Sentence Embedding

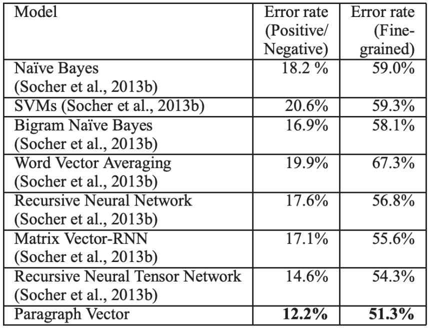
Jianguo Lu

November 4, 2024

Overview

Before Pre-trained Language Models

- Doc2Vec derived from Word2Vec[1]
- Stanford Sentiment Treebank dataset. 8544 sentences for training.



Model	Error rate (Positive/ Negative)	Error rate (Fine-grained)
Naïve Bayes (Socher et al., 2013b)	18.2 %	59.0%
SVMs (Socher et al., 2013b)	20.6%	59.3%
Bigram Naïve Bayes (Socher et al., 2013b)	16.9%	58.1%
Word Vector Averaging (Socher et al., 2013b)	19.9%	67.3%
Recursive Neural Network (Socher et al., 2013b)	17.6%	56.8%
Matrix Vector-RNN (Socher et al., 2013b)	17.1%	55.6%
Recursive Neural Tensor Network (Socher et al., 2013b)	14.6%	54.3%
Paragraph Vector	**12.2%**	**51.3%**

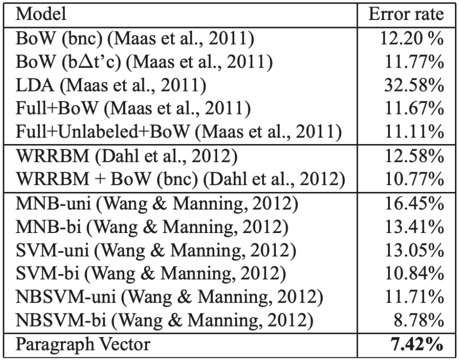
table is from[2]

[1] Andrew M Dai, Christopher Olah, and Quoc V Le. "Document embedding with paragraph vectors". In: *arXiv preprint arXiv:1507.07998* (2015).

[2] Quoc V Le and Tomas Mikolov. "Distributed Representations of Sentences and Documents.". In: *ICML*. vol. 14. 2014, pp. 1188–1196.

Before Pre-trained Language Models

- 10,000 IMDB data. Learn the word vectors and paragraph vectors using 75,000 training documents



Model	Error rate
BoW (bnc) (Maas et al., 2011)	12.20 %
BoW (bΔt'c) (Maas et al., 2011)	11.77%
LDA (Maas et al., 2011)	32.58%
Full+BoW (Maas et al., 2011)	11.67%
Full+Unlabeled+BoW (Maas et al., 2011)	11.11%
WRRBM (Dahl et al., 2012)	12.58%
WRRBM + BoW (bnc) (Dahl et al., 2012)	10.77%
MNB-uni (Wang & Manning, 2012)	16.45%
MNB-bi (Wang & Manning, 2012)	13.41%
SVM-uni (Wang & Manning, 2012)	13.05%
SVM-bi (Wang & Manning, 2012)	10.84%
NBSVM-uni (Wang & Manning, 2012)	11.71%
NBSVM-bi (Wang & Manning, 2012)	8.78%
Paragraph Vector	**7.42%**

- The baseline includes NB and bigram NB
- Trained on small data
- LLMs use entire wiki or entire web.
- Are LLMs better if training data size are similar?

doc2ve

Two versions paragraph vectors (PV)

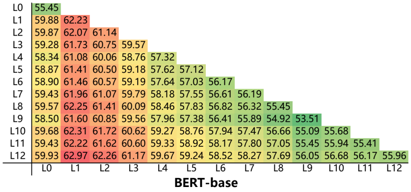
- PV-DM (Distributed Memory)
  - analogous to Word2Vec CBOW.
  - The doc-vectors are obtained by training a neural network on the synthetic task of predicting a center word based an average of both context word-vectors and the full document's doc-vector.
- PV-DBOW (Distributed Bag of Words)
  - analogous to Word2Vec SG.

```
from gensim.models.doc2vec import Doc2Vec, TaggedDocument
from nltk.tokenize import word_tokenize

data = ["I love machine learning. Its awesome.",
        "I love coding in python",
        "I love building chatbots",
        "they chat amagingly well"]

tagged_data = [TaggedDocument(words=word_tokenize(_d.lower()), tags=[str(i)]) for
     i, _d in enumerate(data)]

max_epochs = 100
vec_size = 20
alpha = 0.025

model = Doc2Vec(size=vec_size,
                alpha=alpha,
                min_alpha=0.00025,
                min_count=1,
                dm =1)

model.build_vocab(tagged_data)
```

