ECE365: Introduction to NLP

Spring 2021

Lecture 7: Word Embeddings

[Reading J&M Chapter 6]

Logistics

- Quiz 2 Tuesday 05/04, 05/05

Distributional Hypothesis

Distributional hypothesis, stated by linguist John R. Firth (1957) as:

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

≈ "words that occur in similar contexts tend to have similar meanings"

One of the most successful ideas of modern statistical NLP!

Distributional Semantic Models

- Vector semantics = {distributional idea (defining a word by counting what other words occur in its environment) } + {meaning of a word as a vector (a point in N-dimensions)}

- Popularly called word embeddings
  - Various versions depending on how the vector components are computed
  - Latent Semantic Analysis, Word2vec, GloVe

Why model words as vectors?

- We need to model word meaning
- As a way for computing similarity between words
- Useful for unknown words

Why model words as vectors?

- We need to model word meaning
- As a way for computing similarity between words
- Similar words
  - not synonyms, but share some element of meaning
  - Car- bicycle, cow-horse

Human-rated similarity on scale of 1-10

word1	word2	similarity
vanish	disappear	9.8
behave	obey	7.3
belief	impression	5.95
muscle	bone	3.65
modest	flexible	0.98
hole	agreement	0.3

SimLex -999 (Hill et al. 2015)

Why not use a thesaurus?

- We don’t have a thesaurus for every language
- We can’t have a thesaurus for every year

Distributional Semantics

- What is *tesgüino*?

(a) A bottle of tesgüino is on the table

(b) People like tesgüino.

(c) Don't have tesgüino before you drive.

(d) Tesgüino is made out of corn

- Intuition for algorithm: **Two words are similar if they have similar word contexts.**

- **How do we model context of a word?**

Two kinds of vector models

- *Sparse representation*

  weighted word co-occurrence matrices

- *Dense representation*

  Singular Value Decomposition (LSA)

  Prediction-based models (Word2vec, GloVe)

  Brown clusters

Two kinds of vector models

- *Sparse representation*

  weighted word co-occurrence matrices

Co-occurrence matrices

- Represent how often a word occurs with other words

Term-term matrix or **word-word co-occurrence matrix** or **word-context matrix**

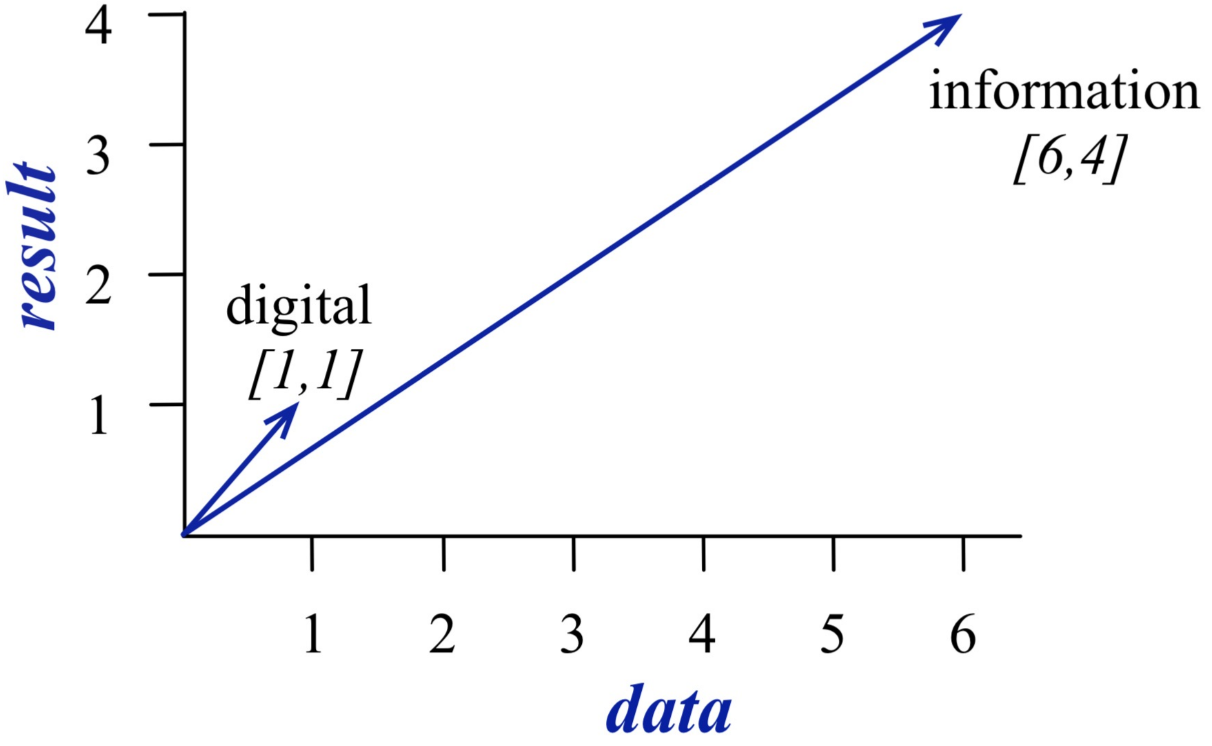
Word-word matrix (or "term-context matrix")

