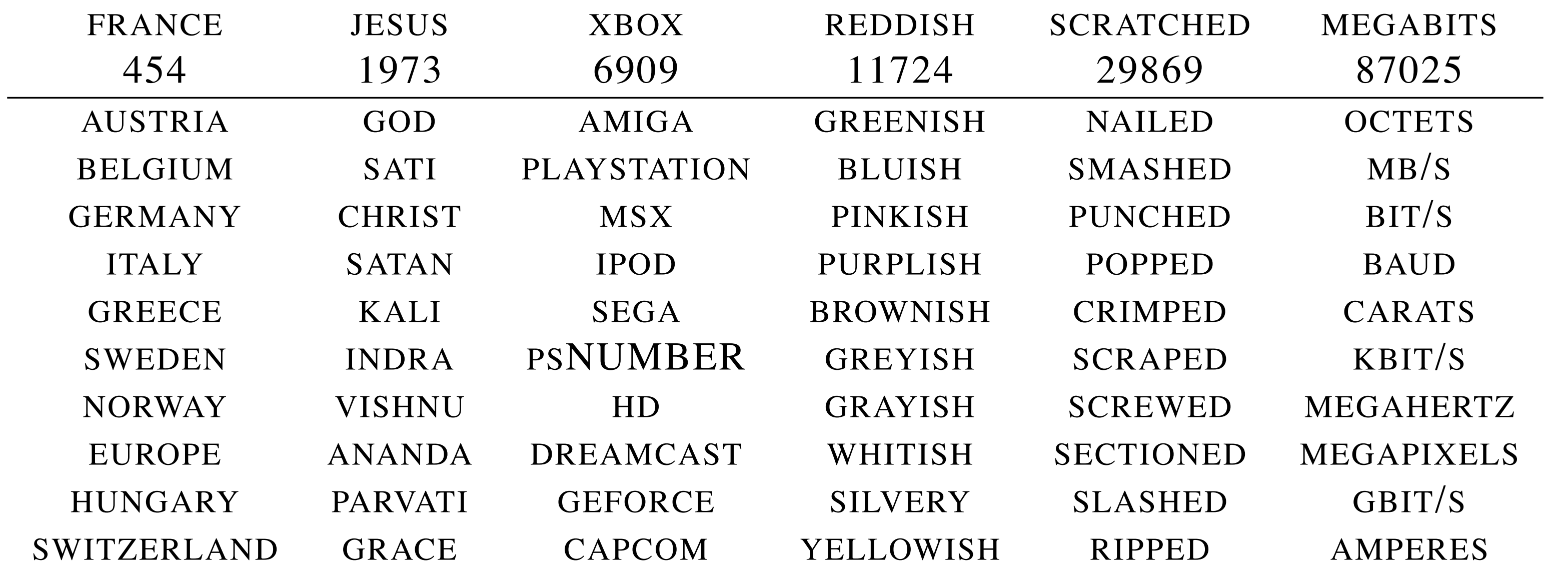
We consider a *window* approach network, as described in Section 3.3.1 and Figure 1, with parameters θ which outputs a score $f_\theta(x)$ given a window of text $x = [w]_1^{d_{win}}$. We minimize the ranking criterion with respect to θ:

$$\theta \mapsto \sum_{x \in \mathcal{X}} \sum_{w \in \mathcal{D}} \max\left\{ 0 \,,\, 1 - f_\theta(x) + f_\theta(x^{(w)}) \right\}, \tag{17}$$

where $\mathcal{X}$ is the set of all possible text windows with d_{win} words coming from our training corpus, $\mathcal{D}$ is the dictionary of words, and $x^{(w)}$ denotes the text window obtained by replacing the central word of text window $[w]_1^{d_{win}}$ by the word w.

Collobert et al. (2011)

- word embedding nearest neighbors:



FRANCE	JESUS	XBOX	REDDISH	SCRATCHED	MEGABITS
454	1973	6909	11724	29869	87025
AUSTRIA	GOD	AMIGA	GREENISH	NAILED	OCTETS
BELGIUM	SATI	PLAYSTATION	BLUISH	SMASHED	MB/S
GERMANY	CHRIST	MSX	PINKISH	PUNCHED	BIT/S
ITALY	SATAN	IPOD	PURPLISH	POPPED	BAUD
GREECE	KALI	SEGA	BROWNISH	CRIMPED	CARATS
SWEDEN	INDRA	PSNUMBER	GREYISH	SCRAPED	KBIT/S
NORWAY	VISHNU	HD	GRAYISH	SCREWED	MEGAHERTZ
EUROPE	ANANDA	DREAMCAST	WHITISH	SECTIONED	MEGAPIXELS
HUNGARY	PARVATI	GEFORCE	SILVERY	SLASHED	GBIT/S
SWITZERLAND	GRACE	CAPCOM	YELLOWISH	RIPPED	AMPERES

Table 7: Word embeddings in the word lookup table of the language model neural network LM1 trained with a dictionary of size 100,000. For each column the queried word is followed by its index in the dictionary (higher means more rare) and its 10 nearest neighbors (using the Euclidean metric, which was chosen arbitrarily).

