Carnegie Mellon University
Language Technologies Institute

11-411/11-611 Natural Language Processing

Distributional Semantics and Word Embeddings

David R. Mortensen (and Kemal Oflazer)

September 29, 2022

Language Technologies Institute

Learning Objectives

- Know the history, definition, and basic insight behind the **distributional hypothesis**.
- Understand why vector models of meaning are useful in NLP.
- List the kinds of vector semantic models that exist
- State what a term-document matrix is and how it relates to vector semantics
- Compare it with a word-word (or word-context) matrix
- Contrast first-order co-occurrence with second-order co-occurrence
- Apply PPMI to solve a problem of raw word counts
- Address problems with PPMI
- Discuss strengths and weaknesses of various similarity metrics for vectors
- Know why and how to obtain dense vectors
- Describe how **word2vec** embeddings are trained and used
    - **Skip-Gram**
    - **CBOW**

Introduction to the Distributional Hypothesis

Word Embeddings have Their Roots In Anthropological Linguistics

- In the early 20th century, many native languages of the Americas were dying
- A group of and anthropologists (Boas, Sapir, Bloomfield, etc.) decided that they needed to describe all of these languages (produce grammars, dictionaries, and texts for them) before they were gone
- Earlier scholars who studied American languages tried to shoehorn them into the grammatical structure of European languages, but this group of researchers saw that they were very different from one another and from, e.g., Latin
- **They wanted to describe languages on their own terms**
- They developed techniques (in some cases, **algorithms**) for discovering meaning and grammatical structure without making reference to other languages
- Pinnacle: **Zellig Harris** (Noam Chomsky's dissertation adviser)

The Distributional Hypothesis: Words' Meanings Follows from Their Contexts

Insight: Want to know the meaning of a word? Find what words occur with it.

- Leonard Bloomfield
- Edward Sapir
- Martin Joos–stated in information-theoretic terms
- Zellig Harris—first complete formalization
    - "oculist and eye-doctor …occur in almost the same environments"
    - "If A and B have almost identical environments we say that they are synonyms."

Firth’s Formulation of the Distributional Hypothesis

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

—J. R. Firth

You Can Tell a Lot about *Beef* from Its Contexts

1	fertility. Organ meats such as	beef	and chicken liver, tongue and hear
2	controlling scours. _HOW TO FEED:	BEEF	AND DAIRY CALVES_ - 0.2 gram Dy
3	ing process discolors the treated	beef	and liquid accumulates in prepackag
4	say. He did say she could get her	beef	and vegetables in cans this summer
5	and feed efficiency of fattening	beef	animals. _HOW TO FEED:_ At the
6	steaks, chops, chicken and prime	beef	as well as Tom's favorite dish, stu
7	ross from him was surmounted by a	beef	barrel with ends knocked out. In t
8	counter of boards laid across two	beef	barrels. There was, of course, no
9	Because Holstein cattle weren't a	beef	breed, they were rarely seen on a
10	2-5 grams of phenothiazine daily;	beef	calves- .5 to 1.5 grams daily depe
11	ties of this drug. _HOW TO FEED:	BEEF	CATTLE (FINISHING RATION)_ - To
12	dairy cows and lesser amounts to	beef	cattle and poultry. About 90 percen
13	raises enough poultry, pigs, and	beef	cattle for most of their needs. Lo
14	on of liver abscesses in feed-lot	beef	cattle. Prevention of bacterial pne
15	pal feed bunk types for dairy and	beef	cattle: (1) Fence-line bunks- catt
16	es feed efficiency. _HOW TO FEED:	BEEF	CATTLE_ - 10 milligrams of diet
17	the rations you are feeding your	beef	, dairy cattle, and sheep are adequa
18	itive business more profitable for	beef	, dairy, and sheep men. The tar
19	o bear. She was ready to kill the	beef	, dress it out, and with vegetables
20	. She had raised a calf, grown it	beef	-fat. She had, with her own work-wea
21	with feeding low-moisture corn in	beef	-feeding programs. Several firms ar
22	he shelf life (at 35 F) of fresh	beef	from 5 days to 5 or 6 weeks. Howeve
23	canned pork products. Tests with	beef	have been largely unsuccessful beca
24	for eggs, pigs to eat garbage, a	beef	herd and wastes of all kinds. Separ
25	their money's worth. A good many	beef	-hungry settlers were accepting the

- This is called a *concordance*.

