FNLP
Distributional Semantics

Yansong Feng
fengyansong@pku.edu.cn

Wangxuan Institute of Computer Technology
Peking University

April 9, 2025

The Meaning of a Word

Q: What's the meaning of life?

The Meaning of a Word

Q: What's the meaning of life?

A: LIFE

The Meaning of a Word

Q: What's the meaning of life?

A: LIFE

– A linguistics joke by Barbara Partee, 1967

The Meaning of a Word

Q: What's the meaning of life?

A: LIFE

– A linguistics joke by Barbara Partee, 1967

- This is hard to answer

The Meaning of a Word

Q: What's the meaning of life?

A: LIFE

– A linguistics joke by Barbara Partee, 1967

- This is hard to answer
- But we did mention it before

Lexcial Semantics

WordNet is a large lexical database of English. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept.

Example (bank)

- S: (n) bank (sloping land (especially the slope beside a body of water)) "*they pulled the canoe up on the bank*"; ...
- S: (n) depository financial institution, ... (a financial institution that accepts deposits and channels the money into lending activities) "*that bank holds the mortgage on my home*"
- ...

Lexcial Semantics

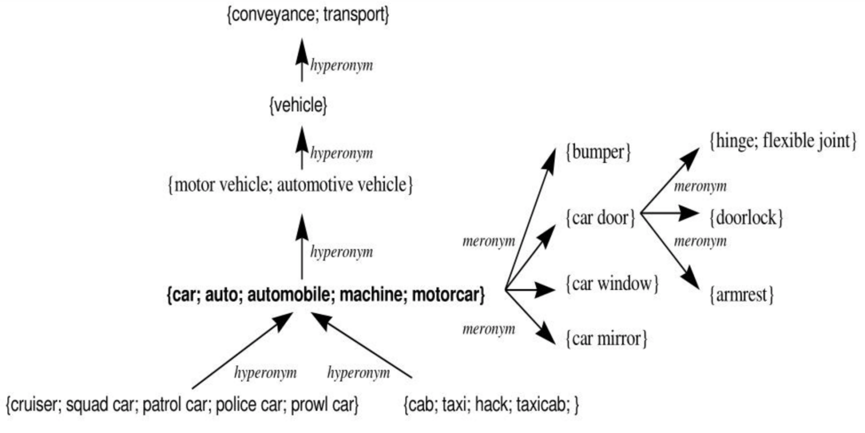
WordNet is a large lexical database of English. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept.

Example (`bank`)

- S: (n) bank (sloping land (especially the slope beside a body of water)) "*they pulled the canoe up on the bank*"; ...
- S: (n) depository financial institution, ... (a financial institution that accepts deposits and channels the money into lending activities) "*that bank holds the mortgage on my home*"
- ...

- Lemma or citation form
- Sense

WordNet


{conveyance; transport}
hyperonym
{vehicle}
hyperonym
{motor vehicle; automotive vehicle}
hyperonym
{car; auto; automobile; machine; motorcar}
hyperonym
hyperonym
{cruiser; squad car; patrol car; police car; prowl car}
{cab; taxi; hack; taxicab; }
meronym
meronym
{bumper}
{car door}
{car window}
{car mirror}
meronym
meronym
{hinge; flexible joint}
{doorlock}
{armrest}

Relations between Senses

- **Synonym**: have the same meaning in certain contexts:
  - automobile / car
  - not always identical in any contexts

Relations between Senses

- **Synonym**: have the same meaning in certain contexts:
  - automobile / car
  - not always identical in any contexts
- **Similarity**: with similar meanings/share some element of meaning, but not identical (not synonyms)
  - car / truck/ bike

Relations between Senses

- **Synonym**: have the same meaning in certain contexts:
  - automobile / car
  - not always identical in any contexts
- **Similarity**: with similar meanings/share some element of meaning, but not identical (not synonyms)
  - car / truck/ bike
- **Word relatedness**: a.k.a. word association, related in any way via a semantic frame or field:
  - coffee / tea are similar
  - coffee / sugar are related

Relations between Senses

- **Synonym**: have the same meaning in certain contexts:
  - automobile / car
  - not always identical in any contexts
- **Similarity**: with similar meanings/share some element of meaning, but not identical (not synonyms)
  - car / truck/ bike
- **Word relatedness**: a.k.a. word association, related in any way via a semantic frame or field:
  - coffee / tea are similar
  - coffee / sugar are related
- **Antonymy**: opposites regarding one aspect of meaning :
  - hot / cold
  - rise / fall

Relations between Senses

- **Synonym**: have the same meaning in certain contexts:
  - automobile / car
  - not always identical in any contexts
- **Similarity**: with similar meanings/share some element of meaning, but not identical (not synonyms)
  - car / truck/ bike
- **Word relatedness**: a.k.a. word association, related in any way via a semantic frame or field:
  - coffee / tea are similar
  - coffee / sugar are related
- **Antonymy**: opposites regarding one aspect of meaning :
  - hot / cold
  - rise / fall
- **Connotation (sentiment)**: affective meanings :
  - copy, replica / fake, knockoff

WordSim353 - Similarity

- identical tokens
- synonym
- antonyms
- first is hyponym of second
- first is hyperonym of second
- sibling terms
- first is part of the second one
- second is part of the first one
- topically related, but none of the above

[Agirre et al., 2009]

word1	word2	similarity
tiger	cat	7.35
tiger	tiger	10.00
plane	car	5.77
train	car	6.31
television	radio	6.77
media	radio	7.42
bread	butter	6.19
cucumber	potato	5.92
doctor	nurse	7.00
professor	doctor	6.62
student	professor	6.81
smart	stupid	5.81
wood	forest	7.73
money	cash	9.15

WordSim353 - Relatedness

word1	word2	relatedness
computer	keyboard	7.62
Jerusalem	Israel	8.46
planet	galaxy	8.11
canyon	landscape	7.53
OPEC	country	5.63
day	summer	3.94
day	dawn	7.53
country	citizen	7.31
planet	people	5.75
environment	ecology	8.81
Maradona	football	8.62
OPEC	oil	8.59
money	bank	8.50
computer	software	8.50

SimLex-999

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

[Hill et al., 2015]

Have Fun

Human

词语相似度和词语相关性有区别吗

Assistant

词语相似度和词语相关性是有区别的。

词语相似度是指两个或多个词语在一定程度上具有相似的含义、结构、发音等特征，可以使用相似度计算方法进行比较。例如，“苹果”和“梨子”都属于水果，它们的相似度较高。

词语相关性是指两个或多个词语在一定程度上具有相互关联，可以使用相关性计算方法进行比较。例如，“苹果”和“水果”之间存在相关性，因为它们都属于水果这一类别。

因此，词语相似度和词语相关性是两个不同的概念，它们可以同时存在，也可以分别存在。

Semantic Field

Words that cover a particular semantic domain or bear structured relations with each other.

Semantic Field

Words that cover a particular semantic domain or bear structured relations with each other.

- hospitals
  - surgeon, scalpel, nurse, medication, hospital
- restaurants
  - waiter, menu, plate, food, menu, chef
- houses
  - door, roof, kitchen, family, bed

Word Meaning

- Words have many senses, thus polysemous
- There are complex relationships among senses, thus words

How to Represent Words?

Naive representation

- The vast majority of rule-based and statistical NLP work regards words as atomic symbols:
  PKU, study, studies, studied, learn, learning, China, china, ...
- Imagine a vector space, this is a vector with one 1 and a lot of 0s:
  $[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, \mathbf{1}, 0, 0, 0, 0]$ in $R^{|vocabulary|}$.

How to Represent Words?

Naive representation

- The vast majority of rule-based and statistical NLP work regards words as atomic symbols:
  PKU, study, studies, studied, learn, learning, China, china, ...
- Imagine a vector space, this is a vector with one 1 and a lot of 0s:
  $[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, \mathbf{1}, 0, 0, 0, 0]$ in $R^{|vocabulary|}$.
- It is huge space:
  $|vocabulary|$ can be very large: 50K (PTB), 13M (Google 1T)

How to Represent Words?

