CSI5386: Natural Language Processing

Assignment 1

Corpus analysis and word embeddings

Lingfeng Zhang(300134245, lzhan278@uottawa.ca)
Yu Sun(8472921, ysun118@uottawa.ca)

The GitHub link for Part_1 code is in https://github.com/RichardChangCA/CSI-5386-Natural-Language-Processing-Assignments
The GitHub link for Part_2 code is in https://github.com/RichardChangCA/word-embeddings-benchmarks

To fulfill this assignment, firstly we are do some research together, like which tool to use for tokenization, what scenario we need to consider and what the different among the embedding model. After discussion, we begin to do the implementation and write the report.
For implementation, Lingfeng mainly is responsible for coding while Yu is for reviewing.
For report, Yu is mainly for writing the report and Lingfeng is for revising it.

Part 1: Corpus processing: tokenization, and word counting

Our goal here is to count the number of occurrences of each token in the corpus. Based on our preliminary research, we have made these decisions by steps for pre-processing:

1. **Remove non-English characters.** Tokenization is language-specific, in our case, we consider English as our language base, to avoid potential noises, we remove all non-English characters from the corpus first to clean the strings.
2. **Break on whitespaces.** English usually uses whitespaces to separate the words, so we could chop the string on whitespaces, but there are few non-breaking cases we need to consider while proceed this step, such as commonly names “San Francisco”, phone numbers (+1) 123-4567, dates (Feb 3, 2020).
3. **Lemmatization.** We should group inflected form of the same word together, such as does, done, doing, did should be transformed to do.
4. **Stemming.** Similar to lemmatization, we should transform inflected form of the same word into its root form, such as cities to city.
5. **Transform different spellings.** Same word in British English has different spellings verse in American English. To group the same token together, we should Americanize those words in British spellings, such as colour to color.
6. **Polysemy word.** We have considered the case that a word that could have different meanings, such as ‘have’ in ‘I have a pen’ verses ‘I have eaten an apple’, ideally, we should group these cases separately by its meanings, but in our case,

we only care the occurrences of each token, so we decided to consider them as one same token.

7. **Remove punctuation characters.** Punctuation characters are not considered as tokens and since we only care about the count of the occurrences of each token in our case, we should remove them. But in English, there are several particular scenarios of using punctuation characters need to be specially taken care of:
   a) Period or comma, such as emails addresses, web urls and IP addresses. These type of character sequences are introduced by computer technology, should be consider as a whole. Depends on what is our goal for tokenization, we should consider whether to keep these as part of our tokens or not. For example, in a scenario that we want to analysis which website is a hit based on tweet, these tokens should be kept. But in other scenarios, such as we want to do a sematic analysis, these tokens might not be needed. In our case here, we decide to keep these.
   b) Hyphen or dash, such as compound term 'part-time' and phone number 613-000-0000. Hyphen is usually used for joining two or more words into a compound term, such compound term should not be broken.
   c) Apostrophe for possession and contractions, such as 'John's' and 'aren't'. Blindly remove apostrophe for such cases will end up with incorrect wordings, so we should replace 'aren't' with 'are not'
   d) Many special characters should be removed simultaneously, like "\n" etc.
8. **Remove stop words.** Depends on what goal we have, removing stop words could be a step that proceed with conscious, and the result will also be affected by how the stop words list is defined. In our case, we decided to not remove stop words for our first analysis, based on the first result, we could design the stop words list and apply this step. Moreover, we used the stop words list from Wikipedia.
9. **Convert strings into lowercase.** We need to convert the tokens into the same case format, so same tokens could be grouped together when case sensitive count method is applied to calculate the occurrences.
10. **Transfer special case into common case.** We converted the special case like "It's" to common cases like "it is". We found the special cases summary list from Wikipedia and use this source file to transfer. Sometimes, different special cases have different common cases, for instance, "ain't" can be transferred into "am not", "are not", "is not", "has not", or "have not" and finaly we decide to use the most frequent one.
11. **Wrap various whitespace into one.** Sometimes in twitter, user may use many whitespaces to represent a single whitespace, so we decided to wrap them together.

In particular, we use NLTK, genism and spacy, some opensource platforms for building python programs to work with natural language data. The results we got are showed below:

a)

Table 1 tokenizer's output for the first 20 sentences in the corpus.