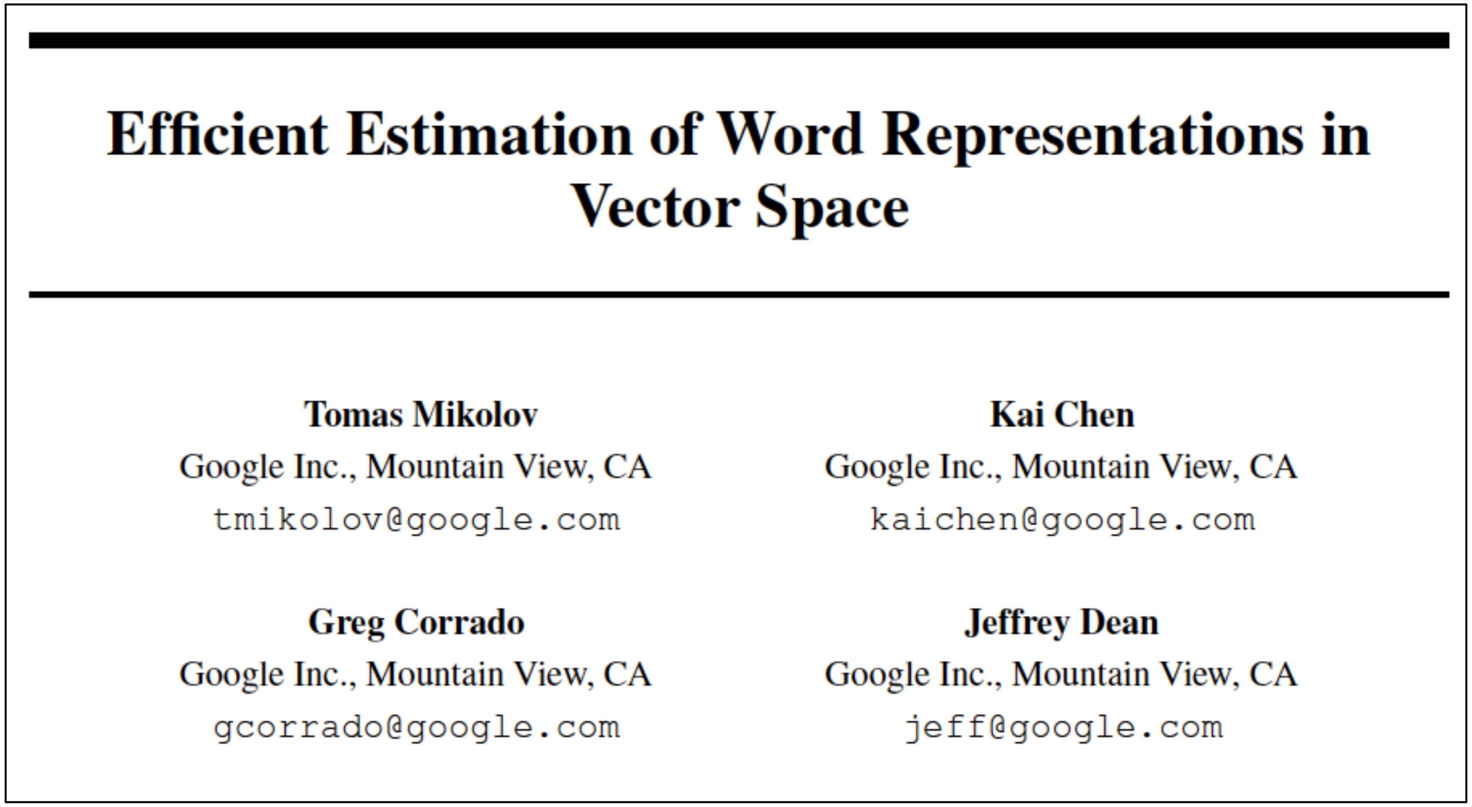
Nomenclature

- “contrastive loss” = connotes the use of a similarity function (I think)

- “pairwise ranking loss” = more general: naming is applicable to any setting with a scoring function (could be similarity)

- these names are somewhat in flux, so it’s best to look at the math in papers to make sure you know exactly what the loss is!

word2vec (Mikolov et al., 2013a)



