Week 3 (Cont.)
Word Embeddings: Prediction-based Approach

Nhung Nguyen
slides courtesy of Phong Le

Recap

- Two major count-based approach methods: term-document matrix and term-term matrix
- Raw frequency is bad
  - Using weighing schemes to "correct" counts
  - Using smoothing to take into account "unseen" events

"Fill in the blanks" (TOEFL or IELTS)

___________ is the study of numbers, equations, functions, and geometric shapes and their relationships.

a. physics

$\boxed{\text{b. mathematics}}$

c. geography

d. theology

- Random guess: 25% accuracy
- Get > 25% accuracy: the *examinee* is supposed to know the meanings of correct words and their contexts.
- If a *computer* successfully fulfill this task, it is supposed to *understand* word meanings

Formalisation

Assumption:

- each word ***w*** ∈ V is represented by a vector **v** ∈ R^d (d is often smaller than 3k)
- there is a mechanism to compute the probability **Pr**(*w*|u_1, u_2, ..., u_l) of the event that a target word *w* appears in a context (u_1, u_2, ..., u_l).

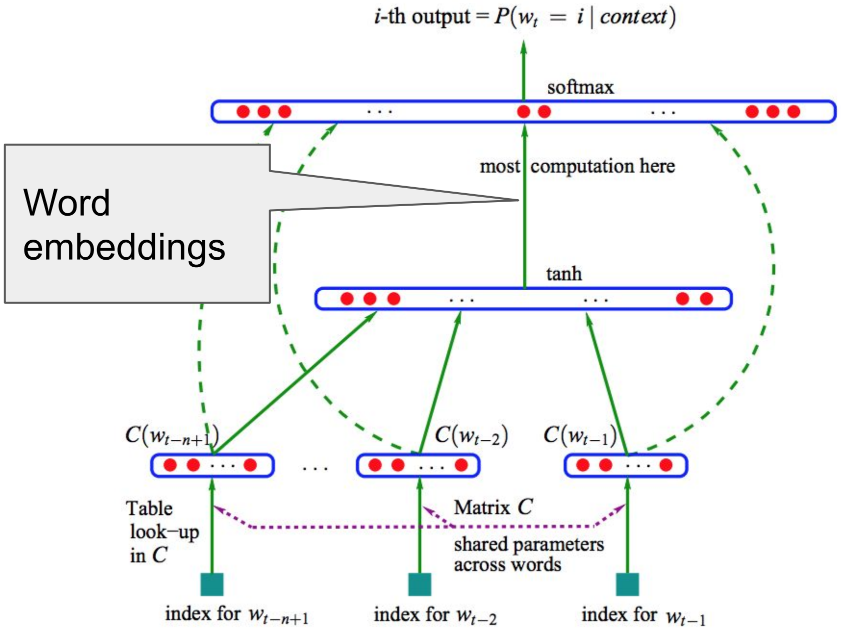
Task: find a vector **v** for each word *w* such that those probabilities are as high as possible for each *w* and its context (u_1, u_2, ..., u_l).

We use a neural network with parameters θ to compute the probability by minimizing the cross-entropy loss

$$L(\theta) = - \sum_{(w,u_1,\ldots,u_l) \in D_{train}} \log Pr(w|u_1, \ldots, u_l)$$

Bengio et al. (2003)

- **Language modelling**: predict a next word from *m-1 previous words* $\Pr(w_t|w_{t-1},\ldots,w_{t-m+1})$
- Each word w is represented by a vector **v**



