Natural Language Processing 1

Lecture 5: Lexical semantics and word embeddings

Katia Shutova

ILLC
University of Amsterdam

Outline.

Semantics

Compositional semantics:

- studies how meanings of phrases are constructed out of the meaning of individual words
- principle of compositionality: meaning of each whole phrase derivable from meaning of its parts
- sentence structure conveys some meaning: obtained by syntactic representation

Lexical semantics:

- studies how the meanings of individual words can be represented and induced

What is lexical meaning?

- recent results in psychology and cognitive neuroscience give us some clues
- but we don't have the whole picture yet
- different representations proposed, e.g.
  - formal semantic representations based on logic,
  - *or* taxonomies relating words to each other,
  - *or* distributional representations in statistical NLP
- but none of the representations gives us a complete account of lexical meaning

How to approach lexical meaning?

- Formal semantics: set-theoretic approach
  e.g., cat′: the set of all cats; bird′: the set of all birds.
- meaning postulates, e.g.

$$\forall x[\text{bachelor}'(x) \rightarrow \text{man}'(x) \wedge \text{unmarried}'(x)]$$

- Limitations, e.g. *is the Pope a bachelor?*
- Defining concepts through enumeration of all of their features in practice is highly problematic
- How would you define e.g. *chair, tomato, thought, democracy*? – impossible for most concepts
- Prototype theory offers an alternative to set-theoretic approaches

How to approach lexical meaning?

- Formal semantics: set-theoretic approach
  e.g., cat$'$: the set of all cats; bird$'$: the set of all birds.
- meaning postulates, e.g.

$$\forall x[\text{bachelor}'(x) \rightarrow \text{man}'(x) \wedge \text{unmarried}'(x)]$$

- Limitations, e.g. *is the Pope a bachelor?*
- Defining concepts through enumeration of all of their features in practice is highly problematic
- How would you define e.g. *chair, tomato, thought, democracy*? – impossible for most concepts
- Prototype theory offers an alternative to set-theoretic approaches

How to approach lexical meaning?

Prototype theory

- introduced the notion of graded semantic categories
- no clear boundaries; no requirement that a property be shared by all members
- certain members of a category are more central or prototypical (i.e. instantiate the prototype)

  furniture: chair is more prototypical than stool

- Categories form around prototypes; new members added on basis of resemblance to prototype

Eleanor Rosch 1975. *Cognitive Representation of Semantic Categories* (J Experimental Psychology)

How to approach lexical meaning?

Semantic relations

Hyponymy: IS-A

dog is a hyponym of *animal*
animal is a hypernym of *dog*

- hyponymy relationships form a taxonomy
- works best for concrete nouns

How to approach lexical meaning?

Semantic relations

Meronomy: PART-OF	e.g., *arm* is a meronym of *body*, *steering wheel* is a meronym of *car*
Synonymy	e.g., *aubergine*/*eggplant*.
Antonymy	e.g., *big*/*little*

Also:

Near-synonymy/similarity	e.g., *exciting*/*thrilling*
	e.g., *slim*/*slender*/*thin*/*skinny*

WordNet: a large-scale lexical resource linking words by their semantic relations.

Polysemy and word senses

The children **ran** to the store
If you see this man, **run**!
Service **runs** all the way to Cranbury
She is **running** a relief operation in Sudan
the story or argument **runs** as follows
Does this old car still **run** well?
Interest rates **run** from 5 to 10 percent
Who’s **running** for treasurer this year?
They **ran** the tapes over and over again
These dresses **run** small

Outline.

Distributional hypothesis

You shall know a word by the company it keeps (Firth)

The meaning of a word is defined by the way it is used (Wittgenstein).

it was authentic scrumpy, rather sharp and very strong

we could taste a famous local product — scrumpy

spending hours in the pub drinking scrumpy

Cornish Scrumpy Medium Dry. £19.28 - Case

Distributional hypothesis

You shall know a word by the company it keeps (Firth)

The meaning of a word is defined by the way it is used (Wittgenstein).

it was authentic scrumpy, rather sharp and very strong

we could taste a famous local product — scrumpy

spending hours in the pub drinking scrumpy

Cornish Scrumpy Medium Dry. £19.28 - Case

Distributional hypothesis

You shall know a word by the company it keeps (Firth)

The meaning of a word is defined by the way it is used (Wittgenstein).

it was authentic scrumpy, rather sharp and very strong

we could taste a famous local product — scrumpy

spending hours in the pub drinking scrumpy

Cornish Scrumpy Medium Dry. £19.28 - Case

Distributional hypothesis

You shall know a word by the company it keeps (Firth)

The meaning of a word is defined by the way it is used (Wittgenstein).

it was authentic scrumpy, rather sharp and very strong

we could taste a famous local product — scrumpy

spending hours in the pub drinking scrumpy

Cornish Scrumpy Medium Dry. £19.28 - Case

Distributional hypothesis

You shall know a word by the company it keeps (Firth)

