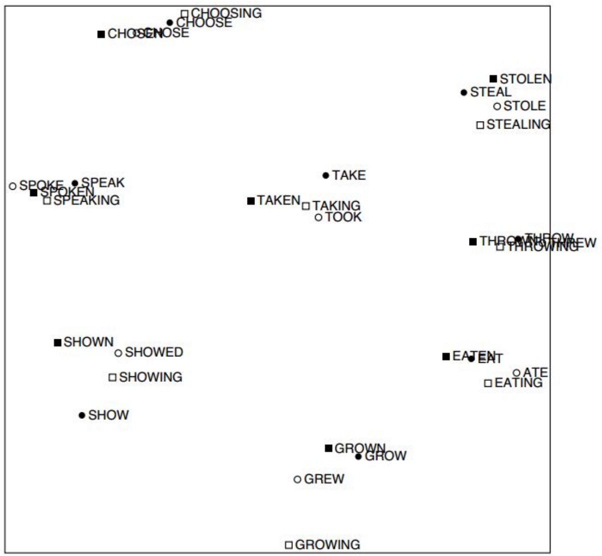
Natural Language Processing (CS5803)

Lecture 3
(Word Representations)

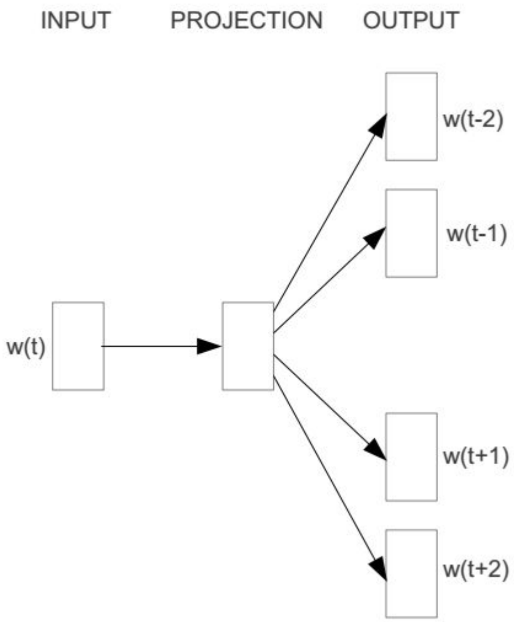
Words as vectors: Word2Vec



- Representation of a word is dictated by other surrounding words
- Assume a fixed length context window
- For example:
  - $[w_{-2}\ \ w_{-1}\ \ c\ w_{1}\ \ w_{2}]$

- Start with random initialization
- Iterate till convergence

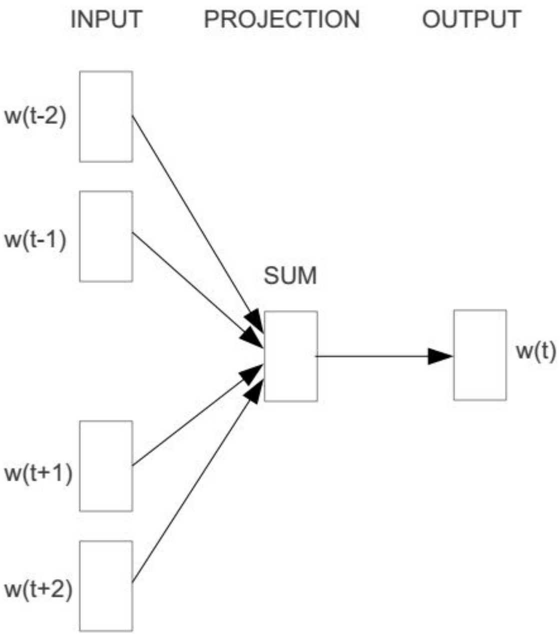
Word2Vec Models: SkipGram (SG)



Skip-gram

- Training sentence:
- ... the algorithm's asymptotic complexity is quadratic...
- w_{-2} w_{-1} c w_1 w_2
- Considering words in a context window of length 5
  - P(context|target)
  - $P([w_{-2}\ w_{-1}\ w_1\ w_2]|c) = ?$

Word2Vec Models: CBOW



- Training sentence:
- ... the algorithm's asymptotic complexity is quadratic...
- w_{-2} w_{-1} c w_{1} w_{2}
- Considering words in a context window of length 5
  - P(target|context)
  - $P(c \mid w_{-2}\ w_{-1}\ w_{1}\ w_{2}) = ?$

Objective function

- Maximize the probability of seen word-context pairs

$$f_p = \frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c} \log p(w_{t+j} \mid w_t)$$

- Where,

$$p(w_O|w_I) = \frac{\exp(dot(v'_{w_O}, v_{w_I}))}{\sum_{w=1}^{W}\exp(dot(v'_{w}, v_{w_I}))}$$