$\Pr(w_t|w_{t-1},\ldots,w_{t-m+1}) = \text{softmax}(\mathbf{Wy})$

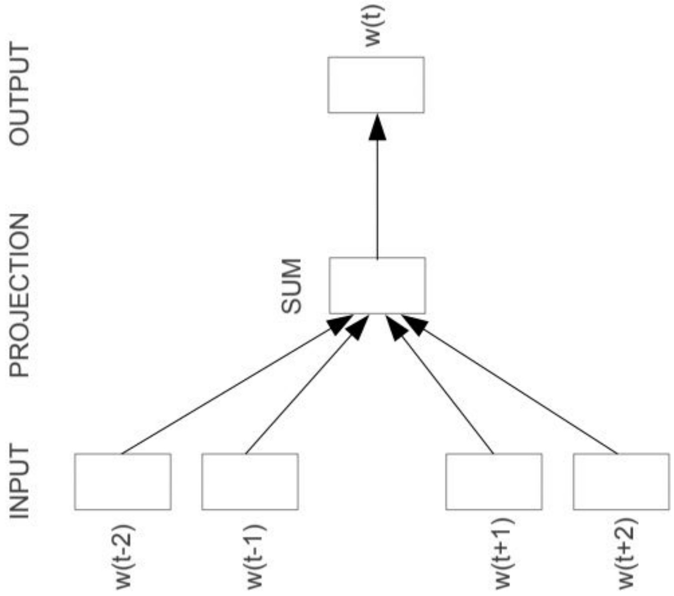
$\mathbf{y} = \tanh(\mathbf{Vx})$

$\mathbf{x} = \text{concat}(\mathbf{w}_{t-1},\ldots,\mathbf{w}_{t-m+1})$

Each *contextual* word w_{t-j} is represented a column of matrix **C**

Mikolov et al. (2013): CBOW (continuous bag of words)

- Predicting a word from the surrounding words (a context is *m words before* and *m words after* each target word w_t)



$Pr(w_t|w_{t-1},\ldots,w_{t-m+1})$ = softmax($\mathbf{Wy}$)

$\mathbf{y}$ = average($\mathbf{w}_{t-1},\ldots,\mathbf{w}_{t-m}, \mathbf{w}_{t+1},\ldots,\mathbf{w}_{t+m}$)

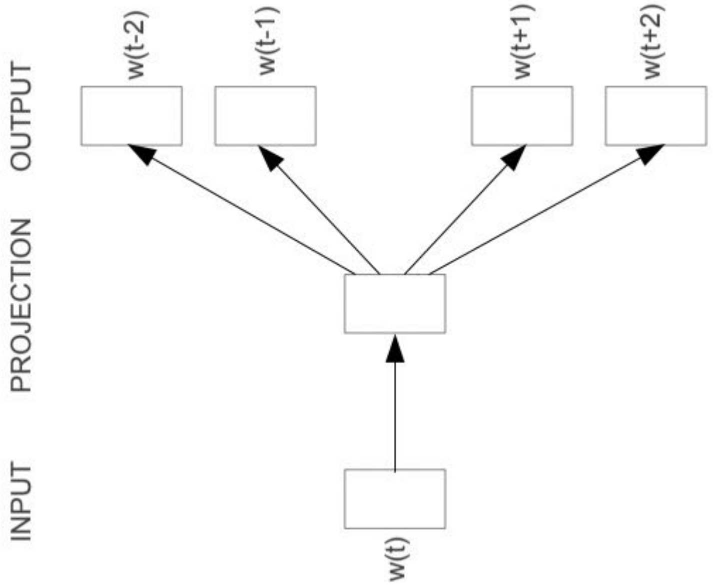
Each *contextual* word w_j is represented a column of matrix **C**

Bengio et al. (2003) vs. CBOW

- The two models look pretty similar
- They however use different types of contexts:
  - Bengio et al. (2003) uses *m-1 words before* the target word → language modelling (predicting a next word given a history)
  - CBOW uses *m words before and m words after* the target words (i.e., a window of size 2m words)
- CBOW is simpler because it has one less layer

Mikolov et al. (2013): Skip-gram

- Predicting surrounding words given a target word (a context is *m words before* and *m words after* each target word w_t)



$\Pr(.|w_t) = \mathrm{softmax}(\mathbf{W}_{OUT}\mathbf{y})$

$\mathbf{y} = \mathbf{w}_t$

Each *target* word w_t is represented by a column of matrix $\mathbf{W}_{IN}$

Mikolov et al. (2013): Skip-gram (cont.)

- Different strategy: we predict contextual words rather than the target word

$$Pr(w_j|w_t)_{\forall j=t-m,\ldots,t-1,t+1,\ldots,t+m} \text{ versus}$$

$$Pr(w_t|w_{t-1},\ldots,w_{t-m},w_{t+1}\ldots,w_{t+m})$$

They have a close relation

$$Pr(w_t|w_{t-1},\ldots,w_{t-m},w_{t+1},\ldots,w_{t+m}) = \frac{Pr(w_{t-1},\ldots,w_{t-m},w_{t+1},\ldots,w_{t+m}|w_t) \times Pr(w_t)}{Pr(w_{t-1},\ldots,w_{t-m},w_{t+1},\ldots,w_{t+m})}$$

$$= U * \prod_{j=\{t-m,\ldots,t+m\}\setminus\{0\}} Pr(w_j|w_t)$$

Uniform distr.

