4.2 Word embedding

In this section, we experiment different hyperparameters of count-based and prediction-based word embeddings. We also compare our word embeddings to other pre-trained ones.

Evaluation

There are two methods to evaluate the quality of word embedding which are intrinsic and extrinsic evaluation. Intrinsic evaluation measures the semantic relatedness of the word embeddings by intermediate subtasks, such as analogy, categorization and relatedness. It's a simple and quick way to compute the score as the performance. And extrinsic evaluation uses word embedding as features to downstream NLP tasks, such as sentiment analysis, machine translation and Named Entity Recognition (NER) and observe the performance of tasks. Different tasks may favor different embeddings, the embeddings have a great result in the task may not have the same result in other tasks. It's a time-consuming evaluation, because we want to assess the relatedness between concepts over a large corpus, we choose similarity/relatedness tasks of intrinsic evaluation as our evaluation task.

We adopt Spearman's rank correlation coefficient in (4.1) as our evaluation function. It calculates the rank difference of two sequences in descending order. Two sequences are highly correlated if the value is close to 1, and less correlated if it is close to -1.

$$r_s = 1 - \frac{6\Sigma_{i=1}^{n} d_i^2}{n(n^2-1)}, -1 \leq r_s \leq 1 \qquad \begin{array}{l} d : \text{difference of two ranks} \\ n : \text{number of word pairs} \end{array} \tag{4.1}$$

We then compare scores with gold standards (human judgement) which are PKU-500 [96], SimLex-999 [97], WordSim-353 similarity (WS-353_sim) [98], Bruni MEN-3000 (MEN-3k) [99], Radinsky MTurk (MTurk-287) [100], WordSim-353 relatedness (WS-353_rel) [98], WordSim-240 (WS-240) [101] and WordSim-353 (WS-353) [102] is the combination of WS-353_sim and WS-353_rel. The Chinese

version of these datasets except PKU-500 and WS-240 are translated by [103]. We divide these datasets to similarity and relatedness datasets after examining the concept pairs.

	Similarity			Relatedness				Combined
Dataset	PKU	SimLex	WS353_sim	MEN	MTurk	WS353_rel	WS240	WS353
Size	494	999	203	2985	287	252	240	353

Table 4.5: Chinese gold standards of concepts similarity and relatedness.

Hyperparameters Comparison

The hyperparameters of count-based word embedding are shown in table 4.6a, and prediction-based hyperparameters are in table 4.6b. We experiment with varying hyperparameters and compare their performance in terms of Spearman correlation. Because the word embedding is used to calculate the relatedness between concepts in our research, the experimental results shown in figures are almost relatedness datasets. The hyperparameters used in each experiment are shown in the lower left of the figure.

As we described in the previous chapter, discovering new words by HMM-based model may find lots of meaningless words. We exclude words with frequency less than 80 in HMM-based model, and less than 30 in non-HMM based model. Despite the low frequency words are excluded, the HMM-based model still performs worse than the non-HMM based one as we expected in Figure 4.3. Therefore, we use non-HMM based model to segment words in both count-based and prediction-based model.

Count-based

Hyperparameter	Experimental values
Frequency weighting	Raw frequency, PMI variants
Window size	1~8
Dimensions	100~1200
Remove first k dimensions	k
Weighting exponent p	-1.5~1.5
Discover new words	yes, no

PMI variants: PMI, PPMI, NPMI, PMI^2, SPPMI

(a) Count-based.

Hyperparameter	Experimental values
Window size	1~8
Dimensions	100~1500
Model	Skip_gram, CBOW
Learning rate (LR)	0.025, 0.05
Sampling rate (SR)	0.1~0.00001
Negative samples (NS)	2, 5, 10
Discover new words	yes, no

(b) Prediction-based.

Table 4.6: Model hyperparameters.

Linear distance weighting The result of linear distance weighting in a fixed context window size is a little bit better than the raw frequency. The average Spearman scores in 8 datasets are 0.283 and 0.279 respectively in count-based model.

Frequency weighting The Figure 4.4 shows that co-occurrence matrix with raw frequency has poor performance than the other weighted methods. Among all the PMI variants, SPPMI performs the best. In the rest of the experiments, frequency weighting is using SPPMI.

Window size Small context window size is better than large one in similarity and relatedness datasets (Figure 4.5). It captures more associations between the target word and the context words, A large window size captures more topics

and domain information, as a result, it performs well in analogy tasks.

Dimensions The result of dimensions are in Figure 4.6. The performance becomes slowly increased when the dimensions is bigger than 500. In our count-based experiments, the performance stops increasing when dimensions $>$ 1000. This result is the same with [104]. They also found that the performance decreases when dimension from 500 to 1000 in count-based model.

