Vector Embeddings

CMSC 473/673 - NATURAL LANGUAGE PROCESSING

Slides modified from Dr. Frank Ferraro

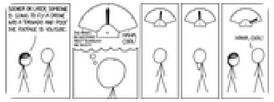
Learning Objectives

Understand the use & creation of dense vector embeddings

Recognize popular vector embeddings

Calculate the distance between vector embeddings

CMSC 473/673 NLP @ UMBC   About   Schedule   Homework▾   Knowledge Checks▾



Evaluating Tech Things

The assignment is due on Tuesday, April 1, 2025 before 11:59PM.
Submission Link: https://blackboard.umbc.edu/ultra/courses/_85408_1/outline/assessment/test/_7438117_1?courseId=_85408_1&gradeitemView=details

Please be sure to double check the academic integrity and generative AI policies listed on the syllabus.

Materials for this assignment:

- GLUE data
- Torch Tutorial from Knowledge Check
- RNN Tutorial
- Character RNN Tutorial
- HW 2 Data Prep Code

Homework 2: Evaluation

https://laramartin.net/NLP-class/homeworks/hw2.html

Learning Objectives

- Calculate common evaluation metrics (recall, precision, F1, accuracy) for multiclass classification.
- Explain when certain evaluation metrics should be modified to fit the problem.
- Create and analyze a simple baseline model.
- Follow code from online for making a neural network.
- Compare the performance of a baseline and a neural model.

Question 1 (24 points): Evaluating by Hand

Each of the following scenarios describes the result of some (made up) classifier: there is a list of the correct labels, and the corresponding predictions. For each of the following situations, compute accuracy, recall, precision, and F1 score. You may verify your answers with code, but **you must show work, or some intermediate steps, to receive full credit** on this problem.

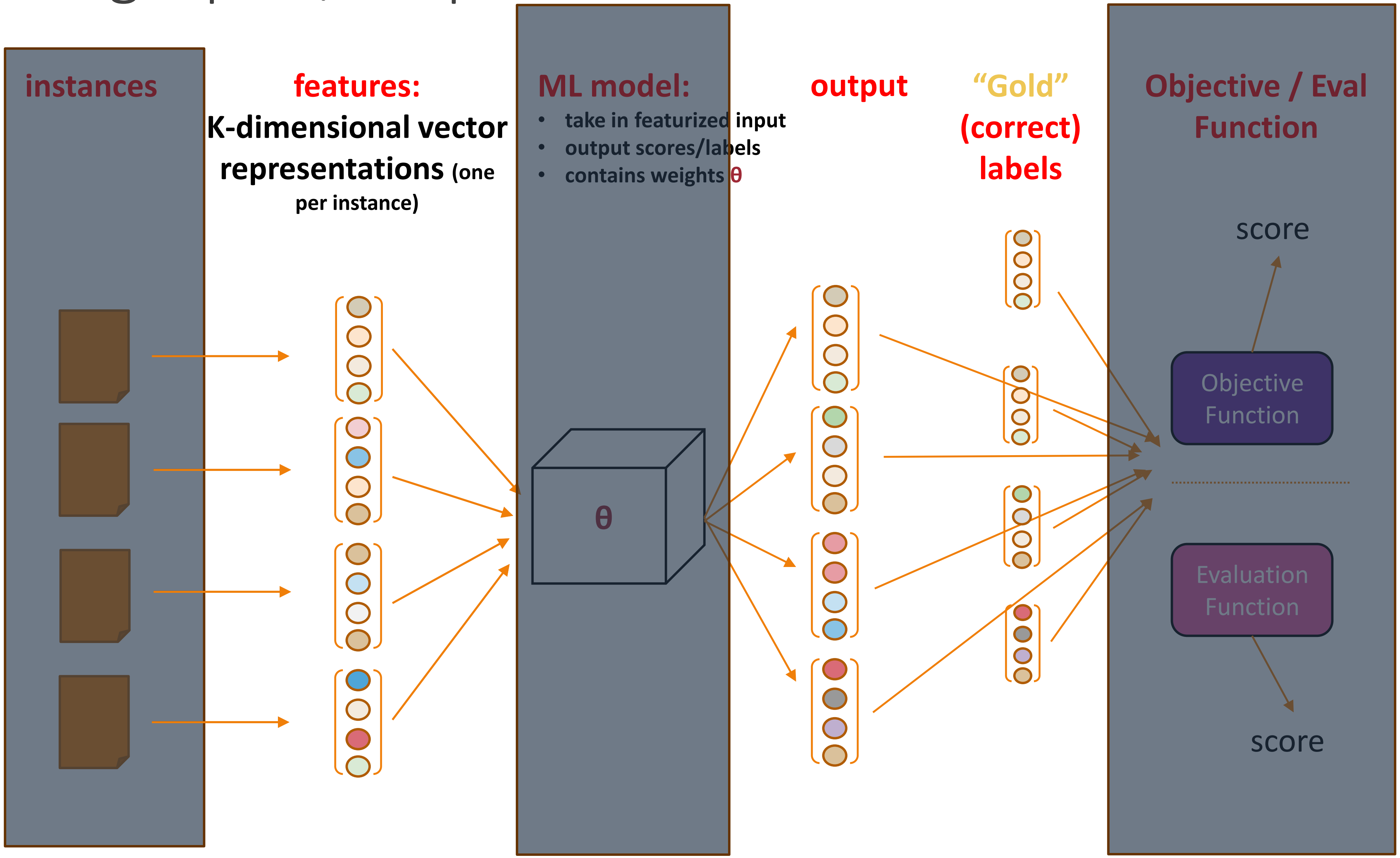
Slides from 2/18 and 2/20 will be the most useful for this question.