Efficient Estimation of Word Representations in Vector Space

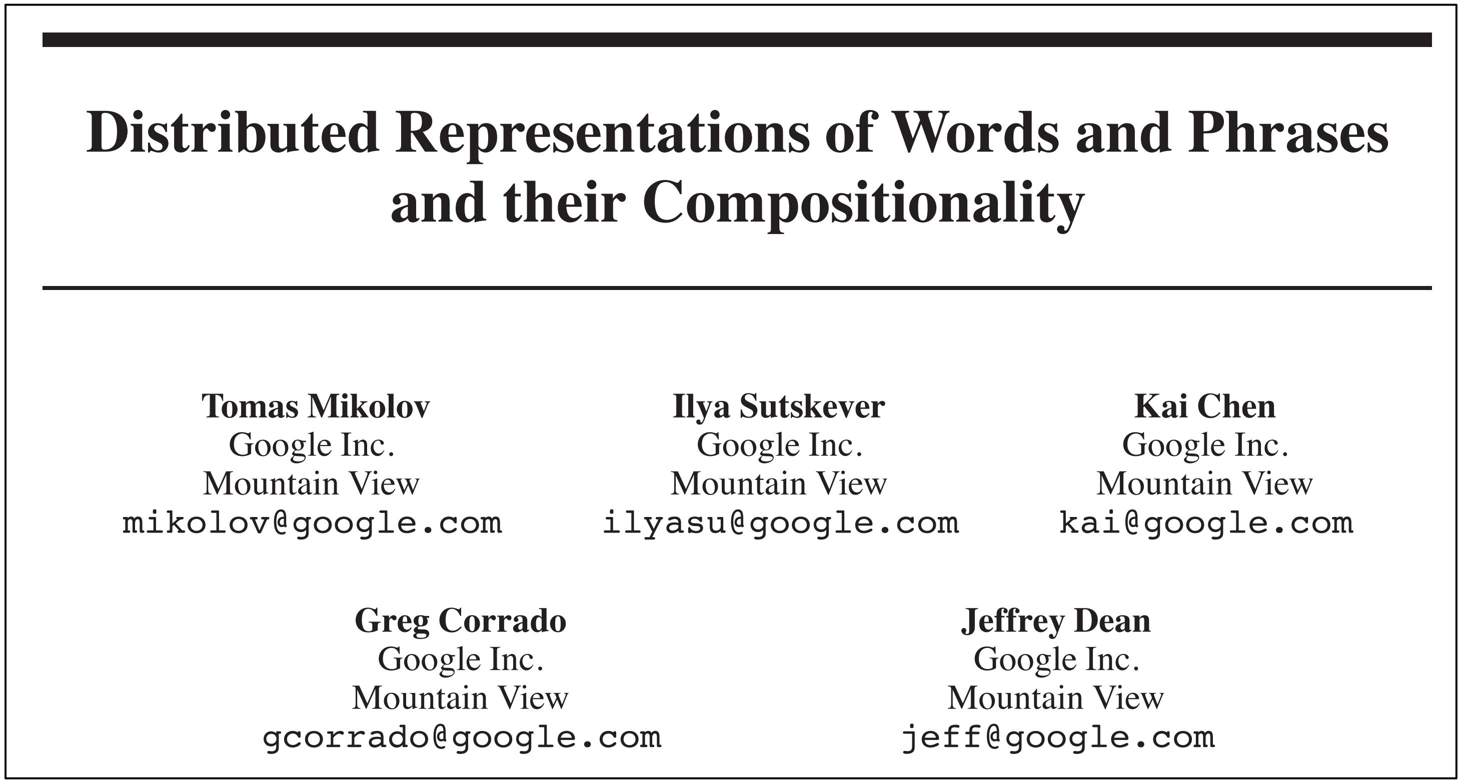
Tomas Mikolov
Google Inc., Mountain View, CA
tmikolov@google.com

Kai Chen
Google Inc., Mountain View, CA
kaichen@google.com

Greg Corrado
Google Inc., Mountain View, CA
gcorrado@google.com

Jeffrey Dean
Google Inc., Mountain View, CA
jeff@google.com

word2vec (Mikolov et al., 2013b)



Distributed Representations of Words and Phrases and their Compositionality

Tomas Mikolov
Google Inc.
Mountain View
mikolov@google.com

Ilya Sutskever
Google Inc.
Mountain View
ilyasu@google.com

Kai Chen
Google Inc.
Mountain View
kai@google.com

Greg Corrado
Google Inc.
Mountain View
gcorrado@google.com

Jeffrey Dean
Google Inc.
Mountain View
jeff@google.com

- word2vec contains two word embedding models:
  - continuous bag of words (CBOW)
  - skip-gram

- both use some part of the text to predict a held-out part

CBOW training data (window size = 5)

corpus (English Wikipedia):

agriculture is the traditional mainstay of the cambodian economy .

but benares has been destroyed by an earthquake .

…

inputs (*x*)	**outputs (*y*)**
{<s>, <s>, is, the}	agriculture
{<s>, agriculture, the, traditional}	is
{agriculture, is, traditional, mainstay}	the
{is, the, mainstay, of}	traditional
{the, traditional, of, the}	mainstay
{traditional, mainstay, the, cambodian}	of
{mainstay, of, cambodian, economy}	the
…	…

skip-gram training data (window size = 5)

corpus (English Wikipedia):

agriculture is the traditional mainstay of the cambodian economy .

but benares has been destroyed by an earthquake .

