

TTIC 31210: Advanced Natural Language Processing

Kevin Gimpel
Spring 2017

Lecture 4: Word Embeddings (2)

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*

Jason Weston[†]

Léon Bottou[‡]

Michael Karlen

Koray Kavukcuoglu[§]

Pavel Kuksa[¶]

NEC Laboratories America

4 Independence Way

Princeton, NJ 08540

RONAN@COLLOBERT.COM

JWESTON@GOOGLE.COM

LEON@BOTTOU.ORG

MICHAEL.KARLEN@GMAIL.COM

KORAY@CS.NYU.EDU

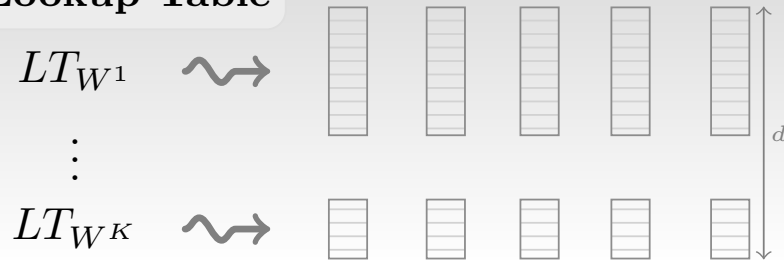
PKUKSA@CS.RUTGERS.EDU

Input Window

Text	cat	sat	on	the	mat
Feature 1	w_1^1	w_2^1	...		w_N^1
⋮					
Feature K	w_1^K	w_2^K	...		w_N^K

word of interest

Lookup Table



Linear



HardTanh



Linear



Collobert et al. Pairwise Ranking Loss

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

- \mathcal{T} is training set of 11-word windows
- \mathcal{V} is vocabulary
- What is going on here?
 - Make actual text window have higher score than all windows with center word replaced by w

Collobert et al. Pairwise Ranking Loss

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

- \mathcal{T} is training set of 11-word windows
- \mathcal{V} is vocabulary
- This still sums over entire vocabulary, so it should be as slow as log loss...
- Why can it be faster?
 - when using SGD, summation \rightarrow sample

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 :

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

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)

- 631M word tokens, 100k vocab size, 11-word input window, 4 weeks of training
- they didn't care about getting good perplexities, just good word embeddings for their downstream NLP tasks
- so a pairwise ranking loss makes sense in this context

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).