- With negative sampling, the objective function becomes:

$$\log\sigma(v'_{w_O} . v_{w_I}) + \sum_{i=1}^{k}\log\sigma(-v'_{w_i} . v_{w_I})$$

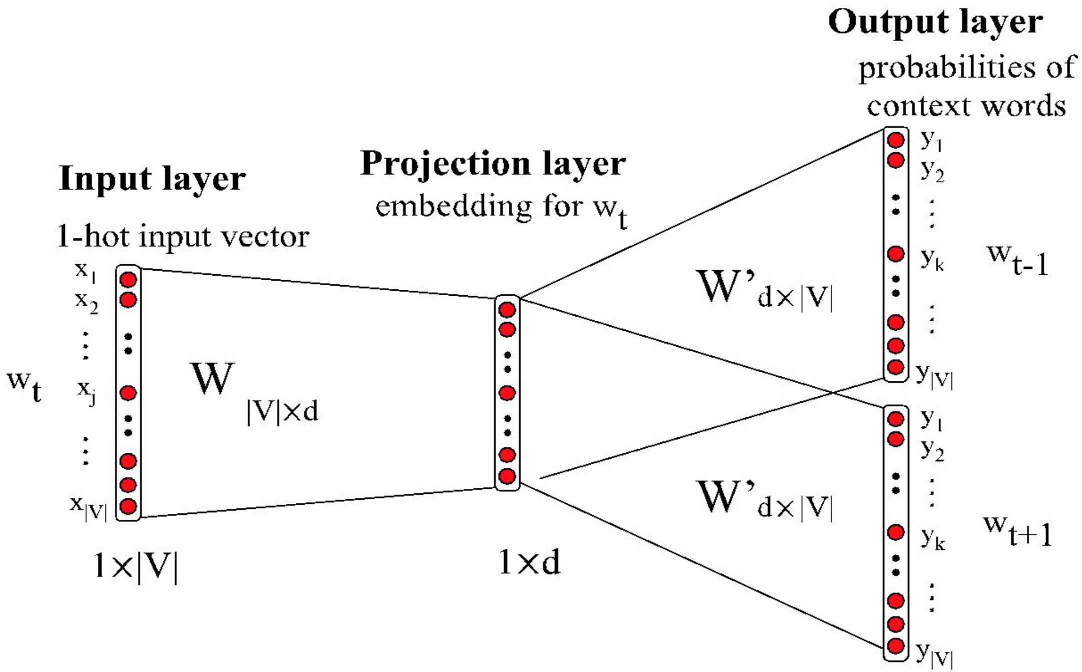
Ref: [“Distributed Representations of Words and Phrases and their Compositionality”](), by Mikolov (2013)

More examples of target and context

#1	natural	language	processing	and	machine	learning	is	fun	and	exciting	#1
	Xκ	Y(c=1)	Y(c=2)								
#2	natural	language	processing	and	machine	learning	is	fun	and	exciting	#2
	Y(c=1)	Xκ	Y(c=2)	Y(c=3)							
#3	natural	language	processing	and	machine	learning	is	fun	and	exciting	#3
	Y(c=1)	Y(c=2)	Xκ	Y(c=3)	Y(c=4)						
#4	natural	language	processing	and	machine	learning	is	fun	and	exciting	#4
		Y(c=1)	Y(c=2)	Xκ	Y(c=3)	Y(c=4)					
#5	natural	language	processing	and	machine	learning	is	fun	and	exciting	#5
			Y(c=1)	Y(c=2)	Xκ	Y(c=3)	Y(c=4)				
#6	natural	language	processing	and	machine	learning	is	fun	and	exciting	#6
				Y(c=1)	Y(c=2)	Xκ	Y(c=3)	Y(c=4)			
#7	natural	language	processing	and	machine	learning	is	fun	and	exciting	#7
					Y(c=1)	Y(c=2)	Xκ	Y(c=3)	Y(c=4)		
#8	natural	language	processing	and	machine	learning	is	fun	and	exciting	#8
						Y(c=1)	Y(c=2)	Xκ	Y(c=3)	Y(c=4)	
#9	natural	language	processing	and	machine	learning	is	fun	and	exciting	#9
							Y(c=1)	Y(c=2)	Xκ	Y(c=3)	
#10	natural	language	processing	and	machine	learning	is	fun	and	exciting	#10
								Y(c=1)	Y(c=2)	Xκ	