doc2ve

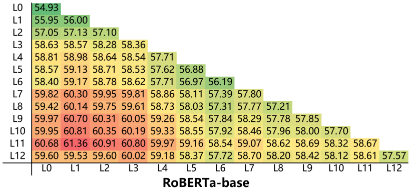
```
for epoch in range(max_epochs):
    print('iteration {0}'.format(epoch))
    model.train(tagged_data,
                total_examples=model.corpus_count,
                epochs=model.iter)
    # decrease the learning rate
    model.alpha -= 0.0002
    # fix the learning rate, no decay
    model.min_alpha = model.alpha

model.save("d2v.model")
print("Model Saved")
```

BERT and Pooling



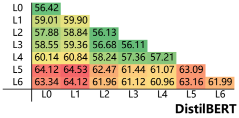
	L0	L1	L2	L3	L4	L5	L6	L7	L8	L9	L10	L11	L12
L0	55.45												
L1	59.88	62.23											
L2	59.87	62.07	61.14										
L3	59.28	61.73	60.75	59.57									
L4	58.34	61.08	60.06	58.76	57.32								
L5	58.87	61.41	60.50	59.18	57.62	57.12							
L6	58.90	61.46	60.57	59.19	57.64	57.03	56.17						
L7	59.43	61.96	61.07	59.79	58.18	57.55	56.61	56.19					
L8	59.57	62.25	61.41	60.09	58.46	57.83	56.82	56.32	55.45				
L9	58.50	61.60	60.85	59.56	57.96	57.38	56.41	55.89	54.92	53.51			
L10	59.68	62.31	61.72	60.62	59.27	58.76	57.94	57.47	56.66	55.09	55.68		
L11	59.43	62.22	61.62	60.60	59.33	58.92	58.17	57.80	57.05	55.45	55.94	55.41	
L12	59.93	62.97	62.26	61.17	59.67	59.24	58.52	58.27	57.69	56.05	56.68	56.17	55.96

BERT-base



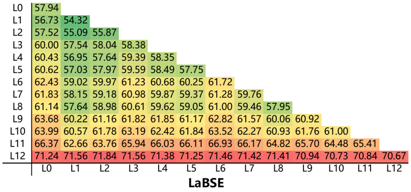
	L0	L1	L2	L3	L4	L5	L6	L7	L8	L9	L10	L11	L12
L0	54.93												
L1	55.95	56.00											
L2	57.05	57.13	57.10										
L3	58.63	58.57	58.28	58.36									
L4	58.81	58.98	58.64	58.54	57.71								
L5	58.57	59.13	58.71	58.53	57.62	56.88							
L6	58.40	59.17	58.78	58.62	57.71	56.97	56.19						
L7	59.82	60.30	59.95	59.81	58.86	58.11	57.39	57.80					
L8	59.42	60.14	59.75	59.61	58.73	58.03	57.31	57.77	57.21				
L9	59.97	60.70	60.31	60.05	59.26	58.54	57.84	58.29	57.78	57.85			
L10	59.95	60.81	60.35	60.19	59.33	58.55	57.92	58.46	57.96	58.00	57.70		
L11	60.68	61.36	60.91	60.80	59.97	59.16	58.54	59.07	58.62	58.69	58.32	58.67	
L12	59.60	59.53	59.60	60.02	59.18	58.37	57.72	58.70	58.20	58.42	58.12	58.61	57.57

RoBERTa-base



	L0	L1	L2	L3	L4	L5	L6
L0	56.42						
L1	59.01	59.90					
L2	57.88	58.84	56.13				
L3	58.55	59.36	56.68	56.11			
L4	60.14	60.84	58.24	57.36	57.21		
L5	64.12	64.53	62.47	61.44	61.07	63.09	
L6	63.34	64.12	61.96	61.12	60.96	63.16	61.99