…

inputs (*x*)	**outputs (*y*)**
agriculture	<s>
agriculture	is
agriculture	the
is	<s>
is	agriculture
is	the
is	traditional
the	is
…	…

skip-gram

- skip-gram objective:

$$\min_{\boldsymbol{\theta}} \sum_{1 \le t \le |\mathcal{T}|} \sum_{-c \le j \le c, j \ne 0} -\log P_{\boldsymbol{\theta}}(x_{t+j} \mid x_t)$$

sum over positions in corpus

sum over context words in window (size = 2c + 1)

$$\min_{\boldsymbol{\theta}} \sum_{1 \leq t \leq |\mathcal{T}|} \sum_{-c \leq j \leq c, j \neq 0} - \log P_{\boldsymbol{\theta}}(x_{t+j} \mid x_t)$$

skip-gram model uses two different vector spaces:

$$P_{\boldsymbol{\theta}}(u \mid v) \propto \exp\{\underset{\substack{\nearrow \\ \text{“output” or} \\ \text{“outside” vector}}}{\boldsymbol{\phi}_u^\top} \underset{\substack{\nwarrow \\ \text{“input” or} \\ \text{“inside” vector}}}{\boldsymbol{\psi}_v}\}$$

$$\boldsymbol{\theta} = \langle \boldsymbol{\phi}, \boldsymbol{\psi} \rangle$$

$$\min_{\boldsymbol{\theta}} \sum_{1 \leq t \leq |\mathcal{T}|} \sum_{-c \leq j \leq c, j \neq 0} - \log P_{\boldsymbol{\theta}}(x_{t+j} \mid x_t)$$

skip-gram model uses two different vector spaces:

$$P_{\boldsymbol{\theta}}(u \mid v) \propto \exp\{\boldsymbol{\phi}_u^\top \boldsymbol{\psi}_v\}$$

$$\boldsymbol{\theta} = \langle \boldsymbol{\phi}, \boldsymbol{\psi} \rangle$$

which should we use as our word embeddings?

$$\min_{\boldsymbol{\theta}} \sum_{1 \leq t \leq |\mathcal{T}|} \sum_{-c \leq j \leq c, j \neq 0} - \log P_{\boldsymbol{\theta}}(x_{t+j} \mid x_t)$$

normalization requires sum over what?

$$P_{\boldsymbol{\theta}}(u \mid v) \propto \exp\{\boldsymbol{\phi}_u^\top \boldsymbol{\psi}_v\}$$

$$\min_{\boldsymbol{\theta}} \sum_{1 \leq t \leq |\mathcal{T}|} \sum_{-c \leq j \leq c, j \neq 0} - \log P_{\boldsymbol{\theta}}(x_{t+j} \mid x_t)$$

normalization requires sum over entire vocabulary:

$$P_{\boldsymbol{\theta}}(u \mid v) = \frac{\exp\{\boldsymbol{\phi}_u^\top \boldsymbol{\psi}_v\}}{\sum_{w \in \mathcal{V}} \exp\{\boldsymbol{\phi}_w^\top \boldsymbol{\psi}_v\}}$$

Binary Log Loss with Negative Sampling
(Mikolov et al., 2013)

- like what we saw earlier with similarity modeling, now with skip-gram scoring function:

$$\min_{\boldsymbol{\theta}} \sum_{1 \leq t \leq |\mathcal{T}|} \sum_{-c \leq j \leq c, j \neq 0} - \log \sigma(\boldsymbol{\phi}_{x_{t+j}}^{\top} \boldsymbol{\psi}_{x_t}) - \sum_{x \in \mathrm{NEG}} \log \left(1 - \sigma(\boldsymbol{\phi}_{x}^{\top} \boldsymbol{\psi}_{x_t})\right)$$

Binary Log Loss with Negative Sampling
(Mikolov et al., 2013)

- like what we saw earlier with similarity modeling, now with skip-gram scoring function:

$$\min_{\boldsymbol{\theta}} \sum_{1 \leq t \leq |\mathcal{T}|} \sum_{-c \leq j \leq c, j \neq 0} -\log \sigma(\boldsymbol{\phi}_{x_{t+j}}^\top \boldsymbol{\psi}_{x_t}) - \sum_{x \in \text{NEG}} \log \left(1 - \sigma(\boldsymbol{\phi}_x^\top \boldsymbol{\psi}_{x_t})\right)$$

Note: this is not actually what the word2vec code does. The code samples a window size *d* (up to the max window size *c*), then samples a *j* value between –*d* and *d* (such that *j* does not equal 0)

How should we define *NEG*?

- for learning word embeddings, Mikolov et al. (2013) defined *NEG* to be 2-20 words sampled from some distribution
  - e.g., uniform, unigram, or flattened unigram
  - flattened: raise probabilities to power ¾, renormalize to get a distribution
  - why do you think flattening works better than uniform or unigram? **(Q4 on handout)**

Break

Hierarchical Softmax
(Morin and Bengio, 2005)

original skip-gram optimization problem:

$$\min_{\boldsymbol{\theta}} \sum_{1 \leq t \leq |\mathcal{T}|} \sum_{-c \leq j \leq c, j \neq 0} -\log P_{\boldsymbol{\theta}}(x_{t+j} \mid x_t)$$

normalization requires sum over entire vocabulary:

$$P_{\boldsymbol{\theta}}(u \mid v) = \frac{\exp\{\boldsymbol{\phi}_u^\top \boldsymbol{\psi}_v\}}{\sum_{w \in \mathcal{V}} \exp\{\boldsymbol{\phi}_w^\top \boldsymbol{\psi}_v\}}$$

Hierarchical Softmax
(Morin and Bengio, 2005)

- based on a new generative story for $P_{\boldsymbol{\theta}}(u \mid v)$
- but the generative story is so simple!
  - just draw from the conditional distribution
- how can we make it more efficient?