Naive representation

- The vast majority of rule-based and statistical NLP work regards words as atomic symbols:
  PKU, study, studies, studied, learn, learning, China, china, ...
- Imagine a vector space, this is a vector with one 1 and a lot of 0s: $[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, \mathbf{1}, 0, 0, 0, 0]$ in $R^{|vocabulary|}$.
- It is huge space:
  $|vocabulary|$ can be very large: 50K (PTB), 13M (Google 1T)
- Less clear how to deal with **compositionality**
- Still haven't discussed how to do inference

A Motivating Example: POS tagging

Example

- The/det yinkish/adj dripner/noun blorked/verb quastofically/adv into/prep the/det nindin/noun with/prep the/det pidibs/noun.

As a structured prediction problem

- Structured prediction techniques
  - Structured perceptron
  - Conditional random fields
- It is easy to utilize rich features
  - Word form features
  - Morphological features

A Feature-based POS tagger

Features

- Word uni-grams:
- Word bi-grams:
- Prefix strings:
- Suffix strings:

A Feature-based POS tagger

Features for $wi = c_1 ... c_n$ in $... w_{i-2}, w_{i-1}, w_i, w_{i+1}, w_{i+2}$

- Word uni-grams: $w_{i-2}, w_{i-1}, w_i, w_{i+1}, w_{i+2}$
- Word bi-grams:
- Prefix strings:
- Suffix strings:

A Feature-based POS tagger

Features for $wi = c_1...c_n$ in $...w_{i-2}, w_{i-1}, w_i, w_{i+1}, w_{i+2}$

- Word uni-grams: $w_{i-2}, w_{i-1}, w_i, w_{i+1}, w_{i+2}$
- Word bi-grams: $w_{i-2}w_{i-1}, w_{i-1}w_i, w_iw_{i+1}, w_{i+1}w_{i+2}$
- Prefix strings:
- Suffix strings:

A Feature-based POS tagger

Features for $wi = c_1 ... c_n$ in $...w_{i-2}, w_{i-1}, w_i, w_{i+1}, w_{i+2}$

- Word uni-grams: $w_{i-2}, w_{i-1}, w_i, w_{i+1}, w_{i+2}$
- Word bi-grams: $w_{i-2}w_{i-1}, w_{i-1}w_i, w_iw_{i+1}, w_{i+1}w_{i+2}$
- Prefix strings: $c_1, c_1c_2, c_1c_2c_3...$
- Suffix strings:

A Feature-based POS tagger

Features for $wi = c_1 ... c_n$ in $... w_{i-2}, w_{i-1}, w_i, w_{i+1}, w_{i+2}$

- Word uni-grams: $w_{i-2}, w_{i-1}, w_i, w_{i+1}, w_{i+2}$
- Word bi-grams: $w_{i-2}w_{i-1}, w_{i-1}w_i, w_iw_{i+1}, w_{i+1}w_{i+2}$
- Prefix strings: $c_1, c_1c_2, c_1c_2c_3 ...$
- Suffix strings: $c_n, c_{n-1}c_n, c_{n-2}c_{n-1}c_n ...$

A Feature-based POS tagger

Features for $wi = c_1 ... c_n$ in $... w_{i-2}, w_{i-1}, w_i, w_{i+1}, w_{i+2}$

- Word uni-grams: $w_{i-2}, w_{i-1}, w_i, w_{i+1}, w_{i+2}$
- Word bi-grams: $w_{i-2}w_{i-1}, w_{i-1}w_i, w_iw_{i+1}, w_{i+1}w_{i+2}$
- Prefix strings: $c_1, c_1c_2, c_1c_2c_3 ...$
- Suffix strings: $c_n, c_{n-1}c_n, c_{n-2}c_{n-1}c_n ...$

When we can train a simple discriminative sequential tagger:

model	accuracy
LLM	93.61%
LGLM1	94.30%
LGLM2	94.42%

Error Analysis

Freq.	LLM	LGLM1	LGLM2	SR-HMM
0	78.72%	79.77%	80.66%	77.49%
1-5	87.75%	87.95%	88.13%	87.57%
6-10	90.04%	91.04%	91.28%	90.69%
11-100	94.49%	94.94%	94.80%	94.60%
101-1000	95.68%	96.08%	96.12%	96.23%

Tagging accuracies relative to word frequency.

Error Analysis

Freq.	LLM	LGLM1	LGLM2	SR-HMM
0	78.72%	79.77%	80.66%	77.49%
1-5	87.75%	87.95%	88.13%	87.57%
6-10	90.04%	91.04%	91.28%	90.69%
11-100	94.49%	94.94%	94.80%	94.60%
101-1000	95.68%	96.08%	96.12%	96.23%

Classification of low-frequency words is hard.

Error Analysis

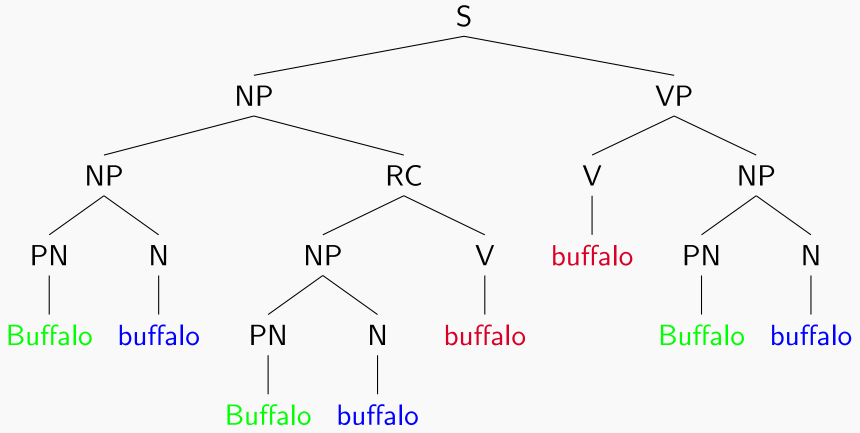
Freq.	LLM	LGLM1	LGLM2	SR-HMM
0	78.72%	79.77%	80.66%	77.49%
1-5	87.75%	87.95%	88.13%	87.57%
6-10	90.04%	91.04%	91.28%	90.69%
11-100	94.49%	94.94%	94.80%	94.60%
101-1000	95.68%	96.08%	96.12%	96.23%

But we still get something for those 0-frequency words.

Problems

- the noun buffalo, an animal
- the city of Buffalo, New York, United States
- the verb to bully, confuse, deceive, or intimidate

Buffalo buffalo Buffalo buffalo buffalo buffalo Buffalo buffalo.



Problems

Synonyms:
adept, expert, good, practiced, proficient, skillful?

Problems

Synonyms:
adept, expert, good, practiced, proficient, skillful?

Missing new words (impossible to keep up to date):

Example (tezgüino)

- A bottle of tezgüino is on the table.
- Everyone likes tezgüino.
- tezgüino makes you drunk.
- We make tezgüino out of corn.

Problems

Synonyms:
adept, expert, good, practiced, proficient, skillful?

Missing new words (impossible to keep up to date):

Example (tezgüino)

- A bottle of tezgüino is on the table.
- Everyone likes tezgüino.
- tezgüino makes you drunk.
- We make tezgüino out of corn.

Example (nindin)

- it was authentic nindin, rather sharp and very strong
- you could taste a famous local product — nindin
- we spent hours in the pub drinking nindin

The Intuition

The contexts suggest

- tezgüino may be a kind of alcoholic beverage made from corn mash.
- nindin is a kind of a drink as well, maybe alcoholic.

The Intuition

The contexts suggest

- tezgüino may be a kind of alcoholic beverage made from corn mash.
- nindin is a kind of a drink as well, maybe alcoholic.

- Use linguistic context to represent the meaning of words and phrases (*partially*).
- Meaning space with dimensions corresponding to elements in the context (*features*).
- Most computational techniques use vectors, or more generally tensors: a.k.a. *semantic space models, vector space models, embeddings*.

Guiding Hypotheses

- **John Firth**, (1957, A synopsis of linguistic theory)
  - You shall know a word by the company it keeps.
  - the complete meaning of a word is always contextual, and no study of meaning apart from context can be taken seriously.
- **Zellig Harris** (1954, Distributional structure)
  - distributional statements can cover all of the material of a language without requiring support from other types of information.

Distributional semantics: family of techniques for representing word meaning based on (linguistic) contexts of use.

Co-occurrence

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.