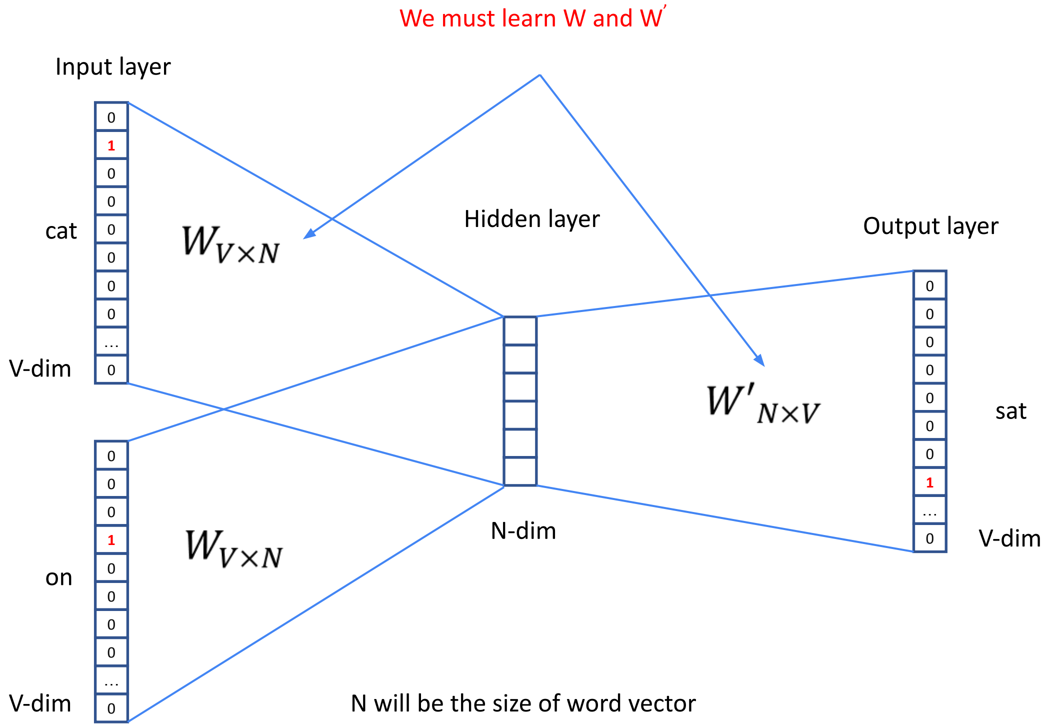
Ref: https://towardsdatascience.com/an-implementation-guide-to-word2vec-using-numpy-and-google-sheets-13445eebd281