Contexts for *Chicken* Are also Informative

1	y the irradiated and refrigerated	chicken	. Acceptance of radiopasteurization
2	torehouse". Glendora dropped a	chicken	and a flurry of feathers, and went
3	will specialize in steaks, chops,	chicken	and prime beef as well as Tom's fa
4	ard as the one concerned with the	chicken	and the egg. Which came first? Is
5	he millions of buffalo and prairie	chicken	and the endless seas of grass that
6	"! "Come on, there's some cold	chicken	and we'll see what else". They wen
7	ves to extend the storage life of	chicken	at a low cost of about 0.5 cent per
8	CHICKEN CADILLAC# Use one 6-ounce	chicken	breast for each guest. Salt and pe
9	ion juice, to about half cover the	chicken	breasts. Bake slowly at least one-
10	d, in butter. Sprinkle over top of	chicken	breasts. Serve each breast on a th
11	around, they had a hard time". #	CHICKEN	CADILLAC# Use one 6-ounce chicken
12	successful, and the shelf life of	chicken	can be extended to a month or more
13	ay from making a cake, building a	chicken	coop, or producing a book, to found
14	, they decided, but a deck full of	chicken	coops and pigpens was hardly suita
15	im. "Johnny insisted on cooking a	chicken	dinner in my honor- he's always bee
16	nutes. Kid Ory, the trombonist	chicken	farmer, is also one of the solid a
17	y Johnson reaching around the wire	chicken	fencing, which half covered the tr
18	yes glittering behind dull silver	chicken	fencing. "That was Tee-wah I was t
19	wine in the pot roast or that the	chicken	had been marinated in brandy, and
20	yed this same game and called it "	Chicken	". He could not go through the f
21	f the Mexicans hiding in a little	chicken	house had passed through his head,
22	I'll never forget him cleaning the	chicken	in the tub". A story, no doubt
23	. Organ meats such as beef and	chicken	liver, tongue and heart are planne
24	p. "Miss Sarah, I can't cut up no	chicken	. Miss Maude say she won't". Aga
25	pot. "What is it"? he asked. "	Chicken	", Mose said, and theatrically licke
26	im"? Adam shook his head. "	Chicken	", Mose said. She was a child too m

You Learn Words by Using Distributional Similarity

Consider

- A bottle of pocarisweat is on the table.
- Everybody likes pocarisweat.
- Pocarisweat makes you feel refreshed.
- They make pocarisweat out of ginger.

What does *pocarisweat* mean?

You Know Pocarisweat by the Company It Keeps

From context words humans can guess *pocarisweat* means a beverage like **coke**.
How do you know?

- Other words can occur in the same context
- Those other words are often for beverages (that you drink cold)
- You assume that *pocarisweat* is probably similar

So the intuition is that **two words are similar if they have similar word contexts**.

Word Vectors from Word Distributions

Why Vector Models of Meaning?

- **Computing similarity between words:**
  - *fast* is similar to *rapid*
  - *tall* is similar to *height*
- **Application 1:** Question answering—
  - Question: “How *tall* is Mt. Everest?”
  - Candidate A: “The official *height* of Mount Everest is 29029 feet.”
- **Application 2:** Plagiarism detection—
  - Mainframes are **primarily** referred to **large** computers with **rapid**, advanced processing capabilities that **can execute** and perform tasks **equivalent to many** Personal Computers ...
  - Mainframes **usually** are referred those computers with **fast**, advanced processing capabilities that could **perform** by itself tasks that **may require a lot of** Personal Computers ...
- **Application 3:** ...

Word Similarity Can Be Used to Detect Plagiarism

MAINFRAMES

Mainframes are primarily referred to large computers with rapid, advanced processing capabilities that can execute and perform tasks equivalent to many Personal Computers (PCs) machines networked together. It is characterized with high quantity Random Access Memory (RAM), very large secondary storage devices, and high-speed processors to cater for the needs of the computers under its service.

Consisting of advanced components, mainframes have the capability of running multiple large applications required by many and most enterprises and organizations. This is one of its advantages. Mainframes are also suitable to cater for those applications (programs) or files that are of very high demand by its users (clients). Examples of such organizations and enterprises using mainframes are online shopping websites such as Ebay, Amazon, and computing-giant

MAINFRAMES

Mainframes usually are referred those computers with fast, advanced processing capabilities that could perform by itself tasks that may require a lot of Personal Computers (PC) Machines. Usually mainframes would have lots of RAMs, very large secondary storage devices, and very fast processors to cater for the needs of those computers under its service.

Due to the advanced components mainframes have, these computers have the capability of running multiple large applications required by most enterprises, which is one of its advantage. Mainframes are also suitable to cater for those applications or files that are of very large demand by its users (clients). Examples of these include the large online shopping websites -i.e. : Ebay, Amazon, Microsoft, etc.

There are Two Kinds of Vector Models

- Sparse vector representations:
  - Mutual-information weighted word **co-occurrence matrices**
- Dense vector representations:
  - **Singular value decomposition** (and Latent Semantic Analysis)
  - **Neural-network-inspired models** (skip-gram, CBOW)
  - **Brown clusters**

Shared Intuition: Words are Vectors of Numbers Representing Meaning

- Model the meaning of a word by “embedding” it in a vector space.
- The meaning of a word is a vector of numbers:
  - Vector models are also called **embeddings**
  - Often, the word *embedding* is reserved for *dense* vector representations
- In contrast, word meaning is represented in many (early) NLP applications by a vocabulary index (“word number 545”; compare to one-hot representations)

Preliminaries: Term-Document Matrices

Term-document Matrix

- Each cell is the count of term t in a document d ($tf_{t,d}$).
- Each document is a count vector in $\mathbb{N}^V$, a column below.