The meaning of a word is defined by the way it is used (Wittgenstein).

it was authentic scrumpy, rather sharp and very strong

we could taste a famous local product — scrumpy

spending hours in the pub drinking scrumpy

Cornish Scrumpy Medium Dry. £19.28 - Case

Scrumpy

WESTONS
CIDERS INSPIRED BY TRADITION
SCRUMPY
CLOUDY CIDER
Alc 7.5% Vol
Slowly fermented and matured in old oak vats for a full and fruity flavour

Distributional hypothesis

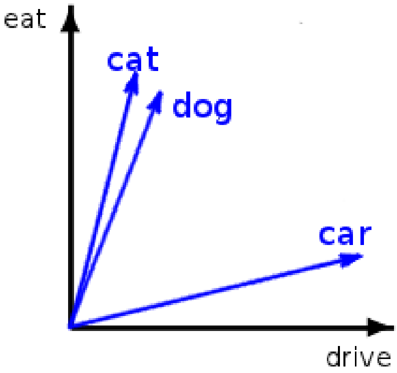
This leads to the distributional hypothesis about word meaning:

- the context surrounding a given word provides information about its meaning;
- words are similar if they share similar linguistic contexts;
- semantic similarity $\approx$ distributional similarity.

The general intuition

- **Distributions** are vectors in a multidimensional semantic space.
- The **semantic space** has dimensions which correspond to possible contexts – features.
- For our purposes, a distribution can be seen as a point in that space (the vector being defined with respect to the origin of that space).
- *scrumpy* [...pub 0.8, drink 0.7, strong 0.4, joke 0.2, mansion 0.02, zebra 0.1...]

Vectors



The notion of context

1 Word windows: *n* words on either side of the lexical item.
Example: n=2 (5 words window):

> | *The prime **minister** acknowledged the* |
> *question.*

minister [the 2, prime 1, acknowledged 1, question 0]

question is 0 because n = 2

Context

2. Lexeme window: as above but using stems.
   Example: n=2 (5 words window):

   *| The prime **minister** acknowledged the | question.*

   minister [the 2, prime 1, acknowledge 1, question 0]

Context

3 Syntactic relations (dependencies). Context for a lexical item is the syntactic dependency structure it belongs to.
Example:

*The prime **minister** acknowledged the question.*

minister [prime 1, acknowledge 1]

minister [prime_mod 1, acknowledge_subj 1]

Context weighting

1. **Binary model**: if context *c* co-occurs with word *w*, value of vector $\vec{w}$ for dimension *c* is 1, 0 otherwise.
2. **Basic frequency model:** the value of vector $\vec{w}$ for dimension *c* is the number of times that *c* co-occurs with *w*.
3. **Characteristic model**: Weights given to the vector components express how *characteristic* a given context is for word *w*.

Characteristic model

- Weights given to the vector components express how *characteristic* a given context is for word w.
- Pointwise Mutual Information (PMI)

$$PMI(w,c) = \log \frac{P(w,c)}{P(w)P(c)} = \log \frac{P(w)P(c|w)}{P(w)P(c)} = \log \frac{P(c|w)}{P(c)}$$

$$P(c) = \frac{f(c)}{\sum_k f(c_k)}, \qquad P(c|w) = \frac{f(w,c)}{f(w)},$$

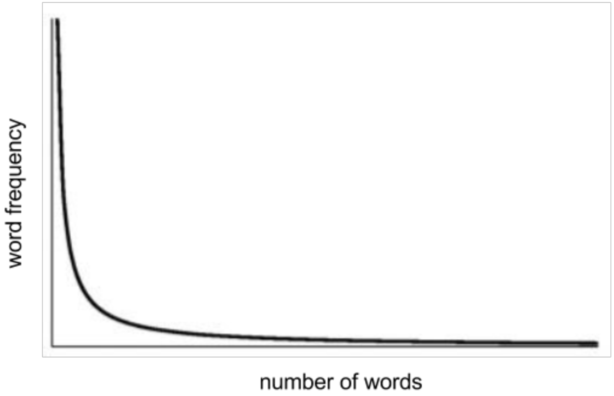
$$PMI(w,c) = \log \frac{f(w,c)\sum_k f(c_k)}{f(w)f(c)}$$

$f(w,c)$: frequency of word w in context c
$f(w)$: frequency of word w in all contexts
$f(c)$: frequency of context c

What semantic space?

- Entire vocabulary.
  - + All information included – even rare contexts
  - - Inefficient (100,000s dimensions). Noisy (e.g. *002.png|thumb|right|200px|graph*). Sparse
- Top *n* words with highest frequencies.
  - + More efficient (2000-10000 dimensions). Only ‘real’ words included.
  - - May miss out on infrequent but relevant contexts.
- Dimensionality reduction using matrix factorization
  - + Very efficient (200-500 dimensions). Captures generalisations in the data.
  - - The resulting matrices are not interpretable.