1	['save', 'bbc', 'world', 'service', 'from', 'savage', 'cuts','http://www.petitionbuzz.com/petitions/savews']
2	['a', 'lot', 'of', 'people', 'always', 'make', 'fun', 'about', 'the', 'end', 'of', 'the', 'world', 'but', 'the', 'question', 'is', '..', "'are", 'u', 'ready', 'for', 'it', '?', '..']
3	['rethink', 'group', 'positive', 'in', 'outlook', ':', 'technology', 'staffing', 'specialist', 'the', 'rethink', 'group', 'expects', 'revenues', 'to', 'be', '“', 'marg', '...', 'http://bit.ly/hfjtmy']
4	['"', 'zombie', '"', 'fund', 'manager', 'phoenix', 'appoints', 'new', 'ceo', ':', 'phoenix', 'buys', 'up', 'funds', 'that', 'have', 'been', 'closed', 'to', 'new', 'business', 'and', '...', 'http://bit.ly/dxrlh5']
5	['latest', ':', ':', 'top', 'world', 'releases', 'http://globalclassified.net/2011/02/top-world-releases-2/']
6	['cdt', 'presents', 'alice', 'in', 'wonderland', '-', 'catonsville', 'dinner', 'has', 'posted', '"', 'cdt', 'presents', 'alice', 'in', 'wonderland', '"', 'to', 'the', '...', 'http://fb.me/gmicayt3']
7	['territory', 'manager', ':', 'location', ':', 'calgary', ',', 'alberta', ',', 'canada', 'job', 'category', ':', 'bu', '...', 'http://bit.ly/e3o7mt', '#', 'jobs']
8	['i', 'cud', 'murder', 'sum1', 'today', 'n', 'not', 'even', 'flinch', 'i', 'am', 'tht', 'fukin', 'angry', 'today']
9	['bbc', 'news', '-', 'today', '-', 'free', 'school', 'funding', 'plans', '"', 'lack', 'transparency', '"', '-', 'http://news.bbc.co.uk', '/', 'today', '/', 'hi', '/', 'today', '/', 'newsid_9389000/9389467.stm', '\xa0', '…']
10	['manchester', 'city', 'council', 'details', 'saving', 'cuts', 'plan', ':', 'http://bbc.in/fypypc', '...', 'depressing', '.', 'apparently', 'we', '’re', '4th', 'most', 'deprived', '&', 'top', '5', 'hardest', 'hit']
11	['http://bit.ly/e0ujdp', ',', 'if', 'you', 'are', 'interested', 'in', 'professional', 'global', 'translation', 'services']
12	['fitness', 'first', 'to', 'float', 'but', 'is', 'not', 'the', 'full', 'service', 'model', 'dead', '?', 'http://bit.ly/evfleb']
13	['david', 'cook', '!', 'http://bit.ly/fkj2gk', 'has', 'the', 'mostest', 'beautiful', 'smile', 'in', 'the', 'world', '!']
14	['piss', 'off', '.', 'cnt', 'stand', 'lick', 'asses']
15	['beware', 'the', 'blue', 'meanies', ':', 'http://bit.ly/hu8ijz', '#', 'cuts', '#', 'thebluemeanies']
16	['como', 'perde', 'os', 'dentes', 'no', 'world', 'of', 'warcraft', '-', 'via', 'alisson', 'http://ow.ly/1bebpo']
17	['how', 'exciting', '!', 'rt', '@bunchesuk', ':', 'hello', '!', 'what', 'has', 'happening', 'in', 'your', 'world', '?', 'we', 'are', 'all', 'gearing', 'up', 'for', '#', 'valentines', 'with', 'bouquets', 'flying', 'out', 'the', 'door', '.']
18	['i', 'had', 'very', 'much', 'appreciate', 'it', 'if', 'people', 'would', 'stop', 'broadcasting', 'asking', 'me', 'to', 'add', 'people', 'on', 'bbm', '.']
19	['@samanthaprabu', 'sam', 'i', 'knw', 'u', 'r', 'a', 'cricket', 'fan', 'r', 'u', 'watching', 'any', 'of', 'the', 'world', 'cup', 'matches']
20	['john', 'baer', ':', 'who', 'did', 'not', 'see', 'this', 'coming', '?', ':', 'to', 'those', 'who', 'know', 'ed', 'and', 'midge', 'rendell', '-', 'heck', ',', 'to', 'the', 'philly', 'world', 'at', 'la', '...', 'http://bit.ly/ii6weo']

b)