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

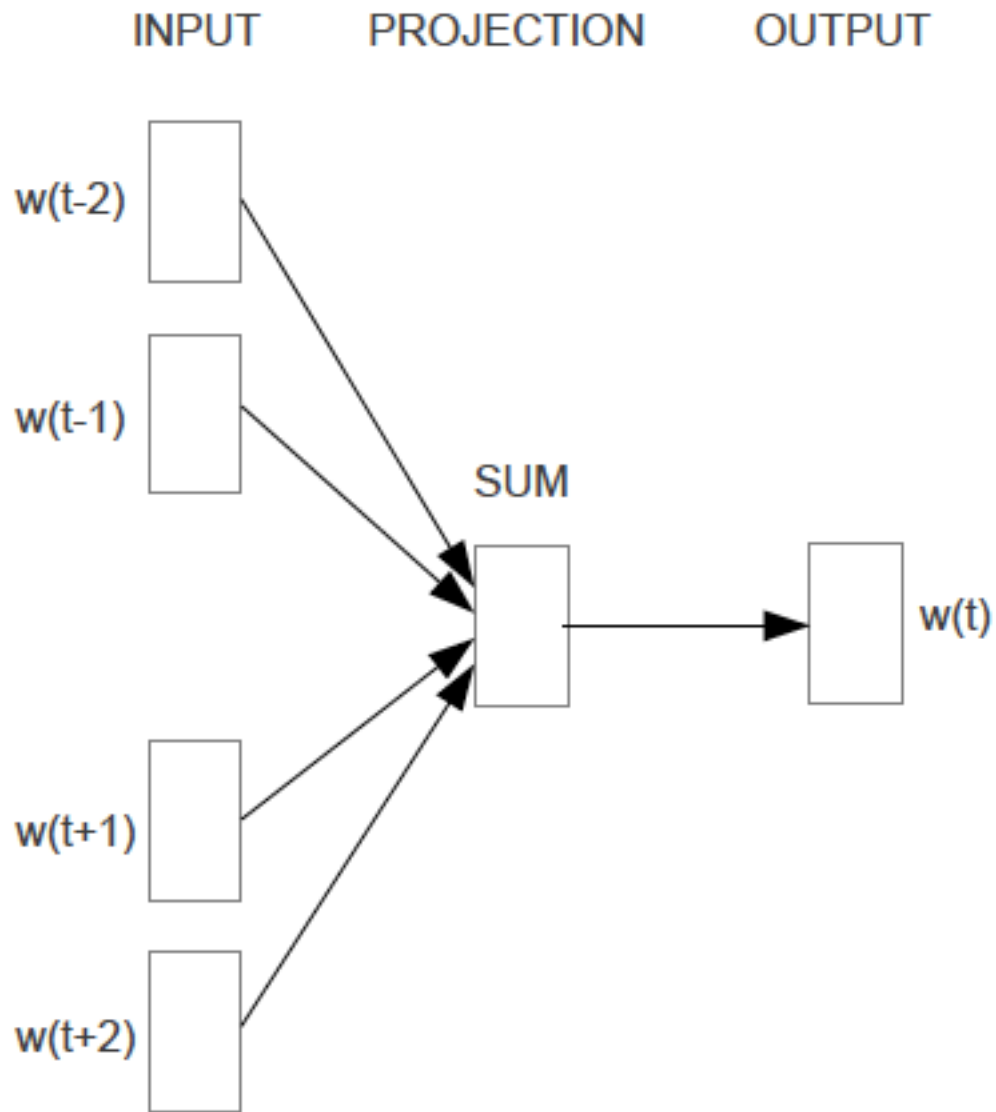
Tomas Mikolov
Google Inc.
Mountain View
mikolov@google.com