Constant (safely removed when training and testing)

word2vec

- Skip-gram model
- "a baby step in Deep Learning but a giant leap towards Natural Language Processing"
- can capture linear relational meanings (i.e., analogy):

```
king - man + woman = queen
```

word2vec (cont.)

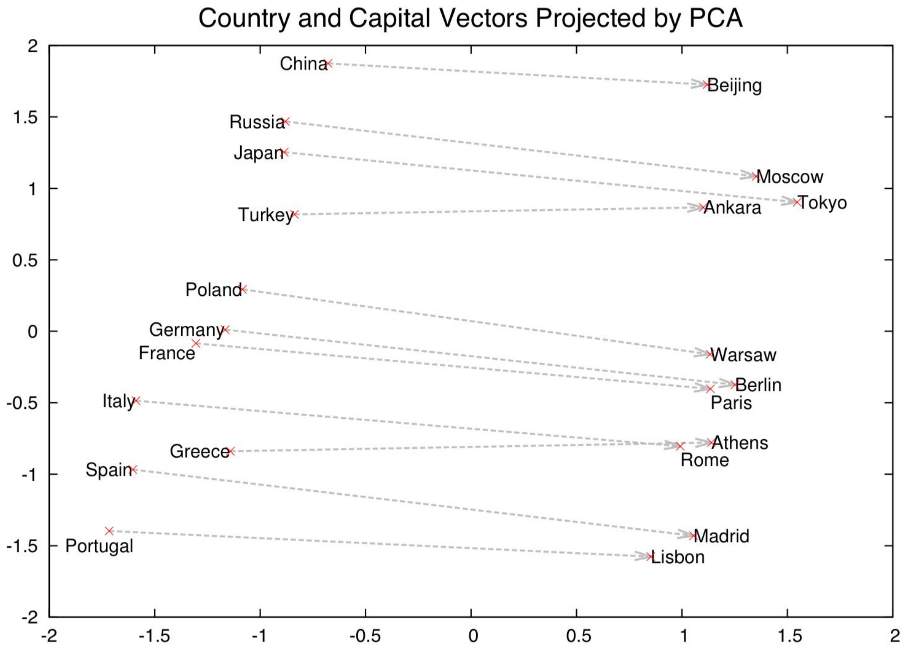
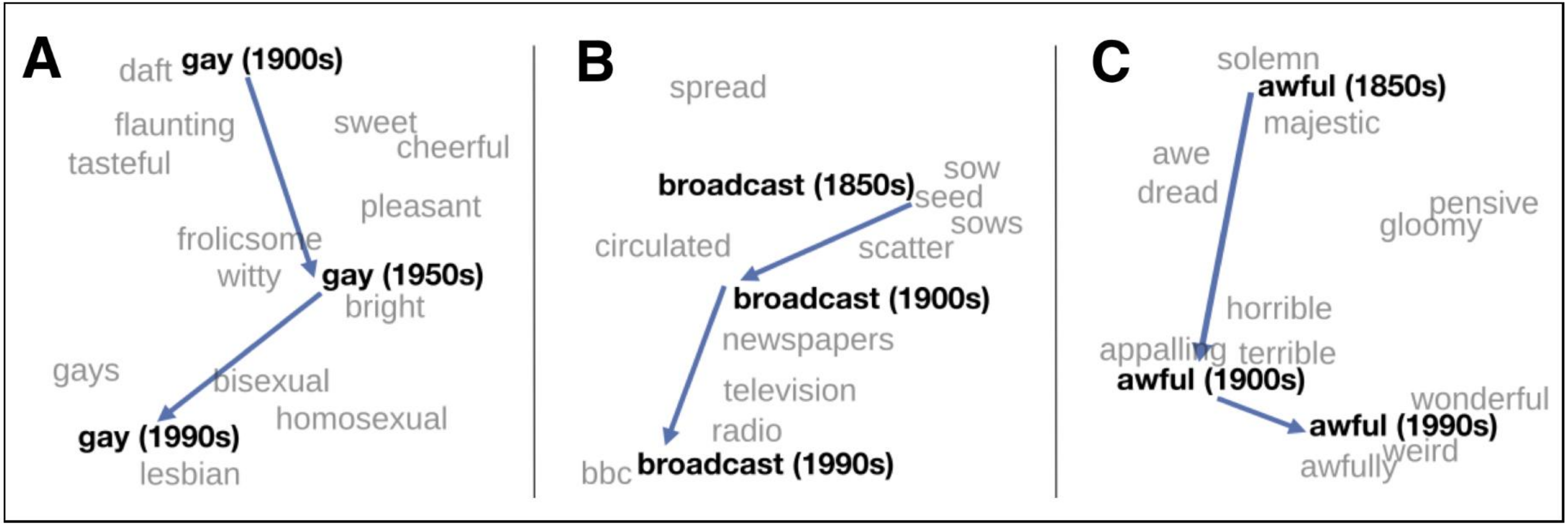