Word frequency: Zipfian distribution



What semantic space?

- Entire vocabulary.
  - + All information included – even rare contexts
  - - Inefficient (100,000s dimensions). Noisy (e.g. *002.png|thumb|right|200px|graph*). Sparse.
- Top *n* words with highest frequencies.
  - + More efficient (2000-10000 dimensions). Only ‘real’ words included.
  - - May miss out on infrequent but relevant contexts.
- Dimensionality reduction using matrix factorization
  - + Very efficient (200-500 dimensions). Captures generalisations in the data.
  - - The resulting matrices are not interpretable.

An example noun

- *language*:

0.54::other+than+English	0.44::of+instruction
0.53::English+as	0.44::speaker+of
0.52::English+be	0.42::pron+speak
0.49::english	0.42::colon+English
0.48::and+literature	0.42::be+English
0.48::people+speak	0.42::language+be
0.47::French+be	0.42::and+culture
0.46::Spanish+be	0.41::arabic
0.46::and+dialects	0.41::dialects+of
0.45::grammar+of	0.40::percent+speak
0.45::foreign	0.39::spanish
0.45::germanic	0.39::welsh
0.44::German+be	0.39::tonal

An example adjective

- *academic*:

0.52::Decathlon	0.36::reputation+for
0.51::excellence	0.35::regalia
0.45::dishonesty	0.35::program
0.45::rigor	0.35::freedom
0.43::achievement	0.35::student+with
0.42::discipline	0.35::curriculum
0.40::viceresident+for	0.34::standard
0.39::institution	0.34::at+institution
0.39::credentials	0.34::career
0.38::journal	0.34::Career
0.37::journal+be	0.33::dress
0.37::vocational	0.33::scholarship
0.37::student+achieve	0.33::prepare+student
0.36::athletic	0.33::qualification

Polysemy

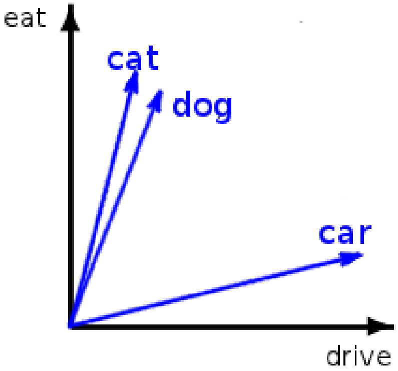
- Distribution for *pot*, as obtained from Wikipedia.

0.57::melt
0.44::pron+smoke
0.43::of+gold
0.41::porous
0.40::of+tea
0.39::player+win
0.39::money+in
0.38::of+coffee
0.33::amount+in
0.33::ceramic
0.33::hot

0.32::boil
0.31::bowl+and
0.31::ingredient+in
0.30::plant+in
0.30::simmer
0.29::pot+and
0.28::bottom+of
0.28::of+flower
0.28::of+water
0.28::food+in

Calculating similarity in a distributional space

- Distributions are vectors, so distance can be calculated.



Measuring similarity

- Cosine:

$$cos(\theta) = \frac{\sum v1_k * v2_k}{\sqrt{\sum v1_k^2} * \sqrt{\sum v2_k^2}} \tag{1}$$

- The cosine measure calculates the angle between two vectors and is therefore length-independent.
- Other measures include Euclidean distance etc.

The scale of similarity: some examples

house – building 0.43
gem – jewel 0.31
capitalism – communism 0.29
motorcycle – bike 0.29
test – exam 0.27
school – student 0.25
singer – academic 0.17
horse – farm 0.13
man –accident 0.09
tree – auction 0.02
cat –county 0.007

Words most similar to *cat*

as chosen from the 5000 most frequent nouns in Wikipedia.

1 cat	0.29 human	0.25 woman	0.22 monster
0.45 dog	0.29 goat	0.25 fish	0.22 people
0.36 animal	0.28 snake	0.24 squirrel	0.22 tiger
0.34 rat	0.28 bear	0.24 dragon	0.22 mammal
0.33 rabbit	0.28 man	0.24 frog	0.21 bat
0.33 pig	0.28 cow	0.23 baby	0.21 duck
0.31 monkey	0.26 fox	0.23 child	0.21 cattle
0.31 bird	0.26 girl	0.23 lion	0.21 dinosaur
0.30 horse	0.26 sheep	0.23 person	0.21 character
0.29 mouse	0.26 boy	0.23 pet	0.21 kid
0.29 wolf	0.26 elephant	0.23 lizard	0.21 turtle
0.29 creature	0.25 deer	0.23 chicken	0.20 robot

But what is similarity?