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

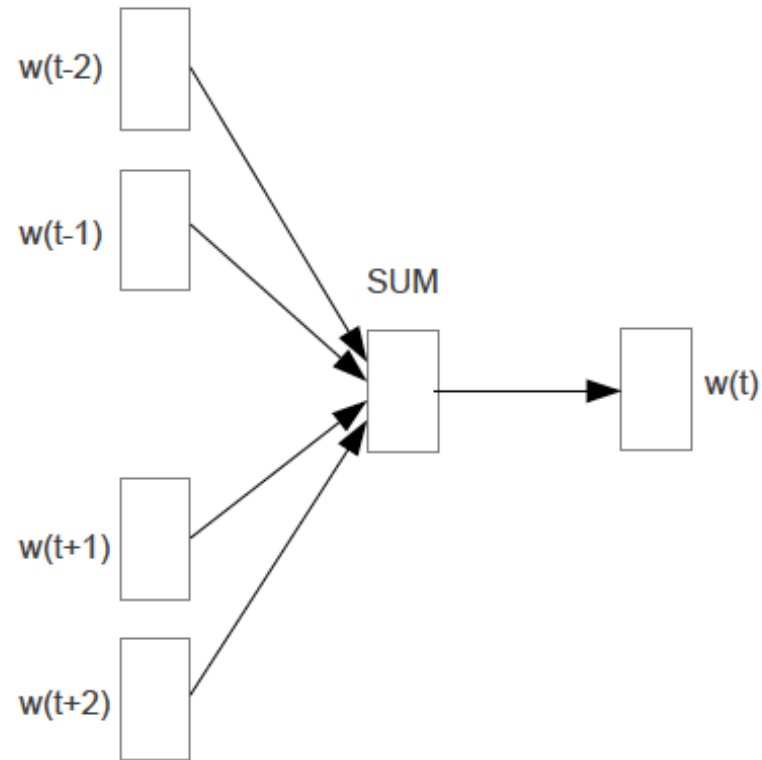


CBOW

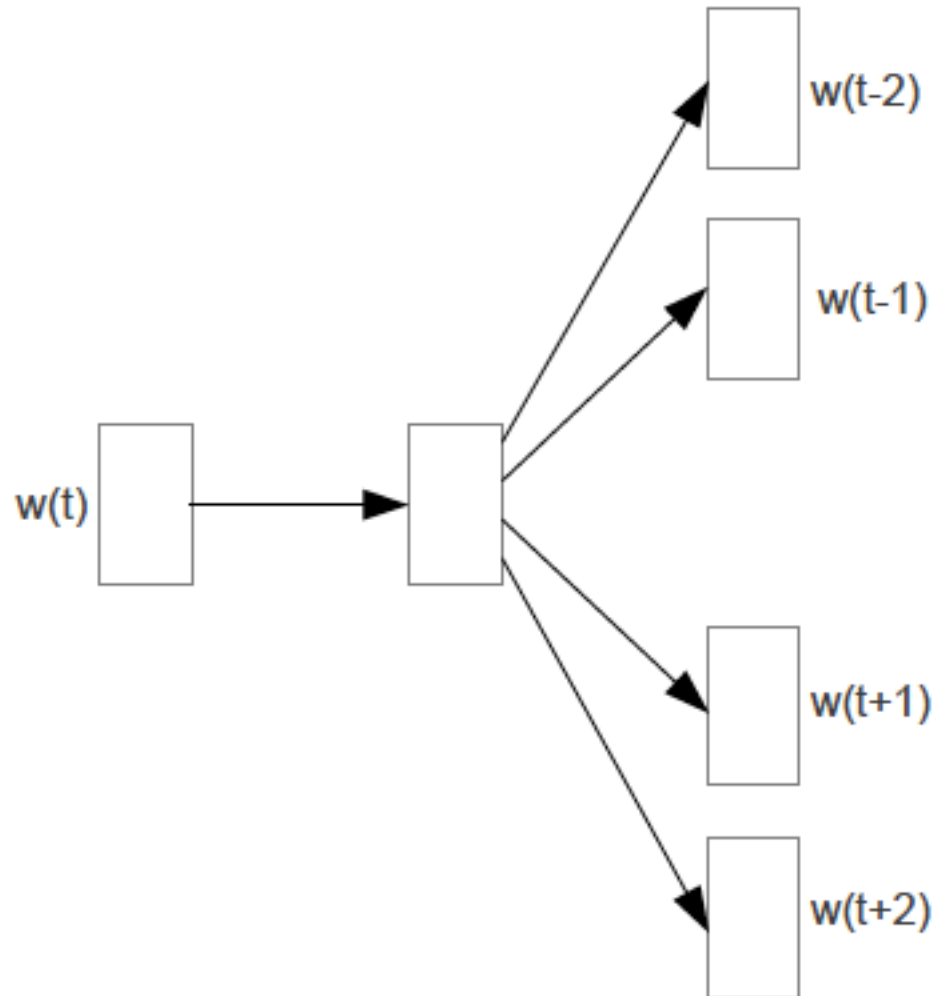
Mikolov et al. (2013a)

CBOW

- like Collobert et al. (2011), except:
 - no hidden layers
 - averages vectors of context words rather than concatenating
 - objective is to classify center word, so it's a multiclass classifier over all possible words (could be very slow!)
 - Collobert et al were essentially just training a binary classifier, where negative examples are drawn randomly from vocab



INPUT PROJECTION OUTPUT



Skip-gram

Mikolov et al. (2013a)


skip-gram

- skip-gram objective:

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



sum over
positions in
corpus



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

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

skip-gram model uses two different vector spaces:

$$P_{\theta}(u | v) \propto \exp\{\phi_u^{\top} \psi_v\}$$

$$\theta = \langle \phi, \psi \rangle$$

“output” or
“outside” vector

“input” or
“inside” vector

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

skip-gram model uses two different vector spaces:

$$P_{\theta}(u \mid v) \propto \exp\{\phi_u^{\top} \psi_v\}$$

$$\theta = \langle \phi, \psi \rangle$$

why?

which should we use as our word embeddings?