Hierarchical Softmax
(Morin and Bengio, 2005)

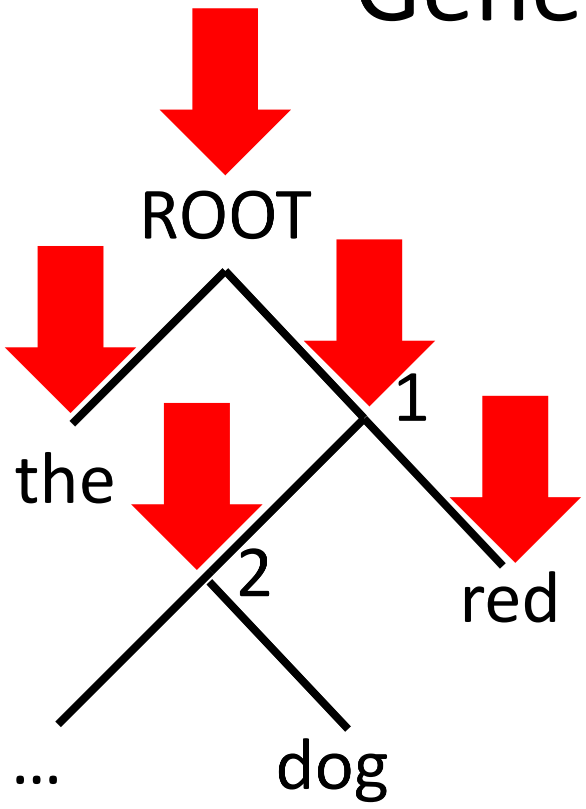
- based on a new generative story for $P_{\boldsymbol{\theta}}(u \mid v)$
- but the generative story is so simple!
  - just draw from the conditional distribution
- how can we make it more efficient?
- we still need it to be true that:

$$\sum_{u \in \mathcal{V}} P_{\boldsymbol{\theta}}(u \mid v) = 1$$

Hierarchical Softmax in word2vec

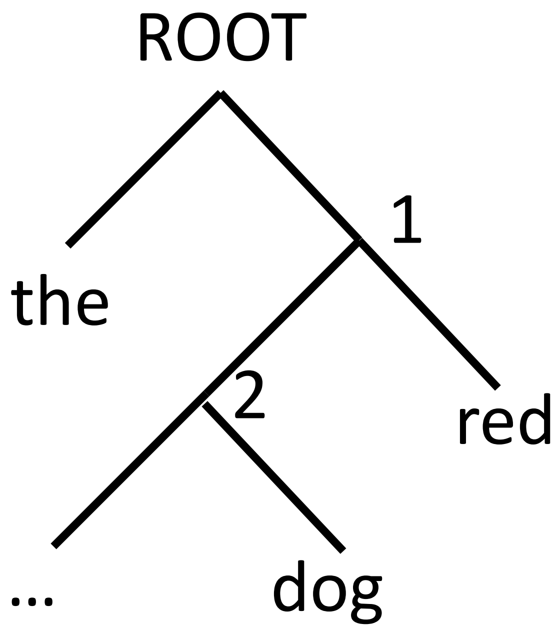
- new generative story for $P_{\boldsymbol{\theta}}(u \mid v)$
  - a random walk through the vocabulary biased by *v*
- idea:
  - build binary tree to represent vocabulary
  - to generate a context word of center word *v*, start at the root and keep flipping biased coins (biased according to *v*) to choose left or right
  - stop on reaching a leaf
- parameters of this generative model are vectors for individual split points in the tree, and input vector of *v*

Generative Story for $P_{\boldsymbol{\theta}}(u \mid v)$



```
flip a coin with Pr(heads) = $\sigma(\boldsymbol{\phi}_{\mathrm{ROOT}}^{\top}\boldsymbol{\psi}_v)$
if tails,
    output “the” as *u*
if heads,
    flip coin w/ Pr(heads) = $\sigma(\boldsymbol{\phi}_{\mathrm{split1}}^{\top}\boldsymbol{\psi}_v)$
if heads,
    output “red” as *u*
if tails,
    flip coin w/ Pr(heads) = $\sigma(\boldsymbol{\phi}_{\mathrm{split2}}^{\top}\boldsymbol{\psi}_v)$
…
```

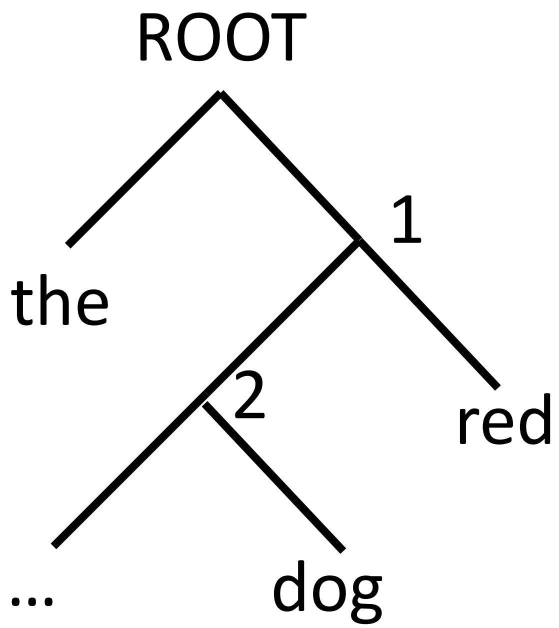
Generative Story for $P_{\boldsymbol{\theta}}(u \mid v)$



```
flip a coin with Pr(heads) = $\sigma(\boldsymbol{\phi}_{\mathrm{ROOT}}^\top \boldsymbol{\psi}_v)$
if tails,
    output “the” as $u$
if heads,
    flip coin w/ Pr(heads) = $\sigma(\boldsymbol{\phi}_{\mathrm{split1}}^\top \boldsymbol{\psi}_v)$
if heads,
    output “red” as $u$
```

