Word Representations

COMP 527

Danushka Bollegala

UNIVERSITY OF LIVERPOOL

Meaning of a word

- lexical semantics
  - The branch in NLP that focuses on how we can represent/process meanings of words.
- If we can handle the meanings of individual words properly, then we can use compositional approaches to construct the meanings of larger constituents such as phrases, sentences, or documents.
  - cf. compositional semantics

Do words have meanings?

Not really.

They just borrow meanings from their neighbours

Distributional Hypothesis

J. R. Firth

"You shall know a word by the company it keeps"

Image credit: www.odlt.org

Quiz

- X is a device that is easy to carry around, you can speak using X, watch the Internet.What could be X?
  - a dog
  - an airplane
  - an iPhone
  - a banana

But is that really true?

- Don’t dictionaries define the meanings of words?
  - Dictionaries define the meanings of words using other words, in a recursive manner.
- Distributional hypothesis provides us with a practical method to learn the meanings of words using large text corpora
- Distributional semantic representations have been successfully used in numerous NLP tasks reporting state-of-the-art performances. Therefore, it must be correct.

Two approaches…

- Distributional Semantic Representations
    - Use the set of words that co-occur with X to represent the meaning of X
    - Sparse and high-dimensional
    - Classical approach for semantic representations
- Distributed Semantic Representations
    - Learn representations that can accurately predict the words that appear in the same context as X.
    - Limited dimensionality(10~1000)
    - Low dimensional and dense
    - Deep learning (to be precise representation learning) methods have been used
    - A more recent/modern approach

Two approaches…

- Distributional Semantic Representations
  - Use the set of words that co-occur with X to represent the meaning of X
  - Sparse and high-dimensional
  - Classical approach for semantic representations
- Distributed Semantic Representations
  - Learn representations that can accurately predict the words that appear in the same context as X.
  - Limited dimensionality(10~1000)
  - Low dimensional and dense
  - Deep learning (to be precise representation learning) methods have been used
  - A more recent/modern approach

Example

- Let us create a representation for “apple”
- S_1=Apples are red.
- S_2=Red apples are delicious.
- S_3=Apples are produced in Washington.

Example

- Let us create a representation for “apple”
- S_1=Apples are red.
- S_2=Red apples are delicious.
- S_3=Apples are produced in Washington.

apple=[(red,2),(delicious,1),(washington,1),(produce,1)]

Applications

- Measure the semantic between apple and orange?
- First, lets create a semantic representation for oranges.
- S_4=Oranges are yellow.
- S_5=Oranges are delicious.
- S_6=Oranges are produced in California.

orange=[(yellow,1),(delicious,1),(california,1),(produce,1)]

“apple” vs. “orange”

apple=[(red,2),(delicious,1),(washington,1),(produce,1)]

orange=[(yellow,1),(delicious,1),(california,1),(produce,1)]

We can measure the similarity between the two words by the overlapping features/attributes in their respective semantic representations.

Jaccard Coefficient = |apple AND orange| / |apple OR orange|

sim(apple,orange) = 2/6 = 0.3333

Co-occurrence Matrix

- We can arrange the semantic representations we learn for all the words in a corpus as rows in a co-occurrence matrix.
  - rows = semantic representations of words
  - columns = various features/words that co-occur with words

$$\begin{array}{cc} & \begin{array}{cccccc} \text{red} & \text{delicious} & \text{washington} & \text{produce} & \text{yellow} & \text{california} \end{array} \\ \begin{array}{c} \text{apple} \\ \text{orange} \end{array} & \begin{pmatrix} 2 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix} \end{array}$$

Issues of large co-occurrences

- How reliable are large co-occurrences?
- Consider Google hits (no. of pages) for,
  - (car, automobile) = 11,300,000
  - (car, apple) = 49,000,000
- apples are more similar to cars than automobiles???
- We need proper weighting for the co-occurrences (in particular when some words are very common)

Co-occurrence Weighting Measures

- Many methods exist. But they are basically a combination of occurrences of individual words, h(x), h(y), and co-occurrences between two words, h(x,y).
  - weighting function = f(h(x), h(y), h(x,y))
- pointwise mutual information (PMI)

$$\mathrm{PMI}(x,y) = \log\left(\frac{p(x,y)}{p(x)p(y)}\right) = \log\left(\frac{h(x,y)/N}{(h(x)/N) \times (h(y)/N)}\right)$$

Alternatives...

- positive pointwise mutual information (PPMI)
  [Turney+Pantel JAIR'10]
  - PPMI(x,y) = max(0, PMI(x,y))
- shifted pointwise mutual information (SPMI)
  [Levy+Goldberg NIPS'14]
  - SPMI(x,y) = PMI(x,y) - log(k)
  - Here, k is a constant (parameter)

Issues of zero co-occurrences

- Some words never co-occur even in very large corpora.
- If words x and y do not co-occur, then
  - x and y might not be related OR
  - it could be that our corpus was too small and we did not observe their co-occurrences.
- If we have co-occurrence-based semantic representations that have many zeros, then we will have many zero similarity scores, which is not good.
- How can we reduce the number of zeros?

Dimensionality reduction / Low-dimensional projection

- We can reduce the number of features (columns) thereby collapsing similar dimensions.
- This process will reduce the number of zeros.

$$
\begin{array}{c|cccccc|}
 & \text{red} & \text{delicious} & \text{washington} & \text{produce} & \text{yellow} & \text{california} \\
\text{apple} & 2 & 1 & 1 & 1 & 0 & 0 \\
\text{orange} & 0 & 1 & 0 & 1 & 1 & 1
\end{array}
\rightarrow
\begin{array}{c|ccc|}
 & \text{dim 1} & \text{dim 2} & \text{dim 3} \\
\text{apple} & 0.3 & 0.2 & 1 \\
\text{orange} & 1.2 & -0.1 & 0.3
\end{array}
$$

Note that the number of words (rows) does not change. Therefore, we have a representation for all the words that appear in the original co-occurrence matrix.