Figure 2: Two-dimensional PCA projection of the 1000-dimensional Skip-gram vectors of countries and their capital cities. The figure illustrates ability of the model to automatically organize concepts and learn implicitly the relationships between them, as during the training we did not provide any supervised information about what a capital city means.

word2vec (cont.)

Semantic change



Problems: biases (gender, ethnic, …)

- Word embeddings are learned from data → they also capture biases implicitly appearing in the data
- Gender bias:
  - "computer_programmer" is closer to "man" than "woman"
  - "homemaker" is closer to "woman" than "man"
- Ethnic bias:
  - African-American names are associated with unpleasant words (more than European-American names)
- …

→ Debiasing embeddings is a hot (and very needed) research topic

Dealing with unknown words

- Many words are not in dictionaries
- New words are invented everyday
- Solution 1: using a special token #UNK# for all unknown words
- Solution 2: using characters/sub-words instead of words
  - Characters (c-o-m-p-u-t-e-r instead of computer)
  - Subwords (com-omp-mpu-put-ute-ter instead of computer)

Word embeddings in a specific context

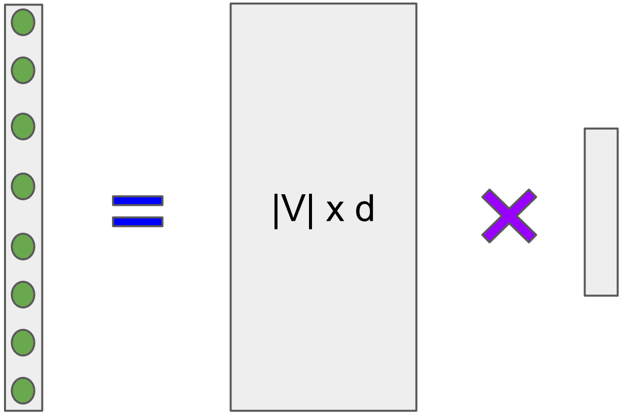
- The meaning of a word standing alone can be different than its meaning in a specific context
  - He lost all of his money when the bank failed.
  - He stood on the bank of Amstel river and thought about his future.
- Solution: $w_{|c} = f(w, c)$
- Solution 1: f is continuous w.r.t. c (contextual embeddings, e.g., ELMO, BERT - next week)
- Solution 2: f is discrete w.r.t. c (e.g., word sense disambiguation - coming up in the next video)

Summary

- Prediction-based approaches require neural network models, which are not intuitive as count-based ones
- Low dimensional vectors (about 200-400 dimensions)
  - Dimensions are not easy to interpret
- Robust performance for NLP tasks

Predicting words: the bottleneck

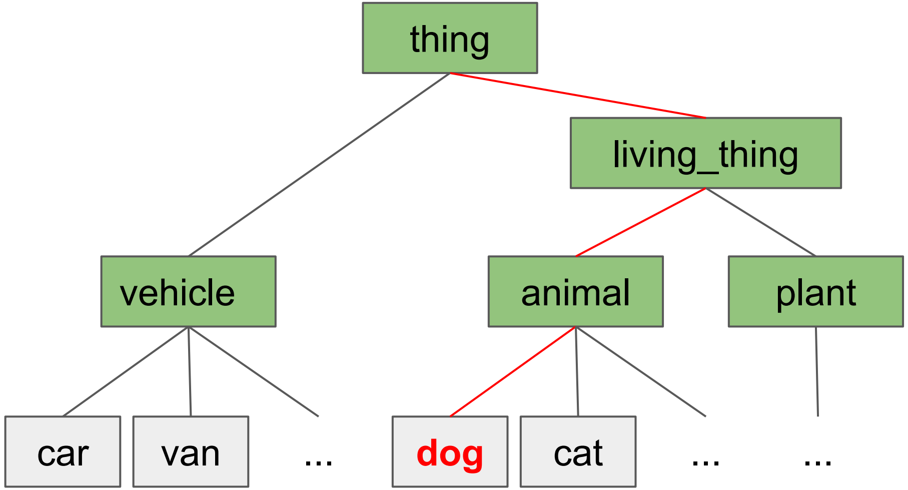
- Predicting a word w requires Pr(w) = softmax(**Wy**) (for simplicity, we ignore conditions)
- Matrix **W** is very large: $\mathbf{W} \in R^{|V| \times d}$, |V| is often 100K to 1M or more
- Computational complexity O(|V| x d)