What is $P_{\boldsymbol{\theta}}(\mathrm{dog} \mid \mathrm{cat})$? **(Q5 on handout)**

Generative Story for $P_{\boldsymbol{\theta}}(u \mid v)$



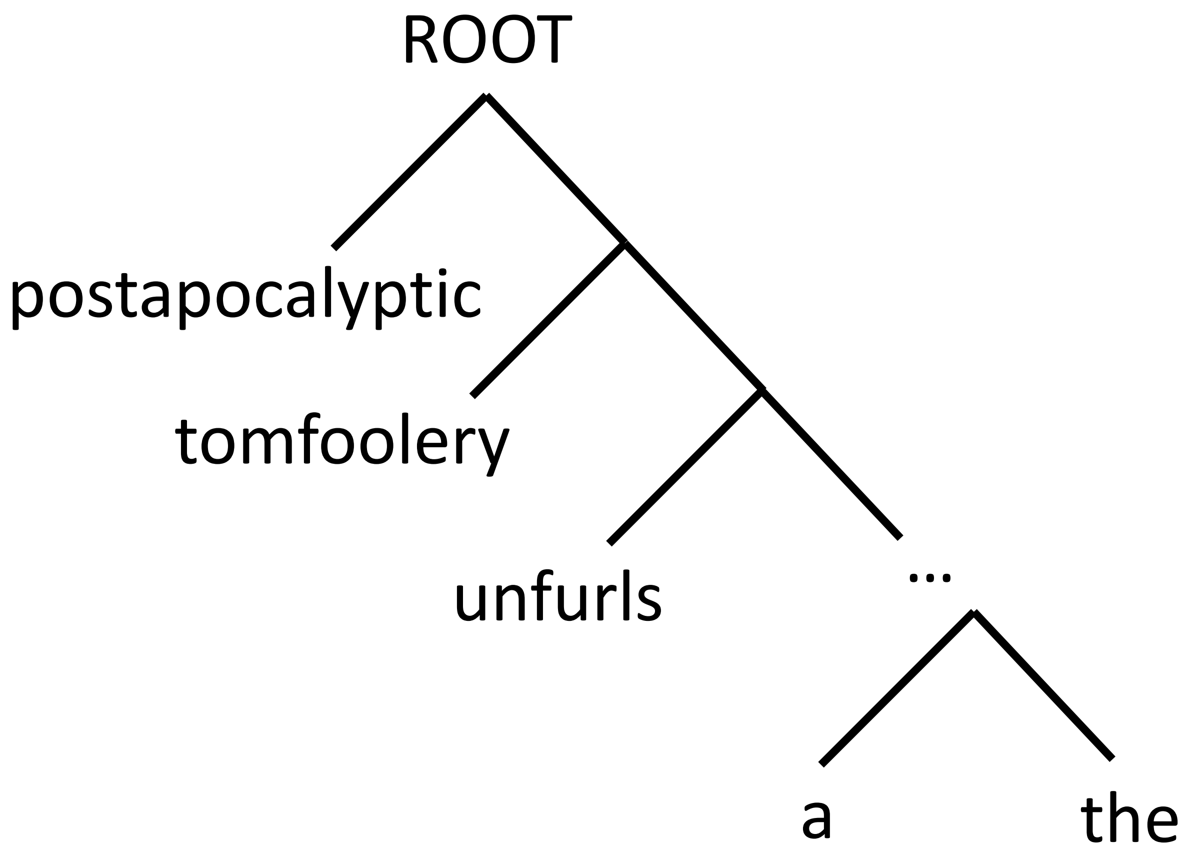
```
flip a coin with Pr(heads) = $\sigma(\boldsymbol{\phi}_{\mathrm{ROOT}}^{\top}\boldsymbol{\psi}_v)$
if tails,
    output "the" as $u$
if heads,
    flip coin w/ Pr(heads) = $\sigma(\boldsymbol{\phi}_{\mathrm{split1}}^{\top}\boldsymbol{\psi}_v)$
if heads,
    output "red" as $u$
```

Can you prove that $\sum_{u \in \mathcal{V}} P_{\boldsymbol{\theta}}(u \mid v) = 1$?