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5
clown	6	117	0	0

Term-document Matrix

- Two documents are similar of their vectors are similar.

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5
clown	6	117	0	0

Term-document Matrix

Each word is a count vector in $\mathbb{N}^D$ — a row below

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5
clown	6	117	0	0

Term-document Matrix

Two words are similar if their vectors are similar.

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5
clown	6	117	0	0

Term-context Matrix for Word Similarity

Two words are similar if their **context vectors** are similar.

	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

This gets us to the main event!

Word-Word Matrices

Word–Word or Word–Context Matrix

- Instead of entire documents, use smaller contexts
  - Paragraph
  - Window of a few words (e.g. 3, 5, 7):

A	soothsayer	bids	you	beware	the	Ides	of	March	.
				w_{t-2}	w_{t-1}	w_t	w_{t+1}	w_{t+2}	

- A word is now defined by a vector over counts of words in context.
  - If a word w_j occurs in the context of w_i, increase $count_{ij}$.
- Assuming we have V words,
  - Each vector is now of length V.
  - The word-word matrix is $V \times V$.

Sample Contexts of ± 7 Words

sugar, a sliced lemon, a tablespoonful of	**apricot**	preserve or 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
⋮						

The Word–Word Matrix

We showed only a 4×6 matrix, but the real matrix is $50,000 \times 50,000$.

- So it is very sparse: Most values are 0.
- That's OK, since there are lots of efficient algorithms for sparse matrices.

The size of windows depends on the goals:

- The smaller the context ($\pm 1 - 3$) , the more syntactic the representation
- The larger the context ($\pm 4 - 10$), the more semantic the representation

Types of Co-occurence between Two Words

First-order co-occurrence (syntagmatic association):

- They are typically nearby each other.
- *wrote* is a first-order associate of *book* or *poem*.

Second-order co-occurrence (paradigmatic association):

- They have similar neighbors.
- *wrote* is a second-order associate of words like *said* or *remarked*.

Positive Pointwise Mutual Information

Problem with Raw Counts

- Raw word frequency is not a great measure of association between words.
- It is very skewed: “the” and “of” are very frequent, but maybe not the most discriminative.
- We would rather have a measure that asks whether a context word is **particularly informative** about the target word.

Positive Pointwise Mutual Information (PPMI)

Pointwise Mutual Information

Pointwise Mutual Information: Do events x and y co-occur more than if they were independent.

$$\mathrm{PMI}(x;y) = \log_2 \frac{p(x,y)}{p(x)p(y)}$$

PMI between two words: Do target word w and context word c co-occur more than if they were independent.

$$\mathrm{PMI}(w;c) = \log_2 \frac{p(w,c)}{p(w)p(c)}$$

Pointwise Mutual Information

- Consider

x	y	$p(x, y)$
0	0	0.1
0	1	0.7
1	0	0.15
1	1	0.05

	$p(x)$	$p(y)$
0	0.8	0.25
1	0.2	0.75

- then
  - $\mathrm{PMI}(x = 0; y = 0) = -1$
  - $\mathrm{PMI}(x = 0; y = 1) = 0.222392$
  - $\mathrm{PMI}(x = 1; y = 0) = 1.584963$
  - $\mathrm{PMI}(x = 1; y = 1) = -1.584963$

- In computational linguistics, PMI has been used for finding collocations and associations between words.

word 1	*word 2*	*count word 1*	*count word 2*	count of co-occurrences	PMI
puerto	rico	1938	1311	1159	10.0349081703
hong	kong	2438	2694	2205	9.72831972408
los	angeles	3501	2808	2791	9.56067615065
carbon	dioxide	4265	1353	1032	9.09852946116
prize	laureate	5131	1676	1210	8.85870710982
san	francisco	5237	2477	1779	8.83305176711
nobel	prize	4098	5131	2498	8.68948811416
ice	hockey	5607	3002	1933	8.6555759741
star	trek	8264	1594	1489	8.63974676575
car	driver	5578	2749	1384	8.41470768304
it	the	283891	3293296	3347	-1.72037278119
are	of	234458	1761436	1019	-2.09254205335
this	the	199882	3293296	1211	-2.38612756961
is	of	565679	1761436	1562	-2.54614706831
and	of	1375396	1761436	2949	-2.79911817902
a	and	984442	1375396	1457	-2.92239510038
in	and	1187652	1375396	1537	-3.05660070757
to	and	1025659	1375396	1286	-3.08825363041
to	in	1025659	1187652	1066	-3.12911348956
of	and	1761436	1375396	1190	-3.70663100173

Positive Pointwise Mutual Information

- PMI ranges from $-\infty$ to $+\infty$
- But the negative values are problematic:
  - Things are co-occurring less than we expect by chance
  - Unreliable without enormous corpora
    - Imagine w_1 and w_2 whose probability is each 10^{-6}.
    - Hard to be sure $p(w_1, w_2)$ is significantly different than 10^{-12}.
  - Furthermore it's not clear people are good at "unrelatedness".
- So we just replace negative PMI values by 0.