Predicting words: the bottleneck (cont.)

<u>Solution 1: Hierarchical softmax</u>

(Group words into a hierarchical structure)



Pr(w=dog) =

Pr(living_thing|thing) — softmax($\mathbf{W}_{thing}\mathbf{y}$)

x Pr(animal|living_thing) — softmax($\mathbf{W}_{living\_thing}\mathbf{y}$)

x Pr(dog|animal) — softmax($\mathbf{W}_{animal}\mathbf{y}$)

- can reduce the computational complexity from O(d|V|) to O(d.log|V|) (why?)
- how to build the structure? (using wordnet, Brown clustering,...)

Predicting words: the bottleneck (cont.)

Solution 2: (negative) sampling

- we need word embeddings, not Pr(w)
- rephrase 1:

$$\boxed{\text{score}(w) > \text{score}(w') \text{ for all } w' \text{ diff. } w}$$

- score(u) = $\mathbf{u}^T\mathbf{y}$ (remind: **y** is the output of the previous layer)
- Loss: $$L(\theta) = \sum_{w \in D_{train}} \underbrace{E_{w' \sim Pr_n(w')}}_{\text{sampling } w'} \underbrace{\left[\max\{0, score(w') - score(w)\} \right]}_{\substack{> 0 \text{ only when score}(w) < \text{score}(w') \\ \to \text{ increase score}(w) \text{ and decrease score}(w')}}$$

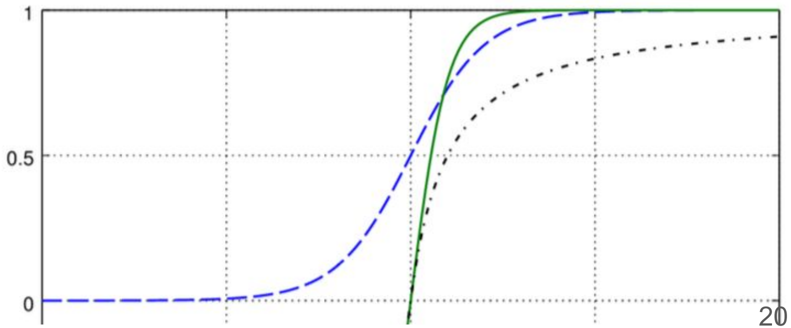
Predicting words: the bottleneck (cont.)

Solution 2: (negative) sampling

- we need word embeddings, not Pr(w)
- rephrase 2: given w and its context c

$$\boxed{\Pr(+|w,v) = 1 \text{ if } v \text{ in } c, = 0 \text{ otherwise}}$$

- Pr(+|w,v) = sigmoid($\mathbf{v}^T\mathbf{y}$) (why sigmoid?)



Predicting words: the bottleneck (cont.)

- Negative sampling:
  - positive examples from data (target word and its contexts)
  - negative examples for a target word: draw a word from the (adjusted) unigram dist. $Pr_n(v)$

positive examples +

t	c
apricot	tablespoon
apricot	of
apricot	jam
apricot	a

negative examples -

t	c	t	c
apricot	aardvark	apricot	seven
apricot	my	apricot	forever
apricot	where	apricot	dear
apricot	coaxial	apricot	if

- Loss: $$L(\theta) = - \sum_{(w,v^+)\in D_{train}} \underbrace{Pr(+|w,v^+)}_{\nearrow \text{ prob of pos. examples}} + \underbrace{E_{v^- \sim Pr_n(v^-)}}_{\text{sampling neg. examples}} \underbrace{[1 - Pr(+|w,v^-)]}_{\searrow \text{ prob of neg. examples}}$$