Lecture 6: Word Embeddings

The Distributional Hypothesis

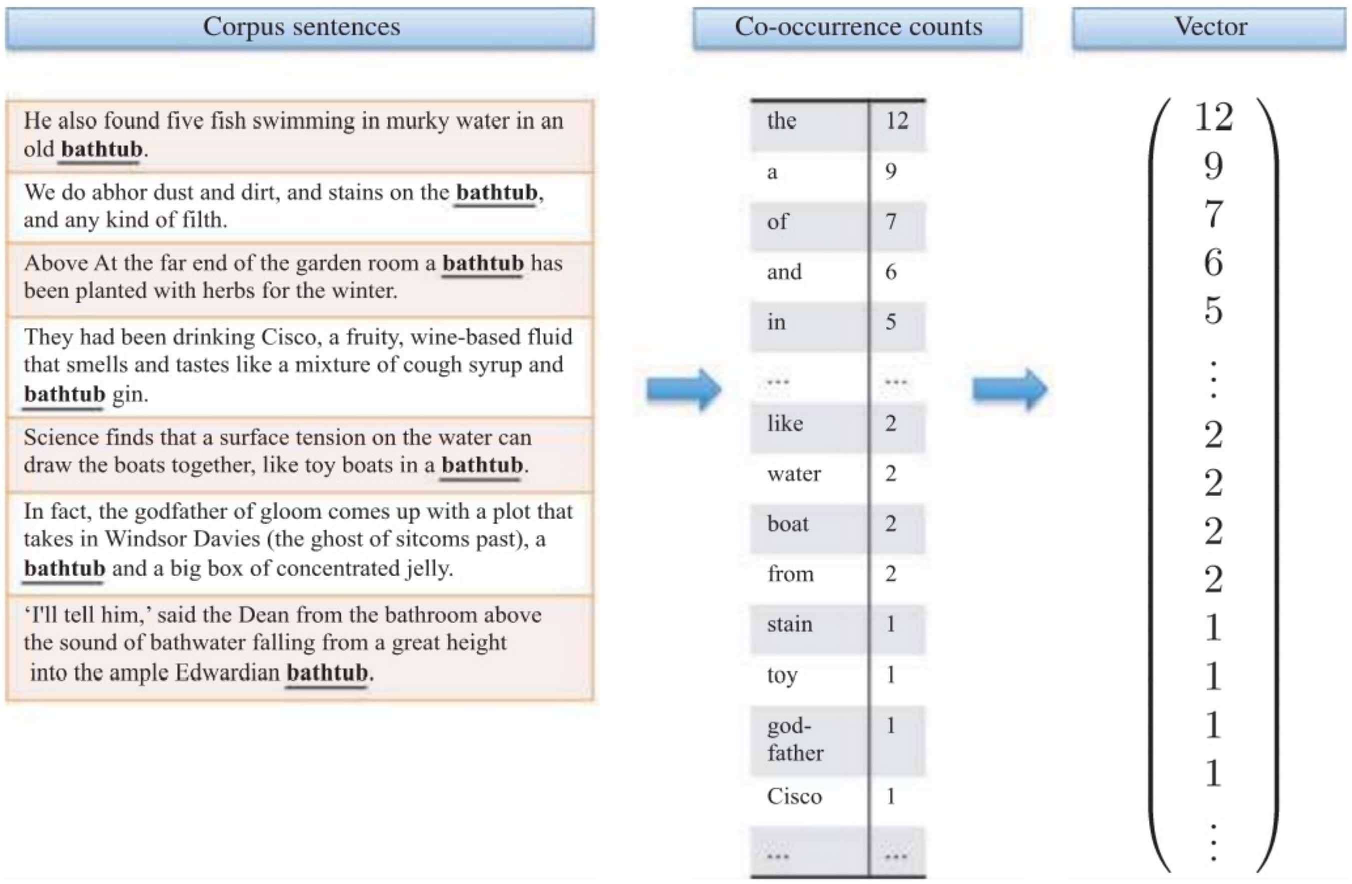
- **The distributional hypothesis:** words that are used and occur in the same contexts tend to have similar meanings (Harris, 1954)

- Distributional semantics represents the meaning of words as a distribution over the word's contexts

Word Embeddings: Count-based Models

- Contexts are defined as neighboring words
  - Usually in a window of *+/-K* words

- Dimensions correspond to context wordforms
- Values in entries correspond to counts – the number of times a word and a context word co-occurred



How Do These Vectors Represent Meaning?