Clustering Algorithms

Example (Distributional word clustering)

- Words that appear in similar context tend to have similar meanings. A model based on the word bi-gram context:

$$P(w_i|w_1, ..., w_{i-1}) \approx P(C(w_i)|C(w_{i-1}))P(w_i|C(w_i))$$

Clustering Algorithms

Example (Distributional word clustering)

- Words that appear in similar context tend to have similar meanings. A model based on the word bi-gram context:

$$P(w_i|w_1, ..., w_{i-1}) \approx P(C(w_i)|C(w_{i-1}))P(w_i|C(w_i))$$

How to estimate paramethers?

- Latent variable models + Learning with missing data: Expectation Maximazition (**EM**)

Clustering Algorithms

Example (Distributional word clustering)

- Words that appear in similar context tend to have similar meanings. A model based on the word bi-gram context:

$$P(w_i|w_1, ..., w_{i-1}) \approx P(C(w_i)|C(w_{i-1}))P(w_i|C(w_i))$$

How to estimate paramethers?

- Latent variable models + Learning with missing data: Expectation Maximazition (**EM**)

Powerful tools:

- `http://www.statmt.org/moses/giza/GIZA++.html`
- `https://github.com/percyliang/brown-cluster`

How to Represent Words?

So far

- Naive representation: One-hot vectors in $R^{|vocabulary|}$
- Not so naive representation: Word clustering

How to Represent Words?

So far

- Naive representation: One-hot vectors in $R^{|vocabulary|}$
- Not so naive representation: Word clustering

Distributional representation

[0, 0, 0, 0, 0, 0, 0, 0, ..., 0, 0, 1, 0, 0, 0, 0]

$$\downarrow$$

[0.456, 0.193, 5.391, 1.235, -93.0]

How to Represent Words?

So far

- Naive representation: One-hot vectors in $R^{|vocabulary|}$
- Not so naive representation: Word clustering

Distributional representation

$$[0, 0, 0, 0, 0, 0, 0, 0, ..., 0, 0, 1, 0, 0, 0, 0]$$
$$\downarrow$$
$$[0.456, 0.193, 5.391, 1.235, -93.0]$$

- Model a word by embedding it in a vector space.

How to Represent Words?

So far

- Naive representation: One-hot vectors in $R^{|vocabulary|}$
- Not so naive representation: Word clustering

Distributional representation

$$[0, 0, 0, 0, 0, 0, 0, 0, ..., 0, 0, 1, 0, 0, 0, 0]$$
$$\downarrow$$
$$[0.456, 0.193, 5.391, 1.235, -93.0]$$

- Model a word by embedding it in a vector space.
- The representation of a word is a vector of real numbers.

How to Represent Words?

So far

- Naive representation: One-hot vectors in $R^{|vocabulary|}$
- Not so naive representation: Word clustering

Distributional representation

[0, 0, 0, 0, 0, 0, 0, 0, ..., 0, 0, 1, 0, 0, 0, 0]

$\downarrow$

[0.456, 0.193, 5.391, 1.235, -93.0]

- Model a word by embedding it in a vector space.
- The representation of a word is a vector of real numbers.
- We expect much lower dimensions than $R^{|vocabulary|}$

How to Represent Words?

So far

- Naive representation: One-hot vectors in $R^{|vocabulary|}$
- Not so naive representation: Word clustering

Distributional representation

[0, 0, 0, 0, 0, 0, 0, 0, ..., 0, 0, 1, 0, 0, 0, 0]

$\downarrow$

[0.456, 0.193, 5.391, 1.235, -93.0]

- Model a word by embedding it in a vector space.
- The representation of a word is a vector of real numbers.
- We expect much lower dimensions than $R^{|vocabulary|}$
- Often called Embedding Models or Vector Space Models (VSM)

How to Represent Words?

So far

- Naive representation: One-hot vectors in $R^{|vocabulary|}$
- Not so naive representation: Word clustering

Distributional representation

$$[0, 0, 0, 0, 0, 0, 0, 0, ..., 0, 0, 1, 0, 0, 0, 0]$$
$$\downarrow$$
$$[0.456, 0.193, 5.391, 1.235, -93.0]$$

- Model a word by embedding it in a vector space.
- The representation of a word is a vector of real numbers.
- We expect much lower dimensions than $R^{|vocabulary|}$
- Often called Embedding Models or Vector Space Models (VSM)
- Vectors are also called embeddings.

Various Vector Models

Sparse vector representations

- One-hot word representations
- Mutual-information weighted word co-occurrence matrices

Dense vector representations

- Brown clusters, giza++ clusters
- Dimension reduction
- Neural-network-inspired models (skip-grams, CBOW, ...)

Lexical semantics

- fast is similar to rapid
- tall is similar to height

- fast is similar to rapid
- tall is similar to height

When question answering

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

Lexical semantics

- fast is similar to rapid
- tall is similar to height

When question answering

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

Knowledge based question answering

- Identifying or solving relations or properties in knowledge bases
- mapping natural language expressions with canonical expressions in knowledge bases

Guiding hypotheses

- **John Firth**, (1957, A synopsis of linguistic theory)
  - You shall know a word by the company it keeps.
  - the complete meaning of a word is always contextual, and no study of meaning apart from context can be taken seriously.
- **Zellig Harris** (1954, Distributional structure)
  - distributional statements can cover all of the material of a language without requiring support from other types of information.

Idea

- To produce dense vector representations based on the context/use of words.
- Main approaches: count-based, prediction-based, and task-based.

How to represent words?

Idea

- To produce dense vector representations based on the context/use of words.
- Main approaches: count-based, prediction-based, and task-based.

Count-based methods

- Define a basis vocabulary C of context words.
- Define a word window size w.
- Count the basis vocabulary words occurring w words to the left or right of each instance of a target word in the corpus.
- Form a vector representation of the target word based on these counts.

An example

Corpus

```
...  and the cute kitten purred and then ...
...  the cute furry cat purred and miaowed ...
...  that the small kitten miaowed and she ...
...  the loud furry dog ran and bit ...
```

Basis Vocabulary: {..., bit, cute, furry, loud, miaowed, purred, ...}.
kitten's context words: {cute, purred, small, miaowed, ...}.
cat's context words: {furry, purred, ...}.
dog's context words: {furry, ran, ...}.

An example

Example

```
...  and the cute kitten purred and then ...
...  the cute furry cat purred and miaowed ...
...  that the small kitten miaowed and she ...
...  the loud furry dog ran and bit ...
```

Basis Vocabulary: $\{$..., bit, cute, furry, loud, miaowed, purred, ...$\}$.

kitten$=$ [0, 1, 0, 0, 1, 1, 0, 1]

cat$=$ [0,1,1,0,1,0,0,0]

dog$=$ [1,0,1,1,0,0,1,0]

An example

Example

```
...  and the cute kitten purred and then ...
...  the cute furry cat purred and miaowed ...
...  that the small kitten miaowed and she ...
...  the loud furry dog ran and bit ...
```

Basis Vocabulary: {..., bit, cute, furry, loud, miaowed, purred, ...}.
kitten= [0, 1, 0, 0, 1, 1, 0, 1]
cat= [0,1,1,0,1,0,0,0]
dog= [1,0,1,1,0,0,1,0]

- More to consider when counting
  - stop word list? the, that, and, ...
  - lemmatization? miaow, miaowed, dog, dogs, ...

An example

Example

```
...  and the cute kitten purred and then ...
...  the cute furry cat purred and miaowed ...
...  that the small kitten miaowed and she ...
...  the loud furry dog ran and bit ...
```

Basis Vocabulary: {..., bit, cute, furry, loud, miaowed, purred, ...}.

kitten= [0, 1, 0, 0, 1, 1, 0, 1]

cat= [0,1,1,0,1,0,0,0]

dog= [1,0,1,1,0,0,1,0]

- More to consider when counting
  - stop word list? the, that, and, ...
  - lemmatization? miaow, miaowed, dog, dogs, ...
- Size of windows depends on your goals
  - Shorter windows → more syntactic representation
  - Longer windows → more semantic representation

Raw counts?

Raw word frequency is not a great measure of association between words:

- **the** and **of** are very frequent, e.g., appearing in almost every document
- but **maybe/definitely** not the most discriminative *features*

Raw counts?

Raw word frequency is not a great measure of association between words:

- **the** and **of** are very frequent, e.g., appearing in almost every document
- but **maybe/definitely** not the most discriminative *features*

If raw counts are not good, then weight them!

