Natural Language Processing

Lecture 12: Lexical Semantics (part I) -
Word Representations and Word Embeddings.

11/30/2020

COMS W4705
Yassine Benajiba

Jabberwocky

- Can you identify what the words in this poem mean?

Beware the jabberwock, my son
the jaws that bite, the claws that catch!

Beware the jubjub bird, and
the frumious bandersnatch!

"Jabberwocky", Lewis Carroll, 1871

Semantic Similarity and Relatedness

- We can often tell that two words are similar or related, even if they aren't exact synonyms.
- "***fast***" is similar to "***rapid***" and ***"speed"***
- "***tall***" is similar to "***high***" and "***height***"
- Question answering:
  - Q: *"How **tall** is Mt. Everest?"*
  - Candidate A: *"The official **height** of Mount Everest is 29029 feet"*

Relatedness

- "***cat***" is more similar to "***dog***" than to "***table***"
- "***table***" is more similar to "***chair***" than to "***dog***"
- "***run***" is more similar to "***fly***" than to "***think***".
- **"cat"** is more similar to "**meow**" than to **"bark".**

Single Word Representation: One-Hot Vector

$$
\begin{array}{ccc}
0 & a & \\
 & \vdots & \\
 & fish & \\
 & \vdots & \\
 & \vdots & \\
 & \vdots & \\
\mathbf{|V|} & zythum &
\end{array}
\begin{bmatrix} \mathbf{0} \\ \vdots \\ \mathbf{1} \\ \vdots \\ \vdots \\ \vdots \\ \mathbf{0} \end{bmatrix}
$$

What about unseen words?

Unknown Words

*A bottle of **tesgüino** is on the table.*
*Everybody likes **tesgüino.***
***Tesgüino** makes you drunk.*
*We make **tesgüino** out of corn.*

Example from Nida, 1975.

- Can you figure out from context what **tesgüino** means?
  - Some kind of alcoholic beverage, maybe beer or whisky.
- Intuition: Two words should be similar if they have similar typical word **contexts**.

How would you represent context?

Distributional Hypothesis

- Wittgenstein ("Philosophical Investigations):

"the meaning of a word is in its use in the language"

- Zelig Harris (1954):

“oculist and eye-doctor … occur in almost the same environments”

“If A and B have almost identical environments we say that they are synonyms.”

- J.R. Firth (1957)

"you shall know a word by the company it keeps!"

Co-occurence Matrix

	⌼	⊞	⊛	⋔	⏊	☊
⌘	51	20	84	0	3	0
⌓	52	58	4	4	6	26
⊠	**115**	**89**	**10**	**42**	**33**	**17**
⊚	59	39	23	4	0	0
⁙	98	14	6	2	1	0
⁂	12	17	3	2	9	27
⬡	11	2	2	0	18	0

sim(⊠,⌘) = 0.770
sim(⊠,⁂) = 0.939
sim(⊠,⌓) = 0.961

- Numbers are co-occurence counts (how often the symbols appear together in context).
- Which symbol is most similar to ⊠?

What it really looks like

	get	see	use	hear	eat	kill
knife	51	20	84	0	3	0
cat	52	58	4	4	6	26
dog	**115**	**89**	**10**	**42**	**33**	**17**
boat	59	39	23	4	0	0
cup	98	14	6	2	1	0
pig	12	17	3	2	9	27
berry	11	2	2	0	18	0

Verb-Object counts

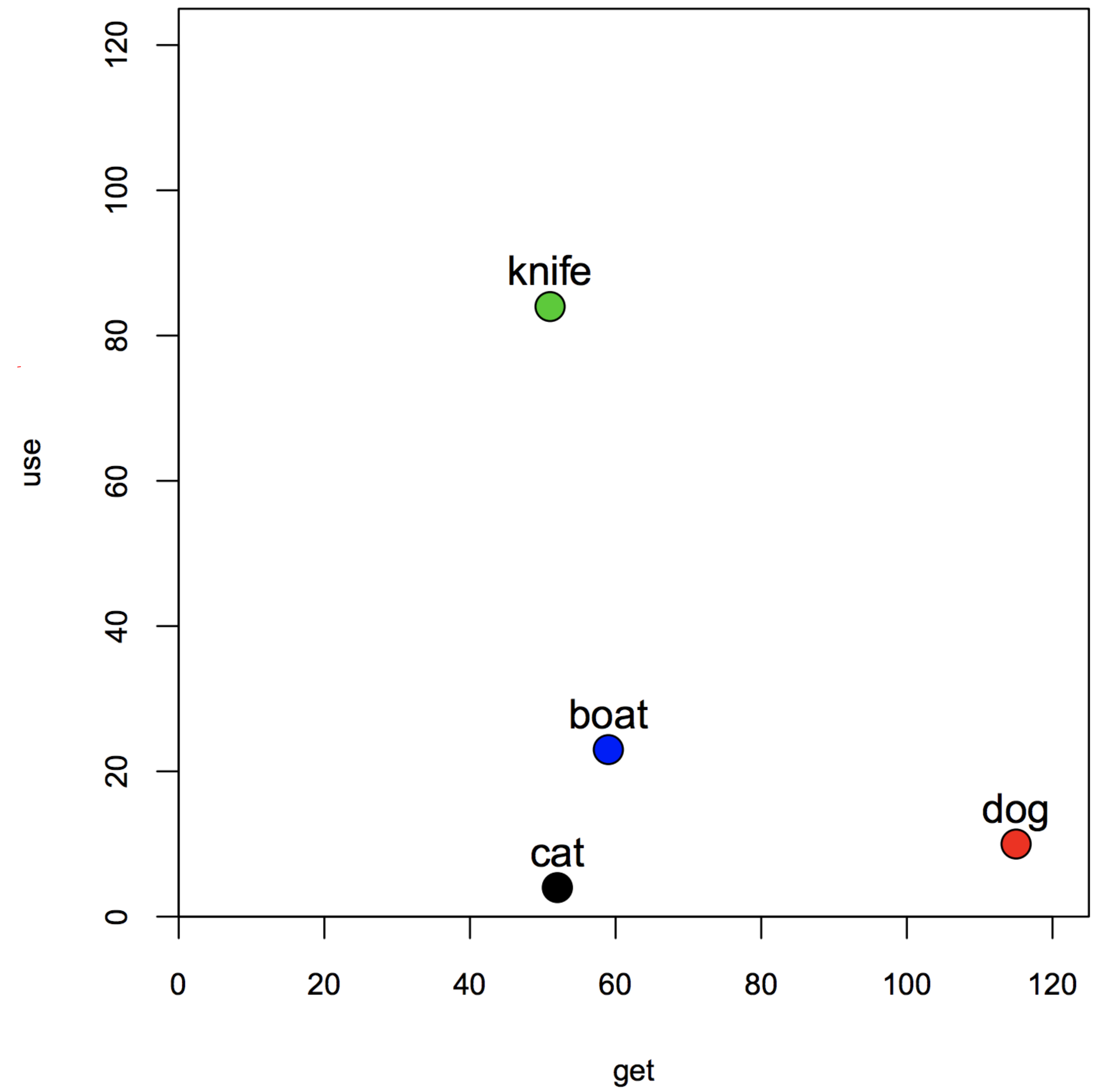
sim(*dog*,*knife*) = 0.770
sim(*dog*,*boat*) = 0.939
sim(*dog*,*cat*) = 0.961

