CS447: Natural Language Processing
http://courses.engr.illinois.edu/cs447

Lecture 17: Vector-space semantics (distributional similarities)

Julia Hockenmaier
juliahmr@illinois.edu
3324 Siebel Center

Where we're at

We have looked at how to obtain the meaning of sentences from the meaning of their words (represented in predicate logic).

Now we will look at how to represent the meaning of words (although this won't be in predicate logic)

We will consider different tasks:
- Computing the semantic similarity of words by representing them in a vector space
- Finding groups of similar words by inducing word clusters
- Identifying different meanings of words by word sense disambiguation

What we're going to cover today

Pointwise mutual information
A very useful metric to identify events that frequency co-occur

Distributional (Vector-space) semantics:
Measure the semantic similarity of words in terms of the similarity of the contexts in which the words appear

- The distributional hypothesis
- Representing words as (sparse) vectors
- Computing word similarities

Using PMI to identify words that "go together"

Discrete random variables

A discrete random variable X can take on values $\{x_1,\ldots, x_n\}$ with probability $p(X = x_i)$

A note on notation:
$p(X)$ refers to the distribution, while $p(X = x_i)$ refers to the probability of a specific value x_i. $p(X = x_i)$ also written as $p(x_i)$

In language modeling, the random variables correspond to words W or to sequences of words $W^{(1)}\ldots W^{(n)}$.

Another note on notation:
We're often sloppy in making the distinction between the *i*-th word [token] in a sequence/string, and the *i*-th word [type] in the vocabulary clear.

Mutual information $I(X;Y)$

Two random variables X, Y are **independent** iff their joint distribution is equal to the product of their individual distributions:

$$p(X, Y) = p(X)p(Y)$$

That is, for all outcomes x, y:

$$p(X=x, Y=x) = p(X=x)p(Y=y)$$

$I(X;Y)$, the **mutual information** of two random variables X and Y is defined as

$$I(X;Y) = \sum_{X,Y} p(X=x, Y=y) \log \frac{p(X=x, Y=y)}{p(X=x)p(Y=y)}$$

Pointwise mutual information (PMI)

Recall that two **events** x, y are **independent** if their joint probability is equal to the product of their individual probabilities:

x,y are independent iff $p(x,y) = p(x)p(y)$

x,y are independent iff $p(x,y)/p(x)p(y) = 1$

In NLP, we often use the pointwise mutual information (PMI) of two outcomes/events (e.g. words):

$$PMI(x, y) = \log \frac{p(X=x, Y=y)}{p(X=x)p(Y=y)}$$

Using PMI to find related words

Find pairs of words w_i, w_j that have high pointwise mutual information:

$$PMI(w_i, w_j) = \log \frac{p(w_i, w_j)}{p(w_i)p(w_j)}$$

Different ways of defining $p(w_i, w_j)$ give different answers.

Using PMI to find "sticky pairs"

$p(w_i, w_j)$: probability that w_i, w_j are adjacent

Define $p(w_i, w_j) = p(\text{“}w_i w_j\text{”})$

High PMI word pairs under this definition:

Humpty Dumpty, Klux Klan, Ku Klux, Tse Tung,
avant garde, gizzard shad, Bobby Orr, mutatis mutandis,
Taj Mahal, Pontius Pilate, ammonium nitrate,
jiggery pokery, anciens combattants, fuddle duddle,
helter skelter, mumbo jumbo
(and a few more)

Back to lexical semantics…

Different approaches to lexical semantics

Lexicographic tradition:
- Use lexicons, thesauri, ontologies
- Assume words have discrete word senses:
  bank1 = financial institution; bank2 = river bank, etc.
- May capture explicit relations between word (senses):
  "dog" is a "mammal", etc.

Distributional tradition:
- Map words to (sparse) vectors that capture corpus statistics
- Contemporary variant: use neural nets to learn dense vector "embeddings" from very large corpora
  (this is a prerequisite for most neural approaches to NLP)
- This line of work often ignores the fact that words have multiple senses or parts-of-speech

Vector representations of words

"Traditional" distributional similarity approaches represent words as sparse vectors [today's lecture]
- Each dimension represents one specific context
- Vector entries are based on word-context co-occurrence statistics (counts or PMI values)

Alternative, dense vector representations:
- We can use Singular Value Decomposition to turn these sparse vectors into dense vectors (Latent Semantic Analysis)
- We can also use neural models to explicitly learn a dense vector representation (embedding) (word2vec, Glove, etc.)

Sparse vectors = most entries are zero
Dense vectors = most entries are non-zero

Distributional Similarities

Measure the semantic similarity of words
in terms of the similarity of the contexts
in which the words appear

Represent words as vectors

Why do we care about word similarity?

Question answering:

Q: "How *tall* is Mt. Everest?"
Candidate A: "The official *height* of Mount Everest is 29029 feet"

"tall" is similar to *"height"*

Why do we care about word similarity?