- In distributional semantics, very broad notion: synonyms, near-synonyms, hyponyms, taxonomical siblings, antonyms, etc.
- Correlates with a psychological reality.
- Test via correlation with human judgments on a test set:
  - Miller & Charles (1991)
  - WordSim
  - MEN
  - SimLex
- Correlation of 0.8 or more.

Distributional methods are a usage representation

- Distributions are a good conceptual representation if you believe that ‘the meaning of a word is given by its usage’.
- Corpus-dependent, culture-dependent, register-dependent.
  Example: similarity between *policeman* and *cop*: 0.23

Distribution for *policeman*

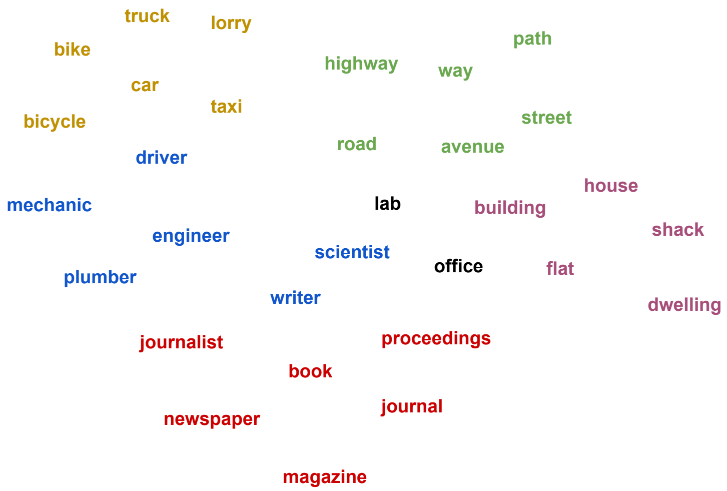
policeman
0.59::ball+poss
0.48::and+civilian
0.42::soldier+and
0.41::and+soldier
0.38::secret
0.37::people+include
0.37::corrupt
0.36::uniformed
0.35::uniform+poss
0.35::civilian+and
0.31::iraqi
0.31::lot+poss
0.31::chechen
0.30::laugh
0.29::and+criminal
0.28::incompetent
0.28::pron+shoot
0.28::hat+poss
0.28::terrorist+and
0.27::and+crowd
0.27::military
0.27::helmet+poss
0.27::father+be
0.26::on+duty
0.25::salary+poss
0.25::on+horseback
0.25::armed
0.24::and+nurse
0.24::job+as
0.24::open+fire

Distribution for *cop*

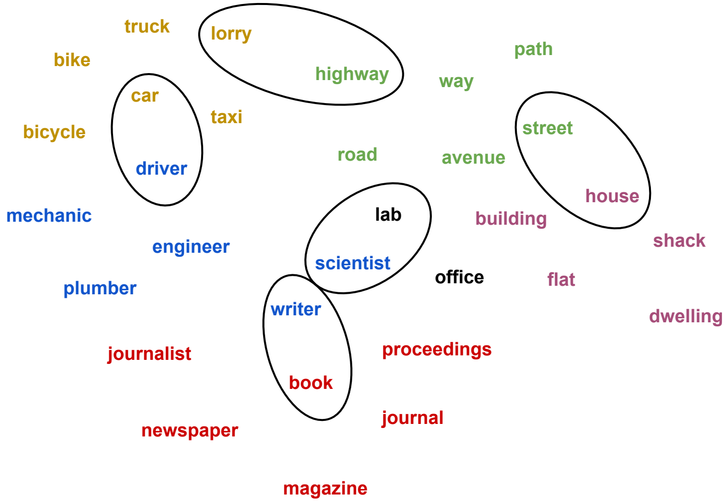
cop

0.45::crooked	0.27::investigate+murder
0.45::corrupt	0.26::on+force
0.44::maniac	0.25::parody+of
0.38::dirty	0.25::Mason+and
0.37::honest	0.25::pron+kill
0.36::uniformed	0.25::racist
0.35::tough	0.24::addicted
0.33::pron+call	0.23::gritty
0.32::funky	0.23::and+interference
0.32::bad	0.23::arrive
0.29::veteran	0.23::and+detective
0.29::and+robot	0.22::look+way
0.28::and+criminal	0.22::dead
0.28::bogus	0.22::pron+stab
0.28::talk+to+pron	0.21::pron+evade

Clustering nouns



Clustering nouns



Outline.

Distributional semantic models

1. Count-based models:
   - Explicit vectors: dimensions are elements in the context
   - **long sparse** vectors with **interpretable** dimensions
2. Prediction-based models:
   - Train a model to predict plausible contexts for a word
   - learn word representations in the process
   - **short dense** vectors with **latent** dimensions

Sparse vs. dense vectors

Why dense vectors?