- Row vector $\mathbf{x}_{\mathbf{dog}}$ describes usage of *dog* as a grammatical object in the corpus.
- Can be seen as coordinates in n-dimensional Euclidean space.

Geometric Interpretation

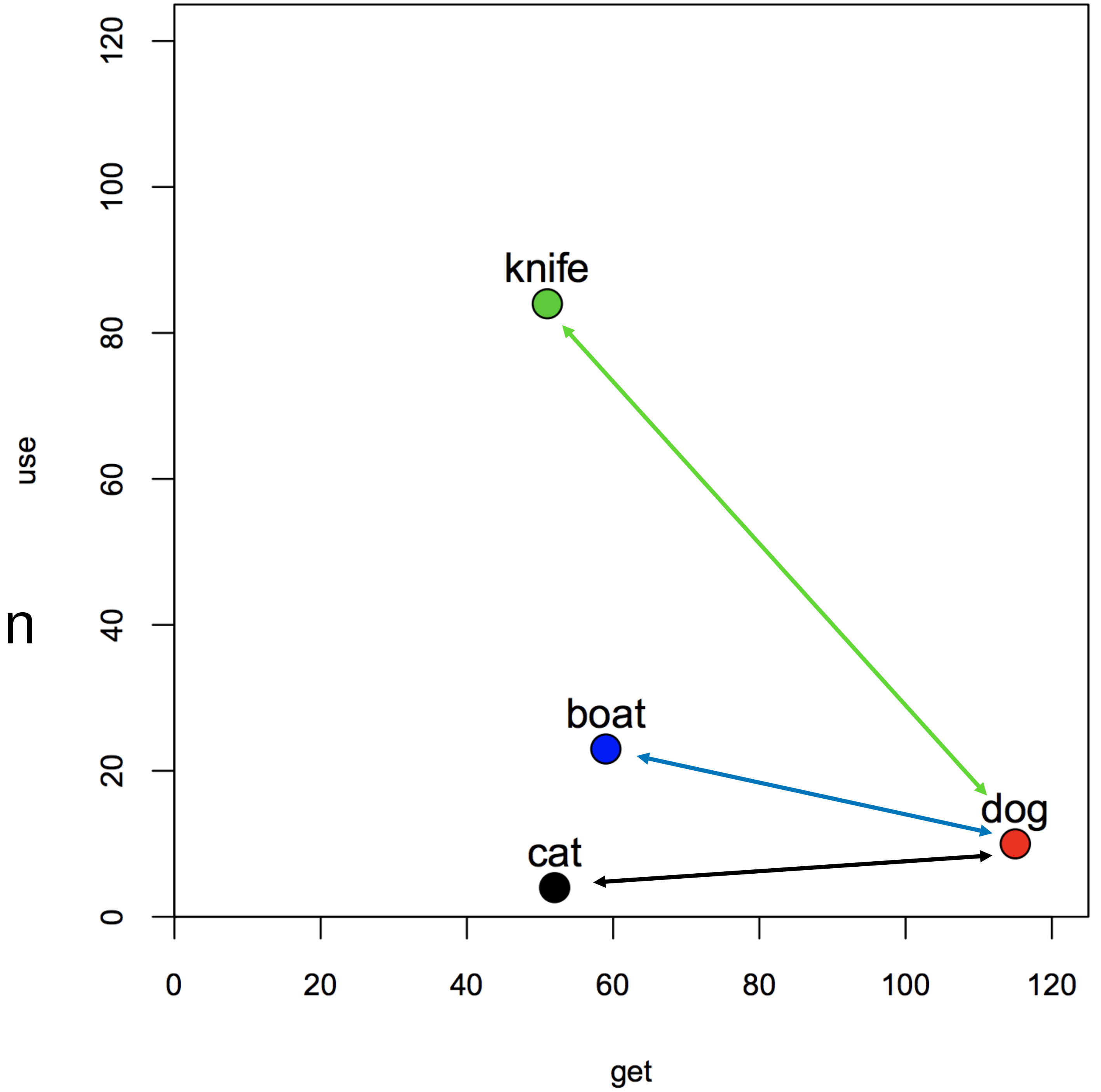
- Row vector $\mathbf{x}_{\mathbf{dog}}$ describes usage of *dog* in the corpus.
- Can be seen as coordinates in *n*-dimensional Euclidean space.
- Illustrated for two dimensions "get" and "use".

