Unsupervised Word Sense Disambiguation on top of word2vec

Sagar B Dollin, 2020HCS7012,
MSc. Cognitive Science, IIT Delhi

Motivation

Like unisense vectors, sense-specific vectors should be closely aligned to words in that sense. (Chen et al., 2014; Huang et al., 2012; Le and Mikolov, 2014; Neelakantan et al., 2015)

Main reference:

title={A Simple Approach to Learn Polysemous Word Embeddings}, author={Yifan Sun and Nikhil Rao and Weicong Ding}

Link: https://arxiv.org/abs/1707.01793v1

Approach

- This model is built on top of Gensim word2vec
- The vector embeddings of word2vec are uni-sense, we will call them as base embeddings
- Our aim is to construct contextual embeddings, which can be constructed using the base embeddings

How to construct Contextual Embeddings?

Example:

1) I was sitting by the side of the River **bank**.
2) I need to withdraw my savings from the **bank**.

Notice, how the context changes the meaning of the word in the above examples.

Contextual Embeddings

bank1 =W[sitting,bank]*base_vector(sitting) + W[side,bank]*base_vector(side) + W[River,bank]*base_vector(river)

and,

bank2 = W[need,bank]*base_vector(need) + W[withdraw,bank]*base_vector(withdraw) + W[savings,bank]*base_vector(savings)

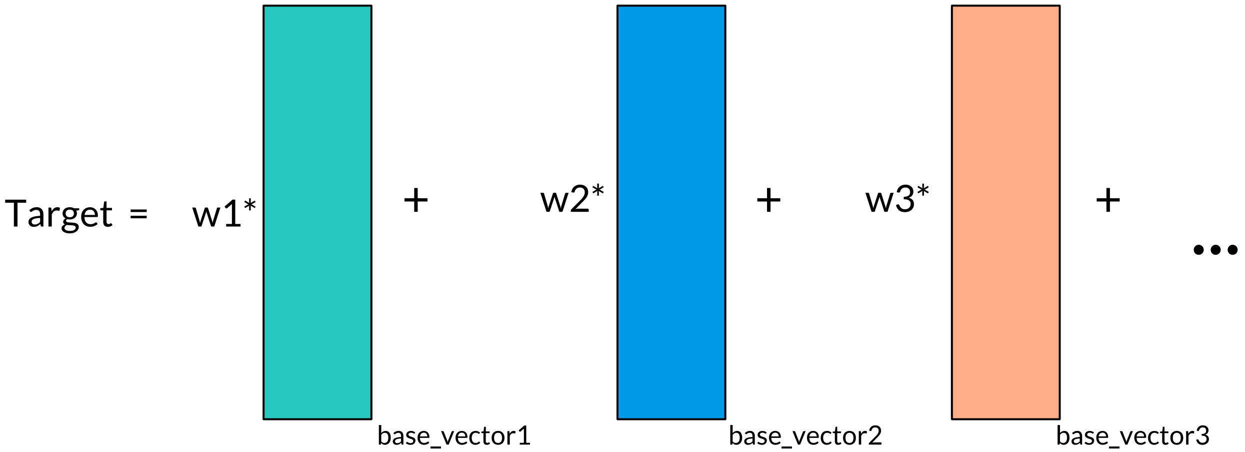
Here,

W[contex_word,target_word] defines the relevance of that context word.

W = 1 , means high relevance

W=0, means no relevance

Visualizing contextual embedding



But What is W?

- A Matrix W of V*V dimensions where V is the total vocabulary of the selected corpus. In the paper above, W[i,j] is defined as,

W[i,j]=co-occurence of words i and j /(freq._of_word_i * freq_of_word_j)

- Equations apart, W is a relevance matrix that tells how relevant a context is for a target word.
- In example 1, **river** is more **relevant** to **bank** than other context words.

So to sum up!

- contextual embedding of a word as a linear combination of its contexts.(Yifan Sun et al.)