Plagiarism detection

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.

Why do we care about word contexts?

What is tezgüino?

A bottle of tezgüino is on the table.
Everybody likes tezgüino.
Tezgüino makes you drunk.
We make tezgüino out of corn.
(Lin, 1998; Nida, 1975)

The contexts in which a word appears
tells us a lot about what it means.

The Distributional Hypothesis

Zellig 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."

John R. Firth 1957:

You shall know a word by the company it keeps.

The contexts in which a word appears
tells us a lot about what it means.

Words that appear in similar contexts have similar meanings

Exploiting context for semantics

Distributional similarities (vector-space semantics):
Use the set of contexts in which words (= word types) appear to measure their similarity

Assumption: Words that appear in similar contexts (*tea, coffee*) have similar meanings.

Word sense disambiguation (future lecture)
Use the context of a particular occurrence of a word (token) to identify which sense it has.

Assumption: If a word has multiple distinct senses (e.g. *plant*: *factory* or *green plant*), each sense will appear in different contexts.

Distributional similarities

Distributional similarities

Distributional similarities use the set of contexts in which words appear to measure their similarity.

They represent each word *w* as a vector $\mathbf{w}$

$$\mathbf{w} = (w_1, \ldots, w_N) \in \mathbf{R}^N$$

in an N-dimensional vector space.

- Each dimension corresponds to a particular context c_n
- Each element w_n of $\mathbf{w}$ captures the degree to which the word *w* is associated with the context c_n.
- w_n depends on the co-occurrence counts of *w* and c_n

The similarity of words *w* and *u* is given by the similarity of their vectors $\mathbf{w}$ and $\mathbf{u}$

Documents as contexts

Let's assume our corpus consists of a (large) number of documents (articles, plays, novels, etc.)

In that case, we can define the contexts of a word as the sets of documents in which it appears.

Conversely, we can represent each document as the (multi)set of words which appear in it.

- Intuition: Documents are similar to each other if they contain the same words.
- This is useful for information retrieval, e.g. to compute the similarity between a query (also a document) and any document in the collection to be searched.

Term-Document Matrix

	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

A Term-Document Matrix is a 2D table:

- Each cell contains the frequency (count) of the term (word) *t* in document *d*: $tf_{t,d}$
- Each column is a vector of counts over words, representing a document
- Each row is a vector of counts over documents, representing a word

Term-Document Matrix

	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

Two documents are similar if their vectors are similar
Two words are similar if their vectors are similar

What is a 'context'?

There are many different definitions of context that yield different kinds of similarities:

Contexts defined by nearby words:

How often does *w* appear near the word *drink*?
Near = "*drink* appears within a window of ±k words of *w*", or "*drink* appears in the same document/sentence as *w*"
This yields fairly broad thematic similarities.

Contexts defined by grammatical relations:

How often is (the noun) *w* used as the subject (object) of the verb *drink*? (Requires a parser).
This gives more fine-grained similarities.

Using nearby words as contexts

- Decide on a fixed vocabulary of N context words $c_1..c_N$
  Context words should occur frequently enough in your corpus that you get reliable co-occurrence counts, but you should ignore words that are too common ('stop words': *a, the, on, in, and, or, is, have,* etc.)
- Define what 'nearby' means
  For example: *w* appears near *c if* c appears within ±5 words of *w*
- Get co-occurrence counts of words *w* and contexts *c*
- Define how to transform co-occurrence counts of words *w* and contexts *c* into vector elements w_n
  For example: compute (positive) PMI of words and contexts
- Define how to compute the similarity of word vectors
  For example: use the cosine of their angles.

Defining and counting co-occurrence

Defining co-occurrences:

- **Within a fixed window**: v_i occurs within ±n words of *w*
- **Within the same sentence:** requires sentence boundaries
- **By grammatical relations**:
  v_i occurs as a subject/object/modifier/… of verb *w*
  (requires parsing - and separate features for each relation)

Counting co-occurrences:

- f_i as **binary features** (1,0): *w* does/does not occur with v_i
- f_i as **frequencies**: w occurs *n* times with v_i
- f_i as **probabilities**:
  e.g. f_i is the probability that v_i is the subject of *w*.

Getting co-occurrence counts

Co-occurrence as a binary feature:

Does word w ever appear in the context c? (1 = yes/0 = no)

	arts	boil	data	function	large	sugar	water
apricot	0	1	0	0	1	1	1
pineapple	0	1	0	0	1	1	1
digital	0	0	1	1	1	0	0
information	0	0	1	1	1	0	0

Co-occurrence as a frequency count:

How often does word w appear in the context c? (0…n times)

	arts	boil	data	function	large	sugar	water
apricot	0	1	0	0	5	2	7
pineapple	0	2	0	0	10	8	5
digital	0	0	31	8	20	0	0
information	0	0	35	23	5	0	0

Typically: 10K-100K dimensions (contexts), very sparse vectors

Counts vs PMI

Sometimes, low co-occurrences counts are very informative, and high co-occurrence counts are not:

- Any word is going to have relatively high co-occurrence counts with very common contexts (e.g. "it", "anything", "is", etc.), but this won't tell us much about what that word means.
- We need to identify when co-occurrence counts are more likely than we would expect by chance.