a) (4 points total) A binary classification result, where the correct labels are `[T, T, F, T, F, T, F, T]` and the predicted labels are `[T, F, T, T, F, F, F, T]`. Assume `T` means "true" (the desired class) and `F` ("false") is the "default" class.

- (1 pt) Compute accuracy.
- (3 pts) Compute recall, precision, and F1 for `T`.

b) (4 points total) A binary classification result, where the correct labels are `[T, F, F, F, F, F, F, T]` and the predicted labels are `[F, T, F, F, F, F, F, F]`. Assume `T` means "true" (the desired class) and `F` ("false") is the "default" class.

Representing Inputs/Outputs

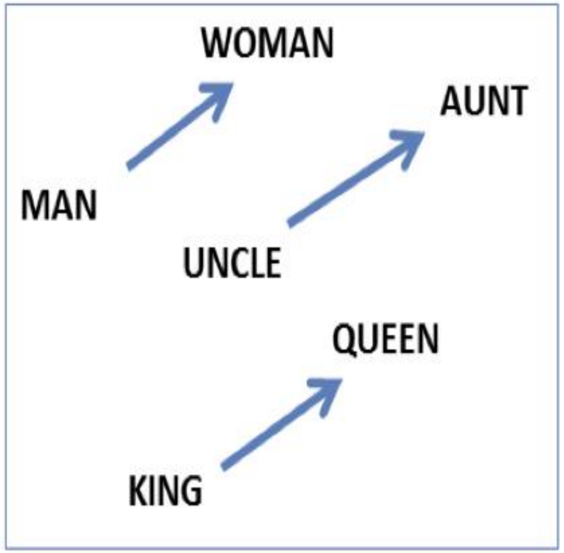


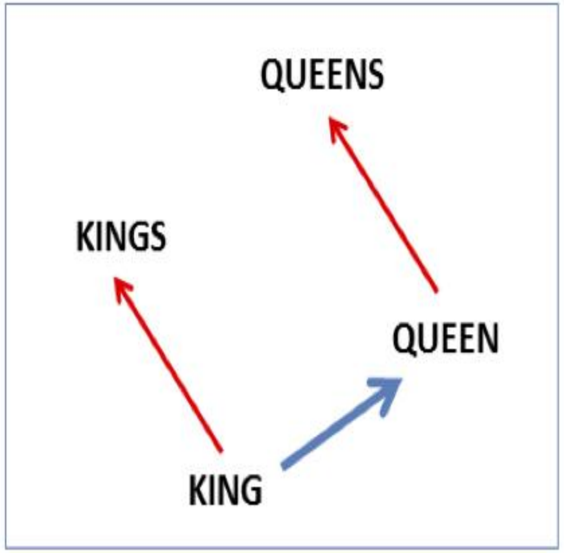
(Some) Properties of Embeddings

Capture “like” (similar) words

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

Capture relationships





vector(*‘king’*) – vector(*‘man’*) + vector(*‘woman’*) ≈ vector(‘queen’)

vector(*‘Paris’*) - vector(*‘France’*) + vector(*‘Italy’*) ≈ vector(‘Rome’)

Mikolov et al. (2013)

Shared Intuition Across Embedding Types

Model the meaning of a word by “embedding” in a vector space

The meaning of a word is a vector of numbers

Contrast: word meaning is represented in many computational linguistic applications by a vocabulary index (“word number 545”) or the string itself

Three Common Kinds of Embedding Models

1. Co-occurrence matrices
2. Matrix Factorization: Singular value decomposition/Latent Semantic Analysis, Topic Models
3. Neural-network-inspired models (skip-grams, CBOW)