$$\text{PPMI}(w, c) = max\left(\log_2 \frac{p(w, c)}{p(w)p(c)}, 0\right)$$

Computing PPMI on a Term-Context Matrix

- We have matrix F with V rows (words) and C columns (contexts) (in general $C = V$)
- f_{ij} is how many times word w_i co-occurs in the context of the word c_j.

$$p_{ij} = \frac{f_{ij}}{\sum_{i=1}^{V}(\sum_{j=1}^{C} f_{ij})}$$

$$p_{i*} = \frac{\sum_{j=1}^{C} f_{ij}}{\sum_{i=1}^{V}(\sum_{j=1}^{C} f_{ij})} \qquad p_{*j} = \frac{\sum_{i=1}^{V} f_{ij}}{\sum_{i=1}^{V}(\sum_{j=1}^{C} f_{ij})}$$

$$pmi_{ij} = \log_2 \frac{p_{ij}}{p_{i*}p_{*j}} \qquad ppmi_{ij} = \max(pmi_{ij}, 0)$$

Worked Example: Computing PPMI from Term-Context Matrix (Part I)

	computer	data	pinch	result	sugar	
apricot	0	0	1	0	1	2
pineapple	0	0	1	0	1	2
digital	2	1	0	1	0	4
information	1	6	0	4	0	11
	3	7	2	5	2	19

$$p(w = information, c = data) = \frac{6}{19} = 0.32$$

$$p(w = information) = \frac{11}{19} = 0.58 \quad p(c = data) = \frac{7}{19} = 0.32$$

$p(w, c)$

	computer	data	pinch	result	sugar	$p(w)$
apricot	0.00	0.00	0.05	0.00	0.05	0.11
pineapple	0.00	0.00	0.05	0.00	0.05	0.11
digital	0.11	0.05	0.00	0.05	0.00	0.21
information	0.05	0.32	0.00	0.21	0.00	0.58

Worked Example: Computing PPMI from Term-Context Matrix (Part II)

$p(w, c)$

	computer	data	pinch	result	sugar	$p(w)$
apricot	0.00	0.00	0.05	0.00	0.05	0.11
pineapple	0.00	0.00	0.05	0.00	0.05	0.11
digital	0.11	0.05	0.00	0.05	0.00	0.21
information	0.05	0.32	0.00	0.21	0.00	0.58
$p(c)$	0.16	0.37	0.11	0.26	0.11	

$$pmi(information, data) = \log_2 \frac{0.32}{0.37 \cdot 0.57} \approx 0.58$$

$PPMI(w, c)$

	computer	data	pinch	result	sugar
apricot	-	-	2.25	-	2.25
pineapple	-	-	2.25	-	2.25
digital	1.66	0.00	-	0.00	-
information	0.00	0.32	-	0.47	-

Issues with PPMI

- PMI is biased toward infrequent events.
- Very rare words have very high PMI values.
- Two solutions:
    - Give rare words slightly higher probabilities
    - Use add-one smoothing (which has a similar effect)

First Solution: Raise Context Probabilities

- Raise the context probabilities to $\alpha = 0.75$:

$$\text{PPMI}_{\alpha}(w, c) = \max(\log_2 \frac{p(w, c)}{p(w)p_{\alpha}(c)}, 0)$$

$$p_{\alpha}(c) = \frac{p(c)^{\alpha}}{\sum_c p(c)^{\alpha}}$$

- This helps because $p_{\alpha}(c) > p(c)$ for rare c.
- Consider two context words $p(a) = 0.99$ and $p(b) = 0.01$
- $p_{\alpha}(a) = \frac{0.99^{0.75}}{0.99^{0.75}+0.01^{0.75}} = 0.97 \qquad p_{\alpha}(b) = \frac{0.99^{0.75}}{0.99^{0.75}+0.01^{0.75}} = 0.03$

Second Solution: Use Laplace Smoothing

Add-2 Smoothed $Count(w, c)$

	computer	data	pinch	result	sugar
apricot	2	2	3	2	3
pineapple	2	2	3	2	3
digital	4	3	2	3	2
information	3	8	2	6	2

$p(w, c)$ Add-2

	computer	data	pinch	result	sugar	$p(w)$
apricot	0.03	0.03	0.05	0.03	0.05	0.20
pineapple	0.03	0.03	0.05	0.03	0.05	0.20
digital	0.07	0.05	0.03	0.05	0.03	0.24
information	0.05	0.14	0.03	0.10	0.03	0.36
$p(c)$	0.19	0.25	0.17	0.22	0.17	

Comparing PPMI and add-2 Smoothed PPMI

PPMI(w, c)

	computer	data	pinch	result	sugar	
apricot	-	-	**2.25**	-	**2.25**	
pineapple	-	-	**2.25**	-	**2.25**	
digital	1.66	0.00	-	0.00	-	
information	0.00	**0.32**	-	**0.47**	-	-