We therefore want to use PMI values instead of raw frequency counts:

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

But this requires us to define $p(\mathrm{w},\mathrm{c})$, $p(\mathrm{w})$ and $p(\mathrm{c})$

Word-Word Matrix

Context: ± 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

Resulting word-word matrix:

f(w, c) = how often does word w appear in context c:
"information" appeared six times in the context of "data"

	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	

$$p_{ij} = \frac{f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}}$$

$$p(w_i) = \frac{\sum_{j=1}^{C} f_{ij}}{N}$$

$$p(c_j) = \frac{\sum_{i=1}^{W} f_{ij}}{N}$$

Count(w,context)

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

p(w=information, c=data) = 6/19 = .32
p(w=information) = 11/19 = .58
p(c=data) = 7/19 = .37

p(w,context)

	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(context)	0.16	0.37	0.11	0.26	0.11	

Computing PMI of *w* and *c*: Using a fixed window of ± k words

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

N: How many tokens does the corpus contain?
$f(w) \le N$: How often does w occur?
$f(w, c) \le f(w,)$: How often does w occur with c in its window?
$f(c) = \sum_w f(w, c) \le N$: How many tokens have c in their window?

$p(w) = f(w)/N$
$p(c) = f(c)/N$
$p(w, c) = f(w, c)/N$

Computing PMI of *w* and *c*: *w* and *c* in the same sentence

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

N: How many sentences does the corpus contain?
$f(w) \le N$: How many sentences contain w?
$f(w, c) \le f(w)$: How many sentences contain w and c?
$f(c) \le N$: How many sentences contain c?

$p(w) = f(w)/N$
$p(c) = f(c)/N$
$p(w, c) = f(w, c)/N$

Using grammatical features

Observation: verbs have 'selectional preferences':

E.g. "eat" takes edible things as objects and animate entities as subjects.

Exceptions: metonymy (*"The VW honked at me"*) and metaphors: *"Skype ate my credit"*

This allows us to induce noun classes:

Edible things occur as objects of "eat".

In general, nouns that occur as subjects/objects of specific verbs tend to be similar.

This also allows us to induce verb classes:

Verbs that take the same class of nouns as arguments tend to be similar/related.

Example: frequencies of grammatical relations

64M word corpus, parsed with Minipar (Lin, 1998)

	cell
sbj of *absorb*	1
sbj of *adapt*	1
sbj of *behave*	1
...	...
mod of *abnormality*	3
mod of *anemia*	8
...	
obj of *attack*	6
obj of *call*	11
...	

Measuring association with context

- Every **element** f_i of the co-occurrence vector corresponds to some **word *w'*** (and possibly a relation *r*):
  e.g. (r,w')= (obj-of, *attack*)
- The **value** of f_i should indicate the **association strength** between (*r, w'*) and *w*.
- What **value** should feature f_i for word *w* have?
  Probability $P(f_i \mid w)$: f_i will be high for any frequent feature (regardless of w)

Frequencies vs. PMI

Objects of 'drink' *(Lin, 1998)*

	Count	**PMI**
bunch beer	2	12.34
tea	2	11.75
liquid	2	10.53
champagne	4	11.75
anything	3	5.15
it	3	1.25

Positive Pointwise Mutual Information

PMI is negative when words co-occur less than expected by chance.

This is unreliable without huge corpora:

With $P(w_1) \approx P(w_2) \approx 10^{-6}$, we can't estimate whether $P(w_1,w_2)$ is significantly different from 10^{-12}

We often just use positive PMI values, and replace all PMI values < 0 with 0:

Positive Pointwise Mutual Information (PPMI):

PPMI(w,c) = PMI(w,c) if PMI(w,c) > 0

= 0 if PMI(w,c) ≤ 0

PMI and smoothing

PMI is biased towards infrequent events:

If $P(\mathrm{w}, \mathrm{c}) = P(\mathrm{w}) = P(\mathrm{c})$, then $\mathrm{PMI}(\mathrm{w},\mathrm{c}) = \log(1/P(\mathrm{w}))$

So $\mathrm{PMI}(\mathrm{w}, \mathrm{c})$ is larger for rare words w with low $P(\mathrm{w})$.

Simple remedy: Add-k smoothing of $P(\mathrm{w}, \mathrm{c}), P(\mathrm{w}), P(\mathrm{c})$ pushes all PMI values towards zero.