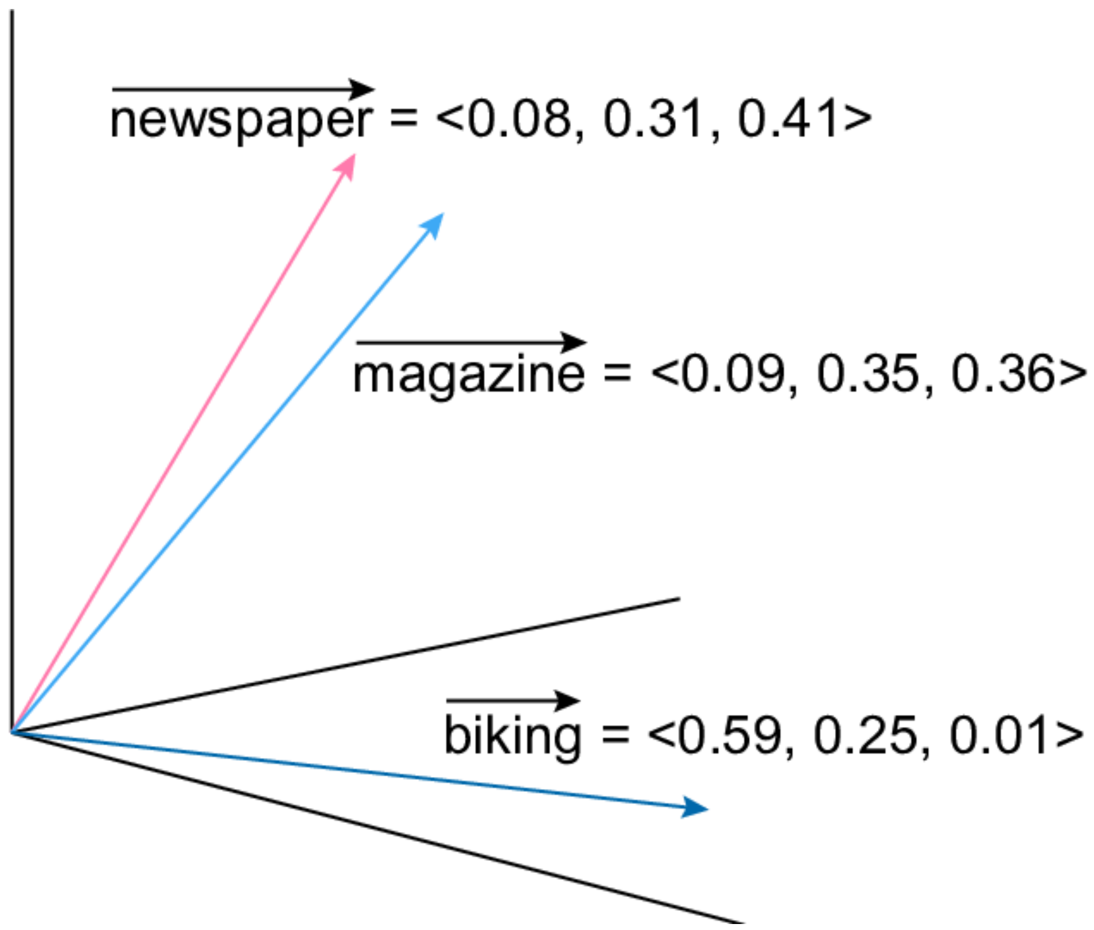
- Similarity can be measured using vector distance metrics
- A popular choice is the “cosine similarity”:

$$\text{similarity}(w, u) = \frac{w \cdot u}{\|w\| \|u\|} = \frac{\sum_{i=1}^{n} w_i u_i}{\sqrt{\sum_{i=1}^{n} w_i^2} \sqrt{\sum_{i=1}^{n} u_i^2}}$$

How Do These Vectors Represent Meaning?

- Instead of representing words with 1-hot vectors (words are either the same or unrelated), embed them in a space that reflects their similarity patterns

- But taking neighboring wordforms as features is a bit naïve…



From Word Counts to Dimensionality Reduction

- Distributional semantics makes intuitive sense if we think of the dimensions as representing semantic features
- For example, a *dog* is a mammal, which is terrestrial, a carnivore and often domesticated
  - A *cat* is then more similar to a dog than a goat is, since they share these traits, while goats share only some
- However, neighboring words are considerably less abstract
  - For example, *car* and *automobile* are synonyms; but are represented as distinct dimensions
  - This fails to capture similarity between a word with *car* as a neighbor and a word with *automobile* as a neighbor

From Word Counts to Dimensionality Reduction

- One way to overcome this would be using dimensionality reduction methods
  - Such as singular value decomposition (Schütze, 1993) or the information bottleneck (Pereira et al., 1993)
- These are strong methods, used (with some variation) today as well
- However, we will devote the rest of the chapter to the more recent, prediction-based models