Table 2 the count of token and type

token count	type count	type/token radio
931699	97008	0.1041

c)

Table 3 the first 100 token and its frequency

number	word	frequency	number	word	frequency	number	word	frequency
0	#	30235	33	new	2978	66	…	1389
1	:	23877	34	are	2848	67	so	1388
2	the	21129	35	news	2717	68	an	1372

3	,	18741	36	&	2546	69	25-Jan	1337
4	.	17874	37	from	2472	70	social	1332
5	to	13877	38	this	2439	71	media	1330
6	-	13506	39	be	2271	72	like	1315
7	...	12636	40	/	2185	73	white	1307
8	!	11896	41	have	2182	74	via	1295
9	in	10839	42	will	2128	75	bowl	1289
10	of	10577	43	by	2052	76	get	1287
11	a	10459	44	do	1979	77	about	1280
12	i	8456	45	egyptian	1896	78	but	1269
13	and	8360	46	your	1822	79	2	1227
14	for	7112	47	obama	1814	80	if	1225
15	on	6286	48	state	1802	81	they	1194
16	is	6251	49	me	1789	82	can	1193
17	?	5506	50	us	1775	83	2011	1148
18	"	5395	51	am	1763	84	$	1143
19	(	4612	52	we	1747	85	how	1125
20	)	4544	53	just	1746	86	more	1117
21	rt	4452	54	as	1682	87	de	1097
22	's	4216	55	out	1647	88	union	1057
23	you	4113	56	all	1571	89	people	1048
24	at	3909	57	what	1471	90	he	1025
25	it	3834	58	no	1464	91	who	1020
26	not	3571	59	up	1459	92	security	1014
27	egypt	3269	60	now	1444	93	airport	1000
28	'	3208	61	|	1422	94	love	994
29	with	3145	62	super	1416	95	today	989
30	my	3122	63	world	1412	96	or	986
31	has	3122	64	..	1411	97	president	977
32	that	3023	65	was	1405	98	day	977
						99	u	976

d)We got **61310** tokens which are appeared only once in the corpus.

e) After excluding punctuation and other symbols, we got 771862 tokens the type/token ratio is 0.1256. The first 100 tokens are listed below:

Table 4 the first 100 token and its frequency excluding punctuation

number	word	frequency	number	word	frequency	number	word	frequency
0	the	21129	33	your	1822	66	if	1225
1	to	13877	34	obama	1814	67	they	1194
2	...	12635	35	state	1802	68	can	1193
3	in	10839	36	me	1789	69	2011	1148
4	of	10577	37	us	1775	70	how	1125

5	a	10459	38	am	1763	71	more	1117
6	i	8456	39	we	1747	72	de	1097
7	and	8360	40	just	1746	73	union	1057
8	for	7112	41	as	1682	74	people	1048
9	on	6286	42	out	1647	75	he	1025
10	is	6251	43	all	1571	76	who	1020
11	rt	4452	44	what	1471	77	security	1014
12	's	4216	45	no	1464	78	airport	1000
13	you	4113	46	up	1459	79	love	994
14	at	3909	47	now	1444	80	today	989
15	it	3834	48	super	1416	81	or	986
16	not	3571	49	world	1412	82	president	977
17	egypt	3269	50	..	1410	83	day	977
18	with	3145	51	was	1405	84	u	966
19	has	3122	52	so	1388	85	release	956
20	my	3122	53	…	1388	86	law	955
21	that	3023	54	an	1372	87	one	953
22	new	2978	55	25-Jan	1337	88	time	942
23	are	2848	56	social	1332	89	his	922
24	news	2717	57	media	1330	90	good	888
25	from	2472	58	like	1315	91	video	888
26	this	2439	59	white	1307	92	house	883
27	be	2271	60	via	1295	93	mubarak	873
28	have	2182	61	bowl	1289	94	over	863
29	will	2128	62	get	1287	95	jobs	857
30	by	2052	63	about	1280	96	protests	849
31	do	1979	64	but	1269	97	when	848
32	egyptian	1896	65	2	1227	98	show	844
						99	service	831

f)From the list of words, after excluding stop words, we got 502960 tokens and the type/token ratio is 0.1914. The first 100 tokens are listed below:

Table 5 the first 100 token and its frequency excluding shop words

number	word	frequency	number	word	frequency	number	word	frequency
0	rt	4452	33	protests	849	66	post	519
1	's	4216	34	service	831	67	budget	517
2	egypt	3269	35	says	748	68	home	514
3	news	2717	36	phone	747	69	weather	513
4	egyptian	1896	37	police	726	70	watch	510
5	obama	1814	38	global	713	71	business	501
6	state	1802	39	’s	712	72	top	493

7	super	1416	40	4	710	73	governme nt	481
8	world	1412	41	dog	705	74	food	476
9	25-Jan	1337	42	free	701	75	u.s.	472
10	social	1332	43	back	682	76	right	472
11	media	1330	44	bbc	669	77	online	470
12	white	1307	45	taco	667	78	car	461
13	bowl	1289	46	bell	666	79	organic	459
14	2	1227	47	3	664	80	tcot	455
15	2011	1148	48	protesters	641	81	blog	451
16	union	1057	49	return	638	82	address	447
17	people	1048	50	live	637	83	attack	446
18	securit y	1014	51	rite	626	84	peace	445
19	airport	1000	52	toyota	624	85	10	443
20	love	994	53	special	614	86	mexico	428
21	today	989	54	know	601	87	pakistan	426
22	presid ent	977	55	iran	599	88	big	424
23	releas e	956	56	:)	563	89	5	423
24	law	955	57	think	562	90	help	422
25	video	888	58	ap	555	91	moscow	414
26	house	883	59	health	551	92	museum	414
27	mubar ak	873	60	court	547	93	protest	410
28	jobs	857	61	twitter	544	94	check	404
29	cairo	812	62	man	543	95	life	403
30	job	782	63	crash	534	96	date	396
31	lol	776	64	tv	532	97	jordan	394
32	energy	753	65	cuts	521	98	work	394
						99	nt	392

g)After excluding stop words and punctuation, we got 367611 unique bigrams. The first 100 bigrams are listed below:

Table 6 the first bigrams and its frequency

number	word	frequency	number	word	frequency
0	super bowl	1205	50	hillary clinton	98
1	social media	994	51	bbc world	96
2	state union	933	52	tcot tlot	95
3	taco bell	579	53	egyptian protests	94

4	white house	365	54	media marketing	93
5	union address	357	55	president hosni	93
6	global warming	314	56	egyptian president	92
7	egypt 43855	312	57	today 's	88
8	obama 's	297	58	federal judge	88
9	43855 egypt	294	59	egyptian police	83
10	keith olbermann	275	60	lol rt	82
11	president obama	268	61	current tv	82
12	bowl xlv	209	62	anti government	79
13	world cup	202	63	union speech	79
14	white stripes	201	64	kate middleton	76
15	moscow airport	196	65	bid date	75
16	barack obama	191	66	security forces	73
17	rahm emanuel	189	67	ca nt	71
18	bbc news	181	68	egyptian people	71
19	health care	179	69	weight loss	70
20	united states	175	70	egyptian government	70
21	egypt protests	154	71	moscow 's	69
22	press release	144	72	secretary state	69
23	video --	144	73	airport security	69
24	budget cuts	143	74	share friends	68
25	julian assange	143	75	0 bid	68
26	egypt 's	141	76	domodedovo airport	67
27	's state	139	77	fox news	67
28	customer service	138	78	egyptian embassy	67
29	hosni mubarak	138	79	ai nt	66
30	http://www.bbc.co.uk news	136	80	60 minutes	66
31	president barack	134	81	care law	66
32	youtube video	122	82	international airport	66
33	supreme court	122	83	gabrielle giffords	66
34	tahrir square	121	84	special olympics	65
35	protests egypt	120	85	egyptian army	64
36	glenn beck	117	86	tear gas	64