Remove the first k dimensions In Figure 4.7, the performance of original SVD matrix is lower than removing the first 100 dimensions and the same as removing the first 150 dimensions (not in the figure). The best k is in range 5~10, starts decreasing after dimensions 10. In the similarity tasks, the best k is in the range 20~30 which is different from the relatedness tasks.

Weighting exponent The weighting exponent p in original SVD is 1. From Figure 4.8, the performance of p $>$ 1 is worse than the others. This result is found to be in line with the removing first k dimensions. They both reduce the impact of large singular values (which may contain noise) by removing or reducing them.

Although the best hyperparameters depends on different corpora, their importance can still be divided into three groups as, [Frequency weighting, remove first k dimensions, weighting exponent p], [dimensions, window size, HMM], [linear distance weighting] The best setting in the first group can increase the performance over 10 %. The second group is 3 % ~10 %, while the last group is less than 1 %.

Figure 4.3: Discover new words with or without HMM in count-based model.

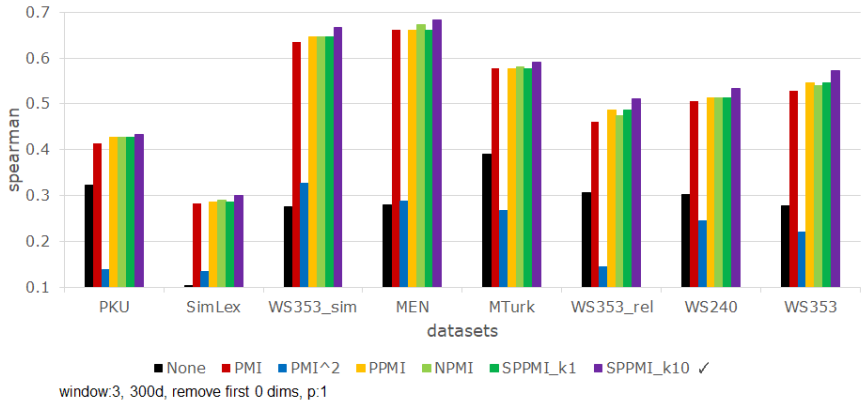

Figure 4.4: Different frequency weighting methods.

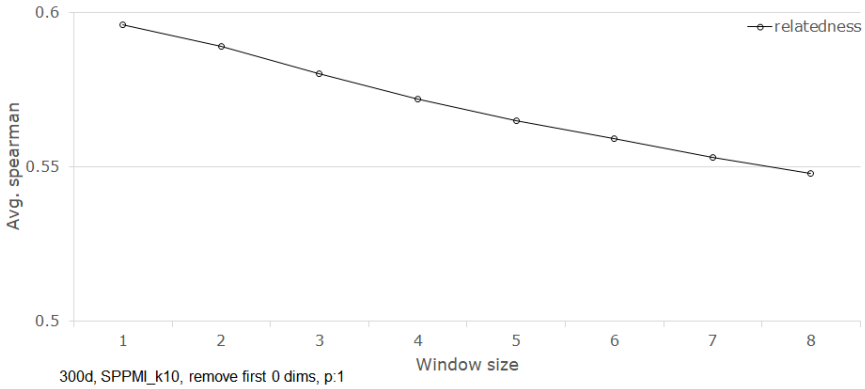

Figure 4.5: Window size.

The experimental result of prediction-based model are shown as figures in appendix C. The context window size and number of dimensions doesn't seem to affect the performance. Different negative samples have similar performance. However, the more negative samples in each training iteration, the more time it spends (appendix Figure 1h). In order to save training time, small window size,

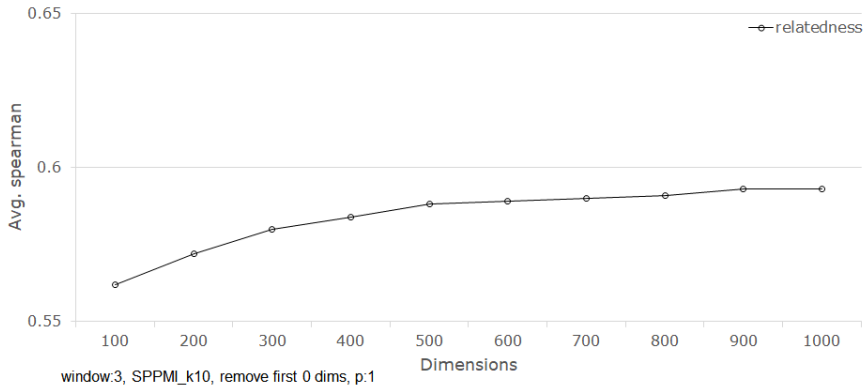

Figure 4.6: Dimensions.