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

normalization requires sum over what?

$$P_{\theta}(u \mid v) \propto \exp\{\phi_u^{\top} \psi_v\}$$

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

normalization requires sum over entire vocabulary:

$$P_{\theta}(u \mid v) = \frac{\exp\{\phi_u^{\top} \psi_v\}}{\sum_{w \in \mathcal{V}} \exp\{\phi_w^{\top} \psi_v\}}$$

Hierarchical Softmax

(Morin and Bengio, 2005)

- based on a new generative story for $P_{\theta}(u | v)$
- 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)

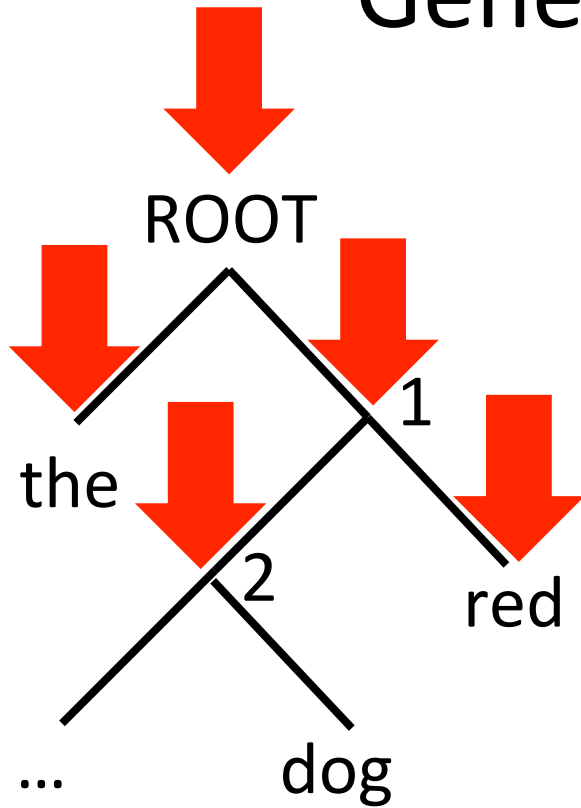
- based on a new generative story for $P_{\theta}(u | v)$
- but the generative story is so simple!
 - just draw from the conditional distribution
- how can we make it more efficient?
- we still need it to be true that:

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

Hierarchical Softmax in word2vec

- new generative story for $P_{\theta}(u | v)$
 - 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 $P_{\theta}(u | v)$



flip a coin with $\Pr(\text{heads}) = \sigma(\phi_{\text{ROOT}}^{\top} \psi_v)$

if tails,

output “the” as u

if heads,

flip coin w/ $\Pr(\text{heads}) = \sigma(\phi_{\text{split1}}^{\top} \psi_v)$

if heads,

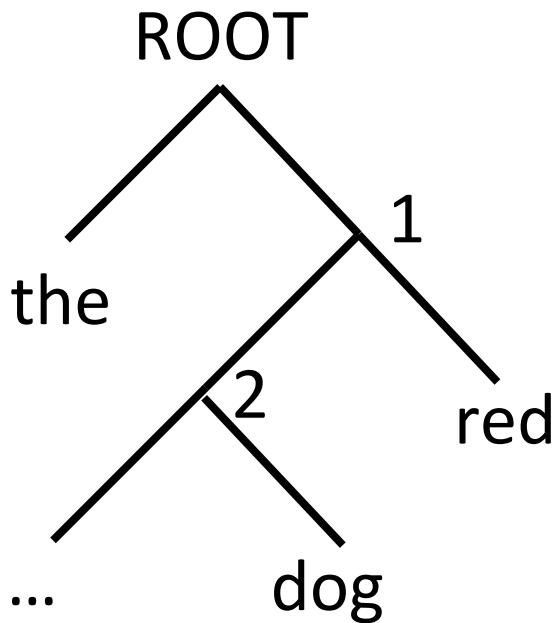
output “red” as u

if tails,

flip coin w/ $\Pr(\text{heads}) = \sigma(\phi_{\text{split2}}^{\top} \psi_v)$

...