37	ap ap	116	87	cowboys stadium	64
38	world news	114	88	global war	63
39	news world	111	89	unemployment rate	63
40	egyptian protesters	109	90	cell phone	62
41	birth certificate	109	91	state tv	61
42	world service	107	92	anthony hopkins	61
43	middle east	107	93	climate change	60
44	egyptian museum	106	94	43855 rt	59
45	release date	105	95	social networking	58
46	blog post	104	96	fifa soccer	58
47	breaking news	103	97	24 hours	58
48	prime minister	103	98	shorty award	58
49	phone hacking	100	99	green bay	57

Part2 Evaluation word embeddings

Our goal here is comparing different word embedding models and evaluating its performance on different dataset. There are two main approaches for learning word embedding, one is context-based method like Word2Vec, while count-based method, like Glove. Based on our preliminary research, we chose 8 pre-trained word embeddings from these two categories:

- **CBOW** is a general Word2Vec model which learn to predict the word by the context[1].
- **Skip-grams** is a general Word2Vec model which is designed to predict the context[1].
- **PDC** (Parallel Document Context) is an extension of CBOW model, by adding an extra document information in parallel direction[2].
- **HDC** (Hierarchical Document Context) is an extension of Skip-grams model, by introducing the document-word prediction layer[2].
- **Glove** a count-based method which combines global matrix factorization and local context window methods[3].
- **LexVec** is also a count based method. It is a matrix factorization method that use WSNS to factorize the PPMI matrix into two lower rank matrices[4].
- **SG_GoogleNews** is a pre-trained Word2Vec model trained on part of Google News dataset.
- **Conceptnet_bumberbatch** is a pre-trained word embedding model by using ConceptNet and distributional semantics[5].

The pre-trained model and their parameters we used are listed below:

	Source	dimension	corpus

CBOW	https://vsmlib.readthedocs.io/en/latest/tutorial/getting_vectors.html#pre-trained-vsms	250	Wikipedia
Skip-grams	https://vsmlib.readthedocs.io/en/latest/tutorial/getting_vectors.html#pre-trained-vsms	250	Wikipedia
PDC	http://ofey.me/projects/wordrep/	300	Wikipedia
HDC	http://ofey.me/projects/wordrep/	300	Wikipedia
Glove	https://nlp.stanford.edu/projects/glove/	300	Wikipedia+Gigaword 5
LexVec	https://github.com/alexandres/lexvec	300	commoncrawl-W+C
SG_GoogleNews	https://code.google.com/archive/p/word2vec/	300	Google News
Conceptnet_numberbatch	https://github.com/commonsense/conceptnet-numberbatch	300	

To evaluate the performance of the models, we used two metrics:

- **Word similarity** method is based on the distances between words in an embedding space and the original words. The higher the metrics is, the better performance the method has.
- **Word analogy** method is based on the arithmetic operations in a word vector space. Same as similarity evaluation, we are looking for to find a method that has the highest analogy score.

We used given 12 dataset to do the experiment, 8 for similarity task and 4 for analogy task, The results we got are listed below and the bolded values are the highest score in each evaluation metric:

	GloVe	CBOW	Skip-grams	PDC	HDC	SG_GoogleNews	LexVec	Conceptnet_numberbatch
MTurk	0.6332	0.6307	0.6381	0.6723	0.6577	0.6815	0.7116	**0.7197**
MEN	0.7375	0.6911	0.7110	0.7726	0.7603	0.7585	0.8092	**0.8596**
WS353	0.5433	0.6082	0.6210	0.7335	0.7169	0.7000	0.6928	**0.7546**
Rubenstein_and_Goodenough	0.7695	0.7799	0.7589	0.7901	0.8058	0.7608	0.7645	**0.9099**
Rare_Words	0.3670	0.3365	0.3759	0.4724	0.4634	0.4970	0.4894	**0.5454**
SimLex999	0.3705	0.3912	0.3877	0.4269	0.4068	0.4420	0.4193	**0.6505**
TR9856	0.0967	0.1289	0.1541	**0.2073**	0.2071	0.1803	0.1209	0.1328
Google_analogy	0.7174	0.5505	0.6471	**0.7476**	0.7313	0.4018	0.7104	0.3812
MSR	0.6143	0.4928	0.5514	0.5964	0.5644	**0.7119**	0.6011	0.5394
MSR_WordRep	0.2339	0.1557	0.1938	**0.2486**	0.2469	0.1930	0.2320	0.1529

SemEval 2012_2	0.1640	0.1753	0.1986	0.1741	0.1845	0.2041	0.1680	**0.2381**
avg	**0.4770**	**0.4492**	**0.4762**	**0.5311**	**0.5223**	**0.5028**	**0.5199**	**0.5349**

On average, Conceptnet_numberbatch word embedding method achieves the highest score in both word similarity and word analogy.