Add-k smoothing affects low-probability events more, and will therefore reduce the bias of PMI towards infrequent events.

(Pantel & Turney 2010)

Vector similarity

In distributional models, every word is a point in *n*-dimensional space.

How do we measure the similarity between two points/vectors?

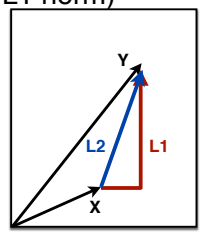
In general:

- **Manhattan distance** (Levenshtein distance, L1 norm)

$$dist_{L1}(\vec{x},\vec{y}) = \sum_{i=1}^{N} |x_i - y_i|$$

- **Euclidian distance** (L2 norm)

$$dist_{L2}(\vec{x},\vec{y}) = \sqrt{\sum_{i=1}^{N} (x_i - y_i)^2}$$



Dot product as similarity

If the vectors consist of simple binary features (0,1), we can use the **dot product** as **similarity metric**:

$$sim_{dot-prod}(\vec{x},\vec{y}) = \sum_{i=1}^{N} x_i \times y_i$$

The dot product is a bad metric if the vector elements are arbitrary features: it prefers **long** vectors

- If one x_i is very large (and y_i nonzero), ***sim(x,y)*** gets very large
  If the number of nonzero x_i and y_is is very large, ***sim(x,y)*** gets very large.
- Both can happen with frequent words.

$$\text{length of } \vec{x} : |\vec{x}| = \sqrt{\sum_{i=1}^{N} x_i^2}$$

Vector similarity: Cosine

One way to define the similarity of two vectors is to use the cosine of their angle.

The cosine of two vectors is their dot product, divided by the product of their lengths:

$$\text{sim}_{cos}(\vec{x},\vec{y}) = \frac{\sum_{i=1}^{N} x_i \times y_i}{\sqrt{\sum_{i=1}^{N} x_i^2}\sqrt{\sum_{i=1}^{N} y_i^2}} = \frac{\vec{x}\cdot\vec{y}}{|\vec{x}||\vec{y}|}$$

sim(**w**, **u**) = 1: **w** and **u** point in the same direction
sim(**w**, **u**) = 0: **w** and **u** are orthogonal
sim(**w**, **u**) = –1: **w** and **u** point in the opposite direction

Kullback-Leibler divergence

When the vectors **x** are probabilities, i.e. $x_i = P(f_i \mid w_x)$, we can measure the distance between the two distributions **P** and **Q**

The standard metric is **Kullback-Leibler divergence** $D(P||Q)$

$$\text{D}(P||Q) = \sum_x P(x) \log \frac{P(x)}{Q(x)}$$

But KL divergence is not very good because it is
- **Undefined** if $P(x)=0$ and $Q(x) \neq 0$.
- **Asymmetric**: $D(P||Q) \neq D(Q||P)$

Jensen/Shannon divergence

Instead, we use the **Jensen/Shannon divergence:** the distance of each distribution from their average.

- **Average** of *P* and *Q*: $Avg_{P,Q}(x) = \frac{P(x)+Q(x)}{2}$
- **Jensen/Shannon divergence** of *P* and *Q*:

$$JS(P||Q) = D(P||Avg_{P,Q}) + D(Q||Avg_{P,Q})$$

- As a distance measure between **x,y** (with $x_i = P(f_i \mid w_x)$)

$$\text{dist}_{JS}(\vec{x},\vec{y}) = \sum_i x_i \log_2 \left(\frac{x_i}{(x_i+y_i)/2}\right) + y_i \log_2 \left(\frac{y_i}{(x_i+y_i)/2}\right)$$

More recent developments

Neural embeddings

There is a lot of recent work on neural-net based word embeddings:

word2vec, https://code.google.com/p/word2vec/

Glove http://nlp.stanford.edu/projects/glove/

etc.

Using the vectors produced by these word embeddings instead of the raw words themselves can be very beneficial for many tasks.

This is currently a very active area of research.

Analogies

It can be shown that for some of these embeddings, the learned word vectors can capture analogies:

Queen::King = Woman::Man

In the vector representation: $queen \approx king - man + woman$

Similar results for e.g. countries and capitals:
Germany::Berlin = France::Paris

"Semantic spaces"?

Does this mean that these vector spaces represent semantics?

Yes, but only to some extent.

- Different context definitions (or embeddings) give different vector spaces with different similarities
- Often, antonyms (hot/cold, etc.) have very similar vectors.
- Vector spaces are not well-suited to capturing hypernym relations (every dog is an animal)

We will get back to that when we talk more about lexical semantics.

Another open problem: how to get from words to the semantics of sentences

Today's key concepts

Distributional hypothesis

Distributional similarities:

word-context matrix
representing words as vectors
positive PMI
computing the similarity of word vectors