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



flip a coin with $\Pr(\text{heads}) = \sigma(\phi_{\text{ROOT}}^{\top} \psi_v)$

if tails,

output “the” as u

if heads,

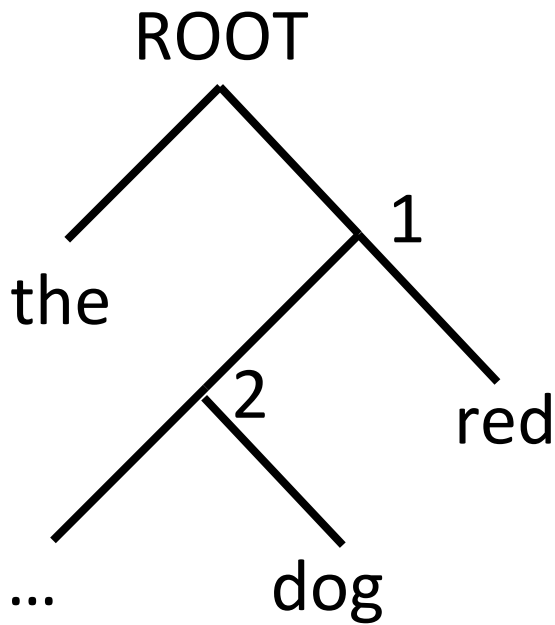
flip coin w/ $\Pr(\text{heads}) = \sigma(\phi_{\text{split1}}^{\top} \psi_v)$

if heads,

output “red” as u

What is the normalization constant?

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



flip a coin with $\Pr(\text{heads}) = \sigma(\phi_{\text{ROOT}}^{\top} \psi_v)$

if tails,

output “the” as u

if heads,

flip coin w/ $\Pr(\text{heads}) = \sigma(\phi_{\text{split1}}^{\top} \psi_v)$

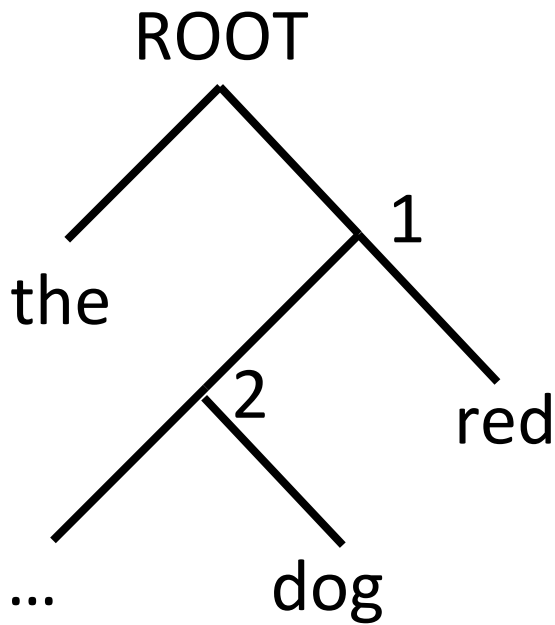
if heads,

output “red” as u

...

What is $P_{\theta}(\text{dog} | \text{cat})$?