Skip-gram

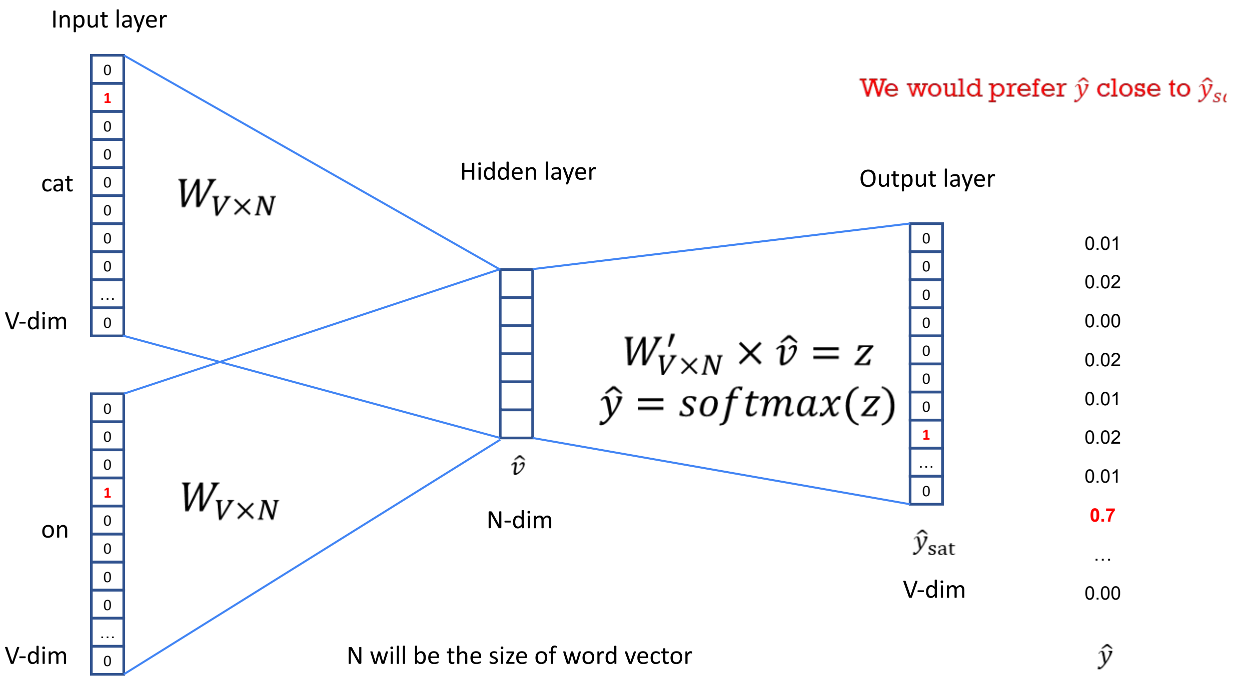


Slide courtesy of Jurafsky & Martin

Steps with example



Steps with example



Learning the representations: Step by step

Calculate Hidden Layer

#	Token	Input - w_t
0	natural	1
1	language	0
2	processing	0
3	and	0
4	machine	0
5	learning	0
6	is	0
7	fun	0
8	exciting	0

1 x 9

np.dot

Weight 1 - W1

0.236	-0.962	0.686	0.785	-0.454	-0.833	-0.744	0.677	-0.427	-0.066
-0.907	0.894	0.225	0.673	-0.579	-0.428	0.685	0.973	-0.070	-0.811
-0.576	0.658	-0.582	-0.112	0.662	0.051	-0.401	-0.921	-0.158	0.529
0.517	0.436	0.092	-0.835	-0.444	-0.905	0.879	0.303	0.332	-0.275
0.859	-0.890	0.651	0.185	-0.511	-0.456	0.377	-0.274	0.182	-0.237
0.368	-0.867	-0.301	-0.222	0.630	0.808	0.088	-0.902	-0.450	-0.408
0.728	0.277	0.439	0.138	-0.943	-0.409	0.687	-0.215	-0.807	0.612
0.593	-0.699	0.020	0.142	-0.638	-0.633	0.344	0.868	0.913	0.429
0.447	-0.810	-0.061	-0.495	0.794	-0.064	-0.817	-0.408	-0.286	0.149

9 x 10

=

Hidden Layer - h
0.236
-0.962
0.686
0.785
-0.454
-0.833
-0.744
0.677
-0.427
-0.066

1 x 10

- Ref: https://towardsdatascience.com/an-implementation-guide-to-word2vec-using-numpy-and-google-sheets-13445eebd281

Learning the representations: Step by step

Calculate y_pred

Hidden Layer - h		Weight 2 - W2										Output Layer	Softmax - y_pred
0.236		-0.868	-0.406	-0.288	-0.016	-0.560	0.179	0.099	0.438	-0.551		1.258	0.218
-0.962		-0.395	0.890	0.685	-0.329	0.218	-0.852	-0.919	0.665	0.968		-1.369	0.016
0.686		-0.128	0.685	-0.828	0.709	-0.420	0.057	-0.212	0.728	-0.690		-1.828	0.010
0.785		0.881	0.238	0.018	0.622	0.936	-0.442	0.936	0.586	-0.020		1.196	0.205
-0.454	**np.dot**	-0.478	0.240	0.820	-0.731	0.260	-0.989	-0.626	0.796	-0.599	=	0.545	0.107
-0.833		0.679	0.721	-0.111	0.083	-0.738	0.227	0.560	0.929	0.017		1.113	0.189
-0.744		-0.690	0.907	0.464	-0.022	-0.005	-0.004	-0.425	0.299	0.757		1.333	0.235
0.677		-0.054	0.397	-0.017	-0.563	-0.551	0.465	-0.596	-0.413	-0.395		-1.528	0.013
-0.427		-0.838	0.053	-0.160	-0.164	-0.671	0.140	-0.149	0.708	0.425		-2.335	0.006
-0.066		0.096	-0.995	-0.313	0.881	-0.402	-0.631	-0.660	0.184	0.487		1 x 9	1 x 9
1 x 10		10 x 9											

For more details regarding weight updates, you may visit the paper "word2vec Parameter Learning Explained"

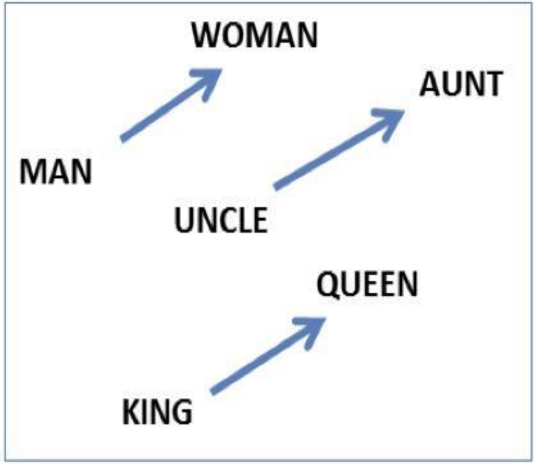
Word2Vec: References

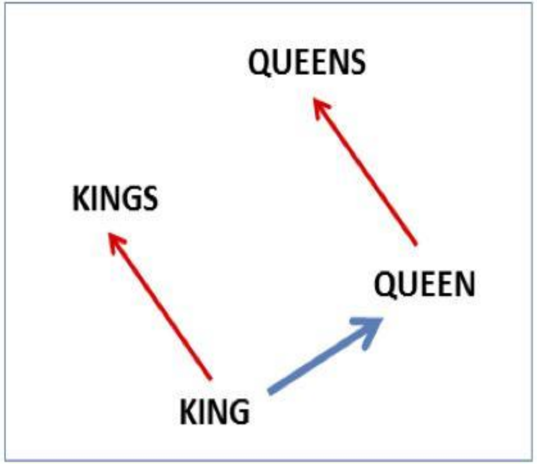
- Distributed Representations of Words and Phrases and their Compositionality
- https://www.geeksforgeeks.org/python-word-embedding-using-word2vec/
- https://radimrehurek.com/gensim/models/word2vec.html

Analogy: Embeddings capture relational meaning!