DistilBERT



	L0	L1	L2	L3	L4	L5	L6	L7	L8	L9	L10	L11	L12
L0	57.94												
L1	56.73	54.32											
L2	57.52	55.09	55.87										
L3	60.00	57.54	58.04	58.38									
L4	60.43	56.95	57.64	59.39	58.35								
L5	60.62	57.03	57.97	59.59	58.49	57.75							
L6	62.43	59.02	59.97	61.23	60.68	60.25	61.72						
L7	61.83	58.15	59.18	60.98	59.87	59.37	61.28	59.76					
L8	61.14	57.64	58.98	60.61	59.62	59.05	61.00	59.46	57.95				
L9	63.68	60.22	61.16	61.82	61.85	61.17	62.82	61.57	60.06	60.92			
L10	63.99	60.57	61.78	63.19	62.42	61.84	63.52	62.27	60.93	61.76	61.00		
L11	66.37	62.66	63.76	65.94	66.03	66.11	66.93	66.17	64.82	65.70	64.48	65.41	
L12	71.24	71.56	71.84	71.56	71.38	71.25	71.46	71.42	71.41	70.94	70.73	70.84	70.67

LaBSE

Figure is from[3]. red color indicates better result

- There are several pooling strategies, e.g., [CLS], last layer.
- The consensus is L1+L12 for BERT
    - Question: How about LLAMA and other LLMs?
    - Is it true for classification tasks?

[3] Junjie Huang et al. "Whiteningbert: An easy unsupervised sentence embedding approach". In: *arXiv preprint arXiv:2104.01767* (2021).

Frequency bias

2D embedding from BERT-base-uncased, BERT-based-cased, ROBERTA-base

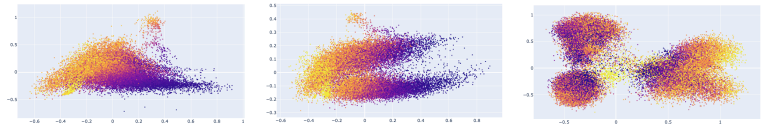
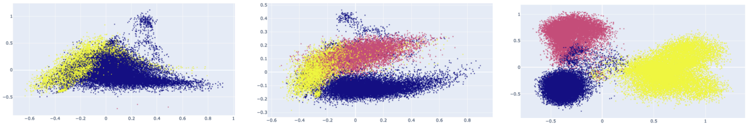


Figure is from[4]. the darker the color, the higher the token frequency.

- Observation: rare words and frequent words are hard to compare

[4] Ting Jiang et al. "Promptbert: Improving bert sentence embeddings with prompts". In: *arXiv preprint arXiv:2201.04337* (2022).

sub-word bias



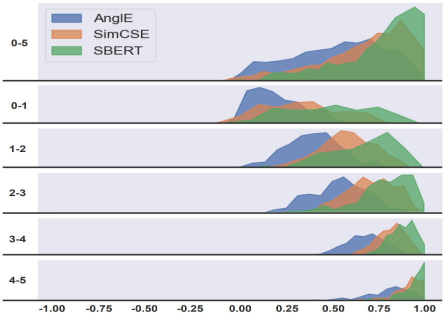
yellow represents subword and red represents the token contains capital letters.

- subwords and words with capital letters are different

Anisotropy

- the learned embeddings occupy a narrow cone in the vector space
- the average of cos similarity is large

$$Anisotropy = \frac{1}{n(n-1)} \sum_{i=1}^{n} \sum_{i \neq j} cos(x_i, x_j) \qquad (1)$$



From[5]

Density plots of cosine similarities between sentence pairs in the STS-B test set. higher ratings indicate a higher degree of similarity.

[5] Xianming Li and Jing Li. *AnglE-optimized Text Embeddings*. 2023. arXiv: 2309.12871 [cs.CL].

Whitening Matrix Transformation

- transforms a vector of random variables with a known covariance matrix into a set of new variables whose covariance is the identity matrix.
- Let $\boldsymbol{x}_i$ be a set of vector representations. The goal is to transform $\boldsymbol{x}_i$ into $\tilde{\boldsymbol{x}}_i$ so that the mean is zero and covariance is identity matrix
- Mean

$$\boldsymbol{\mu} = \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{x}_i \tag{2}$$

- Covariance matrix:

$$\boldsymbol{\Sigma} = \frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{x}_i - \boldsymbol{\mu})^\top(\boldsymbol{x}_i - \boldsymbol{\mu}) \tag{3}$$

- The transformation is done by:

$$\tilde{\boldsymbol{x}}_i = (\boldsymbol{x}_i - \boldsymbol{\mu})\boldsymbol{W} \tag{4}$$

Where $\boldsymbol{W}$ is obtained from SVD:

$$\boldsymbol{W} = \boldsymbol{U}\sqrt{\boldsymbol{\Lambda}^{-1}} \tag{5}$$

$$\boldsymbol{\Sigma} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^\top \tag{6}$$