Pointwise mutual information

- information-theoretic measurement: Do events X and Y co-occur more than if they were independent?

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

Raw counts?

Raw word frequency is not a great measure of association between words:

- **the** and **of** are very frequent, e.g., appearing in almost every document
- but **maybe/definitely** not the most discriminative *features*

If raw counts are not good, then weight them!

Pointwise mutual information

- information-theoretic measurement: Do events X and Y co-occur more than if they were independent?

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

Positive PMI

- It is not clear people are good at unrelatedness.
- We just replace negative PMI values by 0

Motivation behind PMI

Suppose the word *apricot* appear 20 times in our training data (which includes 1000 word tokens in total)

- If we see *apricot* appearing $0.02 * L_d$ times in a document d (which is L_d words-long), you will not feel surprised.
- If we see *apricot* appearing $0.2 * L_d$ times in a document d (which is L_d words-long), you will feel surprised!

Motivation behind PMI

Suppose the word *apricot* appear 20 times in our training data (which includes 1000 word tokens in total)

- If we see *apricot* appearing $0.02 * L_d$ times in a document d (which is L_d words-long), you will not feel surprised.
- If we see *apricot* appearing $0.2 * L_d$ times in a document d (which is L_d words-long), you will feel surprised!
- Under the training data, $20/1000 = 0.02$ will be the chance we see *apricot* in a document. (in a unigram assumption)

Motivation behind PMI

Suppose the word *apricot* appear 20 times in our training data (which includes 1000 word tokens in total)

- If we see *apricot* appearing $0.02 * L_d$ times in a document d (which is L_d words-long), you will not feel surprised.
- If we see *apricot* appearing $0.2 * L_d$ times in a document d (which is L_d words-long), you will feel surprised!
- Under the training data, $20/1000 = 0.02$ will be the chance we see *apricot* in a document. (in a unigram assumption)
- One way to measure the surprise: compute the ratio of observed frequency of *apricot* in a document, e.g., 0.2, against the chance under the independence/unigram assumption, e.g., 0.02

Positive Pointwise Mutual Information

- event X: see word a (e.g., *apricot*) co-occurring with context word w, where w appears l_w times in total
- event Y: see word a (e.g., *apricot*) appearing in the training data with N words in total

$$PMI(X,Y) = \left[\log \frac{\text{count}_w(a)/N}{l_w/N \times \sum_c \text{count}_c(a)/N}\right]_+ = \left[\log \frac{\text{count}_w(a) * N}{\sum_c \text{count}_c(a) * l_w}\right]_+$$

where $[score]_+ = \max(score, 0)$.

Positive Pointwise Mutual Information

- event X: see word a (e.g., *apricot*) co-occurring with context word w, where w appears l_w times in total
- event Y: see word a (e.g., *apricot*) appearing in the training data with N words in total

$$PMI(X,Y) = \left[\log \frac{\text{count}_w(a)/N}{l_w/N \times \sum_c \text{count}_c(a)/N}\right]_+ = \left[\log \frac{\text{count}_w(a) * N}{\sum_c \text{count}_c(a) * l_w}\right]_+$$

where $[score]_+ = \max(score, 0)$.

- if *apricot* appears with almost the same frequency for every context word w, then its PMI scores for evey w will be nearly ZERO
- if *apricot* appears only with one context word w^*, then its PMI scores for this w^* will be a large positive value.
- PPMI may tell us: *where a unigram model is the most wrong :-)*

An Example

Matrix: target words $\times$ context words

	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

- raw counts

An Example

Matrix: target words $\times$ context words

	computer	data	pinch	result	sugar
apricot	0.00	0.00	0.05	0.00	0.05
pineapple	0.00	0.00	0.05	0.00	0.05
digital	0.11	0.05	0.00	0.05	0.00
information	0.05	0.32	0.00	0.21	0.00

- computing $p(.)$

An Example

Matrix: target words $\times$ context words

	computer	data	pinch	result	sugar	p(*word*)
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

- computing $p(x)$

An Example

Matrix: target words $\times$ context words

	computer	data	pinch	result	sugar	p(*word*)
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 $p(y)$

An Example

Matrix: target words $\times$ context words

	computer	data	pinch	result	sugar	p(*word*)
apricot			2.25		2.25	0.11
pineapple			2.25		2.25	0.11
digital	1.66	0.00		0.00		0.21
information	0.00	0.57		0.47		0.58
p(*context*)	0.16	0.37	0.11	0.26	0.11	

- computing $PPMI$
- $\log_2 \frac{1*19}{2*2}$

Weighting PMI

Problems

- PMI is biased toward infrequent events
- Very rare words have very high PMI values
- Solution: Laplace (add-one) smoothing

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

Weighting PMI

Problems

- PMI is biased toward infrequent events
- Very rare words have very high PMI values
- Solution: Laplace (add-one) smoothing

	computer	data	pinch	result	sugar
apricot	0.03	0.03	0.05	0.03	0.05
pineapple	0.03	0.03	0.05	0.03	0.05
digital	0.11	0.05	0.03	0.05	0.03
information	0.05	0.14	0.03	0.10	0.03

Weighting PMI

Problems

- PMI is biased toward infrequent events
- Very rare words have very high PMI values
- Solution: Laplace (add-one) smoothing

	computer	data	pinch	result	sugar	P(*word*)
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	0.11	0.05	0.03	0.05	0.03	0.24
information	0.05	0.14	0.03	0.10	0.03	0.36

Weighting PMI

Problems

- PMI is biased toward infrequent events
- Very rare words have very high PMI values
- Solution: Laplace (add-one) smoothing

	computer	data	pinch	result	sugar	P(*word*)
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	0.11	0.05	0.03	0.05	0.03	0.24
information	0.05	0.14	0.03	0.10	0.03	0.36
p(*context*)	0.19	0.25	0.17	0.22	0.17	

Weighting PMI

Problems

- PMI is biased toward infrequent events
- Very rare words have very high PMI values
- Solution: Laplace (add-one) smoothing

	computer	data	pinch	result	sugar	p(*word*)
apricot			0.56		0.56	0.20
pineapple			0.56		0.56	0.20
digital	0.62	0.00		0.00		0.24
information	0.00	0.58		0.37		0.36
p(*context*)	0.19	0.25	0.17	0.22	0.17	

Cosine Similarity

Given two target words/phrases/sentences/paragraphs/..., we need a way to measure their *similarity*.

Cosine Similarity

Given two target words/phrases/sentences/paragraphs/..., we need a way to measure their *similarity*.

- Distributions are vectors in a multidimensional semantic space, that is, objects with a magnitude (length) and a direction.
- The semantic space has dimensions which correspond to possible contexts.

Cosine Similarity

Given two target words/phrases/sentences/paragraphs/..., we need a way to measure their *similarity*.

- Distributions are vectors in a multidimensional semantic space, that is, objects with a magnitude (length) and a direction.
- The semantic space has dimensions which correspond to possible contexts.
- Take angle between vectors as measure of similarity.
  - (correctly) ignores length of vectors = frequency of words
  - similar angle = similar proportion of context words

Cosine Similarity

Given two target words/phrases/sentences/paragraphs/..., we need a way to measure their *similarity*.

- Distributions are vectors in a multidimensional semantic space, that is, objects with a magnitude (length) and a direction.
- The semantic space has dimensions which correspond to possible contexts.
- Take angle between vectors as measure of similarity.
  - (correctly) ignores length of vectors = frequency of words
  - similar angle = similar proportion of context words
- Cosine of angle is easy to compute.
  - cos = 1 means angle is 0, i.e. very similar
  - cos = 0 means angle is 90, i.e. very dissimilar

$$cos(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u}^\top \mathbf{v}}{||\mathbf{u}|| \cdot ||\mathbf{v}||}$$

Cosine Similarity

Given two target words/phrases/sentences/paragraphs/..., we need a way to measure their *similarity*.