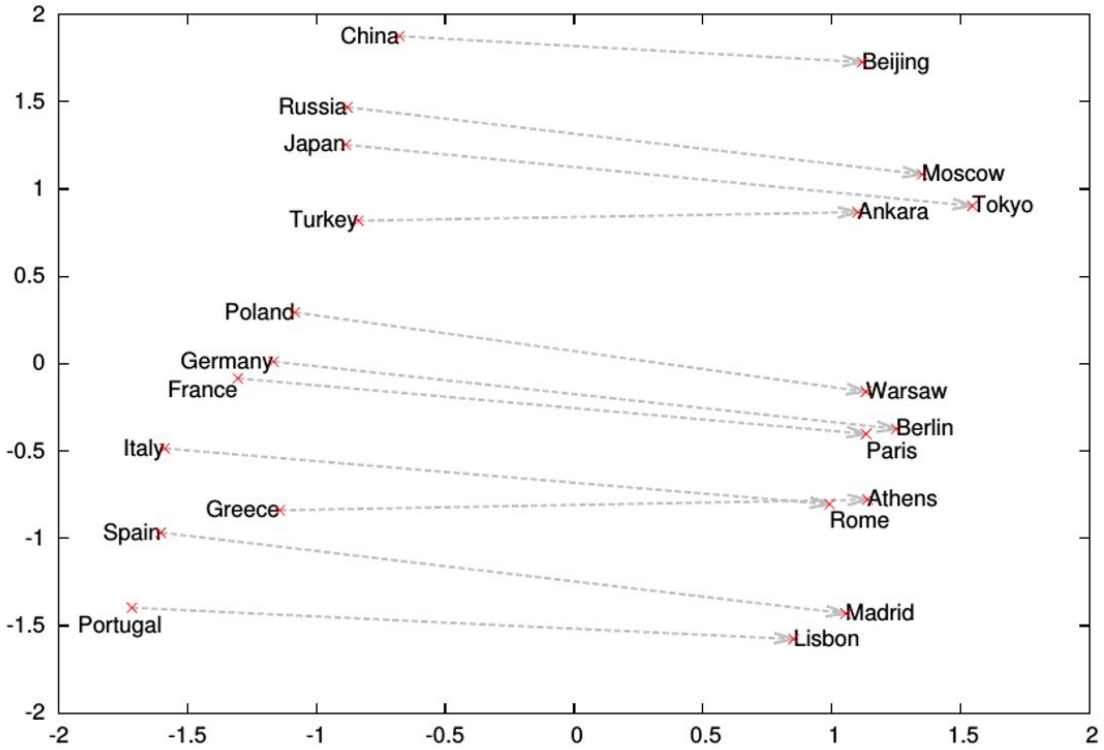
vector(*'king'*) - vector(*'man'*) + vector(*'woman'*) ≈ vector('queen')