Whitening Algorithm[6][7]

$$\tilde{\boldsymbol{x}}_i = (\boldsymbol{x}_i - \boldsymbol{\mu})\boldsymbol{W} \tag{7}$$

Algorithm 1 Whitening-k Workflow

Input: Existing embeddings $\{x_i\}_{i=1}^N$ and reserved dimensionality k

```
1: compute $\mu$ and $\Sigma$ of $\{x_i\}_{i=1}^N$
2: compute $U, \Lambda, U^T = \mathrm{SVD}(\Sigma)$
3: compute $W = (U\sqrt{\Lambda^{-1}})[:, :k]$
4: for $i = 1, 2, \cdots, N$ do
5:     $\widetilde{x}_i = (x_i - \mu)W$
6: end for
```

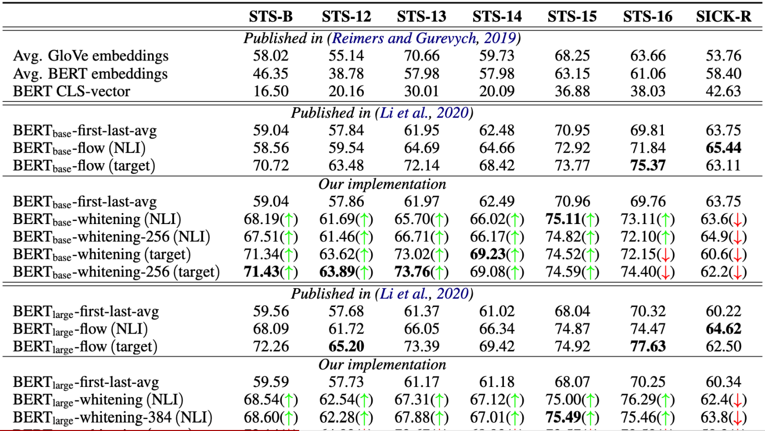
Output: Transformed embeddings $\{\widetilde{x}_i\}_{i=1}^N$

[6] Jianlin Su et al. "Whitening sentence representations for better semantics and faster retrieval". In: *arXiv preprint arXiv:2103.15316* (2021).

[7] Junjie Huang et al. "Whiteningbert: An easy unsupervised sentence embedding approach". In: *arXiv preprint arXiv:2104.01767* (2021).

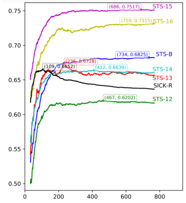
Code and result

```python
def compute_kernel_bias(vecs):
    mu = vecs.mean(axis=0, keepdims=True)
    cov = np.cov(vecs.T)
    u, s, vh = np.linalg.svd(cov)
    W = np.dot(u, np.diag(1 / np.sqrt(s)))
    return W, -mu
```

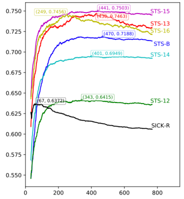


	STS-B	**STS-12**	**STS-13**	**STS-14**	**STS-15**	**STS-16**	**SICK-R**
Published in (Reimers and Gurevych, 2019)							
Avg. GloVe embeddings	58.02	55.14	70.66	59.73	68.25	63.66	53.76
Avg. BERT embeddings	46.35	38.78	57.98	57.98	63.15	61.06	58.40
BERT CLS-vector	16.50	20.16	30.01	20.09	36.88	38.03	42.63
Published in (Li et al., 2020)							
BERT$_{base}$-first-last-avg	59.04	57.84	61.95	62.48	70.95	69.81	63.75
BERT$_{base}$-flow (NLI)	58.56	59.54	64.69	64.66	72.92	71.84	**65.44**
BERT$_{base}$-flow (target)	70.72	63.48	72.14	68.42	73.77	**75.37**	63.11
Our implementation							
BERT$_{base}$-first-last-avg	59.04	57.86	61.97	62.49	70.96	69.76	63.75
BERT$_{base}$-whitening (NLI)	68.19(↑)	61.69(↑)	65.70(↑)	66.02(↑)	**75.11**(↑)	73.11(↑)	63.6(↓)
BERT$_{base}$-whitening-256 (NLI)	67.51(↑)	61.46(↑)	66.71(↑)	66.17(↑)	74.82(↑)	72.10(↑)	64.9(↓)
BERT$_{base}$-whitening (target)	71.34(↑)	63.62(↑)	73.02(↑)	**69.23**(↑)	74.52(↑)	72.15(↓)	60.6(↓)
BERT$_{base}$-whitening-256 (target)	**71.43**(↑)	**63.89**(↑)	**73.76**(↑)	69.08(↑)	74.59(↑)	74.40(↓)	62.2(↓)
Published in (Li et al., 2020)							
BERT$_{large}$-first-last-avg	59.56	57.68	61.37	61.02	68.04	70.32	60.22
BERT$_{large}$-flow (NLI)	68.09	61.72	66.05	66.34	74.87	74.47	**64.62**
BERT$_{large}$-flow (target)	72.26	**65.20**	73.39	69.42	74.92	**77.63**	62.50
Our implementation							
BERT$_{large}$-first-last-avg	59.59	57.73	61.17	61.18	68.07	70.25	60.34
BERT$_{large}$-whitening (NLI)	68.54(↑)	62.54(↑)	67.31(↑)	67.12(↑)	75.00(↑)	76.29(↑)	62.4(↓)
BERT$_{large}$-whitening-384 (NLI)	68.60(↑)	62.28(↑)	67.88(↑)	67.01(↑)	**75.49**(↑)	75.46(↑)	63.8(↓)