- easier to use as features in machine learning (less weights to tune)
- may generalize better than storing explicit counts
- may do better at capturing synonymy:
  - e.g. *car* and *automobile* are distinct dimensions in count-based models
  - will not capture similarity between a word with *car* as a neighbour and a word with *automobile* as a neighbour

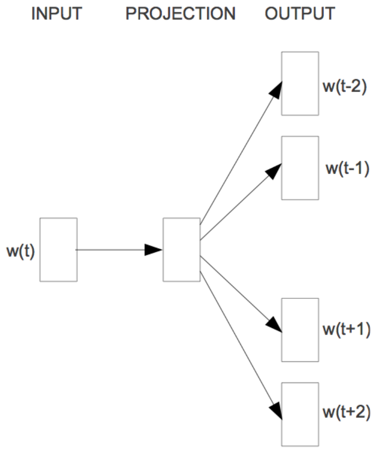
Prediction-based distributional models

Mikolov et. al. 2013. *Efficient Estimation of Word Representations in Vector Space*.

word2vec: **Skip-gram** model

- inspired by work on neural language models
- train a neural network to predict neighboring words
- learn dense embeddings for the words in the training corpus in the process

Skip-gram



given the word as input we want to predict the words in the window (neighboring words)

Slide credit: Tomas Mikolov

Skip-gram

Intuition: words with similar meanings often occur near each other in texts

Given a word $w(t)$:

- Predict each neighbouring word
  - in a context window of $2L$ words
  - from the current word.
- For $L = 2$, we predict its 4 neighbouring words:

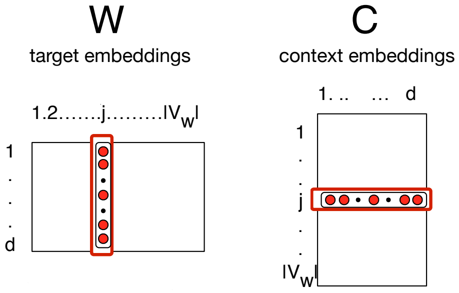
$$[w(t-2), w(t-1), w(t+1), w(t+2)]$$

Skip-gram: Parameter matrices

Learn 2 embeddings for each word $w_j \in V_w$:

- **word embedding** v, in word matrix W
- **context embedding** c, in context matrix C

When the model learns, it tries to make the vector for the target word (from W) similar (by dot product) to the vectors for its true context words (from C), and less similar to vectors for random words (negative samples).



Skip-gram: Setup

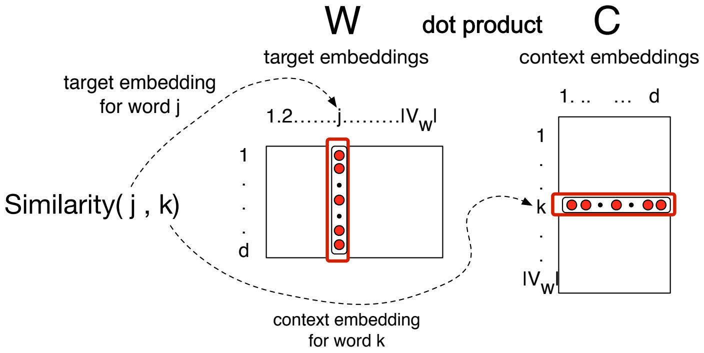
- Walk through the corpus pointing at word $w(t)$, whose index in the vocabulary is j — we will call it w_j
- our goal is to predict $w(t+1)$, whose index in the vocabulary is k — we will call it w_k
- to do this, we need to compute

$$p(w_k|w_j)$$

- Intuition behind skip-gram: to compute this probability we need to compute similarity between w_j and w_k

Skip-gram: Computing similarity

Similarity as dot-product between the target vector and context vector



Slide credit: Dan Jurafsky

Skip-gram: Similarity as dot product

- Remember cosine similarity?

$$cos(v1, v2) = \frac{\sum v1_k * v2_k}{\sqrt{\sum v1_k^2} * \sqrt{\sum v2_k^2}} = \frac{v1 \cdot v2}{||v1||||v2||}$$

It's just a normalised dot product.

- Skip-gram: Similar vectors have a high dot product

$$Similarity(c_k, v_j) \propto c_k \cdot v_j$$

Skip-gram: Compute probabilities

- Compute similarity as a dot product

$$Similarity(c_k, v_j) \propto c_k \cdot v_j$$

- Normalise to turn this into a probability
- by passing through a softmax function:

$$p(w_k|w_j) = \frac{e^{c_k \cdot v_j}}{\sum_{i \in V} e^{c_i \cdot v_j}}$$

Skip-gram: Learning

- Start with some initial embeddings (usually random)
- At training time, walk through the corpus
- iteratively make the embeddings for each word
  - more like the embeddings of its neighbors
  - less like the embeddings of other words.