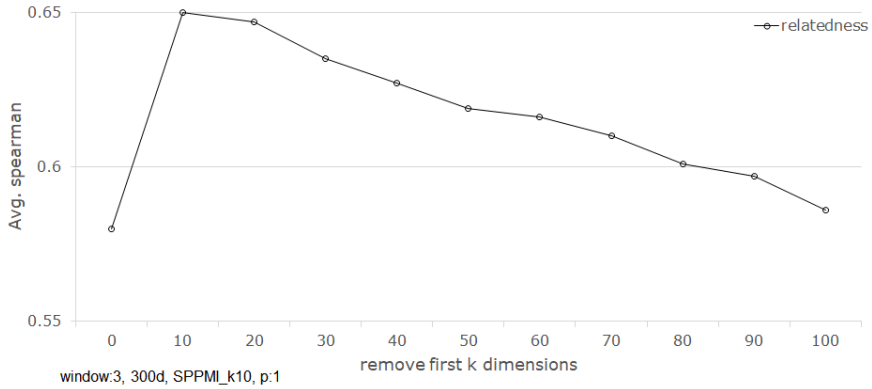

Figure 4.7: Remove first k dimensions.

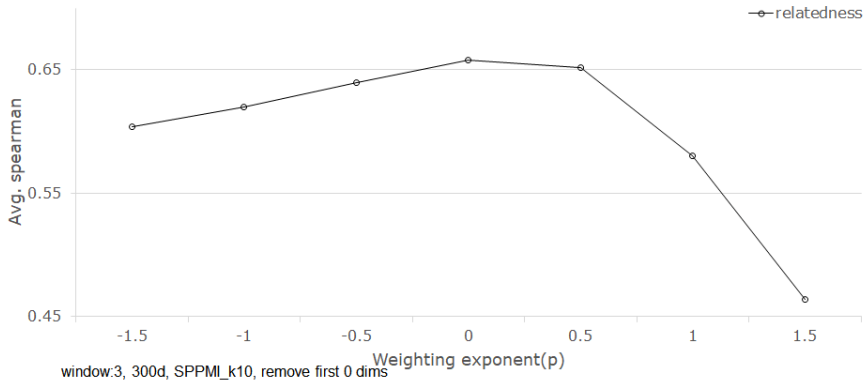

Figure 4.8: Weighting exponent.

dimensions and negative samples are recommended. Skip-gram model performs better than CBOW in the relatedness tasks, but worse in the similarity tasks.

The best settings of hyperparameters and performance are in Figure 4.9 and table 4.7, the best settings in similarity task are in appendix table A.6.

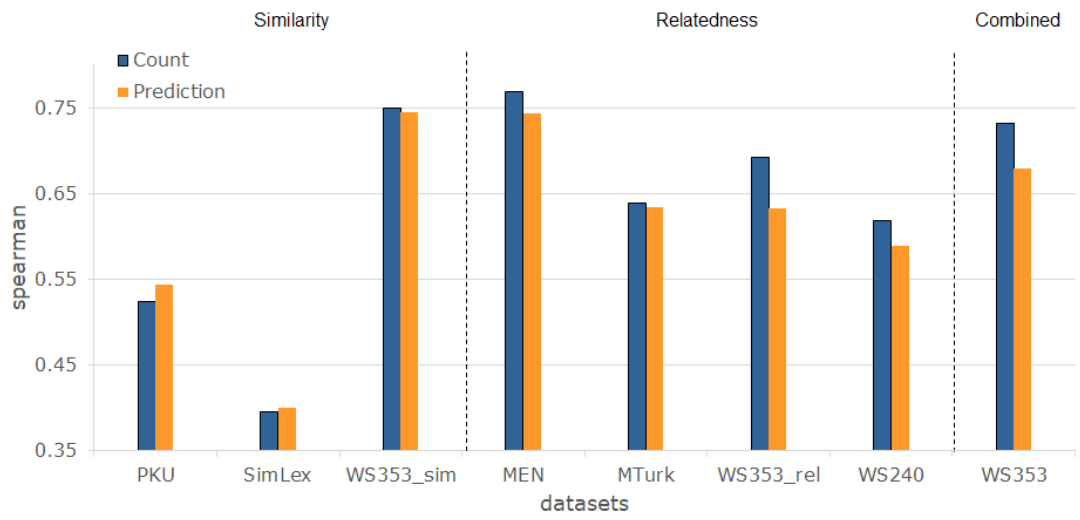

Figure 4.9: Best settings of hyperparameters in count-based and prediction-based model.