Dimensionality reduction methods

- Singular Value Decomposition (SVD)
  - $A = UDV^T$, use U or UD as the lower-dimensional projection
- Principal Component Analysis (PCA)
  - See the lecture on dimensionality reduction
- non-negative matrix factorization (NMF)
  - $A = WH$
- There are many libraries that implement the above (and many more)
  - In Python: numpy, scipy, sklearn

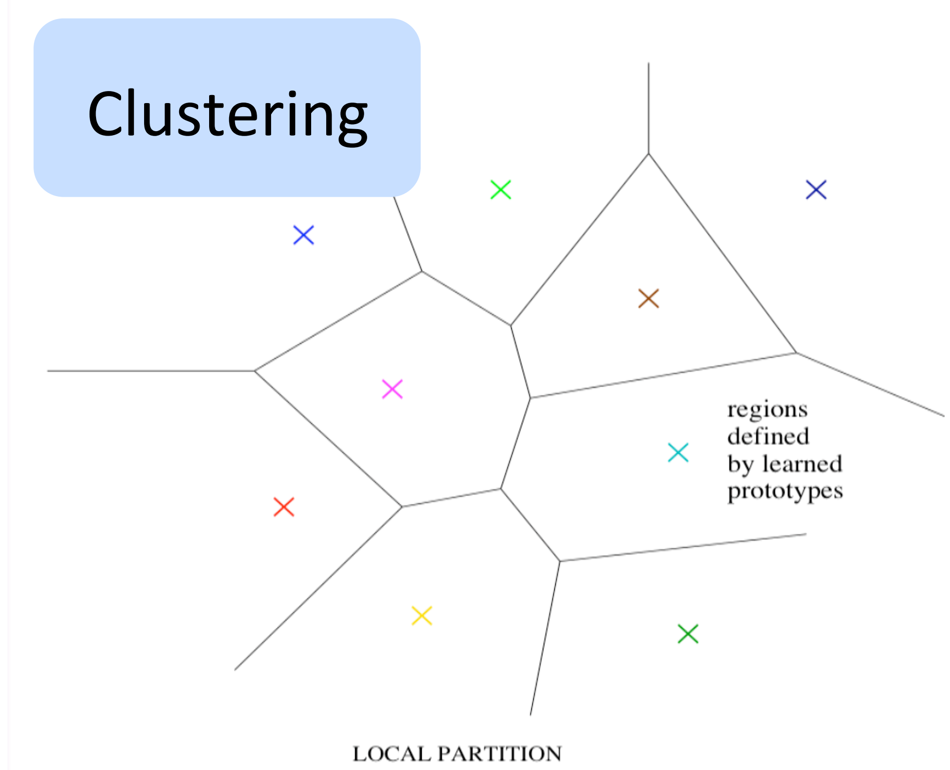
the details...

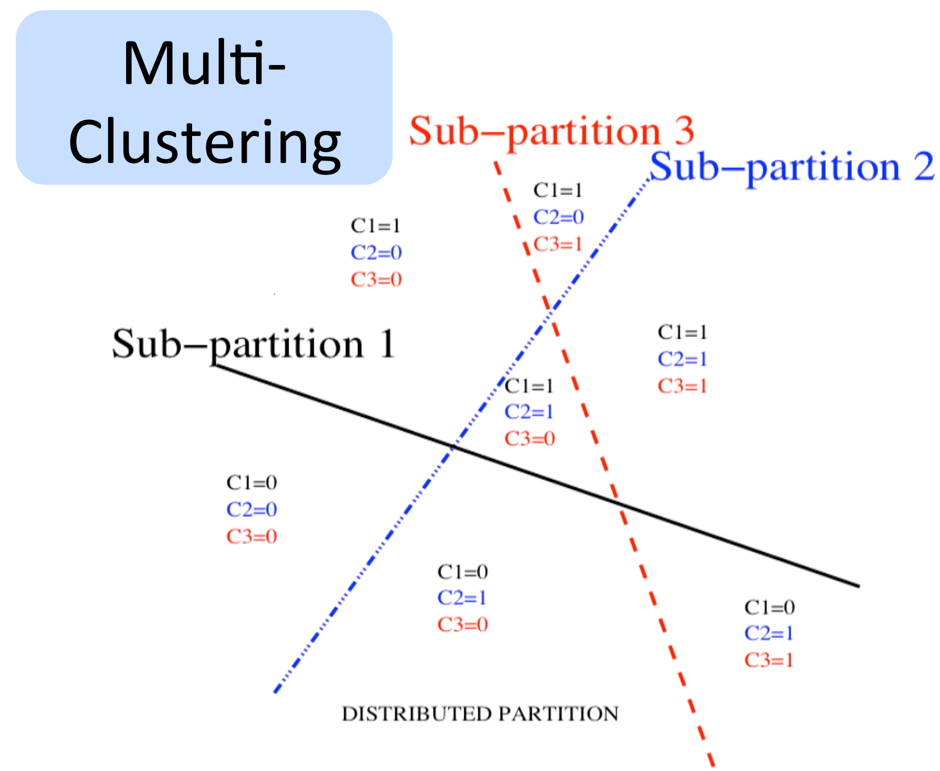
- How to select the “context”?
  - sentence-level co-occurrences
  - proximity window (n words before/after)
  - Words that are connected via dependency relations
- Assign co-occurrence weights inversely proportional to the distance
  - [1/5, 1/4, 1/3, 1/2, *, 1/2, 1/3, 1/4, 1/5]

Two approaches…

- Distributional Semantic Representations
  - Use the set of words that co-occur with X to represent the meaning of X
  - Sparse and high-dimensional
  - Classical approach for semantic representations
- Distributed Semantic Representations
  - Learn representations that can accurately predict the words that appear in the same context as X.
  - Limited dimensionality(10~1000)
  - Low dimensional and dense
  - Deep learning (to be precise representation learning) methods have been used
  - A more recent/modern approach