Skip-gram: Objective

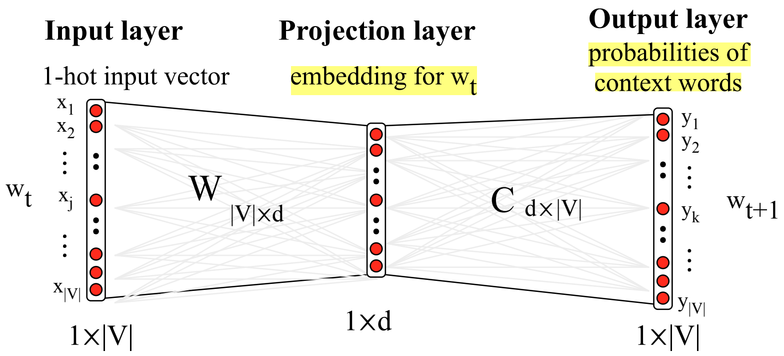
Learn parameters C and W that maximize the overall corpus probability:

$$\arg\max \prod_{(w_j, w_k) \in D} p(w_k | w_j)$$

$$p(w_k | w_j) = \frac{e^{c_k \cdot v_j}}{\sum_{i \in V} e^{c_i \cdot v_j}}$$

$$\arg\max \prod_{(w_j, w_k) \in D} p(w_k | w_j) = \prod_{(w_j, w_k) \in D} \frac{e^{c_k \cdot v_j}}{\sum_{i \in V} e^{c_i \cdot v_j}}$$

Visualising skip-gram as a network



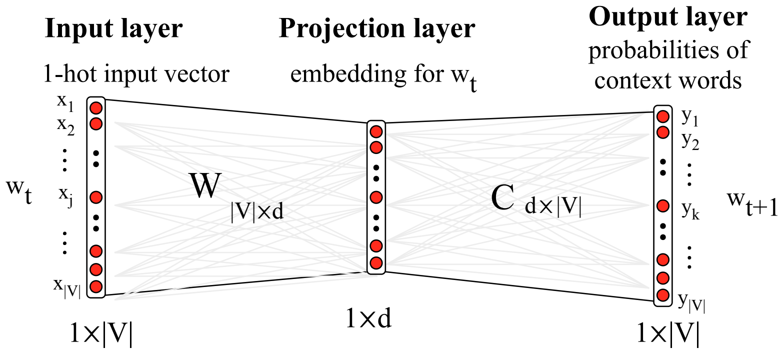
Slide credit: Dan Jurafsky

One hot vectors

- A vector of length |V|
- 1 for the target word and 0 for other words
- So if “bear” is vocabulary word 5
- The one-hot vector is [0,0,0,0,1,0,0,0,0........0]

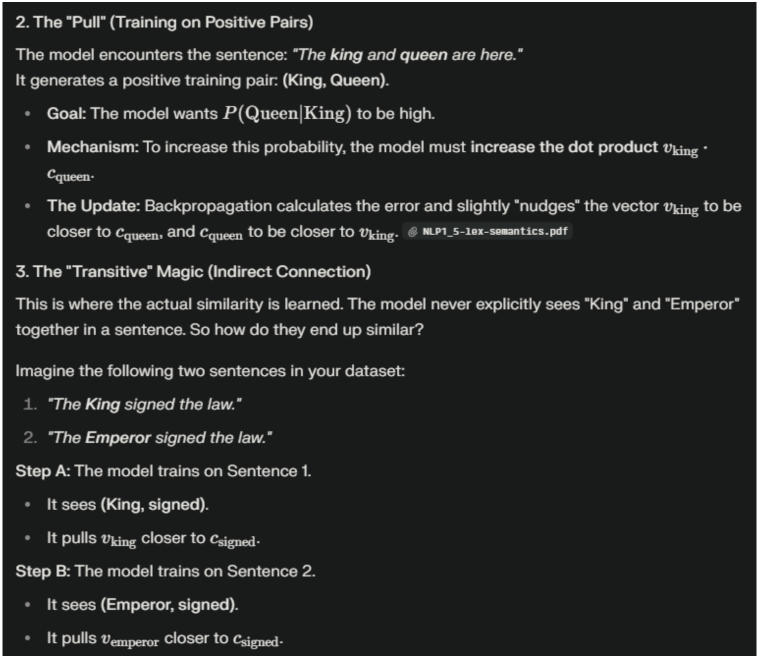
$$\begin{array}{ccccccccccccccccccccc}
w_0 & w_1 & & & & & & & & & w_j & & & & & & & & & & w_{|V|} \\
0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0
\end{array}$$

Visualising skip-gram as a network



Slide credit: Dan Jurafsky

if the number of words is very high it becomes very expensive to compute the den of the prob



2. The "Pull" (Training on Positive Pairs)

The model encounters the sentence: *"The **king** and **queen** are here."*
It generates a positive training pair: **(King, Queen)**.