PPMI(w, c)

	computer	data	pinch	result	sugar
apricot	0.00	0.00	**0.56**	0.00	**0.56**
pineapple	0.00	0.00	**0.56**	0.00	**0.56**
digital	0.62	0.00	0.00	0.00	0.00
information	0.00	**0.58**	0.00	**0.37**	0.00

Measuring Similarity between Word Vectors

Cosine is Used to Measure the Similarity between Word Vectors

- Given two target words represented with vectors $\boldsymbol{v}$ and $\boldsymbol{w}$.
- The **dot product** or **inner product** is usually used as the basis for similarity.

$$\boldsymbol{v} \cdot \boldsymbol{w} = \sum_{i=1}^{N} v_i w_i = v_1 w_1 + v_2 w_2 + \cdots + v_N w_N = |\boldsymbol{v}||\boldsymbol{w}| \cos \theta$$

- $\boldsymbol{v} \cdot \boldsymbol{w}$ is high when two vectors have large values in the same dimensions.
- $\boldsymbol{v} \cdot \boldsymbol{w}$ is low (in fact 0) with zeros in complementary distribution.
- We also do not want the similarity to be sensitive to word-frequency.
- So normalize by vector length and use the cosine as the similarity

$$\cos(\boldsymbol{v}, \boldsymbol{w}) = \frac{\boldsymbol{v} \cdot \boldsymbol{w}}{|\boldsymbol{v}||\boldsymbol{w}|}$$

There Are Other Similarity Measures in the Literature

- **Cosine Similarity**

$$\text{sim}_{cosine}(\mathbf{v}, \mathbf{w}) = \frac{\mathbf{v} \cdot \mathbf{w}}{|\mathbf{v}||\mathbf{w}|}$$

- **Jaccard Similarity**

$$\text{sim}_{Jaccard}(\mathbf{v}, \mathbf{w}) = \frac{\sum_i \min(v_i, w_i)}{\sum_i \max(v_i, w_i)}$$

- **Sørensen-Dice Similarity**

$$\text{sim}_{Dice}(\mathbf{v}, \mathbf{w}) = \frac{2\sum_i \min(v_i, w_i)}{\sum_i (v_i + w_i)}$$

Dense Vectors

Sparse vs. Dense Vectors

- PPMI vectors are
  - **long** (length in 10s of thousands)
  - **sparse** (most elements are 0)
- Alternative: learn vectors which are
  - **short** (length in several hundreds)
  - **dense** (most elements are non-zero)

Dense Vectors Have Three Advantages over Sparse Vectors

1. Short vectors may be **easier to use as features** in machine learning (less weights to tune).
2. Dense vectors may **generalize better** than storing explicit counts.
3. They may do **better at capturing synonymy**:
   - *car* and *automobile* are synonyms
   - But, in sparse vectors, they 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

There are Three Main Methods for Getting Short, Dense Vectors

1. Singular Value Decomposition (SVD)
    - A special case of this is called LSA — **Latent Semantic Analysis**
2. Brown clustering
3. "Neural Language Model"-inspired predictive models.
    - **Word2vec**: skip-gram and continuous bag-of-words (CBOW)
    - **GloVe**
    - **fastText**

Dense Vectors via SVD — Intuition

- Approximate an N-dimensional dataset using fewer dimensions
- By rotating the axes into a new space along the dimension with the most variance
- Then repeat with the next dimension captures the next most variance, etc.
- Many such (related) methods:
    - PCA — principle components analysis
    - Factor Analysis
    - SVD

Embeddings vs. Sparse Vectors

- Dense SVD embeddings sometimes work better than sparse PPMI matrices at tasks like word similarity
- Denoising: low-order dimensions may represent unimportant information
- Truncation may help the models generalize better to unseen data.
- Having a smaller number of dimensions may make it easier for classifiers to properly weight the dimensions for the task.
- Dense models may do better at capturing higher order co-occurrence.

Word2vec: Skip-gram and Continuous Bag of Words (CBOW)

Embeddings Inspired by Neural Language Models

- Skip-gram and CBOW learn embeddings as part of the process of word prediction.
- Train a neural network to predict neighboring words
  - Inspired by neural net language models.
  - In so doing, learn dense embeddings for the words in the training corpus.
- Advantages:
  - Fast, easy to train (much faster than SVD).
  - Available online in the `word2vec` package.
  - Including sets of pretrained embeddings!