$$\mathbf{x}_{\mathbf{dog}} = \mathbf{(115,\ \ 10)}$$



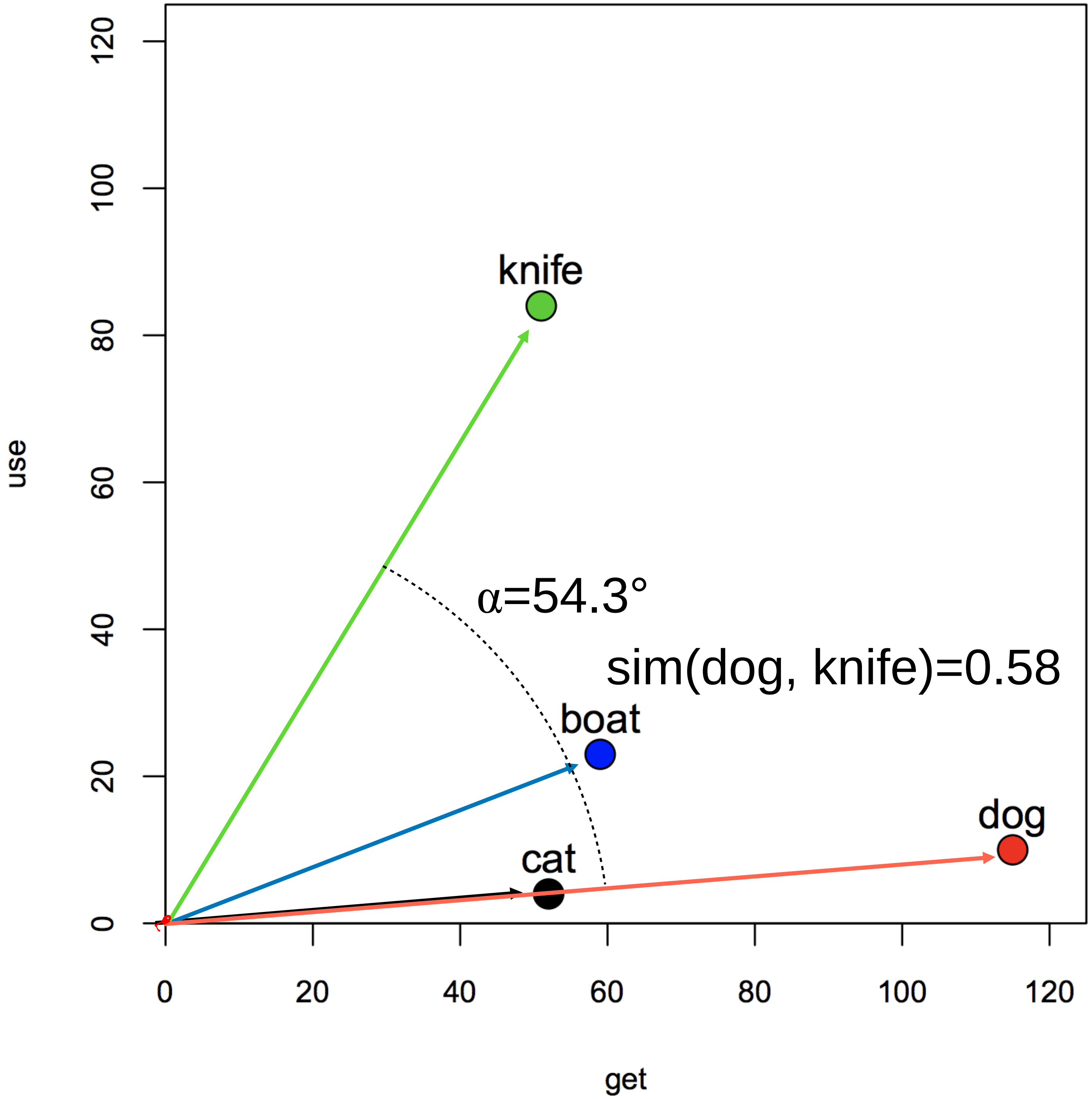
Geometric Interpretation

- How should we compute similarity?
  - First approach: Spatial distance between words.
  - (lower distance = higher similarity)
  - Potential problem: location depends on frequency of noun count(dog) ≈ 2.7 count(cat)



Geometric Interpretation

- How should we compute similarity?
  - Second approach:
    - Direction is more important than location.
    - Normalize "length" $||x_{dog}||$ of vector.
    - or use angle α as distance measure (or cos of these angles).