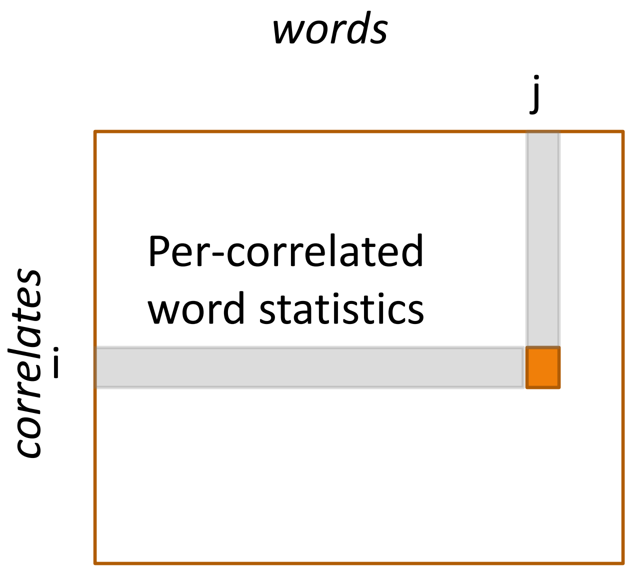
Three Common Kinds of Embedding Models

1. Co-occurrence matrices
2. Matrix Factorization: Singular value decomposition/Latent Semantic Analysis, Topic Models
3. Neural-network-inspired models (skip-grams, CBOW)

Co-occurrence matrices can be used in their own right, but they're most often used as inputs (directly or indirectly) to the matrix factorization or neural approaches

Co-occurrence Matrix

Acquire basic contextual statistics (often counts) for each word type v via *correlate*.



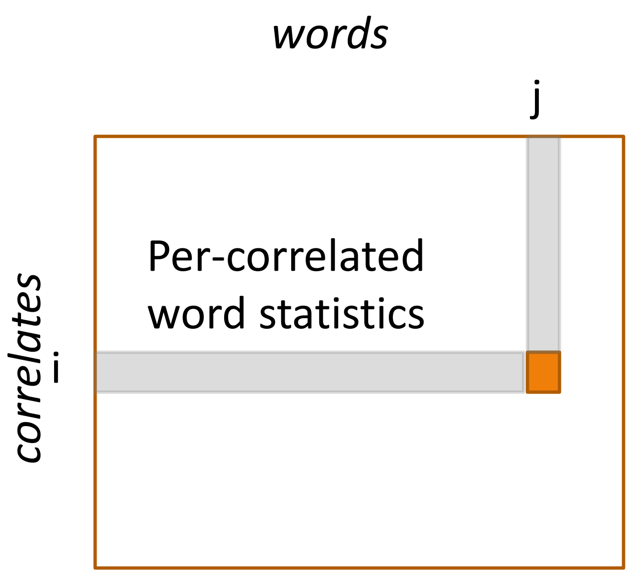
Co-occurrence Matrix

Acquire basic contextual statistics (often counts) for each word type v via *correlate*:

For example:

documents

- Record how often a word occurs in each document



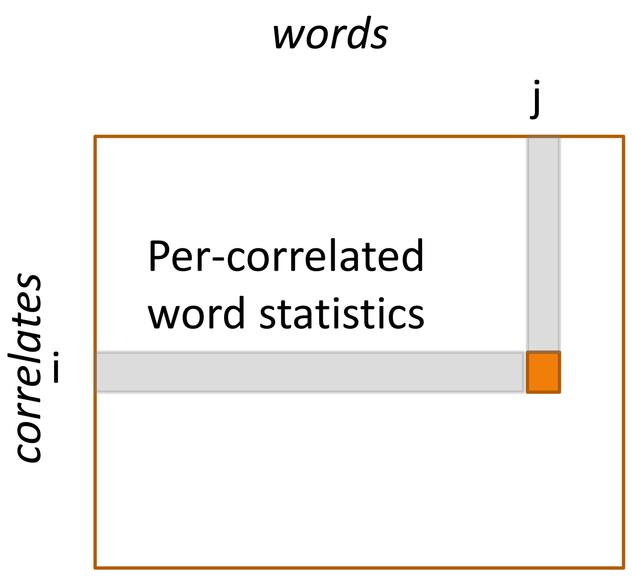
correlates =
documents

Co-occurrence Matrix

Acquire basic contextual statistics (often counts) for each word type v via *correlate*:

For example:

- documents
- surrounding context words
  - Record how often v occurs with other word types u



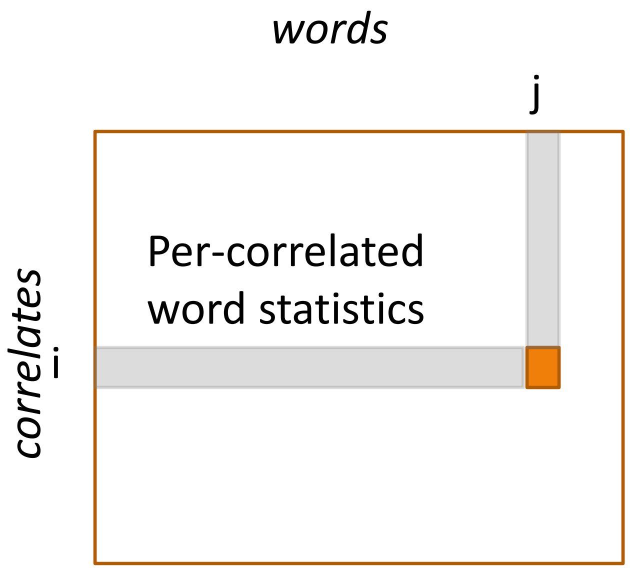
correlates =
word types

Co-occurrence Matrix

Acquire basic contextual statistics (often counts) for each word type v via *correlate*:

For example:

- documents
- surrounding context words
- linguistic annotations (POS tags, syntax)
- …



Assumption: Two words are similar if their vectors are similar

“Acquire basic contextual statistics (often counts) for each word type v”

Two basic, initial counting approaches

- Record which words appear in which documents
- Record which words appear together

These are good first attempts, but with some large downsides

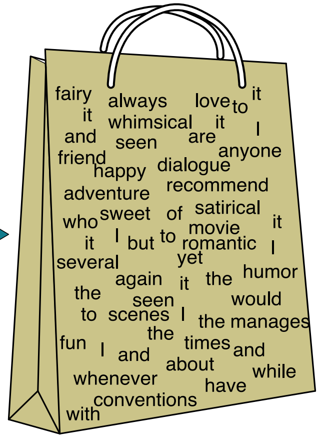
“You shall know a word by the company it keeps!” Firth (1957)

document (↓)-word (→) count matrix

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

basic bag-of-words counting

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!



it	6
I	5
the	4
to	3
and	3
seen	2
yet	1
would	1
whimsical	1
times	1
sweet	1
satirical	1
adventure	1
genre	1
fairy	1
humor	1
have	1
great	1
…	…

“You shall know a word by the company it keeps!” Firth (1957)

document (↓)-word (→) count matrix

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

Assumption: Two documents are similar if their vectors are similar

"You shall know a word by the company it keeps!" Firth (1957)

document (↓)-word (→) count matrix

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

Assumption: Two words are similar if their vectors are similar???

“You shall know a word by the company it keeps!” Firth (1957)

document (↓)-word (→) count matrix

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

Assumption: Two words are similar if their vectors are similar

Issue: Count word vectors are very large, sparse, and skewed!

“You shall know a word by the company it keeps!” Firth (1957)

context (↓)-word (→) count matrix

	apricot	**pineapple**	**digital**	**information**
aardvark	0	0	0	0
computer	0	0	2	1
data	0	10	1	6
pinch	1	1	0	0
result	0	0	1	4
sugar	1	1	0	0

Context: those other words within a small “window” of a target word

a cloud [computer stores digital data on] a remote computer

“You shall know a word by the company it keeps!” Firth (1957)

context (↓)-word (→) count matrix

	apricot	pineapple	digital	information
aardvark	0	0	0	0
computer	0	0	2	1
data	0	10	1	6
pinch	1	1	0	0
result	0	0	1	4
sugar	1	1	0	0

- The size of windows depends on your goals
  - The shorter the windows , the more **syntactic** the representation
    - $\pm$ 1-3 more “syntax-y”
  - The longer the windows, the more **semantic** the representation
    - $\pm$ 4-10 more “semantic-y”

Pointwise Mutual Information (PMI): Dealing with Problems of Raw Counts

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

- It's very skewed: "the" and "of" are very frequent, but maybe not the most discriminative

We'd rather have a measure that asks whether a context word is **particularly informative** about the target word.

- (Positive) Pointwise Mutual Information ((P)PMI)

Pointwise mutual information:

Do events x and y co-occur more than if they were independent?

probability words x and y occur together (in the same context/window)

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

probability that word x occurs

probability that word y occurs

"Noun Classification from Predicate-Argument Structure," Hindle (1990)

"drink it" is more common than "drink wine"

"wine" is a better "drinkable" thing than "it"

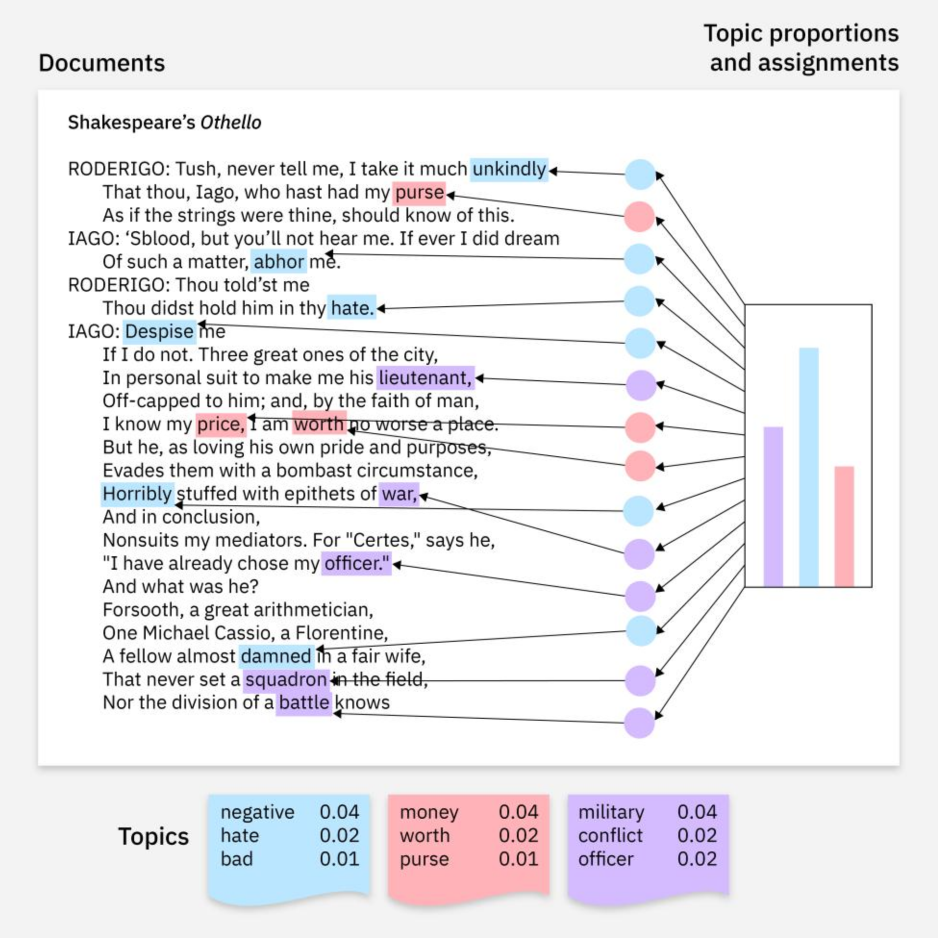
Object of "drink"	Count	PMI
it	3	1.3
anything	3	5.2
wine	2	9.3
tea	2	11.8
liquid	2	10.5