Hyperparameter	best settings
Frequency weighting	SPPMI_k10
Window size	3
Dimensions	700
Remove first k dimensions	6
Weighting exponent p	0.5
Discover new words	no

(a) Count-based.

Hyperparameter	best settings
Window size	2
Dimensions	500
Model	skip_gram
Learning rate (LR)	0.05
Sampling rate (SR)	0.00001
Negative samples (NS)	2
Discover new words	no

(b) Prediction-based.

Table 4.7: Best hyperparameter settings in relatedness task.

Comparison with Other Pre-trained Word Embedding Methods

The information of our and other pre-trained word embeddings are shown in table 4.8. The vocabulary size of fastText_cc is much lower than 2,000,000. It contains English, symbols, numbers and simplified Chinese. All of the pre-trained word embeddings are traditional Chinese except fastText_cc and Numberbatch. The vocabularies in Numberbatch are different from other models. They are concepts not words because they train on ConceptNet which consists of concepts.

Word embedding	Dim	Vocabulary size	Dataset	Model
Our WE model_count	700	249,571	PTT,Wiki	count-based
Our WE model_prediction	500	249,571	PTT,Wiki	skip-gram, CBOW
CKIP_Glove[3]	300	480,551	TCNAGC,ASBC	Glove
CKIP_word2vec	300	473,202	TCNAGC,ASBC	word2vec
fastText_wiki[4] [105]	300	141,676	Wiki	CBOW
fastText_cc[5] [106]	300	2,000,000	CC, Wiki	skip_gram
Numberbatch[6] [5]	300	307 441	CN	CN,word2vec,Glove

Dim : Dimension
WE : Word embedding
CKIP : Chinese Knowledge and Information Processing
TCNAGC : Taiwan's Central News Agency Gigaword Corpus
ASBC : Academia Sinica Balanced Corpus
CC : Common Crawl
CN : ConceptNet

Table 4.8: Pre-trained word embeddings information.

Table 4.9 shows comparison of our word embeddings and other pre-trained ones in relatedness tasks (the result of similarity task is on appendix table A.4). Our count-based model outperforms other models in the similarity/relatedness tasks, and our prediction-based model and Numberbatch are the second. Even if the same dimensions (300d), our model still outperforms other pre-trained ones.

From the table 4.10, the average OOV (vocabulary coverage) of our model is almost the same as fastText_cc despite ours contains less vocabularies With proper corpora pre-processing and some weighting techniques, count-based model can still outperform prediction-based models (except Numberbatch). Besides these pre-trained models, I would

to mention experiments implemented by [107]. They compared the dynamic mod-

[3] https://ckip.iis.sinica.edu.tw/project/embedding
[4] https://fasttext.cc/docs/en/crawl-vectors.html
[5] https://fasttext.cc/docs/en/pretrained-vectors.html
[6] https://github.com/commonsense/conceptnet-numberbatch

els (Elmo, GPT2, BERT) with the static models (skip-gram, CBOW, Glove, fastText) on the similarity tasks. The datasets they used are MEN, WS353_sim, WS353_rel, WS353 and SimLex which are similar to ours (their datasets language is English). They found that dynamic models are not superior to static models.

Word embedding	Dim	Relatedness				Avg.	Combined
		MEN	MTurk	WS353_rel	WS240		WS353
Our WE model_count	700	.769	.639	.692	.618	**.679**	.732
Our WE model_count	300	.757	.631	.663	.609	.665	.709
Our WE model_prediction	500	.744	.634	.633	.589	.650	.669
Our WE model_prediction	300	.744	.634	.632	.583	.648	.685
CKIP_Glove	300	.622	.549	.472	.556	.550	.516
CKIP_word2vec	300	.698	.614	.612	.578	.626	.657
fastText_wiki	300	.371	.351	.091	.277	.272	.242
fastText_cc	300	.624	.574	.546	.480	.556	.608
Numberbatch	300	.758	.703	.636	.538	.659	.700

Table 4.9: Compare to other pre-trained word embeddings on relatedness datasets.

Word embedding	Vocab size	Relatedness				Combined	Avg.(%)
		MEN	MTurk	WS353_rel	WS240	WS353	
Our model	249,571	76	46	15	7	23	**4.1**
CKIP	480,551	129	43	10	16	19	5.3
fastText_wiki	50,184	328	78	27	20	48	12.2
fastText_cc	307,441	77	37	9	7	16	**3.6**
Numberbatch	141,676	483	110	46	24	65	17.7

Table 4.10: Compare to other pre-trained word embeddings (OOV).