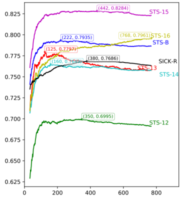
Impact of dimensions



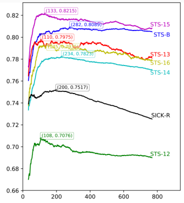
(a) BERT$_{\text{base}}$ (NLI)



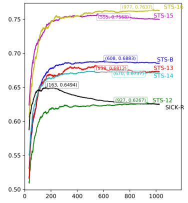
(b) BERT$_{\text{base}}$ (target)



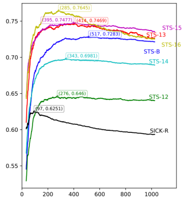
(c) SBERT$_{\text{base}}$-NLI (NLI)



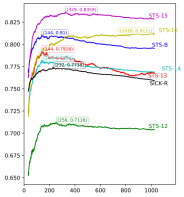
(d) SBERT$_{\text{base}}$-NLI (target)



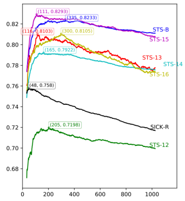
(e) BERT$_{\text{large}}$ (NLI)



(f) BERT$_{\text{large}}$ (target)



(g) SBERT$_{\text{large}}$-NLI (NLI)



(h) SBERT$_{\text{large}}$-NLI (target)

Figure from[8]

[8] Jianlin Su et al. "Whitening sentence representations for better semantics and faster retrieval". In: *arXiv preprint arXiv:2103.15316* (2021).

Hyper-parameters in whitening

normally, whitening is

$$\tilde{\boldsymbol{x}}_i = (\boldsymbol{x}_i - \boldsymbol{\mu})\boldsymbol{U}\boldsymbol{\Lambda}^{-1/2} \tag{8}$$

$$\tag{9}$$

We can add two parameters β and γ:

$$\tilde{\boldsymbol{x}}_i = (\boldsymbol{x}_i - \beta\boldsymbol{\mu})\boldsymbol{U}\boldsymbol{\Lambda}^{-\gamma/2} \tag{10}$$

Where

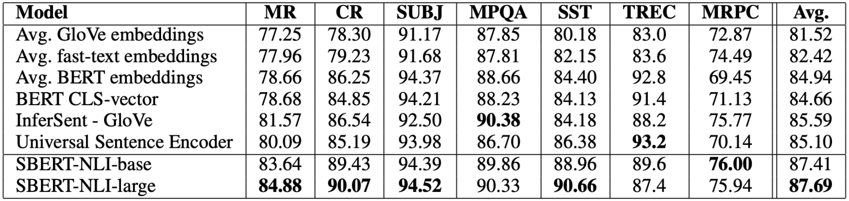
$$\boldsymbol{\mu} = \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{x}_i \tag{11}$$

$$\boldsymbol{\Sigma} = \frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{x}_i - \beta\boldsymbol{\mu})^\top(\boldsymbol{x}_i - \beta\boldsymbol{\mu}) \tag{12}$$

$$= \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^\top \tag{13}$$

Embedding for classification tasks

- Recent papers are evaluated on STS tasks
- SBERT has evaluation on classification tasks
- Observation : improvements are are not as pronouced as in STS tasks
- Whether new methods work on classification tasks?
- Whether LLM embeddings are better than NB + feature selection?