- **Goal:** The model wants $P(\text{Queen}|\text{King})$ to be high.
- **Mechanism:** To increase this probability, the model must **increase the dot product** $v_{\text{king}} \cdot c_{\text{queen}}$.
- **The Update:** Backpropagation calculates the error and slightly "nudges" the vector v_{king} to be closer to c_{queen}, and c_{queen} to be closer to v_{king}. NLP1_5-lex-semantics.pdf

3. The "Transitive" Magic (Indirect Connection)

This is where the actual similarity is learned. The model never explicitly sees "King" and "Emperor" together in a sentence. So how do they end up similar?

Imagine the following two sentences in your dataset:

1. *"The **King** signed the law."*
2. *"The **Emperor** signed the law."*

Step A: The model trains on Sentence 1.

- It sees **(King, signed)**.
- It pulls v_{king} closer to c_{signed}.

Step B: The model trains on Sentence 2.

- It sees **(Emperor, signed)**.
- It pulls v_{emperor} closer to c_{signed}.

Skip-gram with negative sampling

Problem with softmax: expensive to compute the denominator for the whole vocabulary

$$p(w_k|w_j) = \frac{e^{c_k \cdot v_j}}{\sum_{i \in V} e^{c_i \cdot v_j}}$$

Approximate the denominator: **negative sampling**

- At training time, walk through the corpus
- for each target word and positive context
- sample *k* noise samples or negative samples, i.e. other words

Skip-gram with negative sampling

- Objective in training:
  - Make the word like the context words
    lemon, a [tablespoon of apricot preserves or] jam.

    c_1 c_2 w c_3 c_4
  - And not like the *k* negative examples

[cement idle dear coaxial apricot attendant whence forever puddle]

n_1 n_2 n_3 n_4 w n_5 n_6 n_7 n_8

Skip-gram with negative sampling: Training examples

Convert the dataset into word pairs:

- **Positive (+)**

  (apricot, tablespoon)
  (apricot, of)
  (apricot, jam)
  (apricot, or)

- **Negative (-)**

  (apricot, cement)
  (apricot, idle)
  (apricot, attendant)
  (apricot, dear)
  ...

Skip-gram with negative sampling

- instead of treating it as a **multi-class problem** (and returning a probability distribution over the whole vocabulary)
- **return a probability** that word w_k is a valid context for word w_j

$$P(+|w_j, w_k)$$

$$P(-|w_j, w_k) = 1 - P(+|w_j, w_k)$$

Skip-gram with negative sampling

- model similarity as dot product

$$Similarity(c_k, v_j) \propto c_k \cdot v_j$$

- turn this into a probability using the **sigmoid function**:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

$$P(+|w_j, w_k) = \frac{1}{1 + e^{-c_k \cdot v_j}}$$

$$P(-|w_j, w_k) = 1 - P(+|w_j, w_k) = 1 - \frac{1}{1 + e^{-c_k \cdot v_j}} = \frac{1}{1 + e^{c_k \cdot v_j}}$$

Skip-gram with negative sampling

- model similarity as dot product

$$Similarity(c_k, v_j) \propto c_k \cdot v_j$$

- turn this into a probability using the **sigmoid function**:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

$$P(+|w_j, w_k) = \frac{1}{1 + e^{-c_k \cdot v_j}}$$

$$P(-|w_j, w_k) = 1 - P(+|w_j, w_k) = 1 - \frac{1}{1 + e^{-c_k \cdot v_j}} = \frac{1}{1 + e^{c_k \cdot v_j}}$$

Skip-gram with negative sampling: Objective

- make the word like the context words
- and not like the negative examples

$$\arg\max \prod_{(w_j,w_k)\in D_+} p(+|w_k, w_j) \prod_{(w_j,w_k)\in D_-} p(-|w_k, w_j)$$

$$\arg\max \sum_{(w_j,w_k)\in D_+} \log p(+|w_k, w_j) + \sum_{(w_j,w_k)\in D_-} \log p(-|w_k, w_j) =$$

$$\arg\max \sum_{(w_j,w_k)\in D_+} \log \frac{1}{1+e^{-c_k\cdot v_j}} + \sum_{(w_j,w_k)\in D_-} \log \frac{1}{1+e^{c_k\cdot v_j}}$$

Skip-gram with negative sampling: Objective

- make the word like the context words
- and not like the negative examples

$$\arg\max \prod_{(w_j,w_k)\in D_+} p(+|w_k, w_j) \prod_{(w_j,w_k)\in D_-} p(-|w_k, w_j)$$

$$\arg\max \sum_{(w_j,w_k)\in D_+} \log p(+|w_k, w_j) + \sum_{(w_j,w_k)\in D_-} \log p(-|w_k, w_j) =$$

$$\arg\max \sum_{(w_j,w_k)\in D_+} \log \frac{1}{1+e^{-c_k \cdot v_j}} + \sum_{(w_j,w_k)\in D_-} \log \frac{1}{1+e^{c_k \cdot v_j}}$$