Cosine Similarity

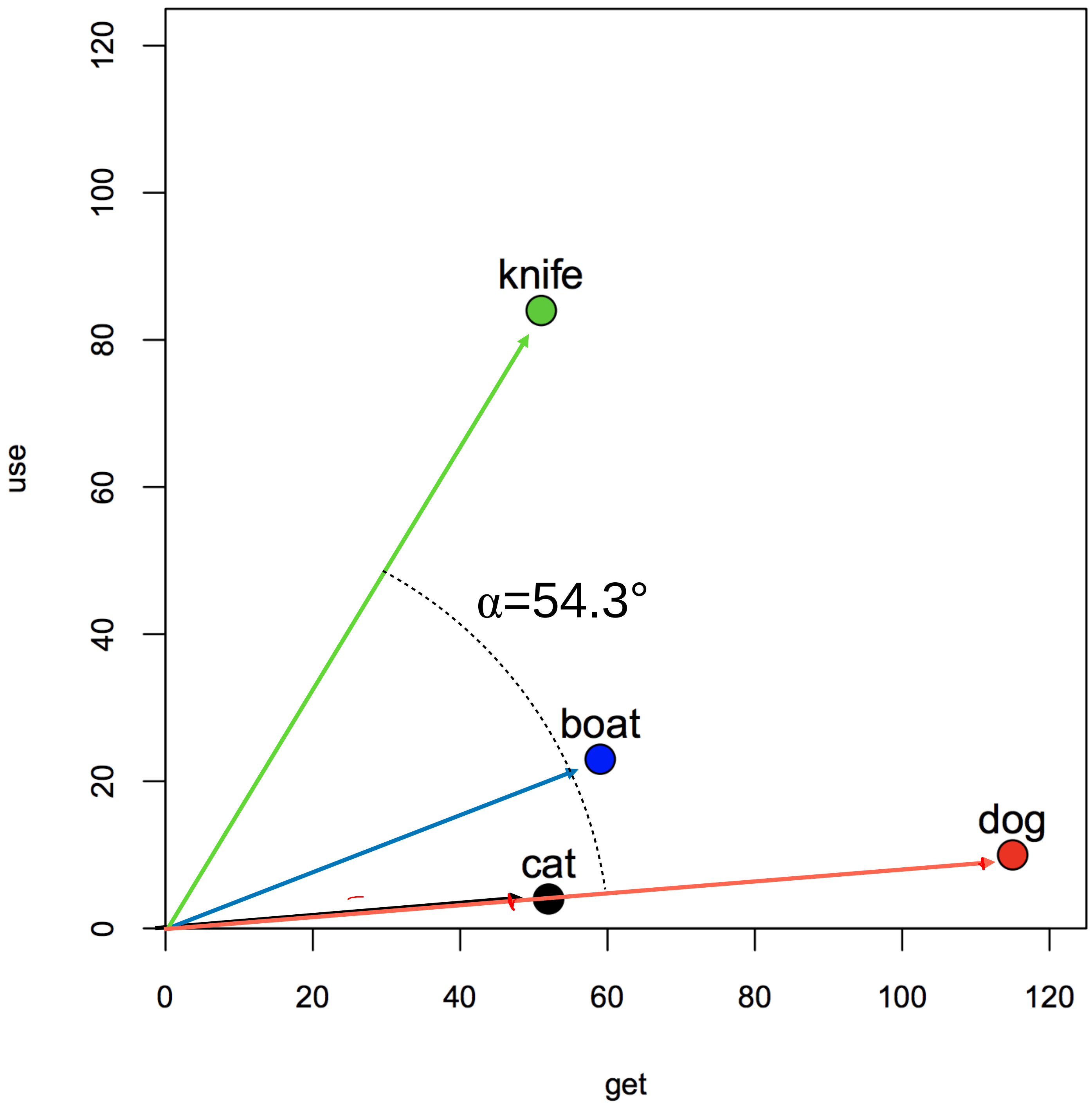
$$sim_{cos}(\mathbf{x},\mathbf{y}) = \frac{\mathbf{x}\cdot\mathbf{y}}{|\mathbf{x}|_2 \cdot |\mathbf{y}|_2} = \frac{\sum_i x_i \cdot y_i}{\sqrt{\sum_i x_i^2}\sqrt{\sum_i y_i^2}}$$

Colinear vectors (same direction):

$$sim_{cos}(\mathbf{x},\mathbf{y}) = 1$$

Orthogonal vectors
(90° angle, no shared attributes):