Model	**MR**	**CR**	**SUBJ**	**MPQA**	**SST**	**TREC**	**MRPC**	**Avg.**
Avg. GloVe embeddings	77.25	78.30	91.17	87.85	80.18	83.0	72.87	81.52
Avg. fast-text embeddings	77.96	79.23	91.68	87.81	82.15	83.6	74.49	82.42
Avg. BERT embeddings	78.66	86.25	94.37	88.66	84.40	92.8	69.45	84.94
BERT CLS-vector	78.68	84.85	94.21	88.23	84.13	91.4	71.13	84.66
InferSent - GloVe	81.57	86.54	92.50	**90.38**	84.18	88.2	75.77	85.59
Universal Sentence Encoder	80.09	85.19	93.98	86.70	86.38	**93.2**	70.14	85.10
SBERT-NLI-base	83.64	89.43	94.39	89.86	88.96	89.6	**76.00**	87.41
SBERT-NLI-large	**84.88**	**90.07**	**94.52**	90.33	**90.66**	87.4	75.94	**87.69**

From[9]

[9] Nils Reimers and Iryna Gurevych. "Sentence-bert: Sentence embeddings using siamese bert-networks". In: *arXiv preprint arXiv:1908.10084* (2019).

Classification data from SentEval

- MR: Sentiment prediction for movie reviews snippets on a five start scale (Pang and Lee, 2005).
- CR: Sentiment prediction of customer product reviews (Hu and Liu, 2004).
- SUBJ: Subjectivity prediction of sentences from movie reviews and plot summaries (Pang and Lee, 2004).
- MPQA: Phrase level opinion polarity classification from newswire (Wiebe et al., 2005).
- SST: Stanford Sentiment Treebank with binary labels (Socher et al., 2013).
- TREC: Fine grained question-type classification from TREC (Li and Roth, 2002).
- MRPC: Microsoft Research Paraphrase Corpus from parallel news sources (Dolan et al., 2004)

Ablation Study

- Definition: Systematically evaluate the contribution or importance of different parts of a system, typically by removing or altering them one at a time and observing the resulting effects.
- Used often in ML

4.4 Ablation Study

To gain a deeper understanding of AnglE, we conducted an ablation study examining different objectives and their effects. The results in table 4 indicate that AnglE shows improved performance with all three objectives. In particular, we observe that AnglE experiences a greater drop in performance without the angle objective than without the in-batch negative (ibn) objective. This suggests that angle optimization is more important than ibn in improving text embedding. Additionally, we find that using the angle objective alone yields performance close to that of using the cosine objective alone, demonstrating the effectiveness of angle optimization. We also evaluated five different pooling strategies and found that the "cls" strategy performed the best. Finally, we compared the ibn with/without identical sentence pair (ISP) detection and found that ibn without ISP detection has about 0.18% performance drop than with. This indicates that ibn with ISP detection is effective.

Model	Spearman's Correlation
Objective	
AnglE-BERT-all	**86.26**
- w/o ibn	86.00
- w/o angle	85.30
only cosine	85.28
only ibn	72.48
only angle	85.15
Pooling Strategy	
cls	**86.26**
cls-last-avg	85.81
last-avg	84.15
last-max	79.76
first-last-avg	81.99

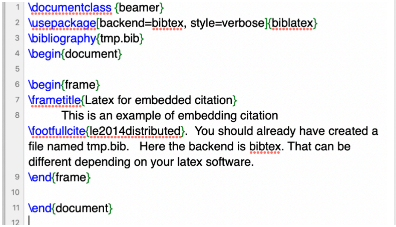
Table 4: Ablation study of AnglE. The results are Spearman's correlations on the STS-B test set.

Table 5: Results of unsupervised and LLM supervised models on the STS-B test set. For ChatGPT, LLaMA, and ChatGLM, we use the gpt-turbo-3.5, 7B LLaMA2, and 6B ChatGLM, respectively.