- Fix a context
  - Paragraph
  - Context window of size k

sugar, a sliced lemon, a tablespoonful of	**apricot**	jam, a pinch each of,
their enjoyment. Cautiously she sampled her first	**pineapple**	and another fruit whose taste she likened
well suited to programming on the digital	**computer**.	In finding the optimal R-stage policy from
for the purpose of gathering data and	**information**	necessary for the study authorized in the

	aardvark	computer	data	pinch	result	sugar	…
apricot	0	0	0	1	0	1	
pineapple	0	0	0	1	0	1	
digital	0	2	1	0	1	0	
information	0	1	6	0	4	0	

	aardvark	computer	data	pinch	result	sugar	…
apricot	0	0	0	1	0	1	
pineapple	0	0	0	1	0	1	
digital	0	2	1	0	1	0	
information	0	1	6	0	4	0	



More common: word-word matrix (or "term-context matrix")

- Each vector of dimension |V|
- The word-word matrix is |V|x|V|

	Aardvark	Computer	Data	Pie	Result	sugar
pineapple	0	2	8	442	9	25
strawberry	0	0	0	1	60	19
digital	0	2	1683	5	85	4
information	0	3325	3982	5	378	13

Co-occurrence vectors for four words in the Wikipedia corpus, showing six of the dimensions

More common: word-word matrix (or "term-context matrix")

- We show only 4 x 6, but real matrix is 50,000 x 50,000
- Very sparse (most values are zero)
- That's OK, since there are lots of efficient algorithms for sparse matrices

	Aardvark	Computer	Data	Pie	Result	sugar
cherry	0	2	8	442	9	25
strawberry	0	0	0	1	60	19
digital	0	2	1683	5	85	4
information	0	3325	3982	5	378	13

Co-occurrence vectors for four words in the Wikipedia corpus, showing six of the dimensions

Reminders from Linear Algebra

$$\text{dot-product}(\vec{v}, \vec{w}) = \vec{v} \cdot \vec{w} = \sum_{i=1}^{N} v_i w_i = v_1 w_1 + v_2 w_2 + ... + v_N w_N$$

vector length $|\vec{v}| = \sqrt{\sum_{i=1}^{N} v_i^2}$

Measuring similarity

- Given word vectors *v* and *w*
- Need to measure their similarity

We know from linear algebra:

Dot-product (inner product) can measure vector similarity

- High when two vectors have large values in same dimensions
- Low (infact 0) for **orthogonal vectors** with zeros in complementary distribution

Problem with dot product

$$\text{dot-product}(\vec{v},\vec{w}) = \vec{v}\cdot\vec{w} = \sum_{i=1}^{N} v_i w_i = v_1 w_1 + v_2 w_2 + \ldots + v_N w_N$$

Vector length:

$$|\vec{v}| = \sqrt{\sum_{i=1}^{N} v_i^2}$$

v_i is the count for word *v* in context *i*
w_i is the count for word *w* in context *i*.

Vectors are longer if they have higher values in each dimension

- Frequent words have higher values in each dimension
- Dot-product then favors frequent words

- That's bad: We don't want similarity between

words sensitive to word frequency (How do we handle this?)

Solution: Cosine Similarity

Divide dot-product by vector lengths!

This turns out to be the cosine of the angle between the vectors.