Local vs. Distributional Representations



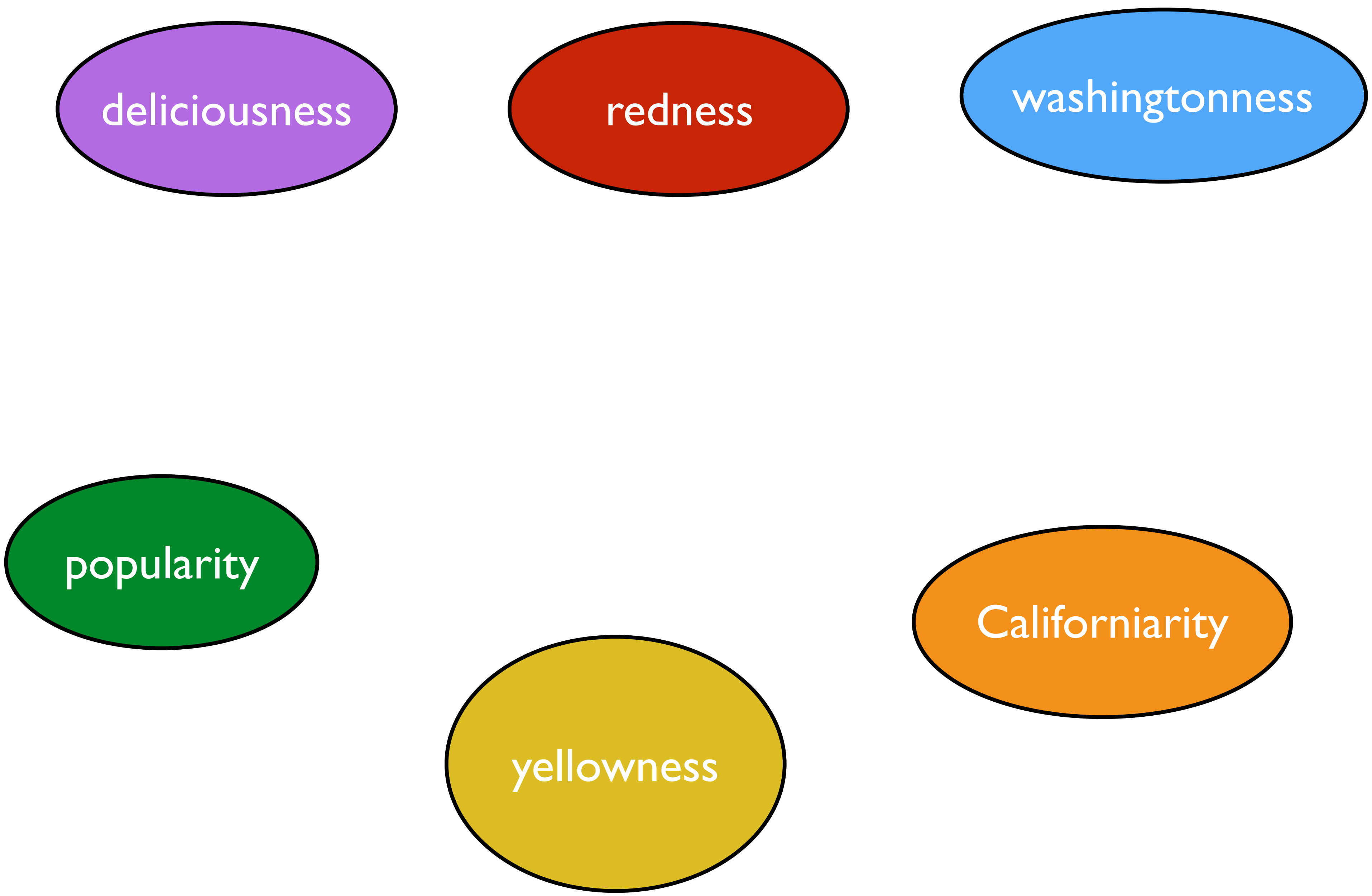


We only require the nearest neigbours when determining the label of a point

With 3 partitions we have 8 regions. All partitions need to be consulted when determining the class of a point (exponential expressiveness 2^n)

slide credit: Yoshua Bengio

Distributed representations



Learning Word Representations

- In distributional word representations we followed
  - a counting-based approach
  - a bottom-up method for learning representations
- On the other hand, in distributed word representations, we first initialize each word with a random vector, and then adjust the elements of those vectors such that they can accurately predict other words that co-occur in their local contexts.
  - prediction-based approach
  - a top-down approach

Example: word2vec

- word2vec is a tool for learning distributed word representations and implements two algorithms
  - skip-gram model
  - continuous bag-of-words(CBOW) model

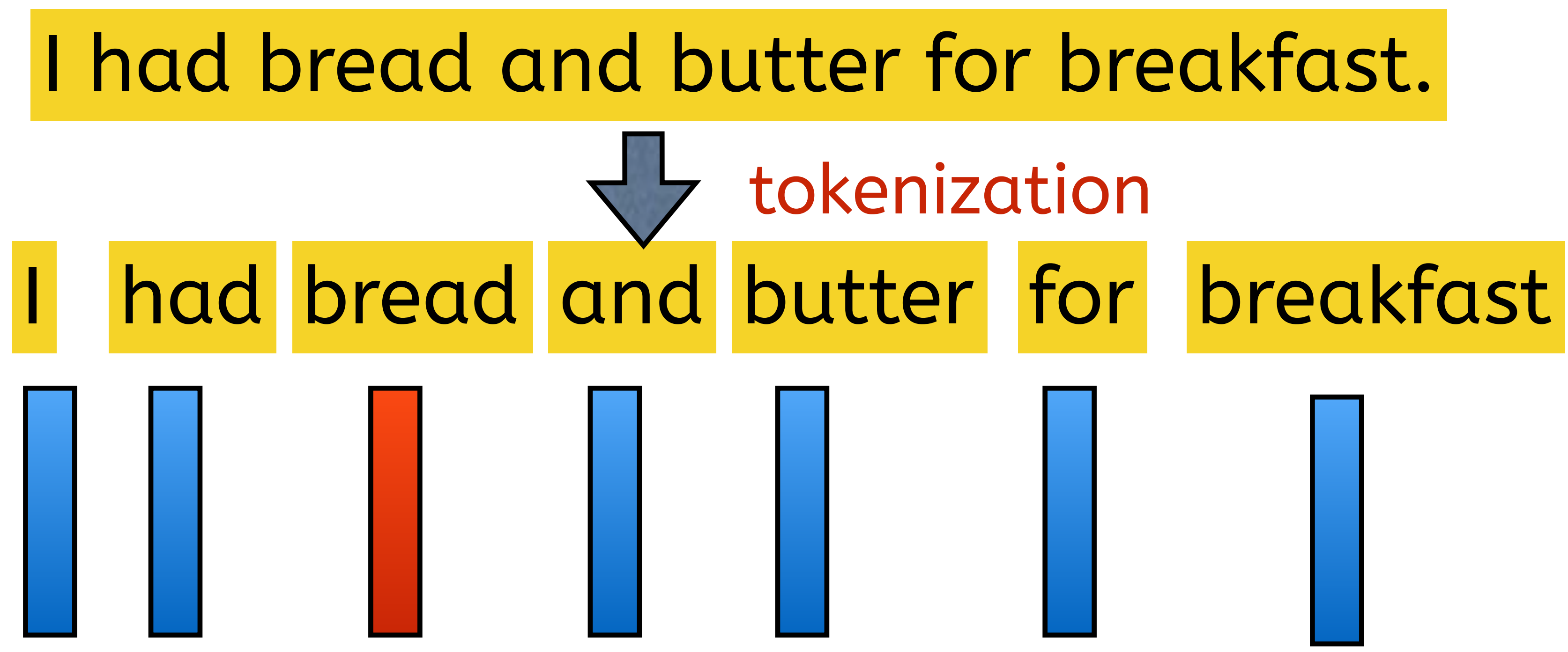
n-gram vs. skip-gram

- n consecutive words are defined as an n-gram
    - eg. I went to school
    - bi-grams = I+went, went+to, to+school
- Skip-grams on the other hand do not need to be consecutive
    - skip bi-grams: I+to, went+school

skip-gram model

I had bread and butter for breakfast.