$$sim_{cos}(\mathbf{x},\mathbf{y}) = 0$$

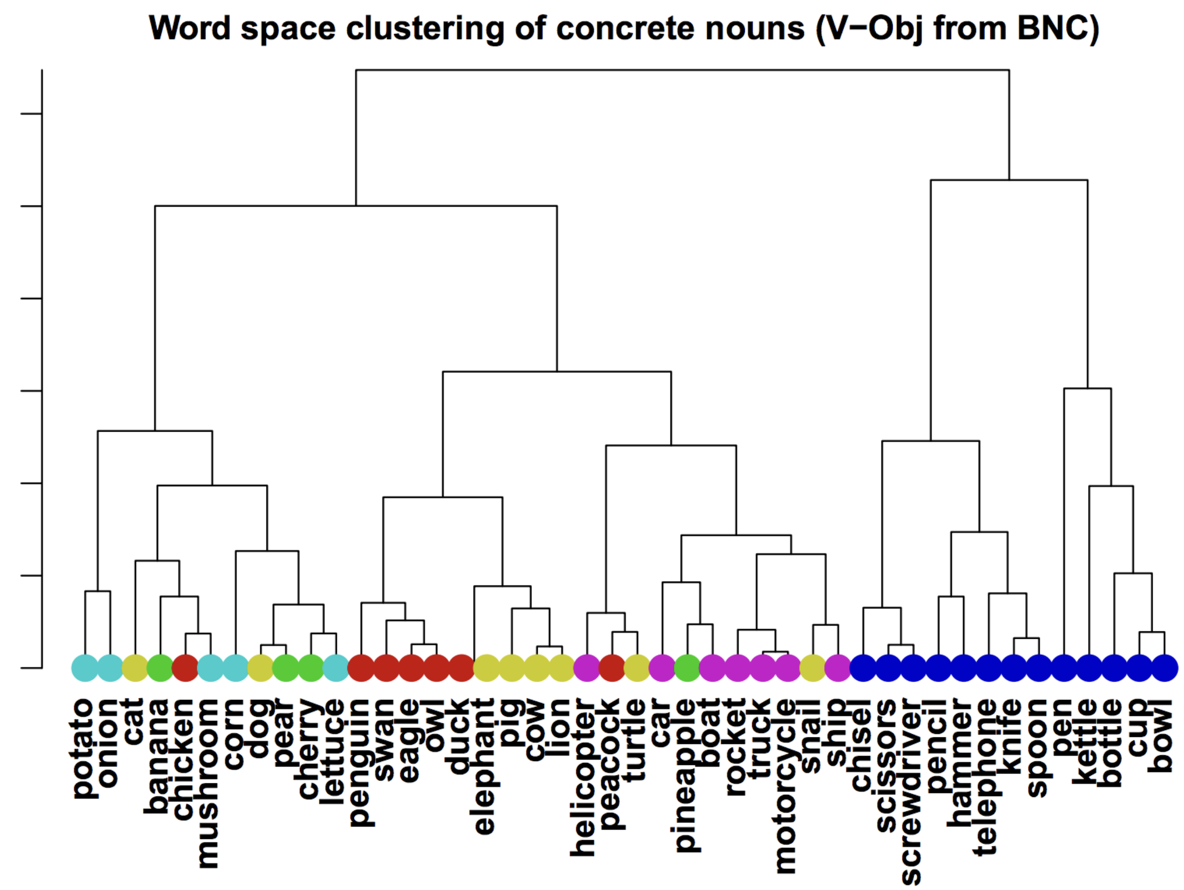


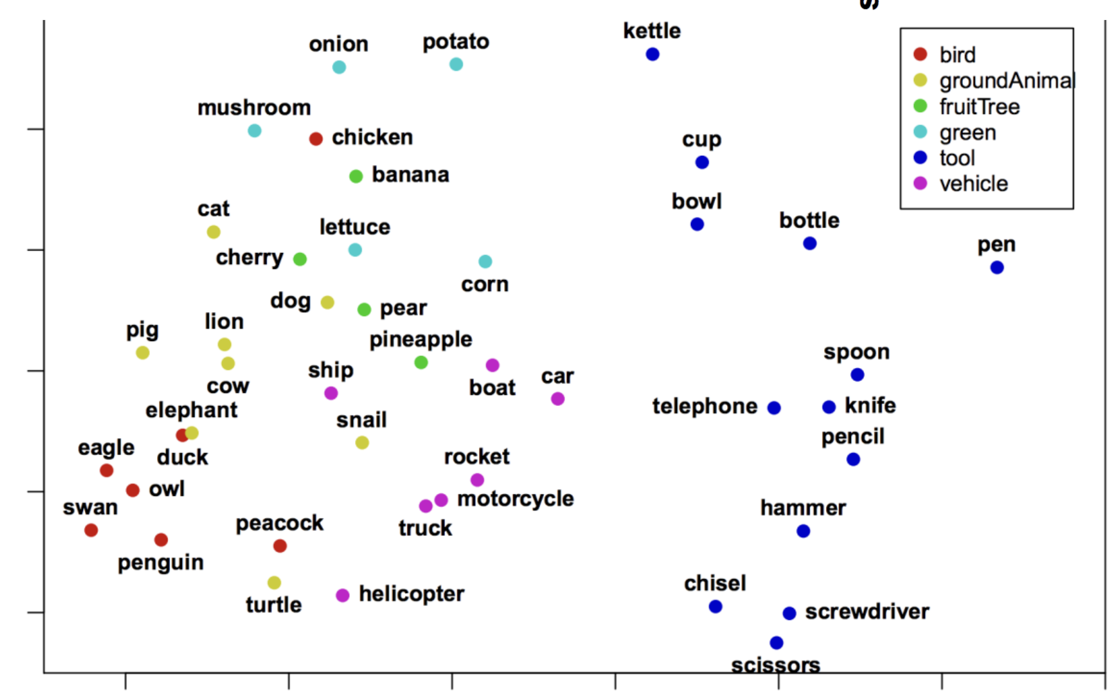
What to do with DSM similarities

- Most similar to **school:**
  country (49.3), church (52.1), hospital (53.1), house (54.4), hotel (55.1), industry (57.0), company (57.0), home (57.7), family (58.4), university (59.0), party (59.4), group (59.5), building (59.8), market (60.3), bank (60.4), business (60.9), area (61.4), department (61.6), club (62.7), town (63.3), library (63.3), room (63.6), service (64.4), police (64.7),...

Clustering and Semantic Maps

- Distributional Similarity/Distance can be used to
  - find nearest neighbors (similar words)
  - cluster related words into hierarchical categories.
  - construct semantic maps.





Variations of Distributional Semantic Models

- A Distributional Semantic Model (DSM) is any matrix M such that each row represents the distribution of a term x across contexts, together with a similarity measurement.

- The previous example shows one particular semantic space (frequency counts of Verb-object co-occurences).

- There are many different models we could choose.

- Different models might capture different "types" of similarity.

Dimensions of Distributional Semantic Models

1. Preprocessing, definition of "terms" (word form, lemmas, POS, ...).

2. Context definition:

   - Type of context (word, syntactic dependents (with or without relation labels labels), remove stop-words, etc.)

   - Size of context window.