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



flip a coin with $\Pr(\text{heads}) = \sigma(\phi_{\text{ROOT}}^{\top} \psi_v)$

if tails,

output “the” as u

if heads,

flip coin w/ $\Pr(\text{heads}) = \sigma(\phi_{\text{split1}}^{\top} \psi_v)$

if heads,

output “red” as u

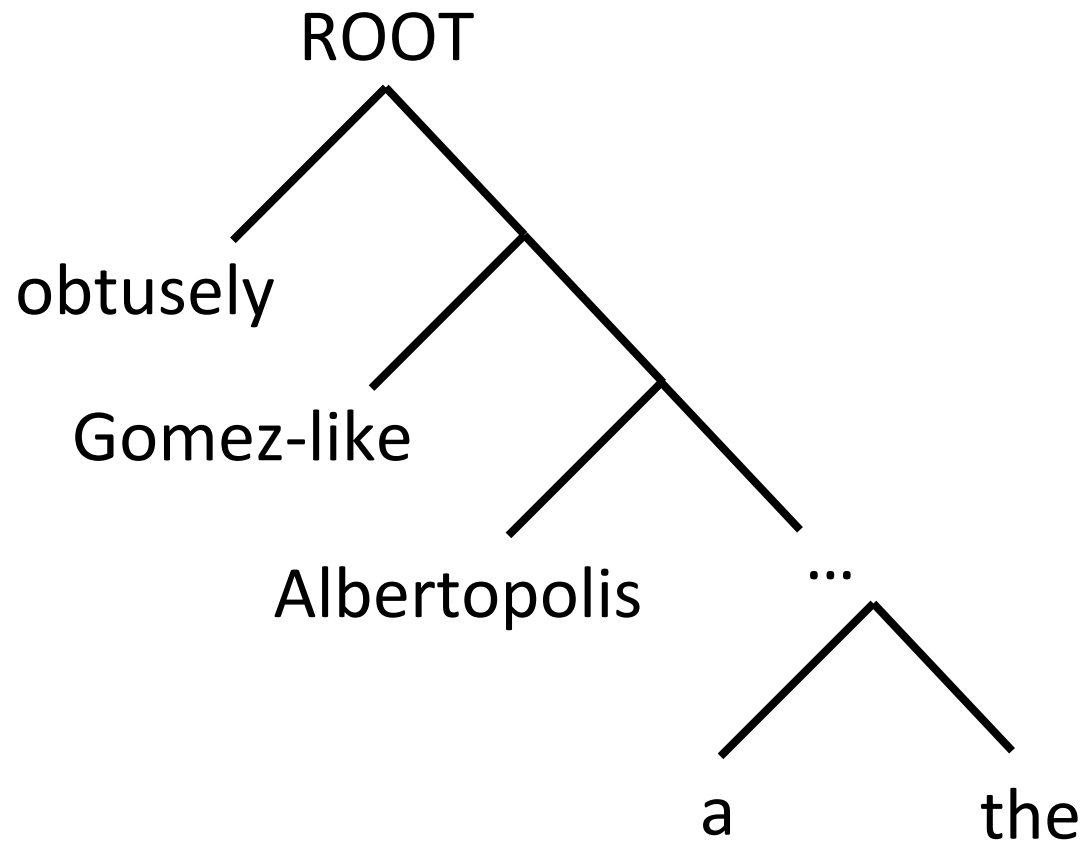
Can you prove that $\sum_{u \in \mathcal{V}} P_{\theta}(u | v) = 1$?