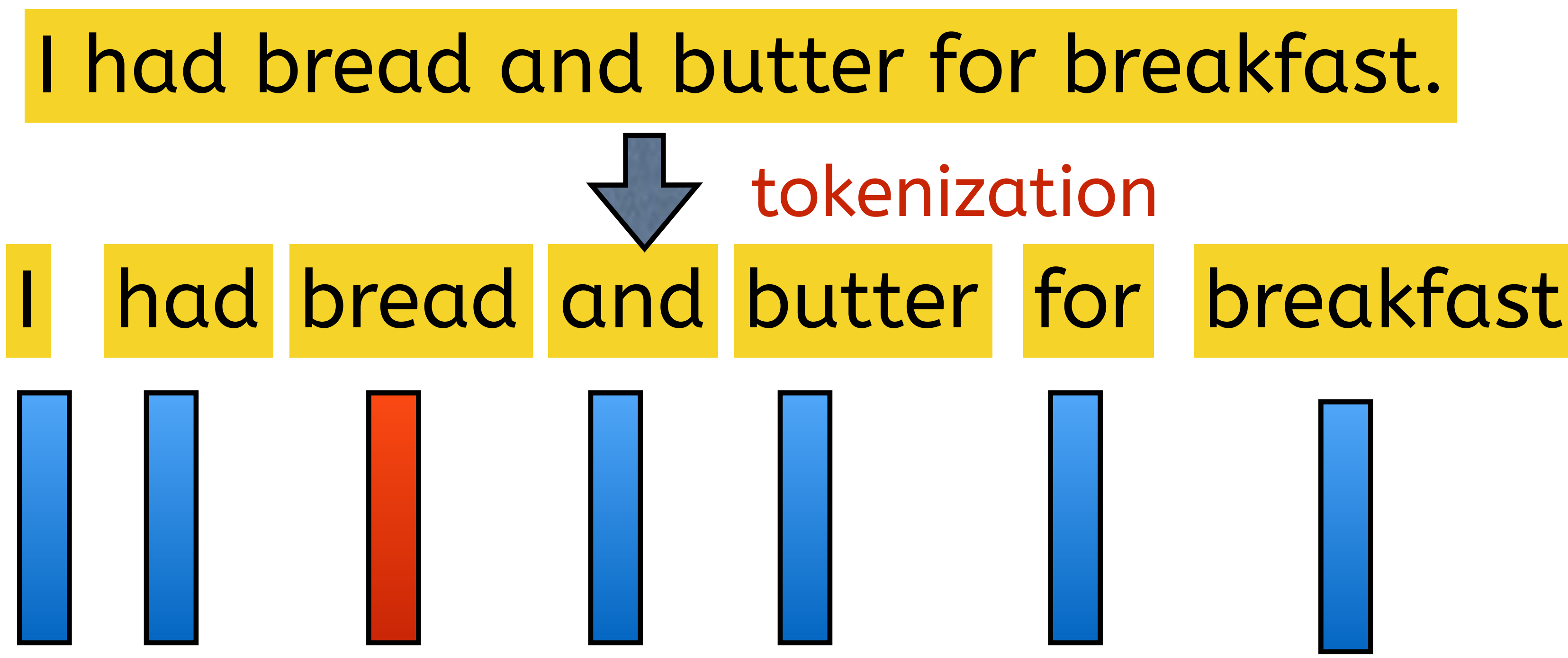
skip-gram model



Each word is assigned with two d-dimensional (random) vectors

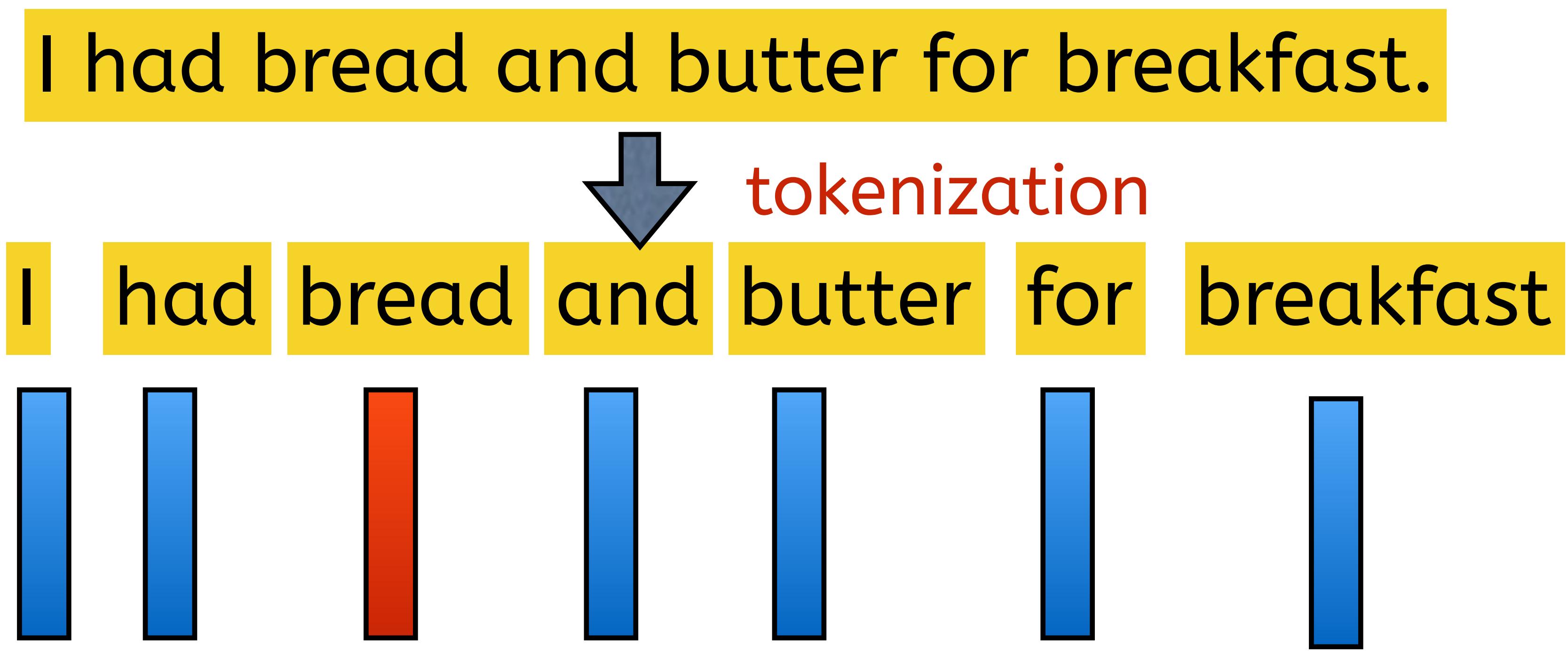
The word that we are interested in learning a semantic representation for has the red vector (target word), and the words that appear in its context are shown in blue vectors (context vectors).