Skip-gram with negative sampling: Objective

- make the word like the context words
- and not like the negative examples

$$\arg\max \prod_{(w_j,w_k)\in D_+} p(+|w_k, w_j) \prod_{(w_j,w_k)\in D_-} p(-|w_k, w_j)$$

$$\arg\max \sum_{(w_j,w_k)\in D_+} \log p(+|w_k, w_j) + \sum_{(w_j,w_k)\in D_-} \log p(-|w_k, w_j) =$$

$$\arg\max \sum_{(w_j,w_k)\in D_+} \log \frac{1}{1+e^{-c_k \cdot v_j}} + \sum_{(w_j,w_k)\in D_-} \log \frac{1}{1+e^{c_k \cdot v_j}}$$

Properties of embeddings

They capture similarity

FRANCE 454	JESUS 1973	XBOX 6909	REDDISH 11724	SCRATCHED 29869	MEGABITS 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

Slide credit: Ronan Collobert

Properties of embeddings

They capture analogy

Analogy task: ***a*** *is to* ***b*** *as* ***c*** *is to* ***d***

The system is given words a, b, c, and it needs to find d.

"apple" is to "apples" as "car"" is to ?
"man" is to "woman" as "king" is to ?

Solution: capture analogy via vector offsets

$$a - b \approx c - d$$

$$man - woman \approx king - queen$$

$$d_w = \operatorname*{argmax}_{d'_w \in V} cos(a - b, c - d')$$

Properties of embeddings

They capture analogy

Analogy task: ***a*** *is to* ***b*** *as* ***c*** *is to* ***d***
The system is given words *a*, *b*, *c*, and it needs to find *d*.

“apple” is to “apples” as “car”‘ is to ?
“man” is to “woman” as “king” is to ?

Solution: capture analogy via vector offsets

$$a - b \approx c - d$$

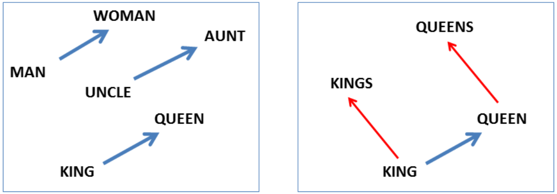
$$man - woman \approx king - queen$$

$$d_w = \operatorname*{argmax}_{d'_w \in V} cos(a - b, c - d')$$

Properties of embeddings

Capture analogy via vector offsets

$$man - woman \approx king - queen$$



Mikolov et al. 2013. *Linguistic Regularities in Continuous Space Word Representations*

Properties of embeddings

They capture a range of semantic relations

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

Mikolov et al. 2013. *Efficient Estimation of Word Representations in Vector Space*

Word embeddings in practice

Word2vec is often used for pretraining in other tasks.

- It will help your models start from an **informed** position
- Requires only **plain text** - which we have a lot of
- Is very **fast** and easy to use
- Already **pretrained** vectors also available (trained on 100B words)

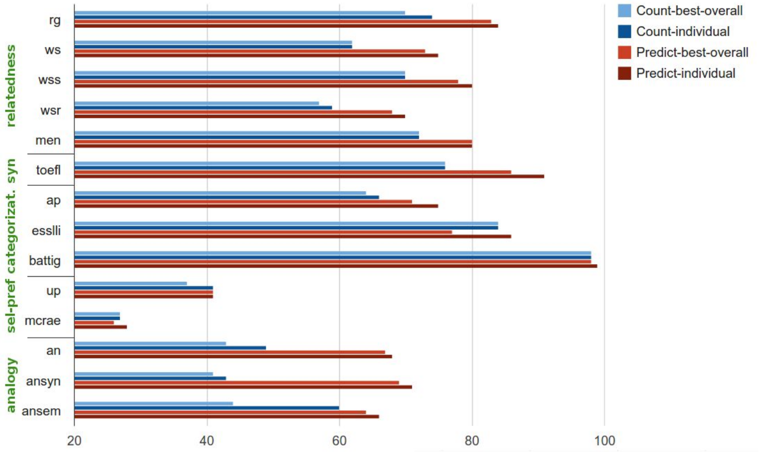
However, for best performance it is important to continue training, fine-tuning the embeddings for a specific task.

Count-based models vs. skip-gram word embeddings

Baroni et. al. 2014. *Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors.*

- Comparison of count-based and neural word vectors on 5 types of tasks and 14 different datasets:
    1. Semantic relatedness
    2. Synonym detection
    3. Concept categorization
    4. Selectional preferences
    5. Analogy recovery

Count-based models vs. skip-gram word embeddings



Some of these findings were later disputed by Levy et. al. 2015. *Improving Distributional Similarity with Lessons Learned from Word Embeddings*

Acknowledgement

Some slides were adapted from Ann Copestake, Dan Jurafsky and Marek Rei