Three Common Kinds of Embedding Models

Learn more in:
- Your project
- Grad assignment (paper)
- Other classes (478/678)

1. Co-occu
2. Matrix Factorization: Singular value decomposition/Latent Semantic Analysis, Topic Models
3. Neural-network-inspired models (skip-grams, CBOW)

Topic Modeling Teaser – Latent Dirichlet Allocation (LDA)



In-depth LDA tutorial:

https://people.eecs.berkeley.edu/~cjrd/static/pdfs/lda_tutorial.pdf

https://www.ibm.com/think/topics/latent-dirichlet-allocation

Three Common Kinds of Embedding Models

1. Co-occurrence matrices
2. Matrix Factorization: Singular value decomposition/Latent Semantic Analysis, Topic Models
3. Neural-network-inspired models (skip-grams, CBOW)

Word2Vec

Mikolov et al. (2013; NeurIPS): "Distributed Representations of Words and Phrases and their Compositionality"

Revisits the context-word approach

Learn a model p(c | w) to predict a context word c from a target word w

Learn two types of vector representations

- $h_c \in \mathbb{R}^E$: vector embeddings for each context word
- $v_w \in \mathbb{R}^E$: vector embeddings for each target word

$$p(c \mid w) \propto \exp(h_c^T v_w)$$

Word2Vec

context (↓)-word (→) count matrix

	apricot	pineapple	digital	information
aardvark	0	0	0	0
computer	0	0	2	1
data	0	10	1	6
pinch	1	1	0	0
result	0	0	1	4
sugar	1	1	0	0