Skip-Gram

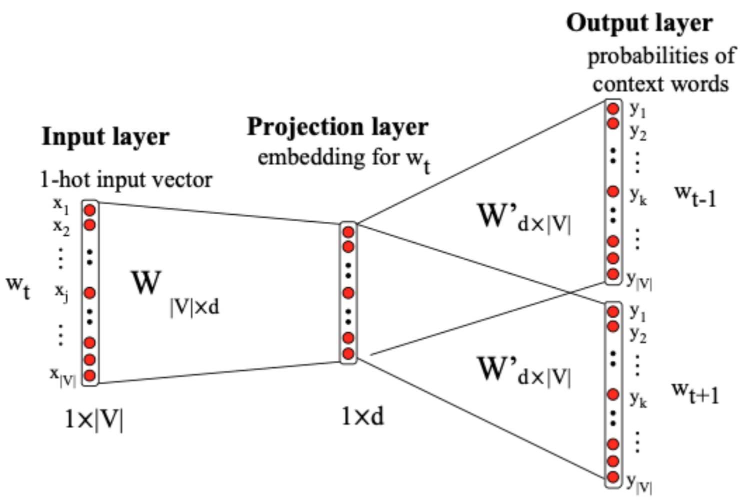
- From the current word t, predict other words in a context window of $2C$ words.
- For example, for $c = 2$, we are given t and we are predicting one of the words in

$$[c_1, c_2, c_3, c_4]$$

A	soothsayer	bids	you	beware	the	Ides	of	March	.
				c_1	c_2	t	c_3	c_4	

.

- We will train a classifier on a binary prediction task:
  - **Is the word c likely to show up near the word t?**
  - **It turns our we do not care about the prediction, but in the classifier itself!**



- **Input Layer** The target word w_t represented as a **one-hot** vector
- **Projection Layer** A vector (embedding) learned for w_t via back-propagation
- **Output Layer** Probability distributions over the vocabulary y for the context words w_{t-1} and w_{t+1}

The Context Classifier

- Suppose we are given a context word c and a center word t with embeddings $\vec{c}$ and $\vec{t}$.
- We want c and t to be “similar”, that is we want

$$similarity(\vec{t}, \vec{c}) = \vec{t} \cdot \vec{c}$$

  to be high.
- We will however frame this in terms of probabilities: we want $p(c \mid t)$ to be high if c is a context word of t and low if it is not.
- But $\vec{t} \cdot \vec{c}$ is not a probability measure.
- Worse, we do not really know what $\vec{t}$ and $\vec{c}$ are.

One-hot Vector Representation

- Since we do not know what the embeddings are, we will initially represent words with *one-hot vectors*.
- Each vector has length $|V|$.
- Each vector has a single entry 1 and all other entries are 0.
- So if "invigilate" is word 5, the one-hot vector is

$$[0, 0, 0, 0, 1, 0, 0, 0, 0, \ldots, 0]$$

The Classifier

- The classifier first "compresses" the input one-hot vector $\vec{t}$ to a d-dimensional vector, $\vec{h}$.
- We achieve this by multiplying a $|V| \times d$ matrix W with $\vec{t}$, to get the $\vec{h}$. For example:

$$\begin{bmatrix} 10 \\ 12 \\ 19 \end{bmatrix}_{d\times 1} = \begin{bmatrix} 17 & 23 & 4 & 10 & 11 \\ 24 & 5 & 6 & 12 & 18 \\ 1 & 7 & 13 & 19 & 25 \end{bmatrix}_{d\times|V|} \times \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}_{|V|\times 1}$$

- We then compute $C_{|V|\times d} \times \vec{h}_{d\times 1}$ to get $\vec{x}_{|V|\times 1}$.
- If $\vec{h}$ was the embedding for t and the rows of C were embedding vectors for context words then

$$x_j = similarity(\vec{h}, \vec{c_j})$$

that is, the elements of $\vec{x}$ are similarities of all words with t (via the embedding $\vec{h}$).

$$\underbrace{\begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_d \end{bmatrix}}_{\vec{h}} = \underbrace{\begin{bmatrix} w_{11} & \dots & w_{1|V|} \\ \vdots & \ddots & \vdots \\ w_{d1} & \dots & w_{d|V|} \end{bmatrix}}_{W} \times \underbrace{\begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ t_{|V|} \end{bmatrix}}_{one-hot\ \vec{t}}$$

$$\underbrace{\begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_{|V|} \end{bmatrix}}_{\text{“one–hot”}\ output} \leftarrow \boxed{\text{SoftMax}} \leftarrow \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{|V|} \end{bmatrix}}_{\vec{x}} = \underbrace{\begin{bmatrix} c_{11} & \dots & c_{1d} \\ \vdots & \ddots & \vdots \\ c_{|V|1} & \dots & c_{|V|d} \end{bmatrix}}_{C} \times \underbrace{\begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_d \end{bmatrix}}_{\vec{h}} \leftarrow$$

Outputs as Probabilities

- Except, the outputs ($\vec{x}$) are not probabilities!
- We use the same scaling idea we used earlier and then use *softmax* .

$$p(c_j \text{ is in the context of } t) = \frac{exp(x_j)}{\sum_i exp(x_i)}$$

Training for Embeddings

- We do not know what W and C are. So we learn them through an iterative process.
- We use a large corpus as a training data
- We also randomly sample the corpus to find words are NOT in the context – negative sampling.