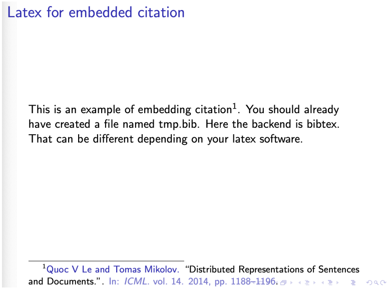
Model	Spearman's
Unsupervised Models	
SimCSE-BERT	76.85
ConSERT-BERT	73.97
DiffCSE-BERT	80.59
LLM-supervised Models	
AnglE-BERT + ChatGPT	81.52
AnglE-BERT + LLaMA	79.29
AnglE-BERT + ChatGLM	81.11
AnglE-BERT + Ensemble	**82.01**

Embedded citation

- Use `beamer` to write slides
- `\bibliography`: include bib file
- `\footfullcite`: create citation in the same slide



```
\documentclass {beamer}
\usepackage[backend=bibtex, style=verbose]{biblatex}
\bibliography{tmp.bib}
\begin{document}

\begin{frame}
\frametitle{Latex for embedded citation}
	This is an example of embedding citation \footfullcite{le2014distributed}.  You should already have created a file named tmp.bib.   Here the backend is bibtex. That can be different depending on your latex software.
\end{frame}

\end{document}

```



Latex for embedded citation

This is an example of embedding citation[1]. You should already have created a file named tmp.bib. Here the backend is bibtex. That can be different depending on your latex software.

[1] Quoc V Le and Tomas Mikolov. "Distributed Representations of Sentences and Documents.". In: *ICML*. vol. 14. 2014, pp. 1188–1196.

Sentence Embedding SOTA

Model	STS12	STS13	STS14	STS15	STS16	STS-B	SICR-R	Avg.
				Unsupervised Models				
GloVe (avg.) †	55.14	70.66	59.73	68.25	63.66	58.02	53.76	61.32
BERT-flow ‡	58.40	67.10	60.85	75.16	71.22	68.66	64.47	66.55
BERT-whitening ‡	57.83	66.90	60.90	75.08	71.31	68.24	63.73	66.28
IS-BERT ‡	56.77	69.24	61.21	75.23	70.16	69.21	64.25	66.58
CT-BERT ‡	61.63	76.80	68.47	77.50	76.48	74.31	69.19	72.05
ConSERT-BERT	64.64	78.49	69.07	79.72	75.95	73.97	67.31	72.74
DiffCSE-BERT	72.28	84.43	76.47	83.90	80.54	80.59	71.23	78.49
SimCSE-BERT	68.40	82.41	74.38	80.91	78.56	76.85	72.23	76.25
LLaMA2-7B ⋆	50.66	73.32	62.76	67.00	70.98	63.28	67.40	65.06
				Supervised Models				
InferSent-GloVe †	52.86	66.75	62.15	72.77	66.87	68.03	65.65	65.01
USE †	64.49	67.80	64.61	76.83	73.18	74.92	76.69	71.22
ConSERT-BERT	74.07	83.93	77.05	83.66	78.76	81.36	76.77	79.37
CoSENT-BERT ⋆	71.35	77.52	75.05	79.68	76.05	78.99	71.19	75.69
SBERT †	70.97	76.53	73.19	79.09	74.30	77.03	72.91	74.89
SimCSE-BERT	75.30	84.67	80.19	85.40	80.82	84.25	80.39	81.57
SimCSE-LLaMA2-7B ⋆	78.39	89.95	84.80	88.50	86.04	87.86	81.11	85.24
AnglE-BERT	75.09	85.56	80.66	86.44	82.47	85.16	81.23	82.37
AnglE-LLaMA2-7B	**79.00**	**90.56**	**85.79**	**89.43**	**87.00**	**88.97**	**80.94**	**85.96**

[10]

- report the Spearman's correlation $\rho \times 100$ (an example to remove redundancy)

[10] Xianming Li and Jing Li. *AnglE-optimized Text Embeddings*. 2023. arXiv: 2309.12871 [cs.CL].

Applications of sentence embedding

- Classification
- STS
- RAG

What is RAG (Retrieval Augmented Generation)