3. Feature scaling / term weighting (association measures).

4. Normalization of rows / columns.

5. Dimensionality reduction.

6. Similarity measure.

Effect of context size

Nearest neighbors of *dog*

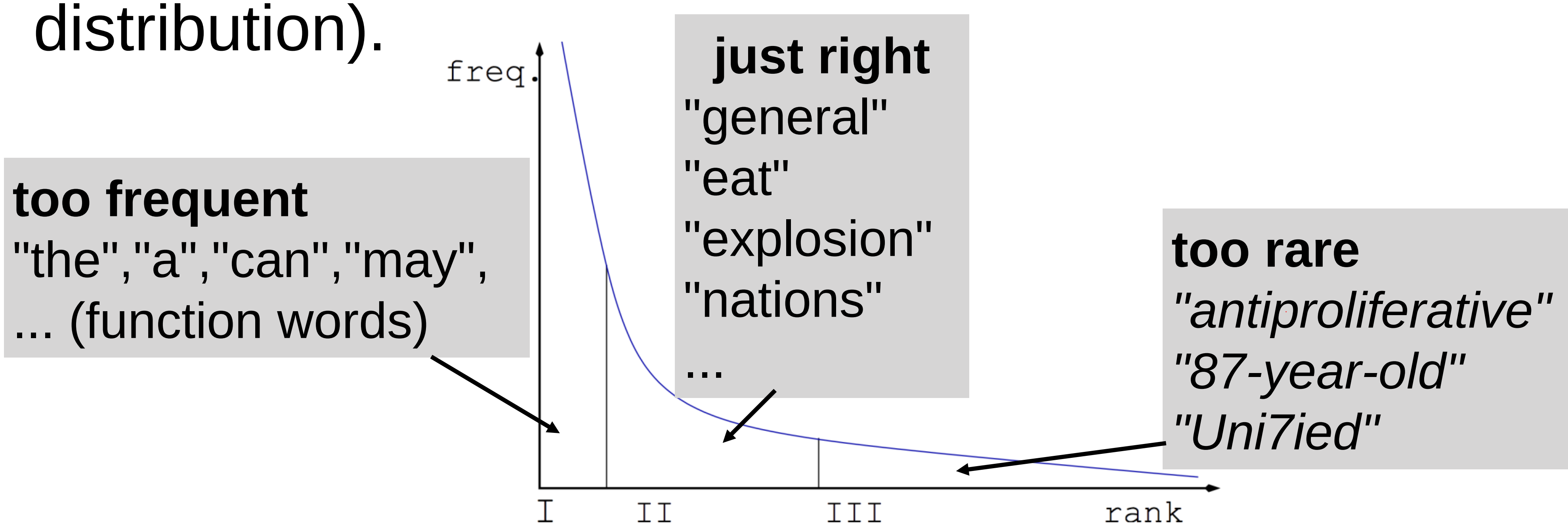
2-word window:
cat, horse, fox, pet, rabbit, pig, animal, mongrel, sheep,pigeon

30-word window:
kennel, puppy, pet, terrier, rottweiler, canine, cat, to bark, Alsatian

Term Weighting

- Problem: Not all context terms are equally relevant to characterize the meaning of a word.

  - Some appear too often, some are too rare (Zipfian distribution).



- One solution: TF*IDF (term frequency * inverse-document frequency)

TF*IDF

- Originates in document retrieval (find document relevant to a keyword). For DSM: 'document' = target word *d*.

- Term frequency: How often does the term *t* appear in the context window of the target word?

$$tf_{t,d} = count(d,t)$$

- Inverse document frequency: For how many words does *t* appear in the context window

$$idf_{t,D} = log\frac{|D|}{|\{d \in D, count(d,t) > 0\}|}$$

- TF*IDF:

$$tf_{t,d} \cdot idf_{t,D}$$

Sparse vs. Dense Vectors

- Full co-occurrence matrix is very big and contains a lot of 0 entries.
  - Potentially inconvenient to store. Slow computation.
  - Synonyms may still contain orthogonal dimensions, which are irrelevant.
- **Word embeddings** are representations of words in a low-dimensional, dense vector space. There are two main approaches:
  - Use matrix decomposition on co-occurence matrix, for example Singular Value Decomposition (SVD).
  - **Learn embeddings using neural networks. Minimal feature-engineering required.**

Learning Word Embeddings with Neural Networks

- The neural network should capture the relationship between a word and its context.

- Two models:
  (Word2Vec, Mikolov et al. 2013)
  - **Skip-Gram model**: Input is a single word.
    Predict a probability for each context word.

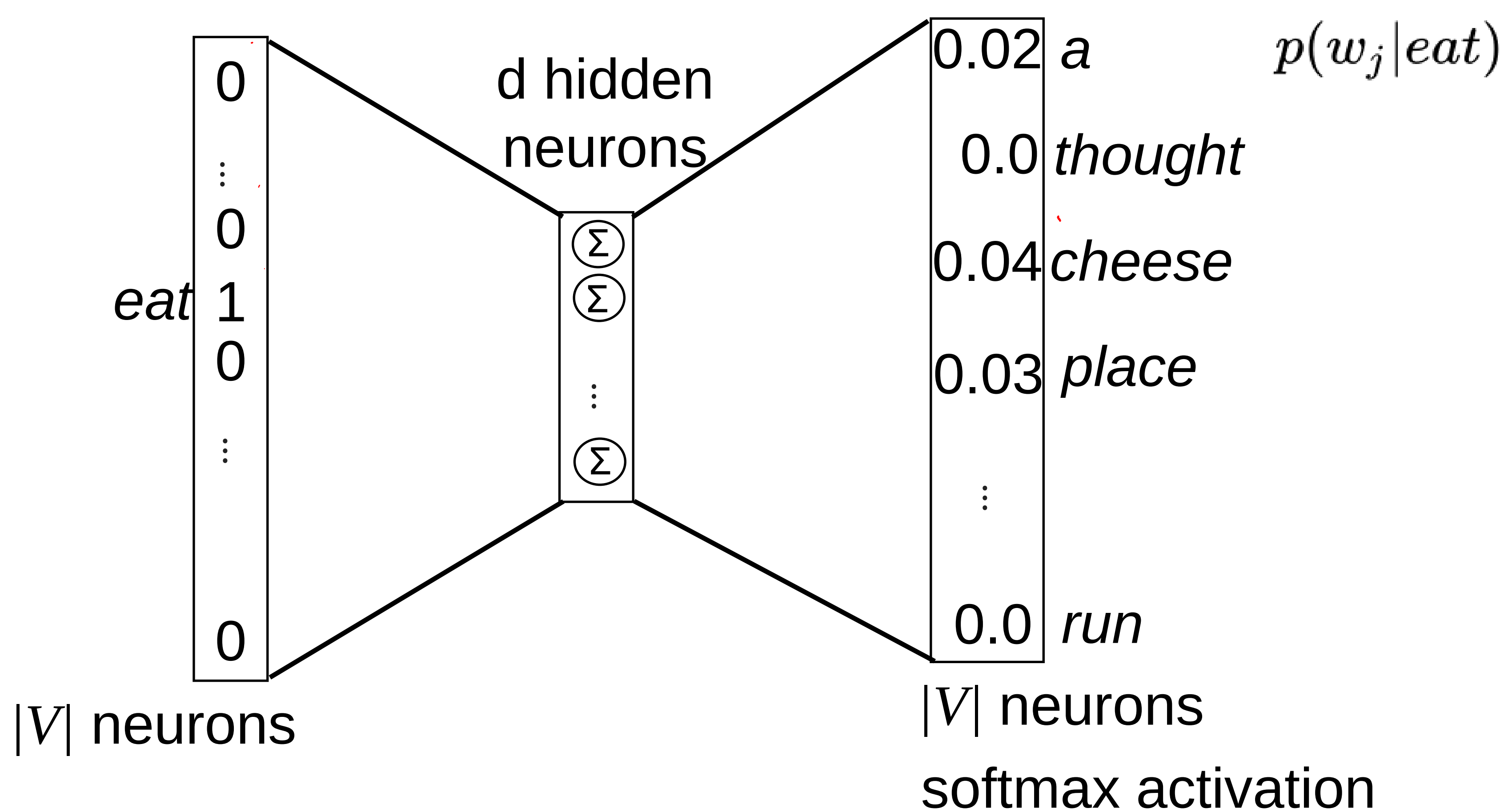
  - **Continuous bag-of-words (BOW):**
    Input is a representation of the context window.
    Predict a probability for each target word.