Hierarchical Softmax for Skip-Gram

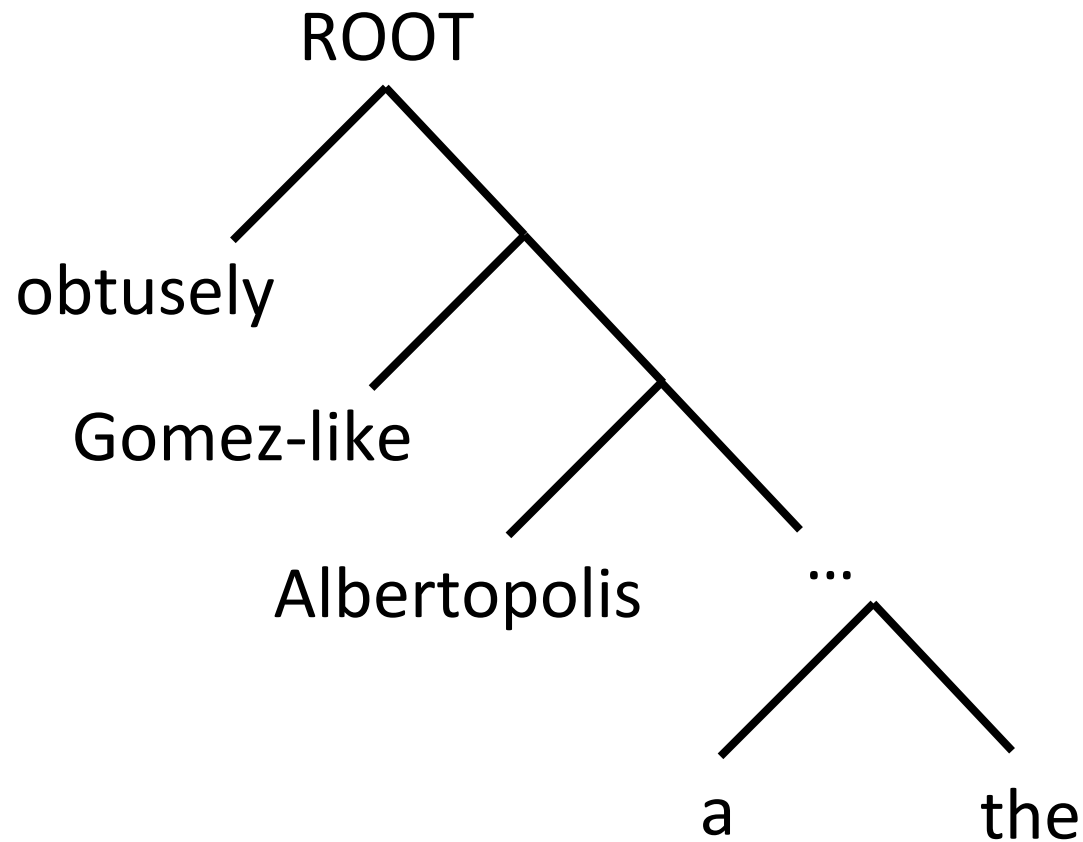
(Mikolov et al., 2013)

- 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?