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

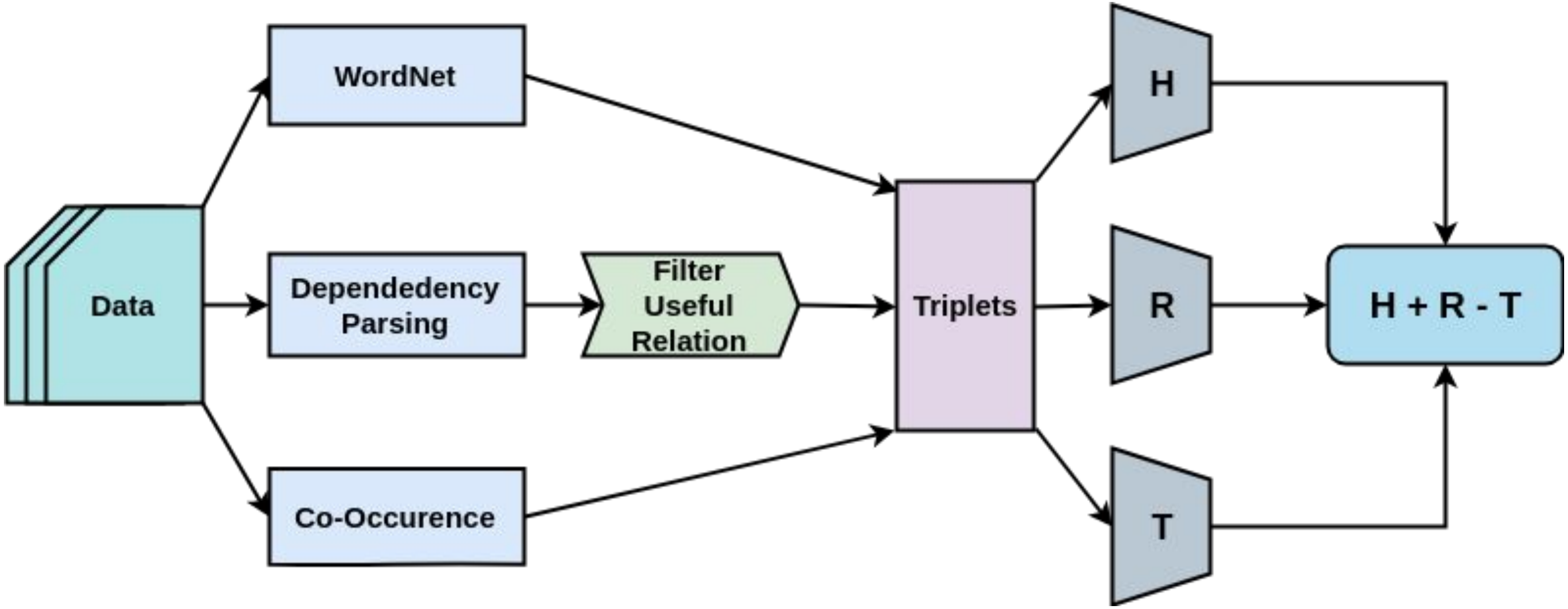




Word analogies



Multicontext representation learning



Evaluation on Word Similarity Task

	W2V	Glove	TED	LG	MCRL-S	MCRL-U
SimVerb-3500	0.116	0.082	0.075	0.253	**0.528**	0.527
MEN-TR-3k	0.571	0.420	0.555	0.562	0.659	**0.662**
RW-STANFORD	0.406	0.334	0.203	**0.479**	0.446	0.420
SIMLEX-999	0.203	0.154	0.181	0.394	**0.502**	0.496
MTurk-771	0.477	0.345	0.440	0.580	0.599	**0.616**
WS-353-ALL	0.566	0.434	0.553	0.581	**0.620**	0.598
MTurk-287	0.539	0.463	0.515	0.566	0.590	**0.601**
WS-353-REL	**0.555**	0.414	0.485	0.436	0.509	0.521
WS-353-SIM	0.611	0.503	0.550	**0.728**	0.722	0.706
VERB-143	0.254	0.121	0.179	0.322	**0.399**	0.396
YP-130	0.303	0.170	0.118	0.273	0.688	**0.702**
RG-65	0.500	0.335	0.272	0.669	**0.788**	0.720
MC-30	0.548	0.308	0.445	**0.777**	0.767	0.747
Average	0.435	0.314	0.352	0.509	**0.601**	0.593
Weighted Average	0.362	0.273	0.298	0.438	**0.559**	0.555

(a) Wikipedia Corpus.

Evaluation Dataset	W2V	Glove	TED	LG	MCRL-S	MCRL-U
SimVerb-3500	0.174	0.145	0.020	0.313	**0.569**	0.550
MTR-3k	0.434	0.346	0.207	0.480	**0.614**	0.609
RW-STANFORD	**0.552**	0.491	0.069	0.550	0.423	0.409
SIMLEX-999	0.165	0.135	0.032	0.330	**0.491**	0.473
MTurk-771	0.367	0.280	0.095	0.463	0.607	**0.610**
WS-353-ALL	0.363	0.322	0.145	0.393	0.547	**0.552**
MTurk-287	0.513	0.334	0.124	0.459	**0.559**	0.541
WS-353-REL	0.272	0.268	0.111	0.285	0.415	**0.422**
WS-353-SIM	0.489	0.406	0.175	0.588	0.668	**0.708**
VERB-143	0.240	0.084	0.127	**0.411**	0.303	0.347
YP-130	0.179	0.197	0.111	0.283	0.706	**0.750**
RG-65	0.478	0.257	0.428	0.639	**0.701**	0.678
MC-30	0.393	0.446	0.379	**0.789**	0.715	0.706
Average	0.355	0.285	0.156	0.460	0.563	**0.566**
Weighted Average	0.342	0.282	0.098	0.422	**0.548**	0.538