A	soothsayer	bids	you	beware	the	Ides	of	March	.
				c_1	c_2	t	c_3	c_4	

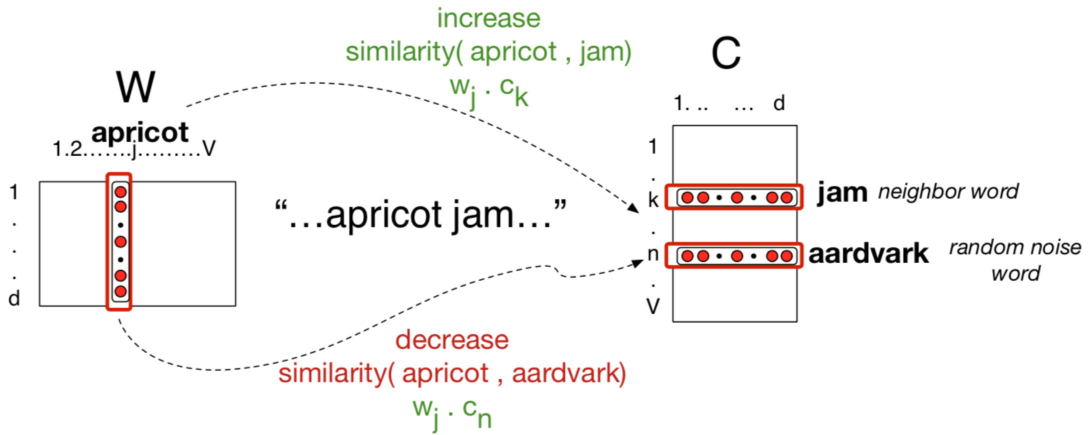
Positive Examples		Negative Examples			
t	c	t	c	t	c
ides	beware	ides	aardvark	ides	twelve
ides	of	ides	puddle	ides	hello
ides	March	ides	where	ides	dear
ides	the	ides	coaxial	ides	forever

- There are many subtleties to sampling the negative examples!

Skip-gram, trained in this way, is called SGNS—**skip-gram with negative sampling**—and is what people usually mean when they say “skip-gram.”

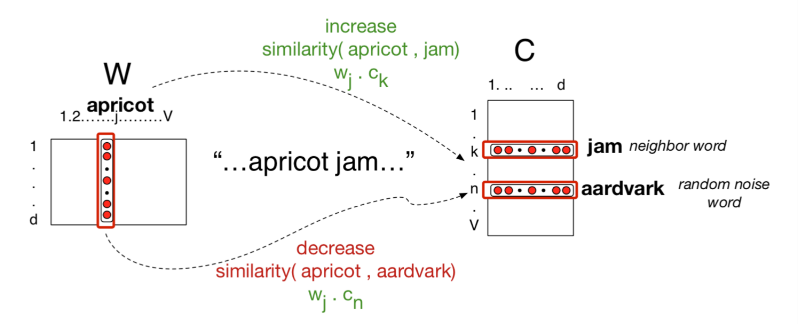
Training for Embeddings

- We define an error function $L(W, C) =$ is the error the classifier makes when presented with the one-hot vectors for a t and a c.
- Note that L has $2 \times |V| \times d$ parameters, the total number of entries of the two matrices.
- We start with randomly initialized W and C matrices.
- The **logistic regression/stochastic gradient descent algorithm** perturbs these parameters (w_{ij} and c_{ij}) in a direction to reduce the error by *backpropagation*, about which we will learn more next lecture!
- Details are not important at this time.


increase
similarity(apricot , jam)
$w_j \cdot c_k$
W
apricot
1.2…….j………V
1
.
.
.
d
"…apricot jam…"
C
1. ..   … d
1
.
k
.
n
.
V
jam *neighbor word*
aardvark *random noise word*
decrease
similarity(apricot , aardvark)
$w_j \cdot c_n$

Extracting the Embeddings

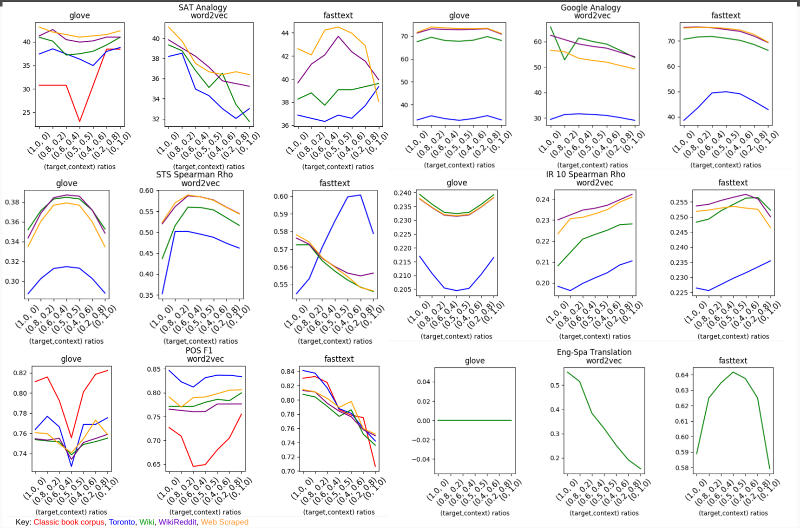
- When the training converges, we have two embeddings for each word w_i
  - **The** i^{th} **column vector of the** W **matrix** (target embeddings), and
  - **The** i^{th} **row vector of the** C **matrix** (context embeddings)



- We can throw away C and just use the column vectors in W as embeddings.
- We can add the two embeddings.
- We can concatenate the two embeddings to get an embedding of size $2d$.