- Distributions are vectors in a multidimensional semantic space, that is, objects with a magnitude (length) and a direction.
- The semantic space has dimensions which correspond to possible contexts.
- Take angle between vectors as measure of similarity.
  - (correctly) ignores length of vectors = frequency of words
  - similar angle = similar proportion of context words
- Cosine of angle is easy to compute.
  - cos = 1 means angle is 0, i.e. very similar
  - cos = 0 means angle is 90, i.e. very dissimilar

$$cos(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u}^\top \mathbf{v}}{||\mathbf{u}|| \cdot ||\mathbf{v}||}$$

- Many other methods to compute similarity

Using Syntax to Define a Word's Context

Zellig Harris (1968)

- The meaning of entities, and the meaning of grammatical relations among them, is related to the restriction of combinations of these entities relative to other entities.
- Two words are similar if they have similar syntactic contexts
- duty and responsibility have similar syntactic distribution:
  - **Modified by adjectives**: additional, administrative, assumed, collective, congressional, constitutional, ...
  - **Objects of verbs**: assert, assign, assume, attend to, avoid, become, breach, ...

Context based on Dependency Parsing

`I have a brown dog`

(have subj I), (I subj-of have), (dog obj-of have), (dog adj-mod brown), (brown adj-mod-of dog), (dog det a), (a det-of dog)

The description of cell

count(cell, subj-of, absorb)=1
count(cell, subj-of, adapt)=1
count(cell, subj-of, behave)=1
...
count(cell, pobj-of, in)=159
count(cell, pobj-of, inside)=16
count(cell, pobj-of, into)=30
...

Context based on Dependency Parsing

- hope (N):
  optimism 0.141, chance 0.137, expectation 0.136, prospect 0.126, dream 0.119, desire 0.118, fear 0.116, effort 0.111, confidence 0.109, promise 0.108
- hope (V):
  would like 0.158, wish 0.140, plan 0.139, say 0.137, believe 0.135, think 0.133, agree 0.130, wonder 0.130, try 0.127, decide 0.125
- brief (N): legal brief 0.139, affidavit 0.103, filing 0.098, petition 0.086, document 0.083, argu- ment 0.083, letter 0.079, rebuttal 0.078, memo 0.077
- brief (A): lengthy 0.256, hour-long 0.191, short 0.173, extended 0.163, frequent 0.162, recent 0.158, short-lived 0.155, prolonged 0.149, week-long 0.149

[Dekang Lin, 1998]

Problems

Similarity = Synonymy?

- Antonyms are basically as distributionally similar as synonyms:
- Distributional similarity is not referential similarity.
- Distinguishing synonyms from antonyms is notoriously hard problem.

brief (A):
lengthy 0.256, hour-long 0.191, short 0.173, extended 0.163, frequent 0.162, recent 0.158, short-lived 0.155, prolonged 0.149, week-long 0.149, occasional 0.146

Document?

	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10
against	0	0	0	1	0	0	3	2	3	0
age	0	0	0	1	0	3	1	0	4	0
agent	0	0	0	0	0	0	0	0	0	0
ages	0	0	0	0	0	2	0	0	0	0
ago	0	0	0	2	0	0	0	0	3	0
agree	0	1	0	0	0	0	0	0	0	0
ahead	0	0	0	1	0	0	0	0	0	0
ain't	0	0	0	0	0	0	0	0	0	0
air	0	0	0	0	0	0	0	0	0	0
aka	0	0	0	1	0	0	0	0	0	0

Each cell: count of word w in a document d:

- Each document is a count vector: a column above
- Each word is a count vector: a row above

Document?

	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10
against	0	0	0	1	0	0	3	2	3	0
age	0	0	0	1	0	3	1	0	4	0
agent	0	0	0	0	0	0	0	0	0	0
ages	0	0	0	0	0	2	0	0	0	0
ago	0	0	0	2	0	0	0	0	3	0
agree	0	1	0	0	0	0	0	0	0	0
ahead	0	0	0	1	0	0	0	0	0	0
ain't	0	0	0	0	0	0	0	0	0	0
air	0	0	0	0	0	0	0	0	0	0
aka	0	0	0	1	0	0	0	0	0	0

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

Document?

	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10
against	0	0	0	1	0	0	3	2	3	0
age	0	0	0	1	0	3	1	0	4	0
agent	0	0	0	0	0	0	0	0	0	0
ages	0	0	0	0	0	2	0	0	0	0
ago	0	0	0	2	0	0	0	0	3	0
agree	0	1	0	0	0	0	0	0	0	0
ahead	0	0	0	1	0	0	0	0	0	0
ain't	0	0	0	0	0	0	0	0	0	0
air	0	0	0	0	0	0	0	0	0	0
aka	0	0	0	1	0	0	0	0	0	0

- Two documents are similar if their vectors are similar
- Two words are similar if their vectors are similar
- The matrix seems not so friendly though

Dimension Reduction

- Too sparse?

- Too sparse?

We got a plan!

Approximate a high-dimensional matrix using fewer dimensions

- first rotating the axes into a new space
- in which the highest order dimension captures the most variance in the original dataset
- and the next dimension captures the next most variance, etc.
- we may only keep a small number of dimensions, e.g., 50, 100, or 300 ...
- Many such (related) methods: principle components analysis (PCA), Factor Analysis, LSA, etc.

Dimensionality Reduction

Why dense vectors?

- Short vectors may be easier to use as features in machine learning
- Dense vectors may generalize better than storing explicit counts

Dense 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.

Latent Semantic Analysis

For a **sparse** word-document co-occurrence matrix A, Latent Semantic Analysis (or Latent Semantic Indexing, LSA, LSI) seeks to find an approximation for A:

$$A \approx \hat{A} = M \times \mathsf{diag}(s) \times C^{\mathsf{T}}$$

- $\hat{A}$ is the approximated matrix, M the *embeddings* of words, C contains representations of documents

Latent Semantic Analysis

For a **sparse** word-document co-occurrence matrix A, Latent Semantic Analysis (or Latent Semantic Indexing, LSA, LSI) seeks to find an approximation for A:

$$A \approx \hat{A} = M \times \mathsf{diag}(s) \times C^{\mathsf{T}}$$

- $\hat{A}$ is the approximated matrix, M the *embeddings* of words, C contains representations of documents
- this can be solved by applying singular value decomposition (SVD) to the sparse matrix A

Latent Semantic Analysis

For a **sparse** word-document co-occurrence matrix A, Latent Semantic Analysis (or Latent Semantic Indexing, LSA, LSI) seeks to find an approximation for A:

$$A \approx \hat{A} = M \times \mathsf{diag}(s) \times C^{\mathsf{T}}$$

- $\hat{A}$ is the approximated matrix, M the *embeddings* of words, C contains representations of documents
- this can be solved by applying singular value decomposition (SVD) to the sparse matrix A
- we could even truncate it to a small dimensions, k.
  - M stores left singular vectors of A
  - C stores left singular vectors of A
  - s stores singular values of A, non-negative and usually organized in decreasing order

Latent Semantic Analysis

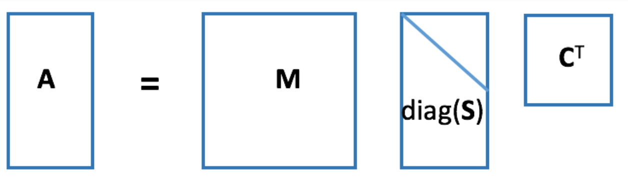
For a **sparse** word-document co-occurrence matrix A, Latent Semantic Analysis (or Latent Semantic Indexing, LSA, LSI) seeks to find an approximation for A:

$$A \approx \hat{A} = M \times \text{diag}(s) \times C^{\mathsf{T}}$$

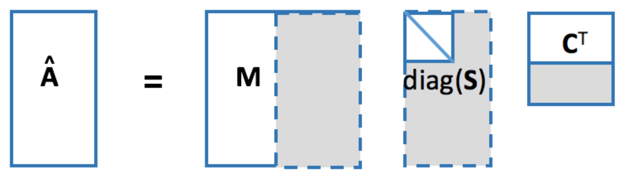
- $\hat{A}$ is the approximated matrix, M the *embeddings* of words, C contains representations of documents
- this can be solved by applying singular value decomposition (SVD) to the sparse matrix A
- we could even truncate it to a small dimensions, k.
  - M stores left singular vectors of A
  - C stores left singular vectors of A
  - s stores singular values of A, non-negative and usually organized in decreasing order
  - if some values in $\text{diag}(s)$ are zeros, then A is *low-rank*
  - we could approximate A by truncating s to keep only k non-negative values