(b) Reviews Corpus

WordSim353: http://www.cs.technion.ac.il/~gabr/resources/data/wordsim353/

Evaluation on semantic textual Similarity Task

Evaluation Dataset	W2V	Glove	TED	LG	MCRL-S	MCRL-U
SICK	0.597	0.544	0.489	0.576	0.639	**0.654**
2012_MSRvid	**0.414**	0.265	0.112	0.103	0.255	0.303
2012_SMTeuroparl	0.471	0.411	**0.512**	0.438	0.458	0.445
2012_surprise.OnWN	**0.551**	0.528	0.316	0.408	0.510	0.536
2013_headlines	0.599	0.570	0.512	0.522	**0.615**	0.604
2014_OnWN	0.431	0.315	0.289	0.437	0.514	**0.538**
2014_headlines	0.556	0.529	0.437	0.475	**0.565**	0.556
2014_images	**0.499**	0.402	0.149	0.258	0.415	0.429
2014_tweet-news	0.655	0.621	0.463	0.555	0.648	**0.656**
2015_headlines	**0.646**	0.629	0.584	0.551	0.641	0.634
2015_images	**0.572**	0.508	0.304	0.416	0.540	0.542
2013_OnWN	0.260	0.097	0.266	0.364	0.465	**0.514**
2012_surprise.SMTnews	**0.410**	0.406	0.102	0.350	0.392	0.406
2015_belief	**0.507**	0.452	0.252	0.309	0.451	0.478
2014_deft-news	0.570	0.497	0.366	0.538	0.570	**0.584**
2016_answer-answer	0.196	0.165	0.193	0.204	0.275	**0.290**
2017_track5.en-en	**0.529**	0.508	0.253	0.416	0.478	0.505
2016_headlines	0.612	0.569	0.549	0.566	**0.652**	0.639
2016_plagiarism	**0.535**	0.434	0.212	0.441	0.476	0.501
2013_FNWN	0.218	0.168	0.156	0.211	0.220	**0.237**
Average	0.491	0.431	0.326	0.407	0.489	**0.503**
Weighted Average	0.544	0.483	0.403	0.474	0.554	**0.568**

(a) Wikipedia Corpus

Evaluation Dataset	W2V	Glove	TED	LG	MCRL-S	MCRL-U
SICK	0.587	0.487	0.403	0.547	0.634	**0.640**
2012_MSRvid	**0.359**	0.168	0.049	0.108	0.301	0.288
2012_SMTeuroparl	0.276	0.240	0.245	0.226	**0.317**	0.314
2012_surprise.OnWN	0.491	0.401	0.260	0.371	0.523	**0.547**
2013_headlines	0.378	0.308	0.339	0.274	**0.535**	0.527
2014_OnWN	**0.557**	0.462	0.326	0.403	0.497	0.520
2014_headlines	0.373	0.303	0.303	0.279	0.512	**0.514**
2014_images	**0.423**	0.291	0.190	0.259	0.420	0.416
2014_tweet-news	0.549	0.443	0.273	0.451	0.629	**0.641**
2015_headlines	0.474	0.429	0.413	0.386	**0.582**	0.581
2015_images	0.518	0.415	0.269	0.423	0.545	**0.567**
2013_OnWN	0.451	0.410	0.279	0.341	0.425	**0.459**
2012_surprise.SMTnews	0.295	0.238	0.055	0.280	0.364	**0.370**
2015_belief	0.397	0.251	0.181	0.259	**0.497**	0.481
2014_deft-news	0.460	0.441	0.347	0.433	**0.532**	0.521
2016_answer-answer	0.221	0.105	0.097	0.202	0.315	**0.322**
2017_track5.en-en	0.491	0.358	0.211	0.383	0.531	**0.539**
2016_headlines	0.356	0.319	0.286	0.295	**0.588**	0.579
2016_plagiarism	0.434	0.286	0.210	0.364	0.489	**0.505**
2013_FNWN	0.203	0.164	0.231	0.163	0.227	**0.242**
Average	0.415	0.326	0.248	0.322	0.473	**0.479**
Weighted Average	0.493	0.397	0.314	0.413	0.539	**0.544**

(b) Reviews Corpus

GloVE

- Stands for GloVe: Global Vectors for Word Representation
  - Emphasizes on co-occurrence with context/probe words
- Learns two representations ($W, \tilde{W}$) for each word

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

- Focus on ratio of co-occurrence probabilities
  - Given words w_i, w_j, and a probe word w_k, model their co-occurrence probability: $F(w_i, w_j, w_k) = P_{ik}/P_{jk}$

GloVE

- Word embeddings are in linear structures
- Natural way of defining F: use vector subtraction, multiplication

$$F\left((w_i - w_j)^T \tilde{w}_k\right) = \frac{P_{ik}}{P_{jk}}$$

Control the form that F can take

$$F\left((w_i - w_j)^T \tilde{w}_k\right) = \frac{F(w_i^T \tilde{w}_k)}{F(w_j^T \tilde{w}_k)}$$

Model P using operations so that role of input and context word can be interchanged later

$$w_i^T \tilde{w}_k = \log(P_{ik}) = \log(X_{ik}) - \log(X_i)$$

Model F as exp(.)

$$w_i^T \tilde{w}_k + b_i + \tilde{b}_k = \log(X_{ik})$$

Introduce bias terms and absorb $\log(X_i)$

$$\boxed{J = \sum_{i,j=1}^{V} f\left(X_{ij}\right)\left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij}\right)^2}$$