skip-gram model



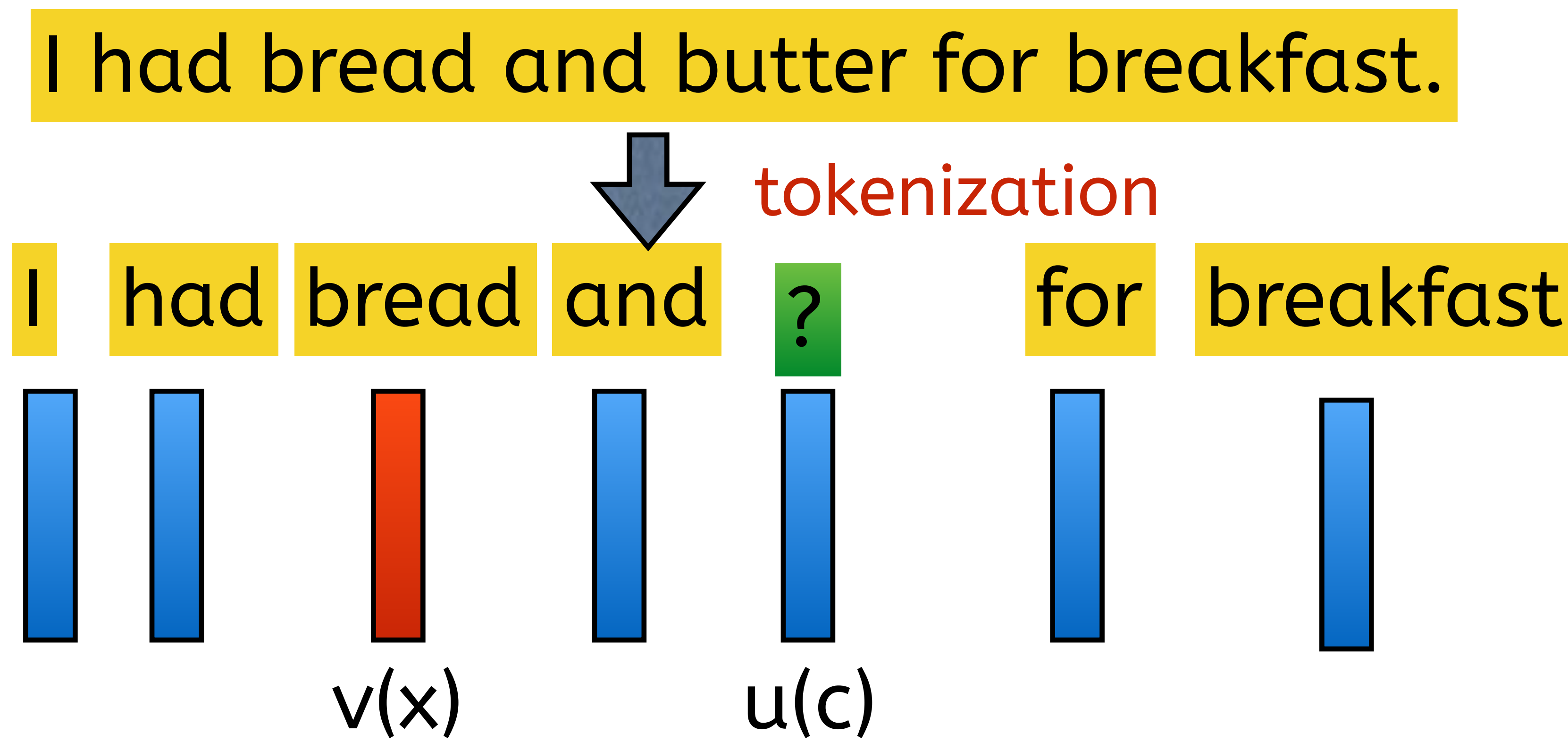
Let us consider the problem of predicting whether the word “butter” appear in the context of “bread”.

skip-gram model



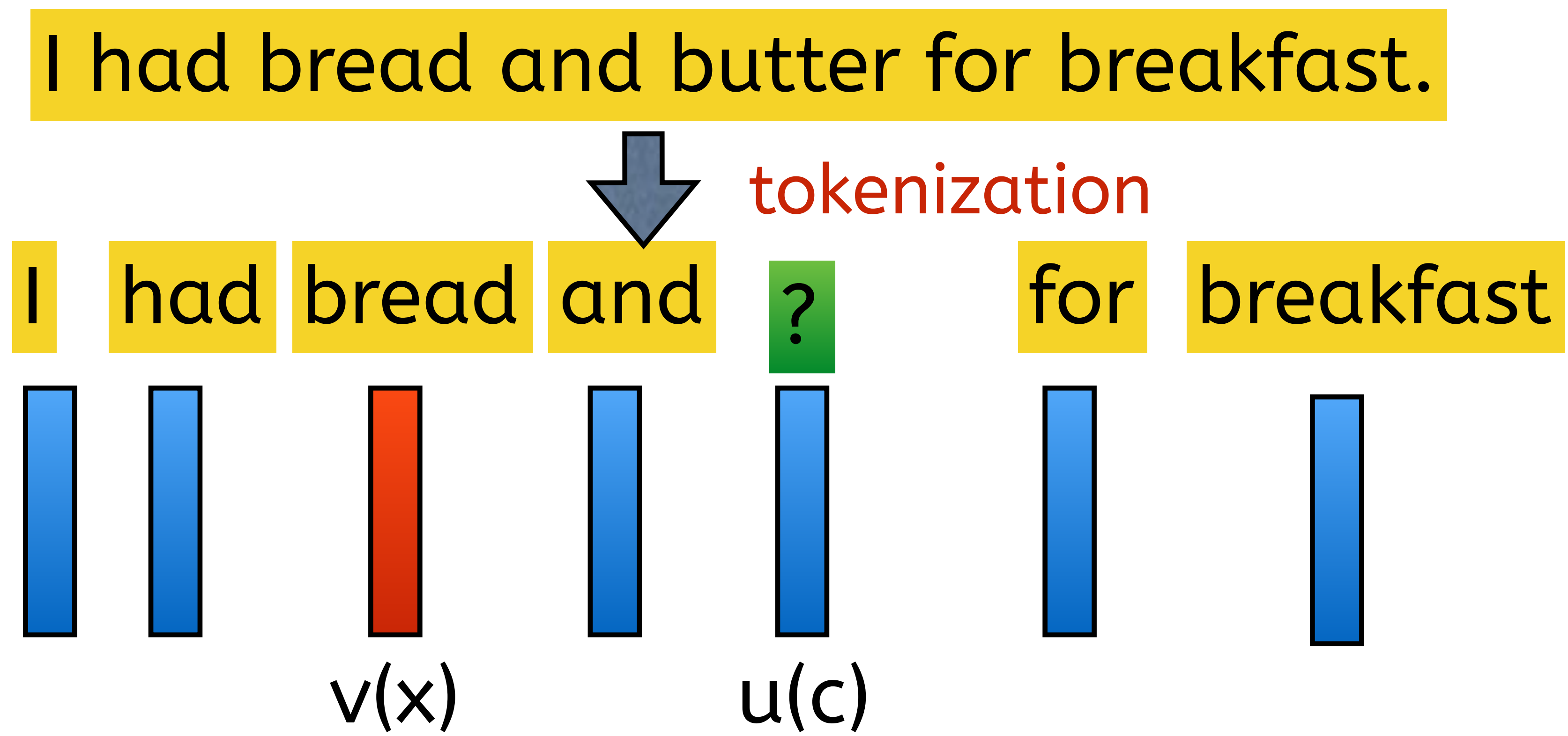
Let us consider the problem of predicting whether the word “butter” appears in the context of “bread”.