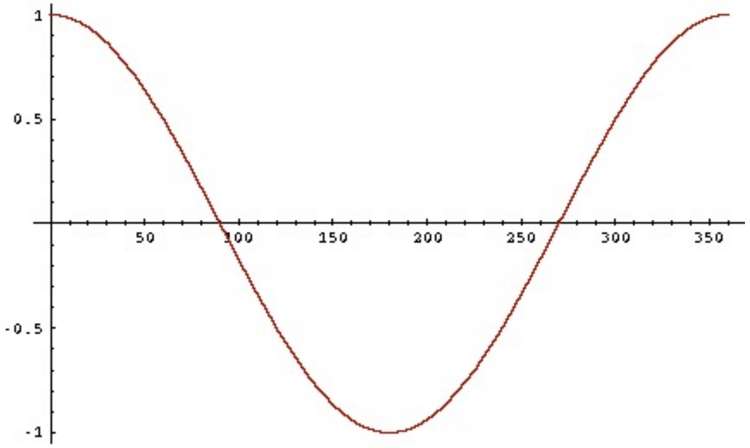
$$\text{cosine}(\vec{v},\vec{w}) = \frac{\vec{v}\cdot\vec{w}}{|\vec{v}||\vec{w}|} = \frac{\sum_{i=1}^{N} v_i w_i}{\sqrt{\sum_{i=1}^{N} v_i^2}\sqrt{\sum_{i=1}^{N} w_i^2}}$$

$$\vec{a}\cdot\vec{b} = |\vec{a}||\vec{b}|\cos\theta$$

$$\frac{\vec{a}\cdot\vec{b}}{|\vec{a}||\vec{b}|} = \cos\theta$$

Cosine as a similarity metric

- -1: vectors point in opposite directions
- +1: vectors point in same directions
- 0: vectors are orthogonal



- Frequency is non-negative, so cosine range 0-1

$$\cos(\vec{v},\vec{w}) = \frac{\vec{v}\bullet\vec{w}}{|\vec{v}||\vec{w}|} = \frac{\vec{v}}{|\vec{v}|}\bullet\frac{\vec{w}}{|\vec{w}|} = \frac{\sum_{i=1}^{N} v_i w_i}{\sqrt{\sum_{i=1}^{N} v_i^2}\sqrt{\sum_{i=1}^{N} w_i^2}}$$

	large	data	computer
apricot	1	0	0
digital	0	1	2
information	1	6	1

Which pair of words is more similar?

cosine(apricot,information) =

cosine(digital,information) =

cosine(apricot,digital) =

But raw frequency is a bad representation

- Frequency is clearly useful; if *sugar* appears a lot near *apricot*, that's useful information.

- But overly frequent words like *the*, *it,* or *they* are not very informative about the context

Need a function that resolves this frequency paradox!

tf-idf: combine two factors

- **tf: term frequency**. frequency count (usually log-transformed):

$$\mathrm{tf}_{t,d} = \begin{cases} 1+\log_{10}\mathrm{count}(t,d) & \text{if } \mathrm{count}(t,d)>0 \\ 0 & \text{otherwise} \end{cases}$$

- **idf: inverse document frequency:**

Total # of docs in collection

$$\mathrm{idf}_i = \log\left(\frac{N}{\mathrm{df}_i}\right)$$

Words like "the" or "good" have very low idf

# of docs that have word i

tf-idf value for word t in document d:

$$w_{t,d} = \mathrm{tf}_{t,d} \times \mathrm{idf}_t$$

So far

- Co-occurrence matrix

- Similarity between vectors

More common: word-word matrix (or "term-context matrix")

- We show only 4 x 6, but real matrix is 50,000 x 50,000
- Very sparse (most values are zero)
- That’s OK, since there are lots of efficient algorithms for sparse matrices

	Aardvark	Computer	Data	Pie	Result	sugar
cherry	0	2	8	442	9	25
strawberry	0	0	0	1	60	19
digital	0	2	1683	5	85	4
information	0	3325	3982	5	378	13

Co-occurrence vectors for four words in the Wikipedia corpus, showing six of the dimensions

Dense Representation

weighted vectors are

- **long** (length |V|= 20,000 to 50,000)
- **sparse** (most elements are zero)

Alternative: learn vectors which are

- **short** (length 200-1000)
- **dense** (most elements are non-zero)

Sparse vs. Dense vectors

- Why dense vectors?
  - Short vectors may be easier to use as features in machine learning (less weights to tune)
  - Dense vectors may generalize better than storing explicit counts
  - They may do better at capturing synonymy:
    - *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

Three Methods for Dense Vectors

- Singular Value Decomposition (SVD)
  - A special case of this is called LSA – Latent Semantic Analysis
- "Neural Language Model"-inspired predictive models
  - skip-grams and CBOW
- Brown clustering

Prediction-based models

- **Word2vec** (Mikolov et al.)
- https://code.google.com/archive/p/word2vec/
- **Fasttext** http://www.fasttext.cc/
- **Glove** (Pennington, Socher, Manning)
- http://nlp.stanford.edu/projects/glove/