Final objective function

GloVE (Summary)

- Stands for GloVe: Global Vectors for Word Representation
  - Emphasizes on co-occurrence with context words
- Learns two representations ($W, \tilde{W}$) for each word
- The prediction problem is given by:

$$w_i^T \cdot \tilde{w}_j + b_i + \tilde{b}_j = \log X_{i,j}$$

- The objective function:

$$J = \sum_{i,j=1}^{V} f(X_{i,j}) \left(w_i^T \cdot \tilde{w}_j + b_i + \tilde{b}_j - \log X_{i,j}\right)^2$$

Embeddings reflect societal bias

- Ask "Paris : France :: Tokyo : x"
  - x = Japan
- Ask "father : doctor :: mother : x"
  - x = nurse
- Ask "man : computer programmer :: woman : x"
  - x = homemaker

Bolukbasi, Tolga, Kai-Wei Chang, James Y. Zou, Venkatesh Saligrama, and Adam T. Kalai. "Man is to computer programmer as woman is to homemaker? debiasing word embeddings." In *Advances in Neural Information Processing Systems*, pp. 4349-4357. 2016.

Embeddings Reflect Societal Bias

Extreme *she*	Extreme *he*
1. homemaker	1. maestro
2. nurse	2. skipper
3. receptionist	3. protege
4. librarian	4. philosopher
5. socialite	5. captain
6. hairdresser	6. architect
7. nanny	7. financier
8. bookkeeper	8. warrior
9. stylist	9. broadcaster
10. housekeeper	10. magician

Gender stereotype *she*-*he* analogies

sewing-carpentry	registered nurse-physician	housewife-shopkeeper
nurse-surgeon	interior designer-architect	softball-baseball
blond-burly	feminism-conservatism	cosmetics-pharmaceuticals
giggle-chuckle	vocalist-guitarist	petite-lanky
sassy-snappy	diva-superstar	charming-affable
volleyball-football	cupcakes-pizzas	lovely-brilliant

Gender appropriate *she*-*he* analogies

queen-king	sister-brother	mother-father
waitress-waiter	ovarian cancer-prostate cancer	convent-monastery

$$S_{(a,b)}(x,y) = \cos\left(\vec{a}-\vec{b}, \vec{x}-\vec{y}\right) \text{ if } \|\vec{x}-\vec{y}\| \leq \delta, \quad 0 \text{ else}$$

Identifying and quantifying bias in word embeddings

- Assumption: The aspect of bias is known. E.g. gender
- Find the "gender" dimension
  - Collect explicit gender-based word pairs (f, m): (woman, man), (mother, father), (gal, guy), (girl, boy), (she, he)
  - Get the gender dimension as (f-m) [How?]
- Collect a set N of gender neutral words

- Compute the gender component in elements from N
  - DirectBias = $(1/|N|)\sum_{w \in N}|\cos(w,g)|$
  - Can be raised to the power c

Identifying and quantifying bias in word embeddings

- How to capture indirect bias?
- Direct bias: component along gender dimension
- Indirect bias: Component along its perpendicular

- Need to find the component to the perpendicular of the "gender" dimension
- Component of vector a along vector b:
  - Scalar Component: $comp_b(a) = (a.b)/|b|$
  - Vector component: $comp_b(a).b$
- $w_g = (w.g)g$, $w_\perp = w - w_g$

- IndirectBias $B(w,v) = (w.v - (w_\perp - v_\perp)/(|w_\perp| - |v_\perp|)) / (w.v)$

$$J = \sum_{i,j=1}^{V} f\left(X_{ij}\right)\left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - log(X_{ij})\right)^2 + \lambda \cos(w_i, g) + \gamma \cos(\tilde{w}_j, g)$$

A simple technique for debiasing GloVE

Identifying and quantifying bias in word embeddings

softball extreme	gender portion	after debiasing
1. pitcher	-1%	1. pitcher
2. bookkeeper	20%	2. infielder
3. receptionist	67%	3. major leaguer
4. registered nurse	29%	4. bookkeeper
5. waitress	35%	5. investigator

football extreme	gender portion	after debiasing
1. footballer	2%	1. footballer
2. businessman	31%	2. cleric
3. pundit	10%	3. vice chancellor
4. maestro	42%	4. lecturer
5. cleric	2%	5. midfielder

Identifying and quantifying bias in word embeddings

Figure 3: Selected words projected along two axes: x is a projection onto the difference between the embeddings of the words *he* and *she*, and y is a direction learned in the embedding that captures gender neutrality, with gender neutral words above the line and gender specific words below the line. Our hard debiasing algorithm removes the gender pair associations for gender neutral words. In this figure, the words above the horizontal line would all be collapsed to the vertical line.

Reference: [Man is to Computer Programmer as Woman is to Homemaker? Debiasing Word Embeddings]() NeurIPS 2016
Another version is [here]().