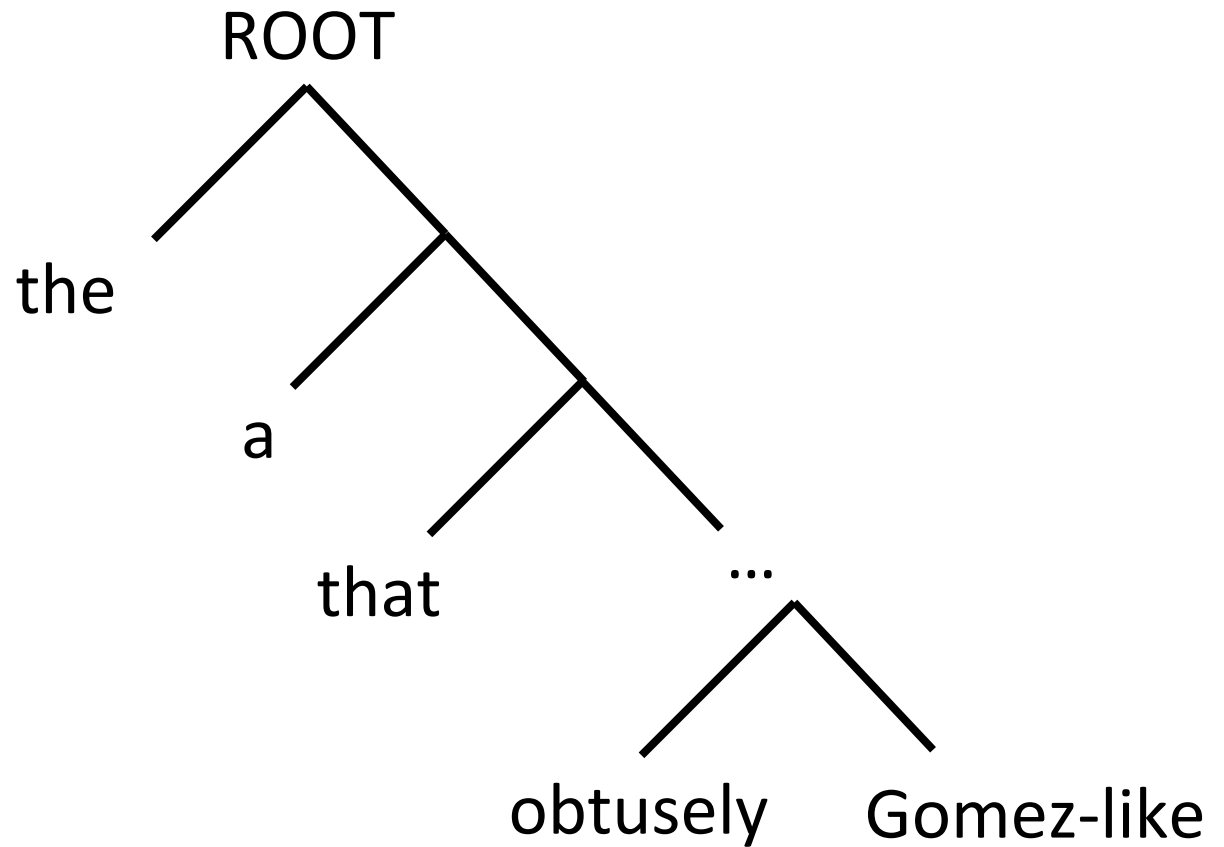
- How about this binary tree?



Why is this tree bad?

Give two reasons.

- 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)

Negative Sampling

(Mikolov et al., 2013)

- rather than sum over entire vocabulary, generate samples and sum over them
- instead of a multiclass classifier, use a binary classifier:

$$\min_{\theta} \sum_{1 \leq t \leq |\mathcal{T}|} \sum_{-c \leq j \leq c, j \neq 0} -\log \sigma(\phi_{x_{t+j}}^{\top} \psi_{x_t}) + \sum_{x \in \text{NEG}} \log \sigma(\phi_x^{\top} \psi_{x_t})$$

- similar to idea of Collobert et al but uses log loss instead of hinge loss

Negative Sampling

(Mikolov et al., 2013)

$$\min_{\theta} \sum_{1 \leq t \leq |\mathcal{T}|} \sum_{-c \leq j \leq c, j \neq 0} -\log \sigma(\phi_{x_{t+j}}^{\top} \psi_{x_t}) + \sum_{x \in \text{NEG}} \log \sigma(\phi_x^{\top} \psi_{x_t})$$

- NEG contains 2-20 words sampled from some distribution
 - e.g., uniform, unigram, or smoothed unigram
 - smoothed: raise probabilities to power $\frac{3}{4}$, renormalize to get a distribution

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`

Other Work on Word Embeddings

- active research area (probably too active)
- other directions:
 - multiple embeddings for a single word corresponding to different word senses
 - using subword information (e.g., characters) in word embeddings
 - tailoring embeddings for different NLP tasks

- Pretrained word embeddings are really useful!
- What about embeddings for phrases and sentences?

Unsupervised Sentence Models

- How are sentence embeddings useful?
 - multi-document summarization
 - automatic essay grading
 - evaluation of text generation systems
 - machine translation
 - entailment/inference

Unsupervised Sentence Models

- how should we evaluate sentence models?
- we consider two kinds of evaluations here:
 - sentence similarity: intrinsic evaluation of sentence embedding space, no additional learned parameters
 - sentence classification: train a linear classifier (logistic regression) using the fixed sentence representation as the input features
 - reporting average accuracies over 6 tasks