	GloVe	CBOW	Skip-grams	PDC	HDC	SG_GoogleNews	LexVec	Conceptnet_numberbatch
MTurk	7	8	6	4	5	3	2	1
MEN	6	8	7	3	4	5	2	1
WS353	8	7	6	2	3	4	5	1
Rubenstein_and_Goodenough	5	4	8	3	2	7	6	1
Rare_Words	7	8	6	4	5	2	3	1
SimLex999	8	6	7	3	5	2	4	1
TR9856	8	6	4	1	2	3	7	5
Google_analogy	3	6	5	1	2	7	4	8
MSR	2	8	6	4	5	1	3	7
MSR_WordRep	3	7	5	1	2	6	4	8
SemEval 2012_2	8	5	3	6	4	2	7	1
avg	**5.91**	**6.64**	**5.73**	**2.91**	**3.55**	**3.82**	**4.27**	**3.18**

In order to find the best solution based on testing results and understand whether these 8 different word embedding technologies have statistically significant difference, we decide to use Friedman test, which is designed for testing k algorithms against n datasets. Calculated from our testing results, $\bar{R} = \frac{k+1}{2} = 4.5$, $n\sum_j (R_j - \bar{R})^2 = 151.46$, $\frac{1}{n(k-1)}\sum_{ij}(R_{ij} - \bar{R})^2 = 6$, the Friedman statistic is 25.24. The critical value for k = 8 and n = 11 at the $\alpha = 0.05$ level is around 14.0, so we reject the null hypothesis that all algorithms perform equally, which means that the average ranks as a whole, at $\alpha = 0.05$ level, shows a significant difference. So, it is necessary to find the best word embedding method in practice.

Conceptnet_numberbatch model learns from unstructured text with skip-gram embedddings model, and uses ConceptNet 5 as its knowledge graph. After comparing the results from distributional semantics word embeddings with relational knowledge ones, we found that Conceptnet_numberbatch which has adopted the combination of both performed better than any of them alone.

One thing should be mentioned is all scores in MSR_WordRep word analogy metric are all low probably because we reduced the size of maximum pairs. For this reason, maybe this kind of evaluation metric had not use sufficient data the evaluate different word embedding methods, but we had to reduce the size of it because running MSR_WordRep with large maximum pairs is extremely slow even we use the powerful computer in the lab.

Appendix: Some understandings about many word embedding models:
From web sources:

Two main algorithms in word2vector: continuous skip-gram and continuous bag-of-words. Both algorithms learn the representation of a word that is useful for prediction of other words in the sentence. Skip-gram is better for infrequent words

Glove is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space.

GloVe is essentially a log-bilinear model with a weighted least-squares objective. The main intuition underlying the model is the simple observation that ratios of word-word co-occurrence probabilities have the potential for encoding some form of meaning. For example, consider the co-occurrence probabilities for target words ice and steam with various probe words from the vocabulary. Here are some actual probabilities from a 6 billion word corpus:

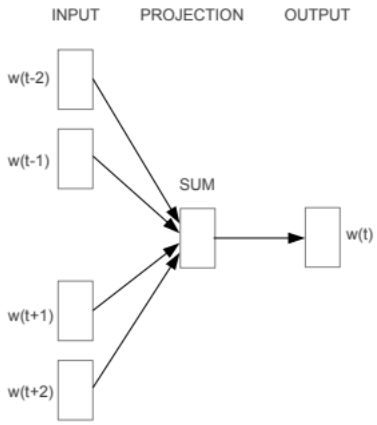
Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

As one might expect, ice co-occurs more frequently with solid than it does with gas, whereas steam co-occurs more frequently with gas than it does with solid. Both words co-occur with their shared property water frequently, and both co-occur with the unrelated word fashion infrequently. Only in the ratio of probabilities does noise from non-discriminative words like water and fashion cancel out, so that large values (much greater than 1) correlate well with properties specific to ice, and small values

(much less than 1) correlate well with properties specific of steam. In this way, the ratio of probabilities encodes some crude form of meaning associated with the abstract concept of thermodynamic phase.