skip-gram model



(x=bread, c=butter) is more natural than (x=bread, c’=cake) in English. Can we learn vectors v(x), u(c), and u(c’) that encode this knowledge?

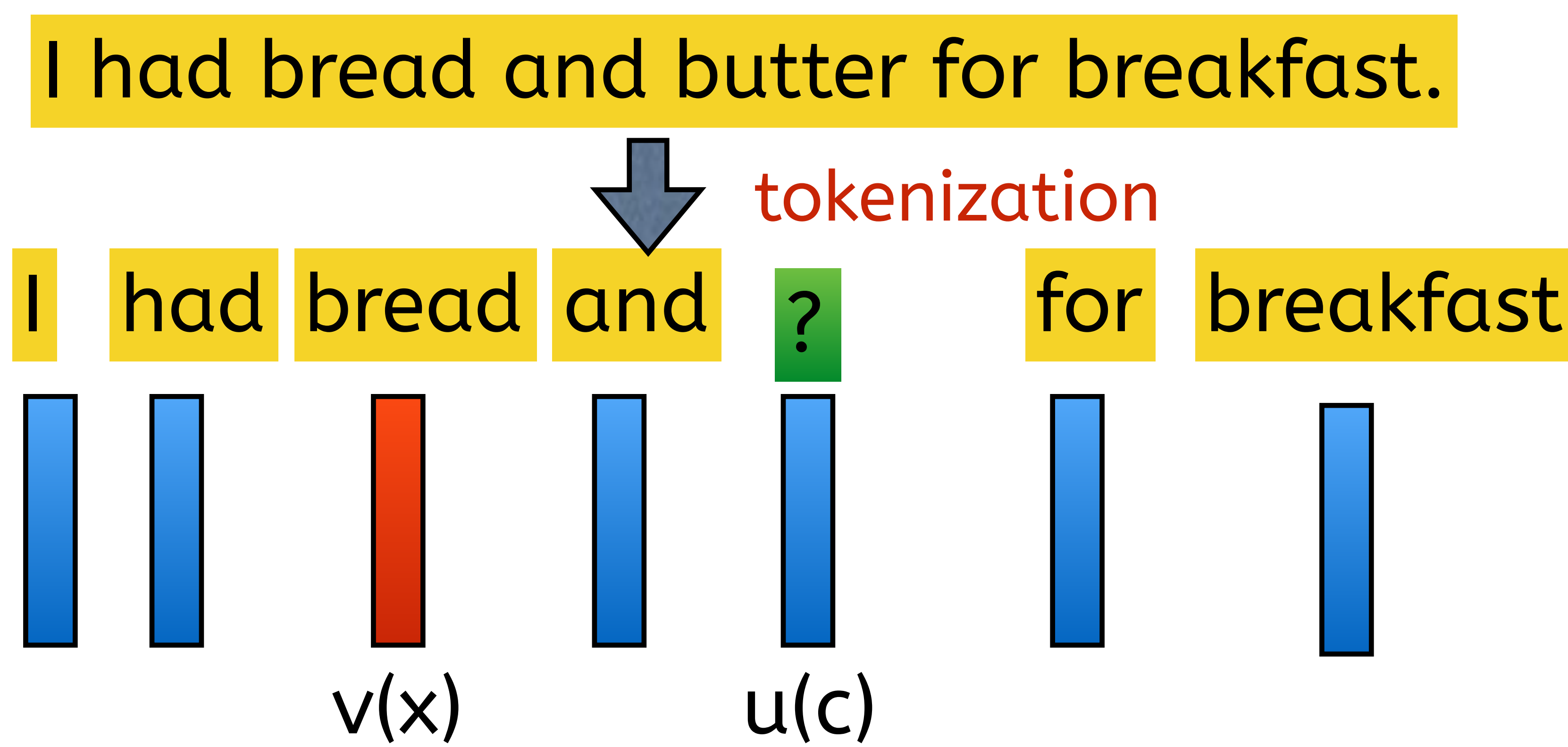
skip-gram model



Solution 1

Let us model the likelihood of a word c co-occurring in the same context with a word x by,
$\text{score}(x,c) = v(x)^T u(c)$

skip-gram model



Solution 1

However, this score is in the range (-∞, +∞), and is not a normalized score. Can we do better?

Log bi-linear model

probability of observing c in the context of x

co-occurrences

$$p(c|x) = \frac{\exp(\boldsymbol{v}(x)^\top \boldsymbol{u}(c))}{\sum_{c' \in \mathcal{V}} \exp(\boldsymbol{v}(x)^\top \boldsymbol{u}(c'))}$$

Normalize by dividing from the sum taken over all words in the vocabulary.

Training is similar to that of logistic regression. If we fix one of u or v, then the function becomes convex in the other. It is similar to solving alternative logistic regression problems.