Context: those other words within a small "window" of a target word

$$\max_{h,v} \sum_{c,w \text{ pairs}} \text{count}(c, w) \log p(c \mid w)$$

Word2Vec

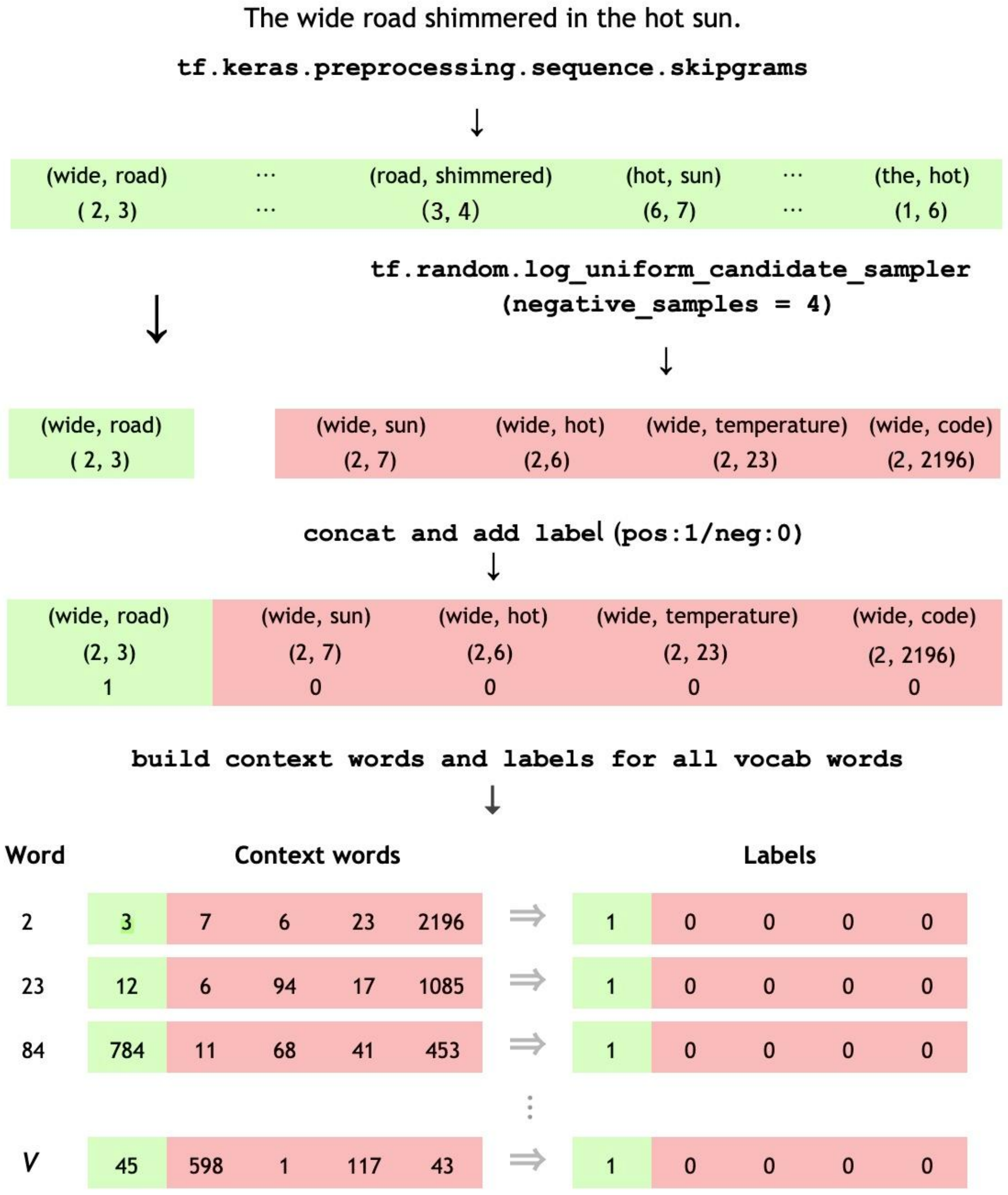
context (↓)-word (→) count matrix

	apricot	pineapple	digital	information
aardvark	0	0	0	0
computer	0	0	2	1
data	0	10	1	6
pinch	1	1	0	0
result	0	0	1	4
sugar	1	1	0	0

Context: those other words within a small "window" of a target word

$$\max_{h,v} \sum_{c,w \text{ pairs}} \text{count}(c,w)\left[h_c^T v_w - \log\left(\sum_u \exp(h_u^T v_w)\right)\right]$$

Example (Tensorflow)



https://www.tensorflow.org/text/tutorials/word2vec

Word2Vec has Inspired a Lot of Work

Off-the-shelf embeddings

- https://code.google.com/archive/p/word2vec/

Off-the-shelf implementations

- https://radimrehurek.com/gensim/models/word2vec.html

Follow-on work

- J. Pennington, R. Socher, and C. D. Manning, "**GLoVe: Global Vectors for Word Representation**," in *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar, 2014, pp. 1532–1543. doi: 10.3115/v1/D14-1162.
  - https://nlp.stanford.edu/projects/glove/
- Many others
- 15000+ citations

FastText

P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "**Enriching Word Vectors with Subword Information**," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017, doi: 10.1162/tacl_a_00051.

Main idea: learn **character n-gram embeddings** for the target word (not context) and modify the word2vec model to use these

Pre-trained models in 150+ languages

- https://fasttext.cc

FastText Details

Main idea: learn **character n-gram embeddings** and for the target word (not the context) modify the word2vec model to use these

Original word2vec:

$$p(c \mid w) \propto \exp(h_c^T v_w)$$

FastText:

$$p(c \mid w) \propto \exp\left(h_c^T \left(\sum_{\text{n-gram } g \text{ in } w} z_g\right)\right)$$

FastText Details

Main idea: learn **character n-gram embeddings** and for the target word (not the context) modify the word2vec model to use these

$$p(c \,|w) \propto \exp\left(h_c^T\left(\sum_{\text{n}-\text{gram } g \text{ in } w} z_g\right)\right)$$

decompose

`fluffy` → `fl flu luf uff ffy fy`

Sub-word units like this have become an important part of today's NLP work!