- Inspired by Neural Language Models **(Bengio et al. 2003)**

Skip-Gram Model

- Input:
  A single word in one-hot representation.

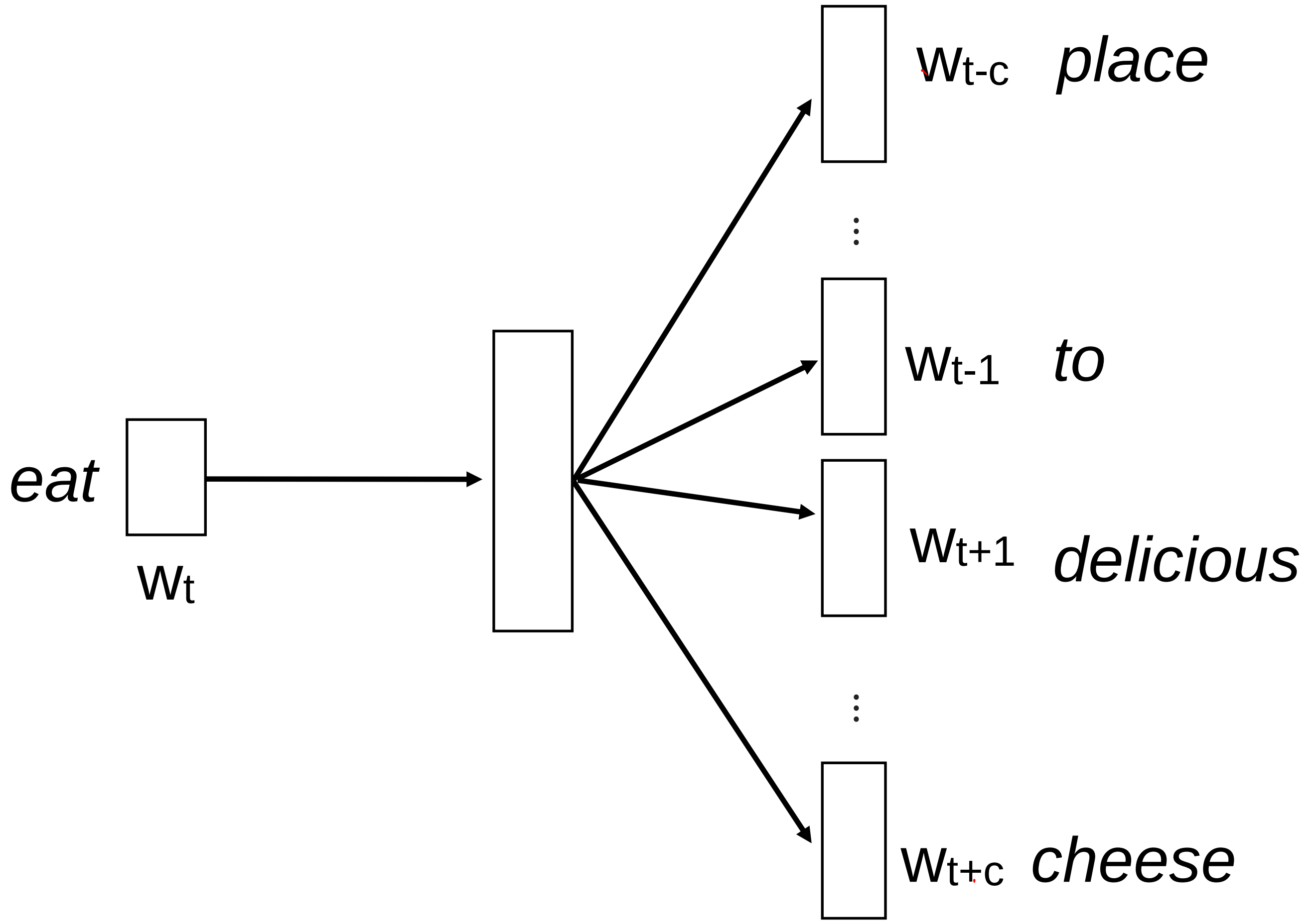
- Output: probability to see any single word as a context word.