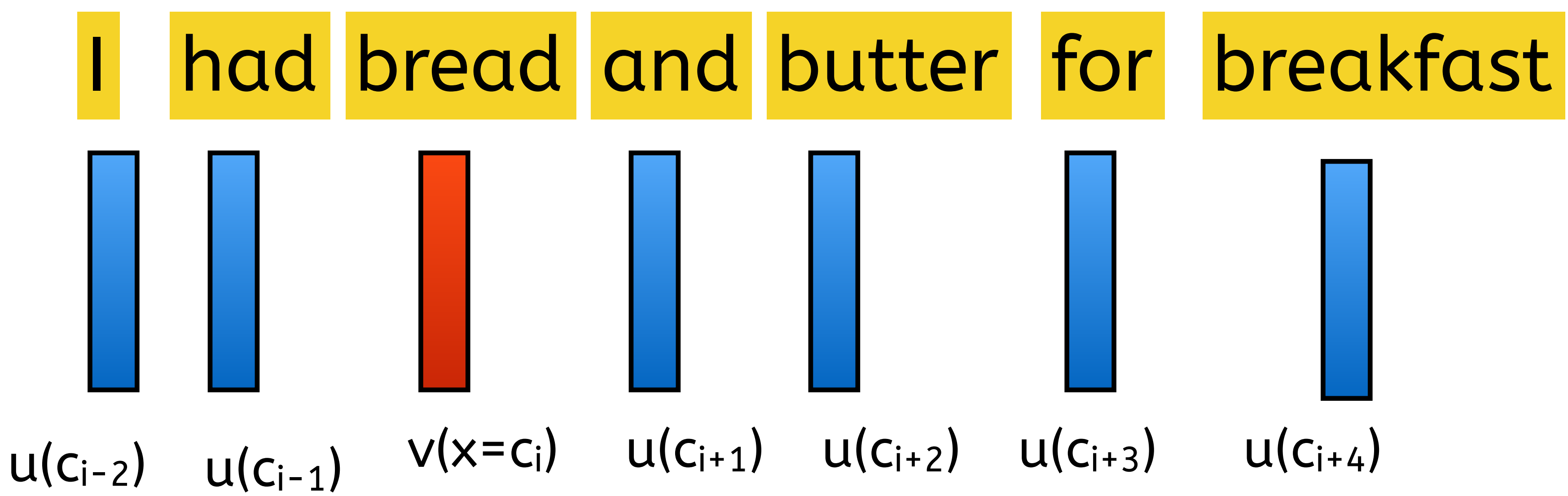
[Mnih+Hinton ICML'07]

What do you mean by bi-linear?

- If a two variable function $f(x,y)$ is linear in each of the variables (when the other variable is fixed), then it is called a bi-linear function.
- Definition of a linear function (a, b are constants)
    - $f(ax + b) = af(x) + f(b)$
- bi-linear function
    - $f(ax+b,y) = af(x,y) + f(b,y)$
- Example of a bi-linear function
    - $f(x,y) = x + y + xy$
- Note that a bi-linear function does not need to be "simultaneously" linear in both arguments.
- After taking the log the function becomes linear = log-linear
- After taking the log the function becomes bi-linear = log-bilinear

Continuos Bag-of-Words Model

- Reverse of the skip-gram model.
- Predict the target word conditioned on the ALL context words



If we limit the context to the two words before and after the target word in a sentence

$p(x|c_{i-1}, c_{i-2}, c_{i+1}, c_{i+2})$

p(bread|I,had,and,butter)

Although CBOW conditions upon all the surrounding contexts, and hence more accurate model than skip-gram, it requires more data to learn in practice, and empirically shows lower performance.

Practical considerations

- The denominator of the log-bilinear form computes the sum over all the words in the vocabulary, which is computationally expensive.
- Techniques for reducing this complexity
  - negative sampling
  - hierarchical softmax

Negative sampling

softmax function

product of logistics

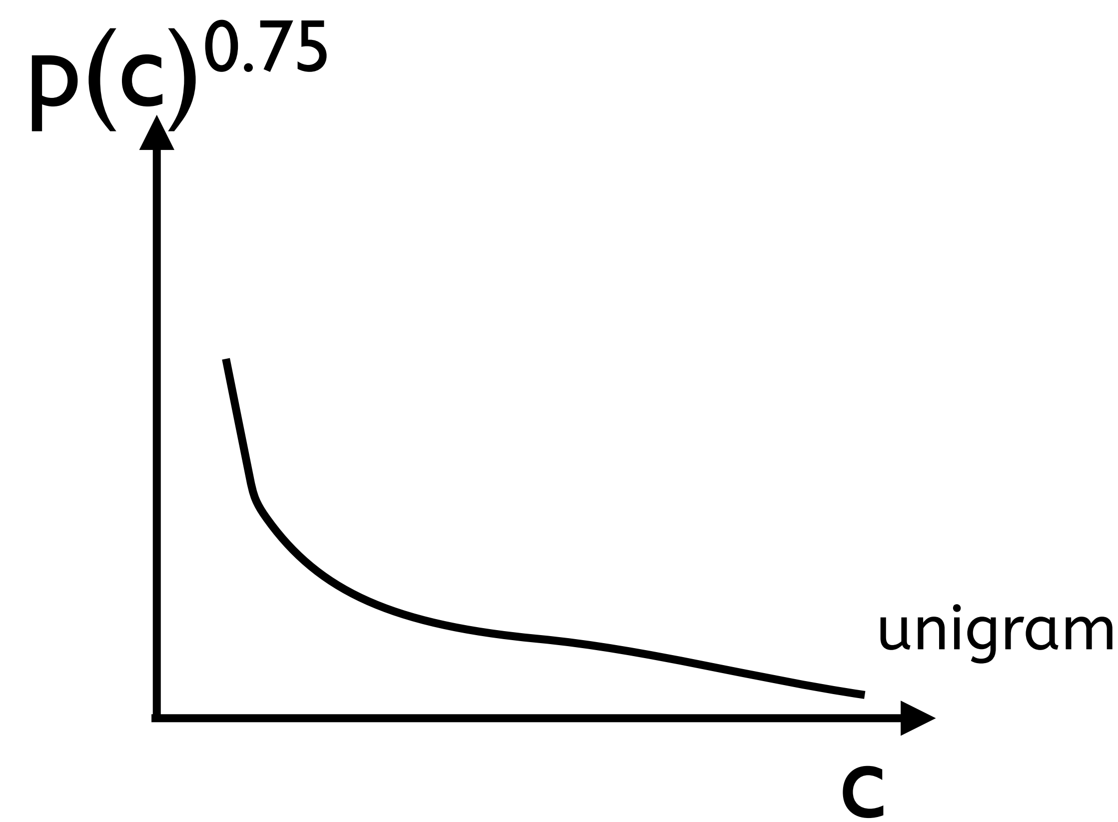
$$p(c|x) = \frac{\exp(\boldsymbol{v}(x)^\top \boldsymbol{u}(c))}{\sum_{c' \in \mathcal{V}} \exp(\boldsymbol{v}(x)^\top \boldsymbol{u}(c'))} \approx \sigma(\boldsymbol{v}(x)^\top \boldsymbol{u}(c)) \prod_{c' \in \mathcal{S}(x)} \sigma(-\boldsymbol{v}(x)^\top \boldsymbol{u}(c'))$$

positive examples

sampled negatives

Noise contrastive estimation
[Mnih+Hinton ICML'12]