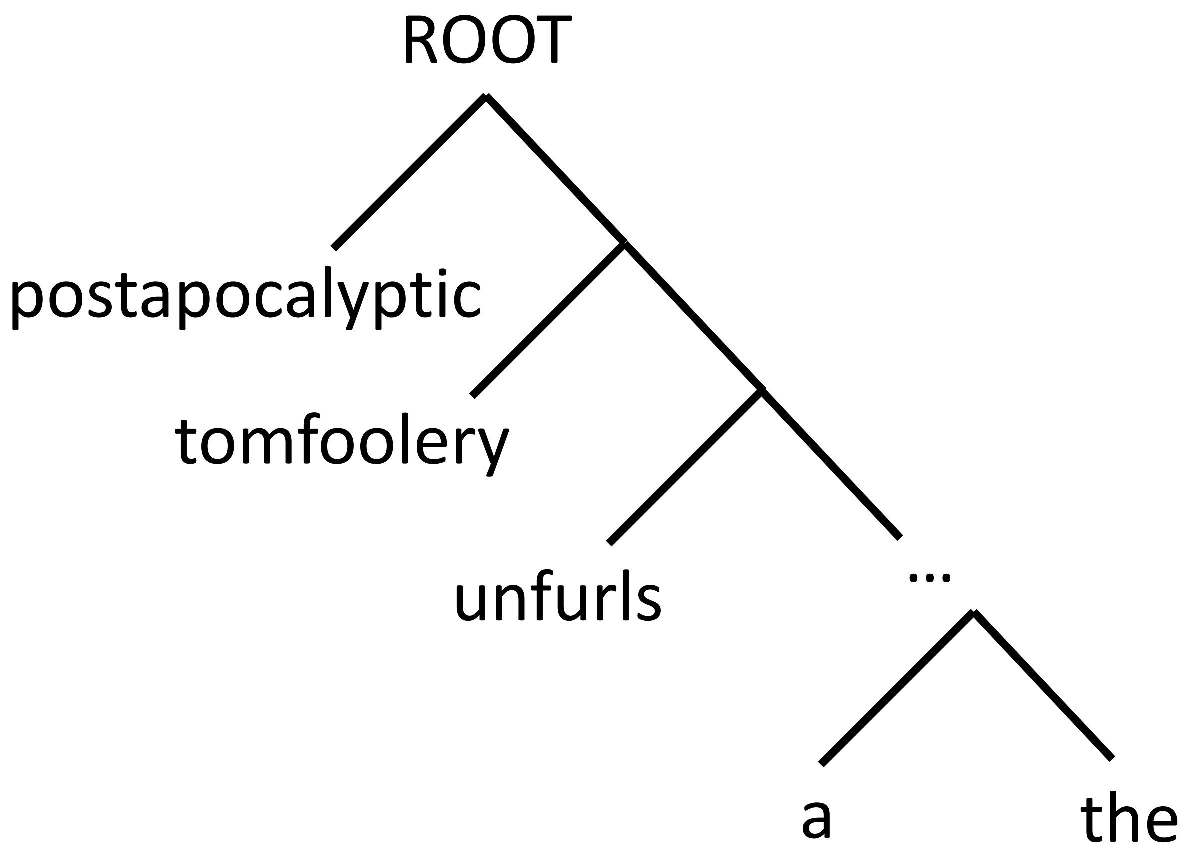
Hierarchical Softmax for Skip-Gram (Mikolov et al., 2013)

- each word has a unique path from ROOT
- rather than learn output vectors for all words, learn output vectors only for internal nodes of the binary tree
- how should we arrange the words into a binary tree?

- How about this binary tree?



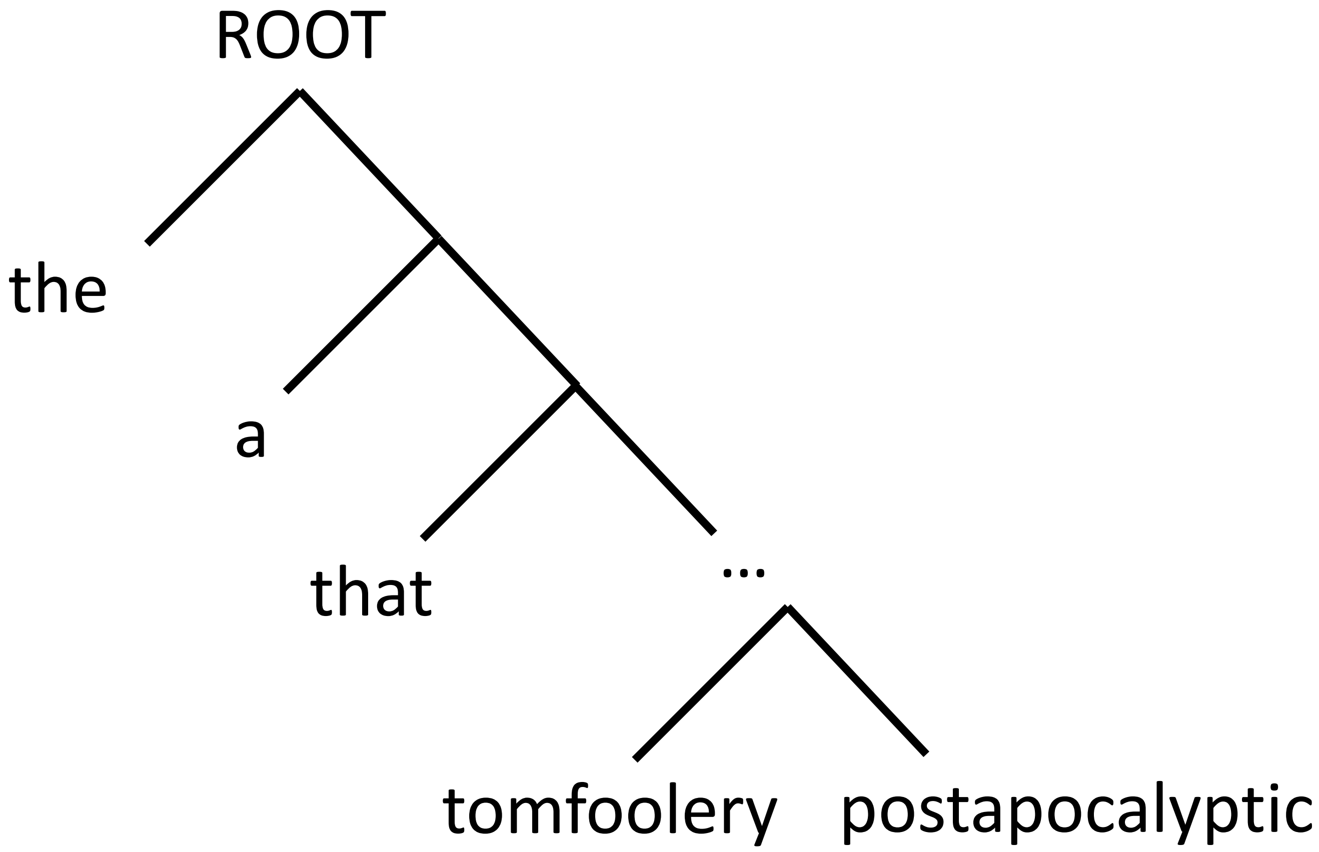
- How about this binary tree?