Latent Semantic Analysis

SVD



truncated SVD



Latent Semantic Analysis

- create a mapping of words and documents into the same low-dimensional space
- could deal with sparseness or even noise in A
- words that occur in similar contexts have similar meanings.
- could deal with synonyms somehow
- still a bag-of-words assumption
- but, need to cope with new words, new documents

Probabilistic Latent Semantic Analysis

There is a probabilistic version, Probabilistic Latent Semantic Analysis (PLSA, Hofmann, 1999)
For every document d with l_d words in a large corpus:

$$p(\mathbf{x}_d, \mathbf{z}|d) = \prod_{i=1}^{l_d} p_{topic}(z_i|d) \cdot p_{word}(x_{d,i}|z_i)$$

$$p(\mathbf{x}_d|d) = \sum_{\mathbf{z} \in \{1,2,\ldots K\}^{l_d}} p(\mathbf{x}_d, \mathbf{z}|d)$$

- we could think each document d is generated by a mixture of K topics
- p_{topic} is a distribution over K topics for each document
- p_{word} is a distribution over the vocabulary for each topic

Probabilistic Latent Semantic Analysis

There is a probabilistic version, Probabilistic Latent Semantic Analysis (PLSA, Hofmann, 1999)

For every document d with l_d words in a large corpus:

$$p(\mathbf{x}_d, \mathbf{z}|d) = \prod_{i=1}^{l_d} p_{topic}(z_i|d) \cdot p_{word}(x_{d,i}|z_i)$$

$$p(\mathbf{x}_d|d) = \sum_{\mathbf{z} \in \{1,2,\dots K\}^{l_d}} p(\mathbf{x}_d, \mathbf{z}|d)$$

- we could think each document d is generated by a mixture of K topics
- p_{topic} is a distribution over K topics for each document
- p_{word} is a distribution over the vocabulary for each topic

Do you remember something we have mentioned before?

Probabilistic Latent Semantic Analysis

There is a probabilistic version, Probabilistic Latent Semantic Analysis (PLSA, Hofmann, 1999)
For every document d with l_d words in a large corpus:

$$p(\mathbf{x}_d, \mathbf{z}|d) = \prod_{i=1}^{l_d} p_{topic}(z_i|d) \cdot p_{word}(x_{d,i}|z_i)$$

$$p(\mathbf{x}_d|d) = \sum_{\mathbf{z} \in \{1,2,\ldots K\}^{l_d}} p(\mathbf{x}_d, \mathbf{z}|d)$$

- we could think each document d is generated by a mixture of K topics
- p_{topic} is a distribution over K topics for each document
- p_{word} is a distribution over the vocabulary for each topic
- But, PLSA cannot assign probability to a text not in the training corpus

Probabilistic Latent Semantic Analysis

There is a probabilistic version, Probabilistic Latent Semantic Analysis (PLSA, Hofmann, 1999)
For every document d with l_d words in a large corpus:

$$p(\mathbf{x}_d, \mathbf{z}|d) = \prod_{i=1}^{l_d} p_{topic}(z_i|d) \cdot p_{word}(x_{d,i}|z_i)$$

$$p(\mathbf{x}_d|d) = \sum_{\mathbf{z} \in \{1,2,...K\}^{l_d}} p(\mathbf{x}_d, \mathbf{z}|d)$$

- we could think each document d is generated by a mixture of K topics
- p_{topic} is a distribution over K topics for each document
- p_{word} is a distribution over the vocabulary for each topic
- But, PLSA cannot assign probability to a text not in the training corpus

Do you remember something we have mentioned before?

Four Kinds of Vector Models

Sparse vector representations

- Mutual-information weighted word co-occurrence matrices

Dense vector representations

- Brown clusters, giza++ clusters
- Neural-network-inspired models (skip-grams, CBOW, ...)

Four Kinds of Vector Models

Sparse vector representations

- Mutual-information weighted word co-occurrence matrices
- Dimension reduction $\rightarrow$ dense vectors

Dense vector representations

- Brown clusters, giza++ clusters
- Neural-network-inspired models (skip-grams, CBOW, ...)

How to Represent Words

Basic Idea

- To produce dense vector representations based on the context/use of words.
- Three main approaches: count-based, prediction-based, task-based.

Basic Idea

- To produce dense vector representations based on the context/use of words.
- Three main approaches: count-based, prediction-based, task-based.

Mikolov et al. (2013a, b) design two models for word vectors to be computationally efficient

How to Represent Words

Basic Idea

- To produce dense vector representations based on the context/use of words.
- Three main approaches: count-based, prediction-based, task-based.

Mikolov et al. (2013a, b) design two models for word vectors to be computationally efficient

`word2vec`

- Predict between every word and its context words!

How to Represent Words

Basic Idea

- To produce dense vector representations based on the context/use of words.
- Three main approaches: count-based, prediction-based, task-based.

Mikolov et al. (2013a, b) design two models for word vectors to be computationally efficient

`word2vec`

- Predict between every word and its context words!

- Continuous bag of words (CBOW) $p(\text{word}|\text{context})$
  - This is very similar to what we have talked about in the feedforward neural language model (Begnio et al., 2003)
- Skip-gram: $p(\text{context}|\text{word})$

Neural Language Model

Basic form of continuous bag of words:

$$p(w_n|w_{n-2}, w_{n-1}, w_{n+1}, w_{n+2}) =$$

$$\text{softmax}(b + \sum_{j=n-2, j\neq n}^{n+2} m_{w_j} A_j + W \tanh(u + \sum_{j=n-2, j\neq n}^{n+2} m_{w_j} T_j))$$

- parameters: b, A, W, T, u, M
- vocabulary size: $V = |\mathcal{V}|$, hidden size: H, word embeddings dim. d
- $b(V)$, $A(d, V)$, $W(V, H)$, $T(d, H)$, $u(H)$, $M(V, d)$

Skip-gram Model (Mikolov et al., 2013)

The key is to predict the context words given the target word, $p(\text{context}|\text{word})$

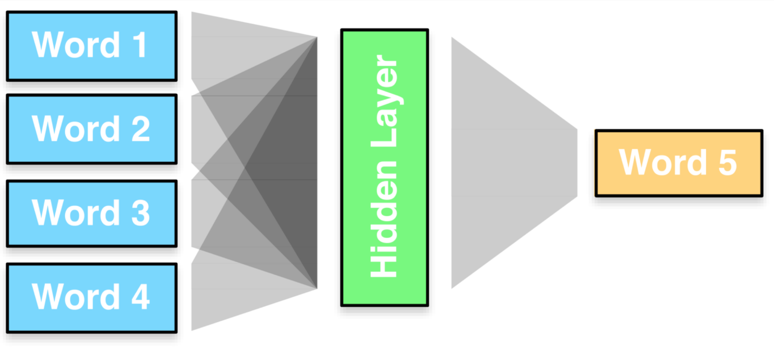
$$p(\text{context} = c|\text{word} = w) = \frac{1}{Z_w} \exp \mathbf{c}_c^{\mathsf{T}} \mathbf{w}_w$$

- two different vectors for each word: one w when it is our target word and one c when it is in the context
- using SGD to estimate the vectors
- computing the normalization term Z_w is expensive, so there are many tricks to perform approximations for efficiency
- need to consider how we define the *context*, k-words window, sentences, paragraphs, or even document

Recap: Language Model as Classification

Treat LM in a classification paradigm:

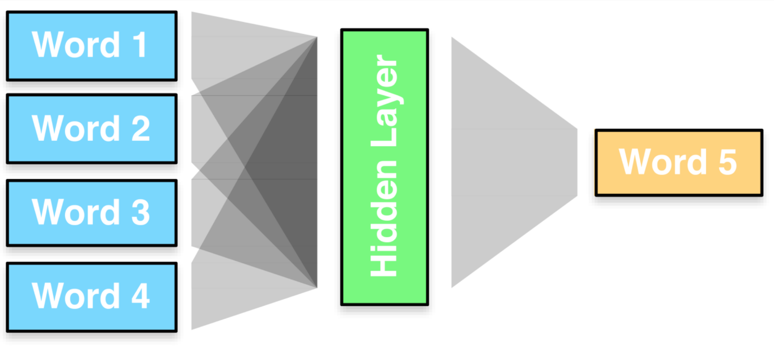
Today, I will talk about language ___



Recap: Language Model as Classification

Treat LM in a classification paradigm:

Today, I will talk about language ___

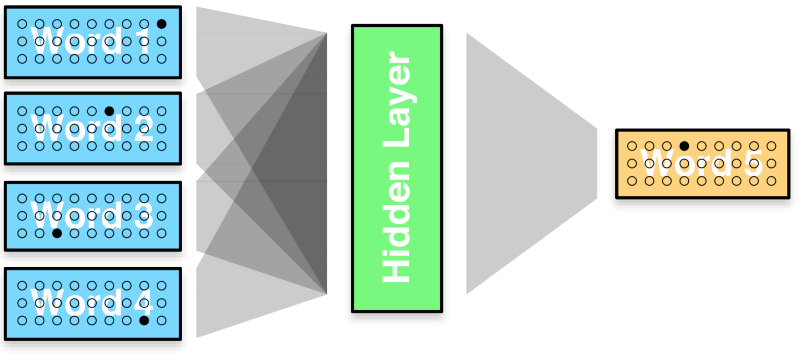


- Using word as itself can be problematic
- Synonyms, Hypernyms, ...