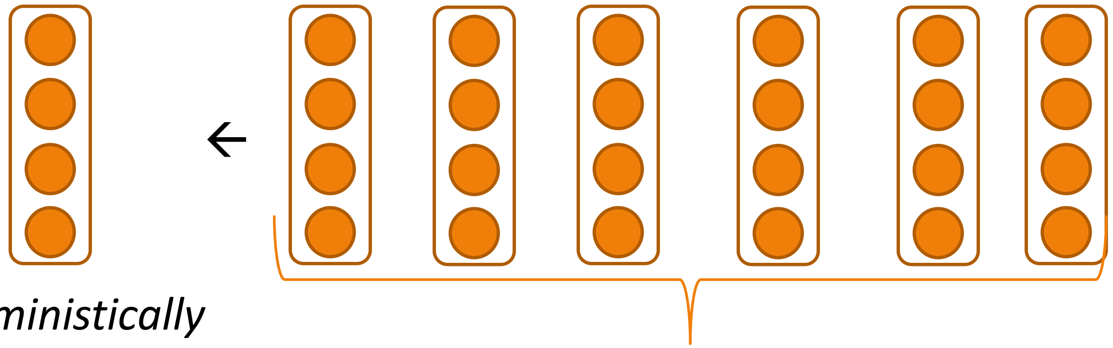
FastText Details

Main idea: learn **character n-gram embeddings** and for the target word (not the context) modify the word2vec model to use these

$$p(c\,|w) \propto \exp\left(h_c^T\left(\sum_{\text{n-gram } g \text{ in } w} z_g\right)\right)$$

decompose

`fluffy` → `fl flu luf uff ffy fy`



To deterministically compute word embeddings

Learn n-gram embeddings


123
SESAME STREET

Contextual Word Embeddings

Word2vec-based models are not context-dependent

- Single word type → single word embedding

If a single word type can have different meanings…

- bank, bass, plant,…

… why should we only have one embedding?

Entire task devoted to classifying these meanings:
Word Sense Disambiguation

Contextual Word Embeddings

Growing interest in this

Off-the-shelf is a bit more difficult

- Download and run a model
- Can't just download a file of embeddings

Two to know about (with code):

- ELMo: "Deep contextualized word representations" Peters et al. (2018; NAACL)
- https://allennlp.org/elmo
- BERT: "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding" Devlin et al. (2019; NAACL)
  - https://github.com/google-research/bert

Evaluating Vector Embeddings

Evaluating Similarity

Extrinsic (task-based, end-to-end) Evaluation:

- Question Answering
- Spell Checking
- Essay grading

“You shall know a word by the company it keeps!” Firth (1957)

context (↓)-word (→) count matrix

	apricot	pineapple	digital	information
aardvark	0	0	0	0
computer	0	0	2	1
data	0	10	1	6
pinch	1	1	0	0
result	0	0	1	4
sugar	1	1	0	0

Context: those other words within a small “window” of a target word

Assumption: Two words are similar if their vectors are similar

Issue: Count word vectors are very large, sparse, and skewed!

Common Evaluation: Correlation between similarity ratings

Input: list of N word pairs $\{(x_1, y_1), \ldots, (x_N, y_N)\}$

- Each word pair (x_i, y_i) has a human-provided similarity score h_i

Use your embeddings to compute an embedding similarity score $s_i = \mathrm{sim}(x_i, y_i)$

Compute the correlation between human and computed similarities

$$\rho = \mathrm{Corr}((h_1, \ldots, h_N), (s_1, \ldots, s_N))$$

Wordsim353: 353 noun pairs rated 0-10

Cosine: Measuring Similarity

Given 2 target words v and w how similar are their vectors?

Dot product or inner product from linear algebra

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

- High when two vectors have large values in same dimensions, low for orthogonal vectors with zeros in complementary distribution

Correct for high magnitude vectors

$$\frac{\vec{a} \cdot \vec{b}}{|\vec{a}||\vec{b}|}$$

Cosine Similarity

Divide the dot product by the length of the two vectors

$$\frac{\vec{a} \cdot \vec{b}}{|\vec{a}||\vec{b}|}$$

This is the cosine of the angle between them

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

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

Cosine Similarity

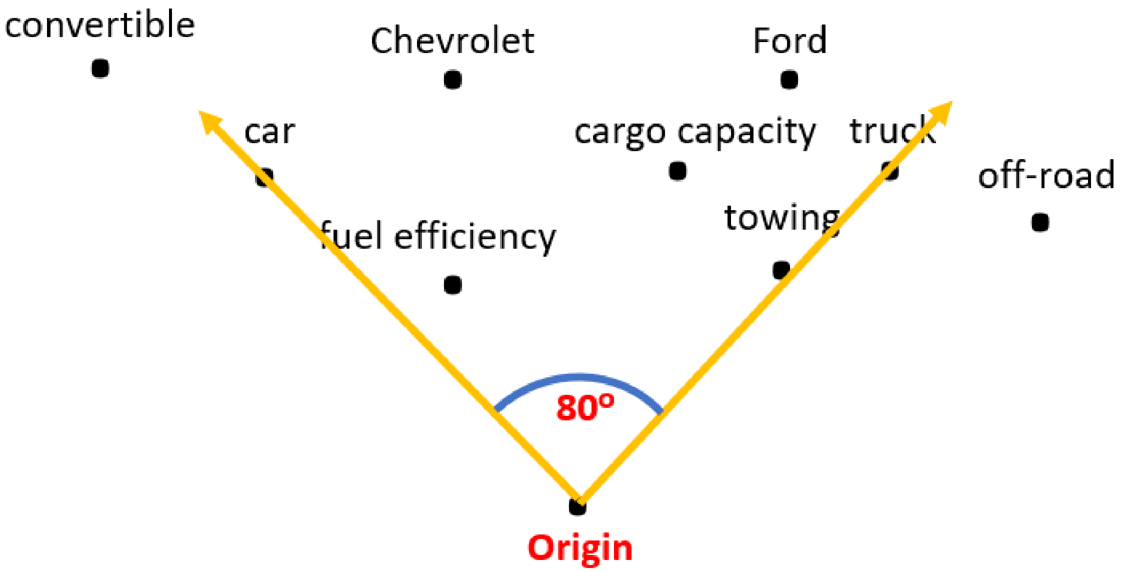
Divide the dot product by the length of the two vectors