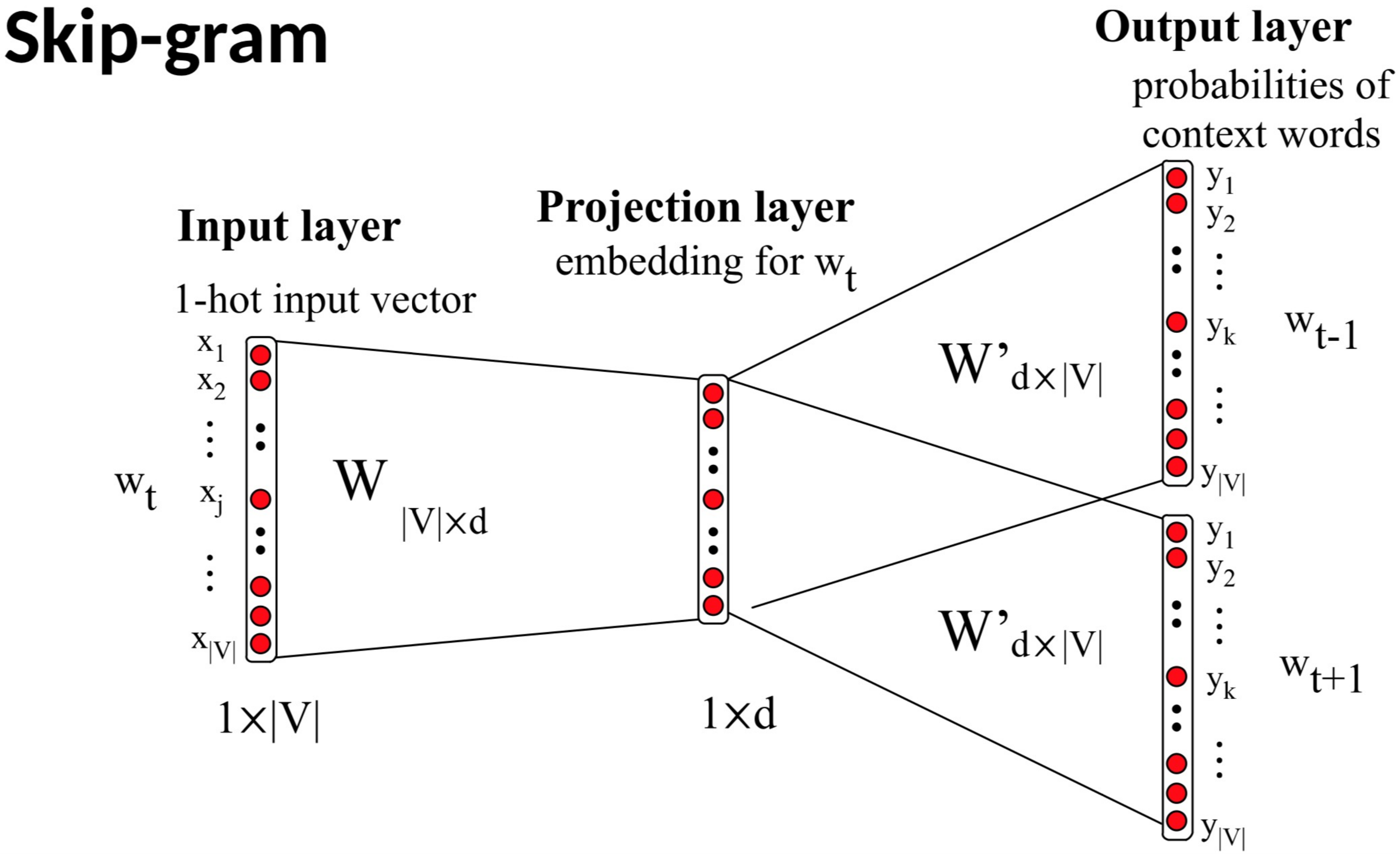
How to learn word2vec (skip-gram) embeddings

- Start with V random 300-dimensional vectors as initial embeddings
- Use logistic regression (the second most basic classifier used in machine learning after naïve Bayes)
  - Take a corpus and take pairs of words that co-occur as positive examples
  - Take pairs of words that don't co-occur as negative examples
  - Train the classifier to distinguish these by slowly adjusting all the embeddings to improve the classifier performance
  - Throw away the classifier code and keep the embeddings.

- Instead of **counting** how often each word *w* occurs near "*apricot*"
- Train a classifier on a binary **prediction** task:
  - Is *w* likely to show up near "*apricot*"?