Recap: Language Model as Classification

Treat LM in a classification paradigm:

Today, I will talk about language ___

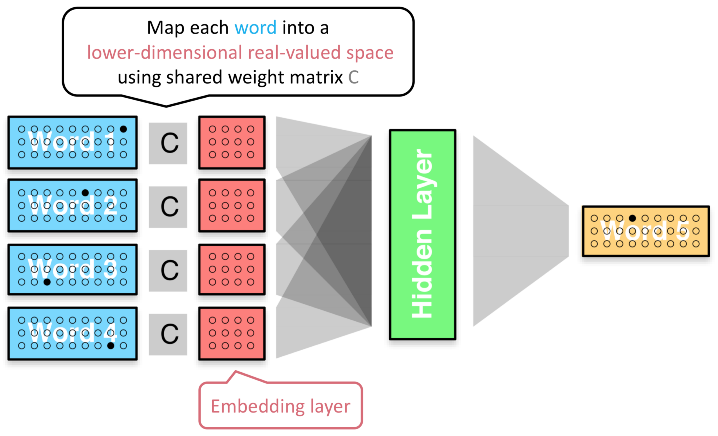


- Vector representations, e.g., one-hot, ...

Recap: Language Model as Classification

Treat LM in a classification paradigm:

Today, I will talk about language ___

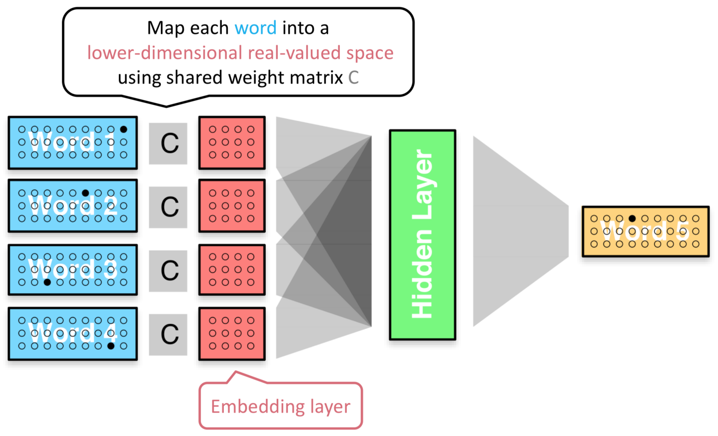


- Continuous vector presentations

Recap: Language Model as Classification

Treat LM in a classification paradigm:

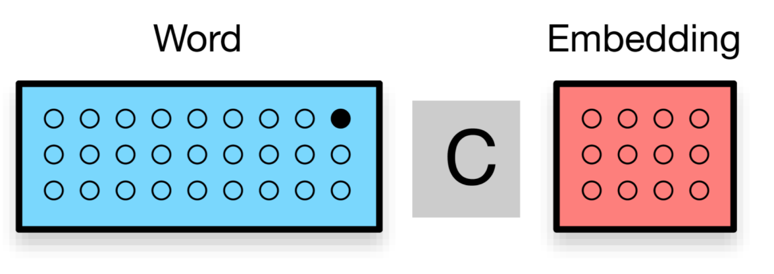
Today, I will talk about language ___



- Continuous vector presentations $\rightarrow$ Lookup table

Treat LM in a classification paradigm:

Today, I will talk about language ___



- Make *lookup table* trainable
- NNs naturally produce word embeddings
- Similar words tend to have similar embeddings: *car*, *van*, ...

Evaluating Word Vectors

Intrinsic Evalutions

- first calculate the similarities between pairs of word vectors, and then calculate the correlation with the judgments of similarity given by humans
- have a TOEFL-like synonym test! e.g., select the one that can best replace *rug* $\rightarrow$ {sofa, ottoman, carpet, hallway}?
- syntactic analogies, e.g., *Peking is to China as Paris is to what?*
- human study, e.g., manual inspection

Evaluating Word Vectors

Extrinsic Evalutions

- Use large unannotated corpus to get your word vectors
- Use them in a text classifier or some other NLP systems
  - Plug in word vectors as *frozen* features, and estimate the other parameters of your model
  - Treat them as parameters of the text classifier; pretraining gives initial values, but they get updated, or *finetuned* during supervised learning.
- Does that system's performance improve?

Count-based vs Prediction-based

Count-based approaches

- Sparse vector representations
- Fast training
- Efficient usage of statistics
- Primarily used to capture word similarity

Prediction-based approaches

- Dense vector representations
- Scales with corpus size
- Inefficient usage of statistics
- Generate improved performance on other tasks
- Can capture complex patterns beyond word similarity

Sparse vectors vs dense vectors

PPMI vectors

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

Prediction learned vectors

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

Why dense?

- Short vectors may be easier to use as features in machine learning
- Dense vectors may generalize better than storing explicit counts

Words may have multiple distinct meanings.

Example (play)

- The new-look play area is due to be completed.
- These districts favor representatives who play to the party base .
- MJ just completed the three-point play for a 66-63 lead .

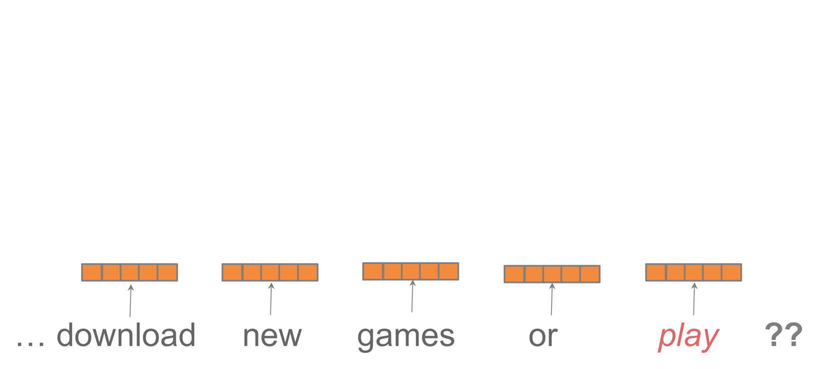
Words may have multiple distinct meanings.

Example (play)

- The new-look play area is due to be completed.
- These districts favor representatives who play to the party base .
- MJ just completed the three-point play for a 66-63 lead .

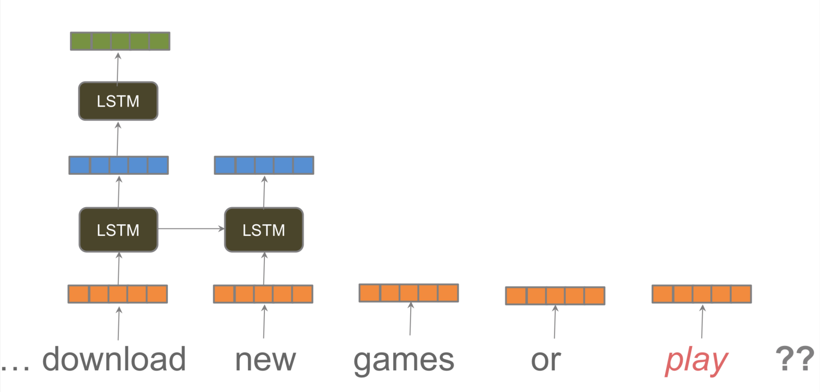
One vector per word is not enough

Contextual Representations



[from Mohit Iyyer]

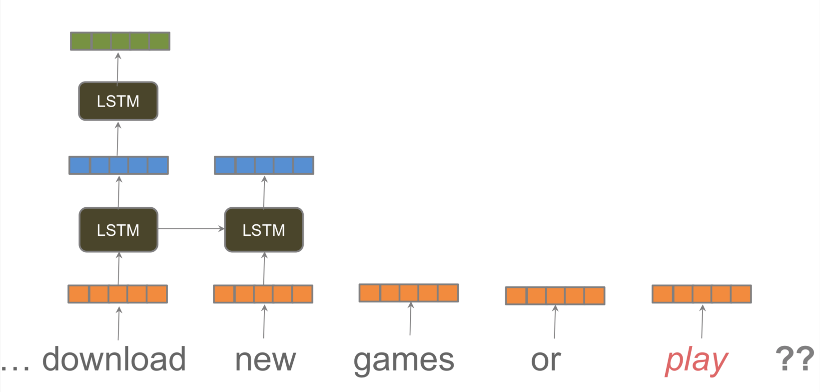
Go Deeper!