The training objective of GloVe is to learn word vectors such that their dot product equals the logarithm of the words' probability of co-occurrence. Owing to the fact that the logarithm of a ratio equals the difference of logarithms, this objective associates (the logarithm of) ratios of co-occurrence probabilities with vector differences in the word vector space. Because these ratios can encode some form of meaning, this information gets encoded as vector differences as well. For this reason, the resulting word vectors perform very well on word analogy tasks, such as those examined in the word2vec package.

The CBOW model architecture tries to predict the current target word (the center word) based on the source context words (surrounding words). Considering a simple sentence, “the quick brown fox jumps over the lazy dog”, this can be pairs of (context_window, target_word) where if we consider a context window of size 2, we have examples like ([quick, fox], brown), ([the, brown], quick), ([the, dog], lazy) and so on. Thus the model tries to predict the target_word based on the context_window words.

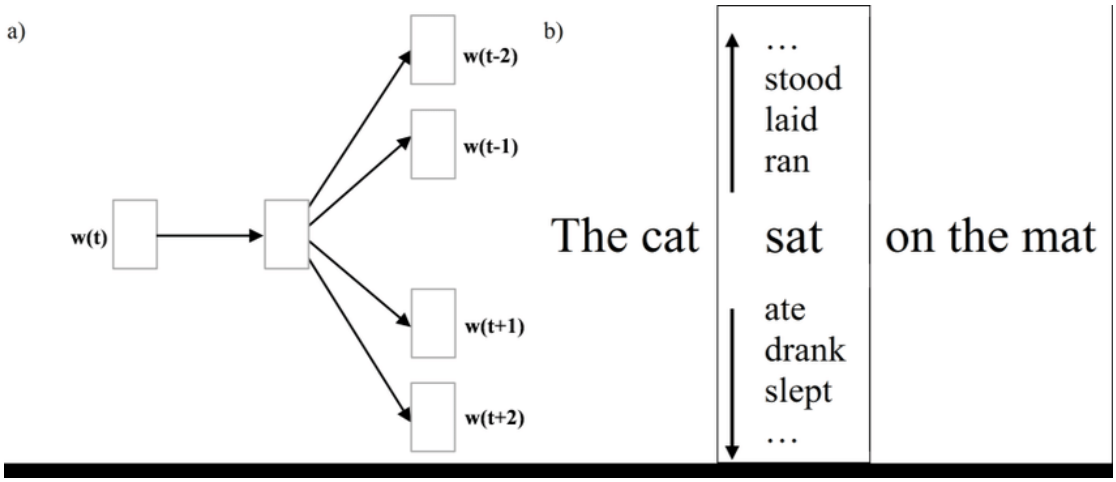


CBOW

model this CBOW architecture now as a deep learning classification model such that we take in the context words as our input, X and try to predict the target word, Y.

Skip-gram is one of the unsupervised learning techniques used to find the most related words for a given word.

Skip-gram is used to predict the context word for a given target word. It’s reverse of CBOW algorithm. Here, target word is input while context words are output. As there is more than one context word to be predicted which makes this problem difficult.



References:
[1] T. Mikolov, K. Chen, G. Corrado, and J. Dean, ‘Efficient estimation of word representations in vector space’, *ArXiv Prepr. ArXiv13013781*, 2013.
[2] F. Sun, J. Guo, Y. Lan, J. Xu, and X. Cheng, ‘Learning word representations by jointly modeling syntagmatic and paradigmatic relations’, in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 2015, pp. 136–145.
[3] J. Pennington, R. Socher, and C. D. Manning, ‘Glove: Global vectors for word representation’, in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.
[4] A. Salle, M. Idiart, and A. Villavicencio, ‘Matrix factorization using window sampling and negative sampling for improved word representations’, *ArXiv Prepr. ArXiv160600819*, 2016.
[5] R. Speer, J. Chin, and C. Havasi, ‘Conceptnet 5.5: An open multilingual graph of general knowledge’, in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.