- Softmax function normalizes the activation of the output neurons to sum up to 1.0.

Skip-Gram Model

- Compute error with respect to each context word.



...a place to ***eat*** *delicious cheese .*

(eat, place)
(eat, to)
(eat, delicious)
(eat,cheese)

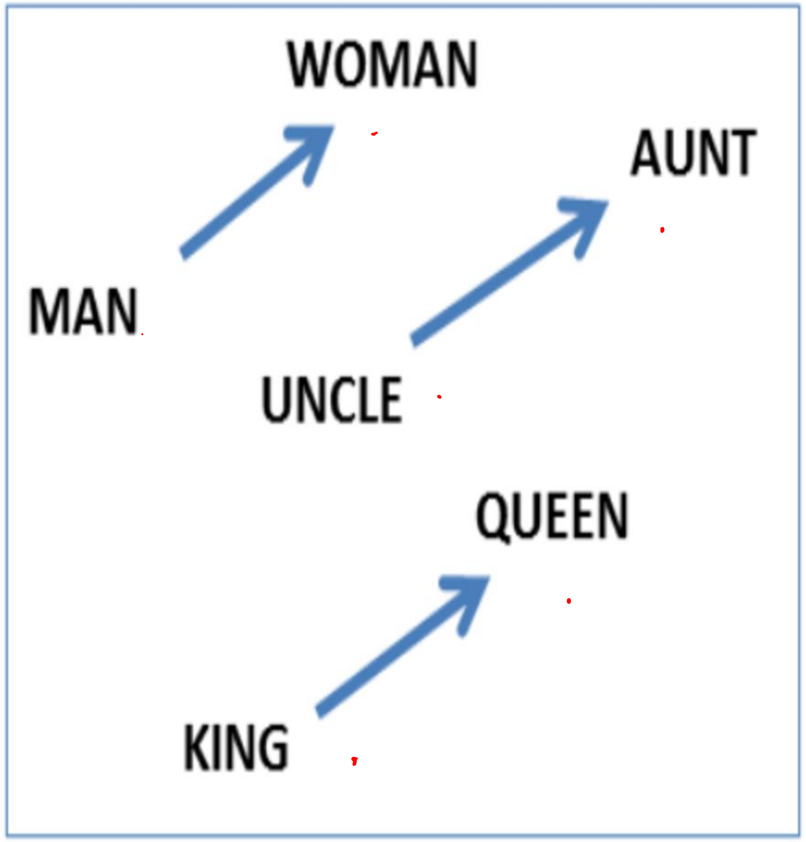
- Combine errors for each word, then use combined error to update weights using back-propagation.

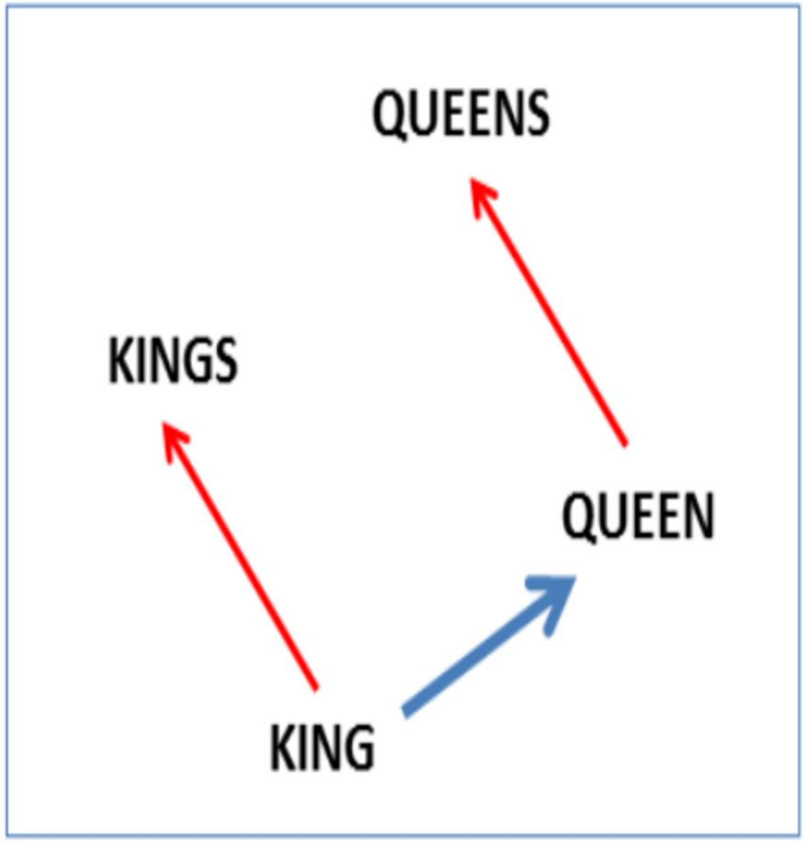
$$error = -\sum_{i=-c, i \neq 0}^{c} \log p(w_{t+i} | w_t)$$

Embeddings are Magic

(Mikolov 2016)

vector('king') - vector('man') + vector('woman') ≈ vector('queen')





Application: Word Pair Relationships

Table 8: *Examples of the word pair relationships, using the best word vectors from Table 4 (Skip-gram model trained on 783M words with 300 dimensionality).*

Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

Using Word Embeddings

- Word2Vec:
  - https://code.google.com/archive/p/word2vec/
- GloVe: Global Vectors for Word Representation
  - https://nlp.stanford.edu/projects/glove/
- Can either use pre-trained word embeddings or train them on a large corpus.

Acknowledgments

- Some content adapted from slides by Kathy McKeown, Dan Jurafsky, Stefan Evert, Marco Baroni