$$\frac{\vec{a} \cdot \vec{b}}{|\vec{a}||\vec{b}|}$$

This is the cosine of the angle between them

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

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



https://upload.wikimedia.org/wikipedia/commons/2/23/CosineSimilarity.png

Example: Word Similarity

$$\cos(x,y) = \frac{\sum_i x_i y_i}{\sqrt{\sum_i x_i^2}\sqrt{\sum_i y_i^2}}$$

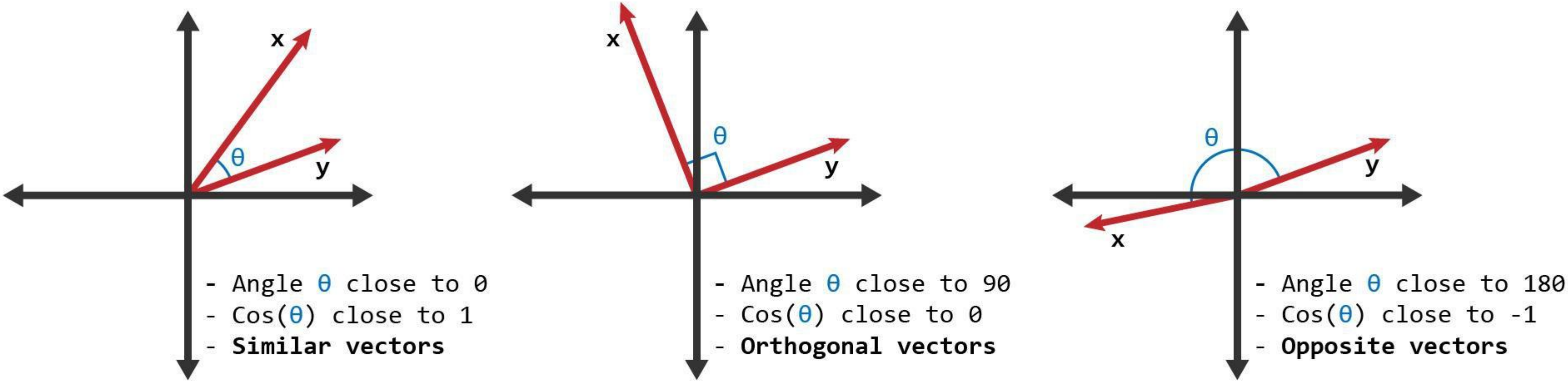
	Dim. 1	Dim. 2	Dim. 3
apricot	2	0	0
digital	0	1	2
information	1	6	1

cosine(apricot,information) = $\frac{2+0+0}{\sqrt{4+0+0}\sqrt{1+36+1}} = 0.1622$

cosine(digital,information) = $\frac{0+6+2}{\sqrt{0+1+4}\sqrt{1+36+1}} = 0.5804$

cosine(apricot,digital) = $\frac{0+0+0}{\sqrt{4+0+0}\sqrt{0+1+4}} = 0.0$

Cosine Similarity Range



https://www.learndatasci.com/glossary/cosine-similarity/

Other Similarity Measures

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

$$\text{sim}_{\text{Jaccard}}(\vec{v},\vec{w}) = \frac{\sum_{i=1}^{N} \min(v_i,w_i)}{\sum_{i=1}^{N} \max(v_i,w_i)}$$

$$\text{sim}_{\text{Dice}}(\vec{v},\vec{w}) = \frac{2\times\sum_{i=1}^{N} \min(v_i,w_i)}{\sum_{i=1}^{N}(v_i+w_i)}$$

$$\text{sim}_{\text{JS}}(\vec{v}||\vec{w}) = D(\vec{v}|\frac{\vec{v}+\vec{w}}{2}) + D(\vec{w}|\frac{\vec{v}+\vec{w}}{2})$$

Adding Morphology, Syntax, and Semantics to Embeddings

- Lin (1998): “Automatic Retrieval and Clustering of Similar Words”
- Padó and Lapata (2007): “Dependency-based Construction of Semantic Space Models”
- Levy and Goldberg (2014): “Dependency-Based Word Embeddings”
- Cotterell and Schütze (2015): “Morphological Word Embeddings”
- Ferraro et al. (2017): “Frame-Based Continuous Lexical Semantics through Exponential Family Tensor Factorization and Semantic Proto-Roles”
- and many more…