[from Mohit Iyyer]

Contextual Representations

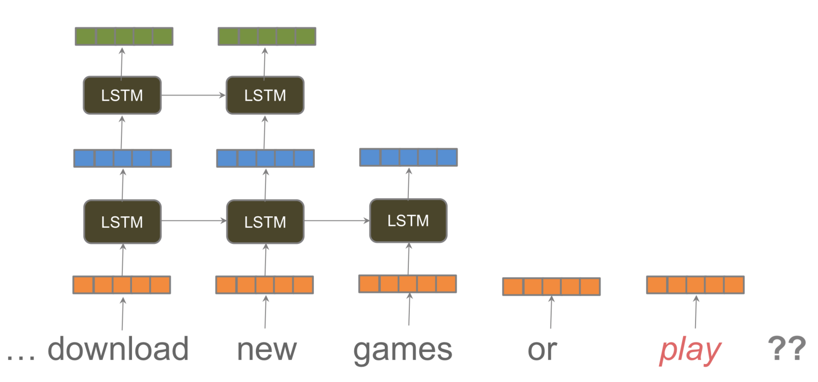
Go Deeper!



[from Mohit Iyyer]

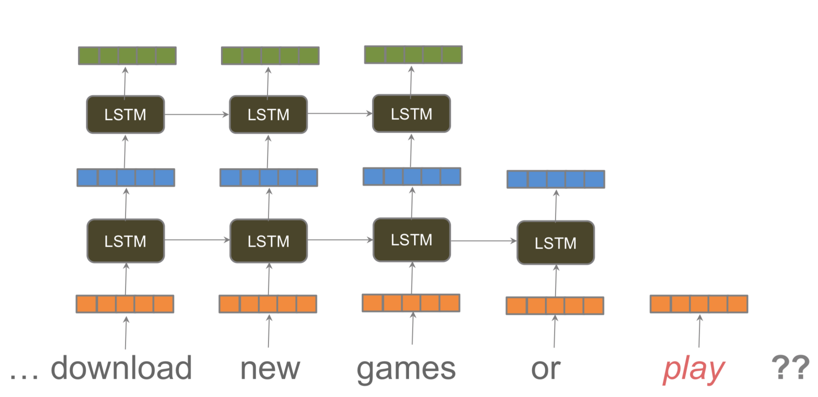
Contextual Representations

Go Deeper!



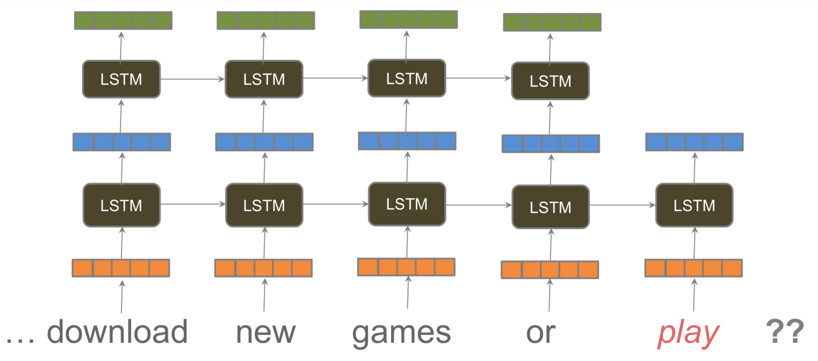
[from Mohit Iyyer]

Go Deeper!



[from Mohit Iyyer]

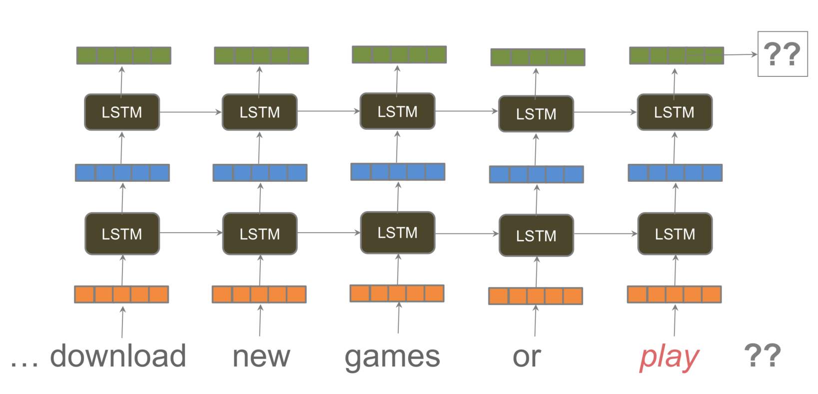
Go Deeper!



[from Mohit Iyyer]

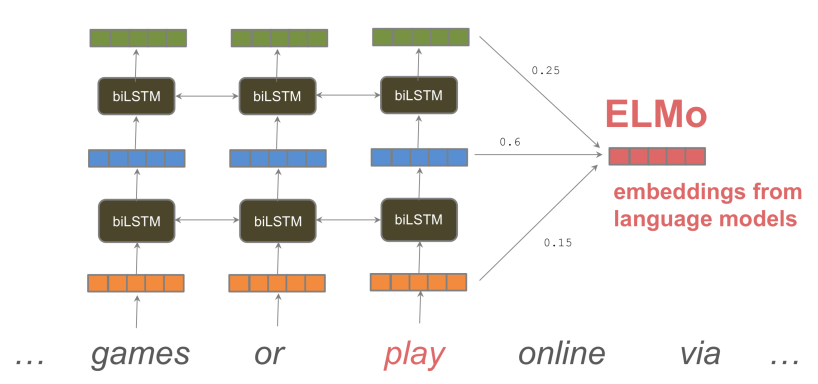
Contextual Representations

Go Deeper!



[from Mohit Iyyer]

Go Deeper!



[from Mohit Iyyer]

Take-away

- One-hot vectors
- PPMI vectors
- Dimention reduction
- Word2vector, Glove
- Elmo, BERT, ...
- ?

Take-away

- One-hot vectors
- PPMI vectors
- Dimention reduction
- Word2vector, Glove
- Elmo, BERT, ...
- ?

- More data, deeper networks
- Pretrain on alternative tasks to language modeling and masked language modeling
- Purposely select what data and/or tasks should learn over time
- Probing the learned representations to determine the extent to which various linguistic, commonsense, world, or domain knowledge is captured

Readings

- Dekang Lin. 1998. Automatic Retrieval and Clustering of Similar Words.
- Tomas Mikolov, Kai Chen, Greg Corrado and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space.
- Dan Jurasky's LSA tutorial: `http://web.stanford.edu/~jurafsky/li15/index.html`
- Chapter 6. Speech and Language Processing: `https://web.stanford.edu/~jurafsky/slp3/6.pdf`
- Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Pasca, Aitor Soroa, A Study on Similarity and Relatedness Using Distributional and WordNet-based Approaches, In Proceedings of NAACL-HLT 2009.
- Thomas Hofmann. Probabilistic latent semantic indexing. In Proc. of SIGIR, 1999.
- Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In NeurIPS, 2013b

Readings

CHAPTER

6 Vector Semantics and Embeddings

荃者所以在鱼，得鱼而忘荃	Nets are for fish; Once you get the fish, you can forget the net.
言者所以在意，得意而忘言	Words are for meaning; Once you get the meaning, you can forget the words

庄子(Zhuangzi), Chapter 26