Why is this tree bad? Give two reasons.

(Q6 on handout)

- How about this binary tree?



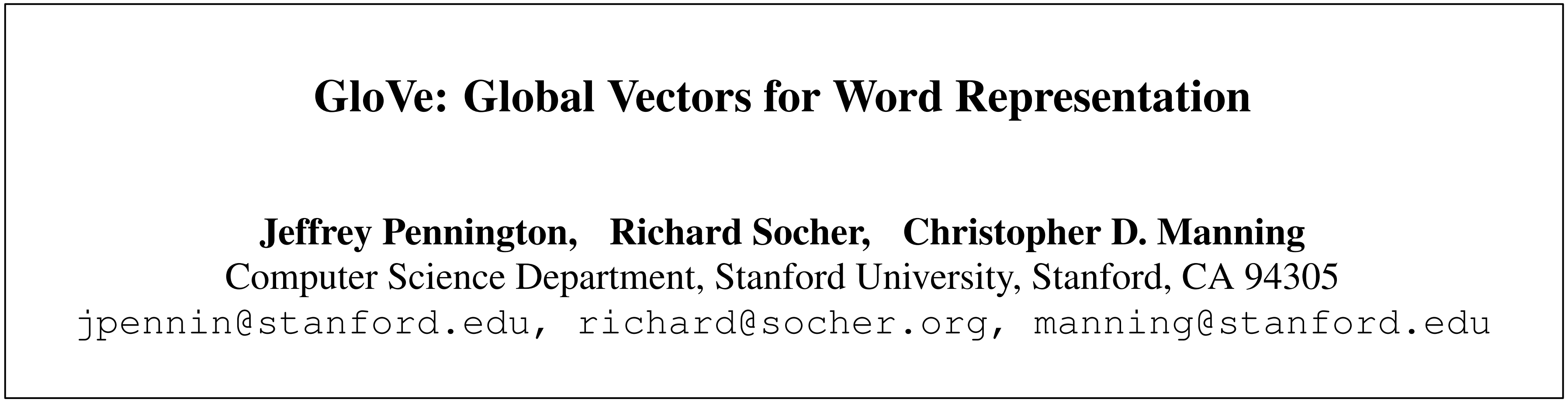
Why is this tree bad?

- word2vec uses Huffman coding (common words have short codes, i.e., are near top of tree)

Alternatives

- Noise-Contrastive Estimation (Gutmann & Hyvarinen, 2010; 2012)
- Applied to language modeling by Mnih & Teh (2012)

GloVe
(Pennington et al., 2014)



GloVe: Global Vectors for Word Representation

Jeffrey Pennington, Richard Socher, Christopher D. Manning
Computer Science Department, Stanford University, Stanford, CA 94305
jpennin@stanford.edu, richard@socher.org, manning@stanford.edu

Summary (1/2)

- we discussed 2 loss functions that avoid the need to iterate over the output space:
  - binary log loss with negative examples
  - pairwise ranking / contrastive hinge loss with negative examples
- we saw how they have been used in training word embeddings:
  - pairwise ranking loss used by Collobert et al.
  - binary log loss used by Mikolov et al.

Summary (2/2)

- results are affected by choice of method for selecting negative examples
  - permits modeler to design a scheme for the task

- we also saw how changing the generative story for the probabilistic model can reduce needed computation (hierarchical softmax)