Project details

- Python programming language
- Base model : Gensim word2vec
- Trained on wiki-corpus
- Evaluation : SCWS(Stanford's Contextual Word Similarities)
- Source(dataset/test set) : Richard Socher - Improving Word Representations Via Global Context And Multiple Word Prototypes.
- My Project available on : https://github.com/SagarDollin/unsupervised_WSD_on_top_of_word2vec

Evaluation and results

- The model was evaluated on **SCWS** test set.
- This is a test set that has two (word,context) pairs and have a similarity measure between two words annotated by human readers, based on the context.
- **Result: The predicted similarity scores (cosine similarity) and ground-truth ranking have an Average Difference = 0.27 (Inaccuracy measure)**
- **Accuracy measure = 1 – 0.27 = 0.73 (approx)**

Compare the evaluation results with original paper (Yifan Sun et al.)

- Note that, here in the image, **our method** refers to the method used by Yifan sun et al
- In the image SCWS scores are in Spearman correlation metric. I used accuracy measure : (1-(avg_diff))
- If we want to compare Acuuracies, we can compare the accuracy I got for SCWS and the Accuracy the original paper's authors got for CWS.
- My accuracy is 0.73 i.e., 73%(SCWS) where as their accuracy is (89-90%) for CWS

Embeddings	dim.	WCR						CWS		SCWS	WSC	
		R1		R2		R3					C1	C2
		Sp.	P@1	Sp.	P@1	Sp.	P@1	AUC	AP	Sp.	Acc	Acc
(Huang et al., 2012)												
Euc. Dist.	50	0.08	0.13	0.24	0.31	0.37	0.45	0.73	0.51	0.35	0.72	0.60
Max Diff.	50	0.07	0.13	0.18	0.25	0.29	0.38	0.73	0.52	0.32	0.67	0.60
Min Diff.	50	0.01	0.09	0.02	0.10	0.01	0.17	0.71	0.53	0.27	0.61	0.60
Intersect dist.	50	0.02	0.36	0.10	0.46	0.07	0.46	0.69	0.47	0.35	0.62	0.60
Angle (cos-sim)	50	0.19	0.29	0.24	0.33	0.34	0.44	0.73	0.51	0.39	0.72	0.60
City block dist.	50	0.08	0.13	0.22	0.30	0.35	0.43	0.73	0.51	0.36	0.68	0.60
Hamming dist.	50	0.15	0.27	0.19	0.31	0.27	0.43	0.72	0.51	0.37	0.68	0.60
Chi Sq.	50	0.10	0.17	0.14	0.20	0.52	0.19	0.72	0.52	0.32	0.67	0.60
(Neelakantan et al., 2015)												
3s 30kmv	50	0.20	0.27	0.25	0.34	0.39	0.49	0.72	0.47	0.53	0.70	0.62
3s 0mv	300	0.22	0.30	0.27	0.38	0.41	0.54	0.66	0.44	0.59	0.70	0.62
3s 30kmv	300	0.20	0.29	0.27	0.39	0.42	0.53	0.69	0.45	0.58	0.70	0.63
10s 1.32cλ 0mv	50	0.21	0.29	0.25	0.35	0.43	0.55	0.71	0.47	0.53	0.71	0.63
10s 1.32cλ 30kmv	50	0.20	0.30	0.24	0.30	0.42	0.52	0.72	0.48	0.51	0.69	0.63
10s 1.32cλ 0mv	300	0.22	0.32	0.27	0.37	0.45	0.58	0.66	0.44	**0.60**	0.69	0.63
(Chen et al., 2014)	200	**0.44**	**0.73**	**0.46**	**0.86**	**0.63**	**0.95**	**0.96**	**0.91**	0.48	0.75	0.66
Our method												
uni=GloVe	100	**0.34**	**0.54**	**0.33**	0.51	**0.46**	**0.61**	0.89	**0.82**	0.57	**0.83**	**0.77**
uni=w2v	50	**0.33**	**0.51**	**0.33**	**0.52**	**0.46**	**0.61**	**0.89**	0.82	**0.61**	**0.81**	**0.76**
uni=w2v	100	0.33	0.51	0.33	**0.52**	0.46	0.61	**0.90**	**0.83**	**0.62**	**0.80**	**0.77**

Image source : [A Simple Approach to Learn Polysemous Word Embeddings]