Throwing Away Context Embeddings is not Always Optimal

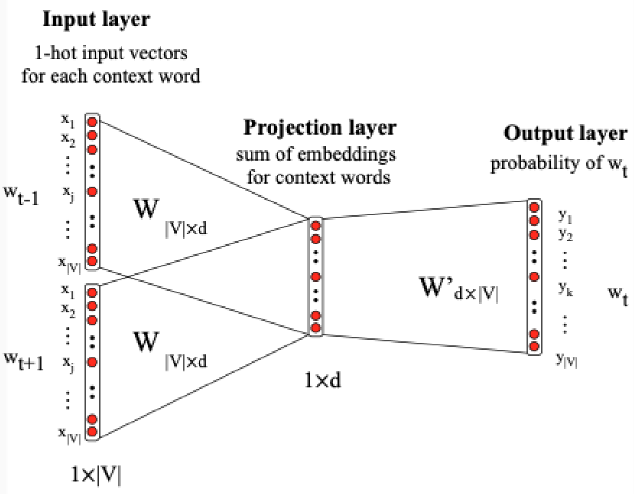
- For skip-gram, the default strategy is often (though not always) best
- For other kinds of embeddings—fastText and GloVe—this is not the case



Robinson, Fulda, and Mortensen (in progress)

CBOW Is Roughly the Mirror Image of Skip-Gram

- **CBOW** stands for **Continuous Bag of Words**
- **Training objective:** Predict the target word, given the context words
- CBOW converges faster than Skip-Gram in training
- CBOW captures syntactic relationships better
- Skip-gram captures semantic relationships better



Tools and Resources

There are Tools and Resources Available for Training and Using Embeddings

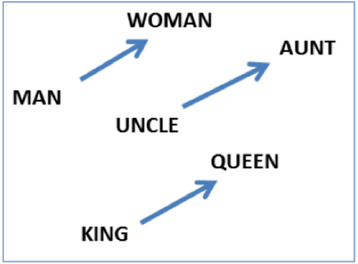
- Pretrained embeddings
    - Skip-gram
    - CBOW
    - fastText
    - GloVe
- Training your own embeddings
    - You can easily train skip-gram, CBOW, and fastText embeddings with **`gensim`**
    - Straightforward Python interface

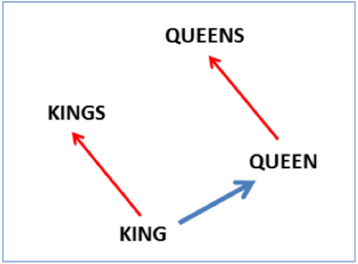
Observations on Embeddings

- Nearest words to some embeddings in the $d-$ dimensional space.

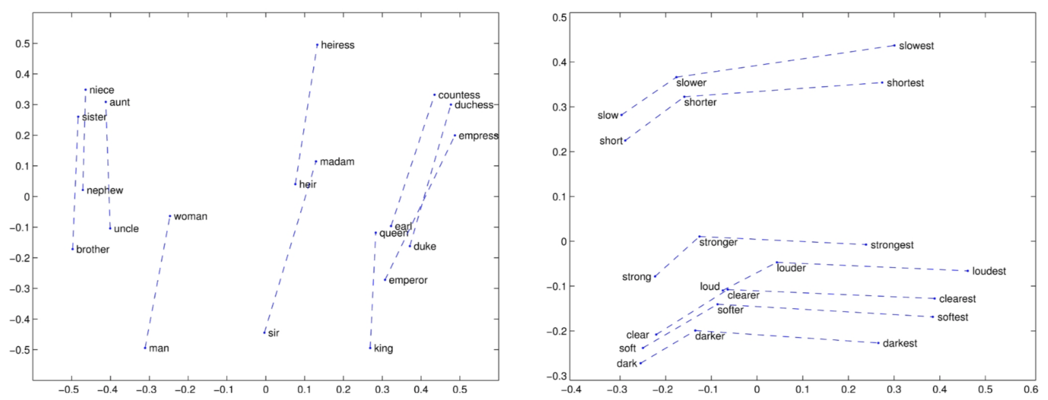
target:	Redmond	Havel	ninjutsu	graffiti	capitulate
	Redmond Wash.	Vaclav Havel	ninja	spray paint	capitulation
	Redmond Washington	president Vaclav Havel	martial arts	grafitti	capitulated
	Microsoft	Velvet Revolution	swordsmanship	taggers	capitulating

- Relation meanings
  - $vector(king) - vector(man) + vector(woman) \approx vector(queen)$
  - $vector(Paris) - vector(France) + vector(Italy) \approx vector(Rome)$





Observations on Embeddings



Questions?