“Part of Speech Induction from Scratch”, Schütze, 1993

“Distributional clustering of English words”, Pereira, Tishby and Lee, 1993

Prediction-based Models

- Idea: instead of directly representing the distribution of a word, we can represent words as a vector from which the distribution of a word can be "decoded"

- **Task:** learn a network to predict a neighboring word from a given word
  - Sometimes called "self-supervision"

- An influential suite of methods for prediction-based embeddings is *word2vec* (Mikolov et al., 2013)

- We will review the basic implementation of the skip-gram model

Skip-gram (Setup)

- Notation:
  - Denote the *j*-th wordform in the vocabulary with x_j
  - Denote the vocabulary size (number of distinct wordforms) with V
  - *N* is a hyperparameter that determines the dimensionality of the embeddings

Skip-gram (Model)

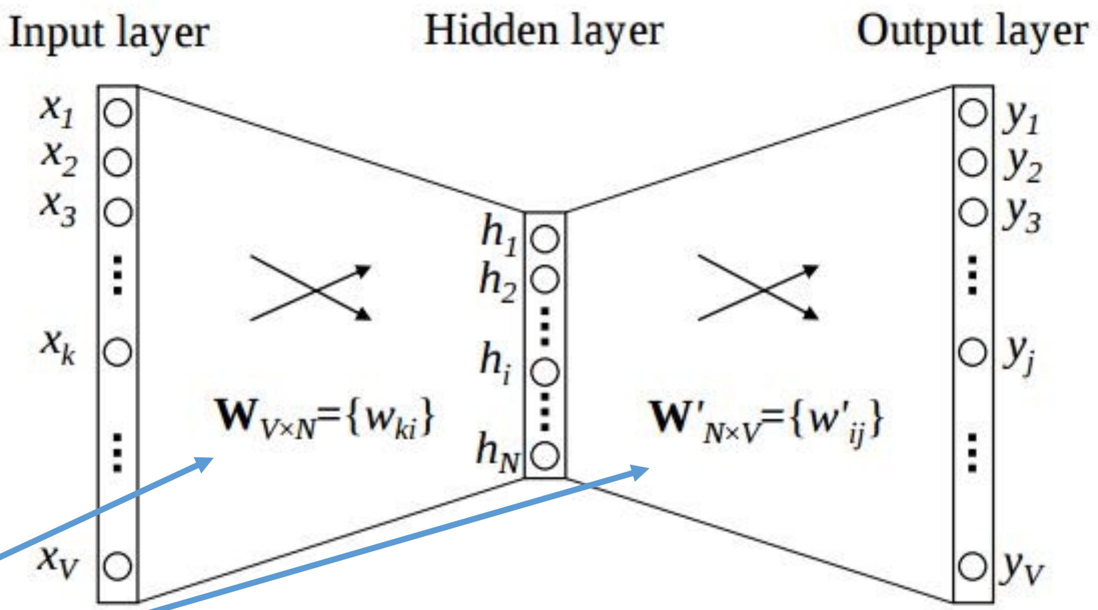
The probability of predicting the neighbor x_j given the word x_k

Soft-max over the network's output

$$P(x_j|x_k) = \frac{e^{w'[:,j]\cdot w[k,:]}}{\sum_{j'=1,\ldots,V} e^{w'[:,j']\cdot w[k,:]}}$$

1-hot vector

Distribution over contexts



W and *W'* are the parameters of the model

Skip-gram (Training)

- Training is carried out by maximizing

$$argmax_{W,W'}\ log[P(text)] =$$

$$argmax_{W,W'} \sum_{(x_j,x_k)\in\ text} log[P(x_j|x_k)]$$

where x_k and x_j are any pair of words no more than K tokens apart

Skip-gram

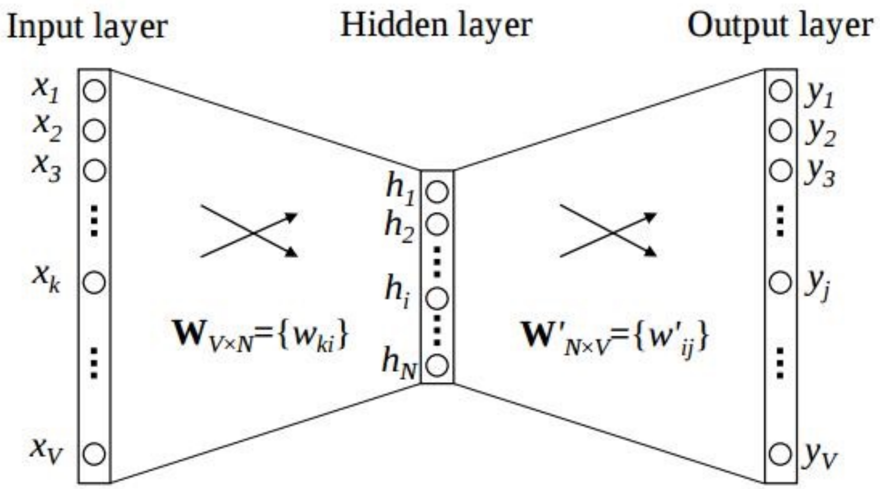
Each wordform now has two embeddings:

input embedding in the input matrix *W*

- Row *j* of the input matrix *W* is the *N* dimensional embedding for word *j* in the vocabulary.

output embedding *v*′, in output matrix W’

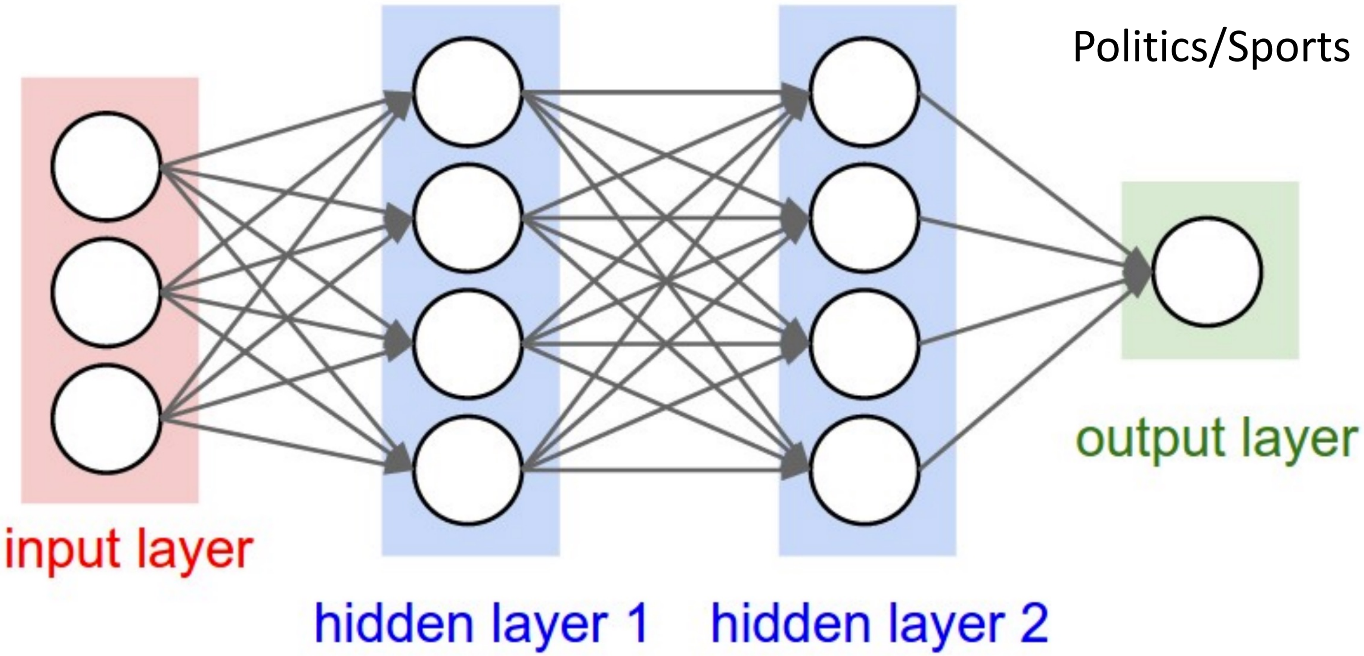
- Column *j* of the output matrix *W*′ is a *N* dimensional vector embedding for word *j* in t vocabulary.
- The input and output embeddings are often concatenated to yield the word embedding for *j*



Word Embeddings as Features

- Word embeddings are often used as features to supervised learning tasks
- For instance, in text classification:

Instead of bag-of-words, one might input the sum of the embeddings of the words in the document



Word Embeddings as Features

- In this case, we don’t really care what the dimensions represent
  - They are just useful feature representations, where the “semantics” of these features are unknown
- In fact, word embeddings are often used as initialization to the first layer of a neural network, and are later updated during training
  - This is sometimes called “fine-tuning”
  - Later in the course…
- Applications cover all aspects of NLP. Essentially any work in NLP that use neural networks (and not only them), will use distributional embeddings to represent the words
  - We will see an example next lesson, and more towards the end of the course