This can be considered as training one-vs-rest classifiers

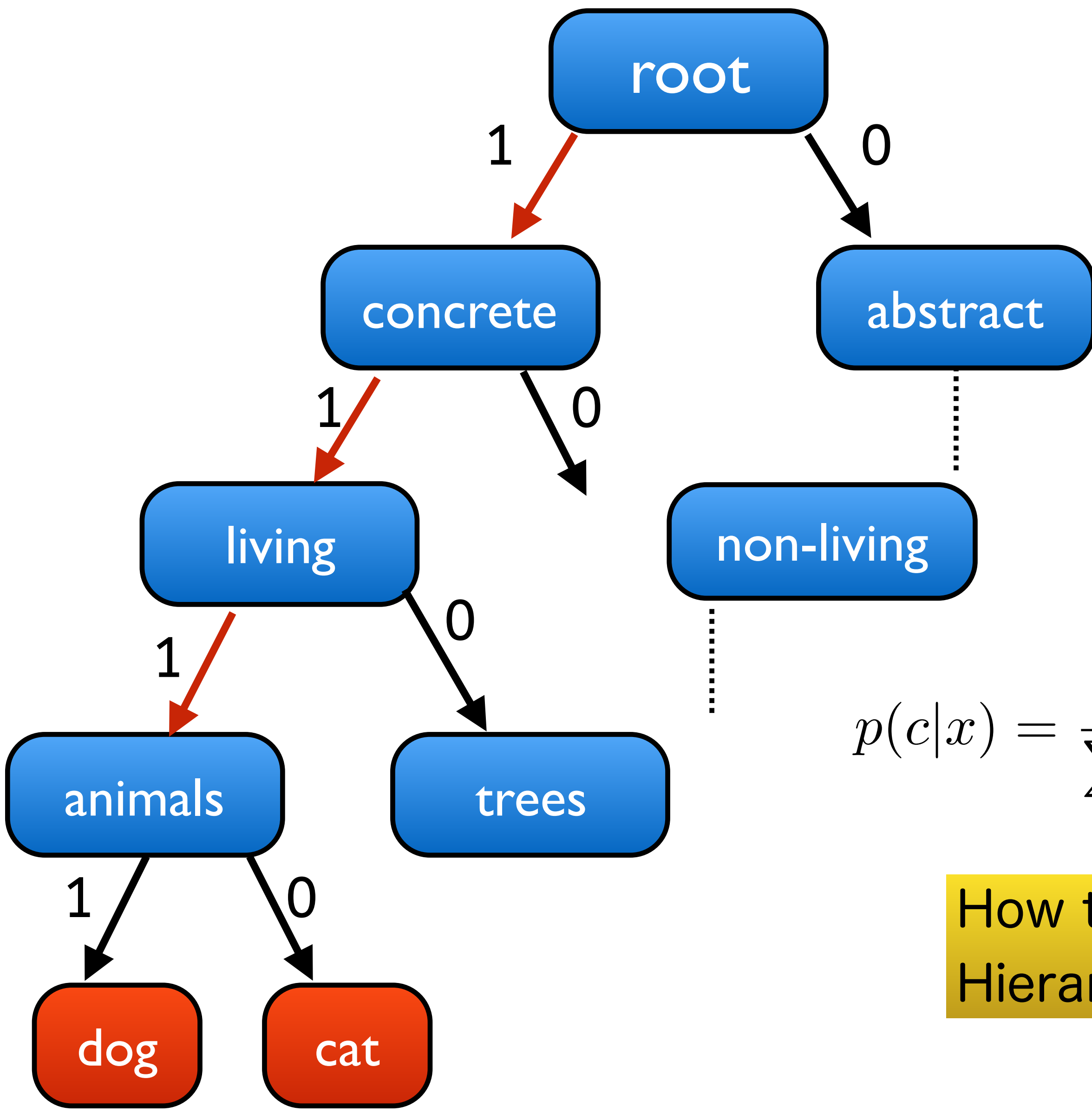


How to sample negative examples?

p(unigram)$^{0.75}$

Hierarchical Softmax

- Instead of considering individual words, we can consider classes of words, thereby reducing the number of terms under the summation.
- We consider a binary tree of classes, and estimate the score at each node using a logistic sigmoid.



p(cat)=p(root)p(concrete|root)p(living|concrete)x
x p(animal|living)p(cat|animals)

Path(cat)={(root,1), (concrete,1), (living,1), (animals,1)}

$$p(c|x) = \frac{\exp(\boldsymbol{v}(x)^\top \boldsymbol{u}(c))}{\sum_{c' \in \mathcal{V}} \exp(\boldsymbol{v}(x)^\top \boldsymbol{u}(c'))} \approx \prod_{(y_i, z_i) \in \mathrm{Path}(c)'} \sigma(z_i \boldsymbol{v}(x)^\top \boldsymbol{u}(y_i))$$

How to create the tree?
Hierarchical clustering, WordNet, Huffman Tree

How to evaluate the learnt word representations?

- We cannot just look at the learnt high dimensional vectors and decide whether they are correct.
- no gold standard for semantic representations
- We must apply the learnt representation in some other task and evaluate the increase/decrease in performance in that task.
- Extrinsic evaluation
  - Semantic similarity measurement
  - Word analogy detection

Word analogies

- Which of the following is analogous to the relation between “water” and “pipe”.
    A. (electricity, wire)
    B. (ice, steam)
    C. (gasoline, pipe)
    D. (sushi, California roll)

Semantic similarity measurement

- Similarity ratings are provided by a group of human annotators (linguists, Amazon mechanical turk).
- The average of all similarity scores per each word pair is considered as the human similarity rating for that word pair.
- We can measure the correlation between human ratings and ratings produced by an algorithm (that uses word representations) to evaluate the accuracy of the learnt word representations
    - Pearson/Spearman correlation coefficients can be used for this purpose
- Higher the correlation the better

Solving word analogies

- If "man" is to "king", the "woman" us to ?
- Procedure
    - Let, x = v(king) - v(man) + v(woman)
    - Compute the cosine similarity between x and all the other words in the vocabulary, and select the most similar word as the answer.