- We don’t actually care about this task
  - But we'll take the learned classifier weights as the word embeddings

How to learn word2vec (skip-gram)



word	**dim0**	**dim1**	**dim2**	**dim3**	···	dim300
today	0.35	-1.3	2.2	0.003		
cat	-3.1	-1.7	1.1	-0.56		
sleep	0.55	3.0	2.4	-1.2		
watch	-0.09	0.8	-1.8	2.9		
bird	2.0	0.16	-1.9	2.3		

Evaluating embeddings

- Have a vocabulary and a set of word embeddings
- Need to know how good the vectors are
- Intrinsic:
  - Compare to human scores on word similarity-type tasks:
  - WordSim-353 dataset

    environment – ecology   8.8
- Extrinsic

Properties of embeddings

Similarity depends on window size C

- C = ±2 The nearest words to *Hogwarts:*
  - *Sunnydale*
  - *Evernight*
- C = ±5 The nearest words to *Hogwarts:*
  - *Dumbledore*
  - *Malfoy*
  - *halfblood*

Properties of embeddings

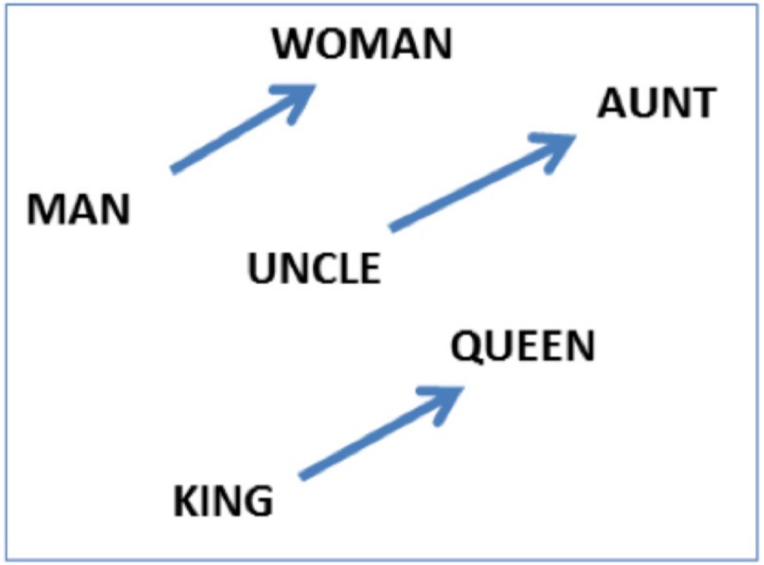
word = “sweden”

Word	Cosine distance
norway	0.760124
denmark	0.715460
finland	0.620022
switzerland	0.588132
belgium	0.585835
netherlands	0.574631
iceland	0.562368
estonia	0.547621
slovenia	0.531408

Analogy: Embeddings capture relational meaning

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

vector(*'Paris'*) - vector(*'France'*) + vector(*'Italy'*) ≈ vector('Rome')



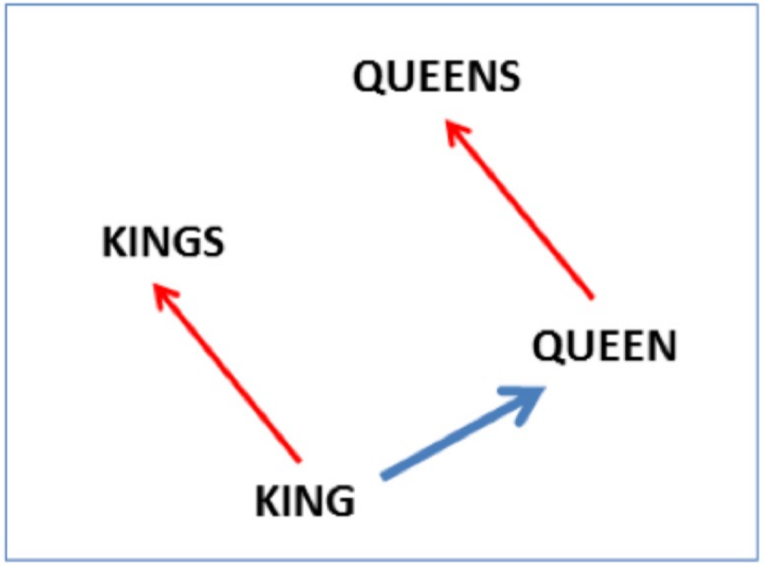


Figure 6.1 A two-dimensional (t-SNE) projection of embeddings for some words and phrases, showing that words with similar meanings are nearby in space. The original 60-dimensional embeddings were trained for sentiment analysis. Simplified from Li et al. (2015).

Summary

- Distributional (vector) Models of meaning
  - Sparse vectors:
    - tf-idf weighted word-word co-occurrence matrix

  - Dense vectors:
    - Prediction-based: Word2vec