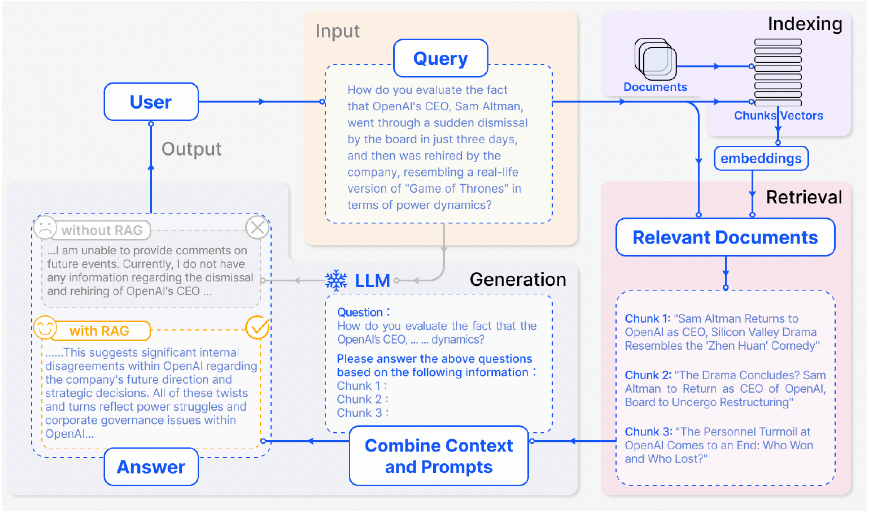

Input
Query
How do you evaluate the fact that OpenAI's CEO, Sam Altman, went through a sudden dismissal by the board in just three days, and then was rehired by the company, resembling a real-life version of "Game of Thrones" in terms of power dynamics?
Indexing
Documents
Chunks
Vectors
embeddings
User
Output
Retrieval
Relevant Documents
without RAG
...I am unable to provide comments on future events. Currently, I do not have any information regarding the dismissal and rehiring of OpenAI's CEO ...
LLM
Generation
Question :
How do you evaluate the fact that the OpenAI's CEO, ... dynamics?
Please answer the above questions based on the following information :
Chunk 1 :
Chunk 2 :
Chunk 3 :
Chunk 1: "Sam Altman Returns to OpenAI as CEO, Silicon Valley Drama Resembles the 'Zhen Huan' Comedy"
Chunk 2: "The Drama Concludes? Sam Altman to Return as CEO of OpenAI, Board to Undergo Restructuring"
Chunk 3: "The Personnel Turmoil at OpenAI Comes to an End: Who Won and Who Lost?"
with RAG
......This suggests significant internal disagreements within OpenAI regarding the company's future direction and strategic decisions. All of these twists and turns reflect power struggles and corporate governance issues within OpenAI...
Answer
Combine Context and Prompts

Fig from[11]

[11] Yunfan Gao et al. "Retrieval-augmented generation for large language models: A survey". In: *arXiv preprint arXiv:2312.10997* (2023).

RAG and Fine-tuning

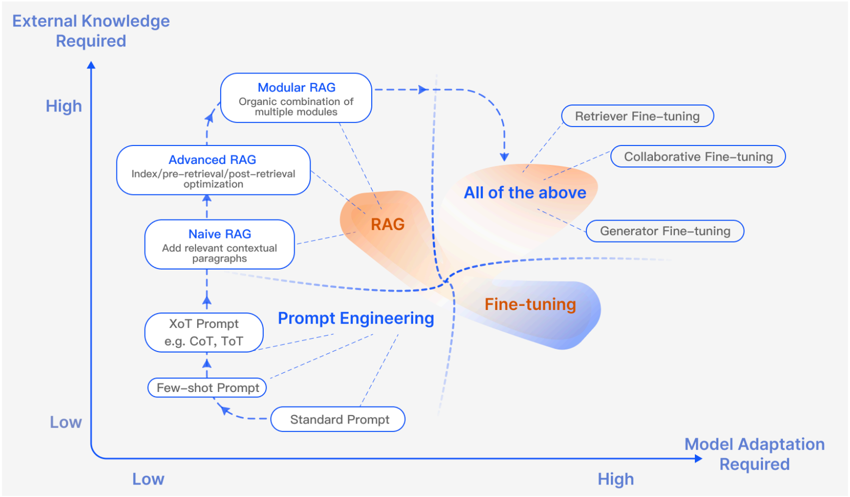

External Knowledge Required
High
Low
Modular RAG
Organic combination of multiple modules
Advanced RAG
Index/pre-retrieval/post-retrieval optimization
Naive RAG
Add relevant contextual paragraphs
XoT Prompt e.g. CoT, ToT
Few-shot Prompt
Standard Prompt
Prompt Engineering
RAG
All of the above
Fine-tuning
Retriever Fine-tuning
Collaborative Fine-tuning
Generator Fine-tuning
Model Adaptation Required
Low
High

Fig from[12]

[12] Yunfan Gao et al. "Retrieval-augmented generation for large language models: A survey". In: *arXiv preprint arXiv:2312.10997* (2023).

The Naive RAG

It includes indexing, retrieval, and generation, which is also characterized as a "Retrieve-Read" framework

- Indexing: text is segmented into smaller, digestible chunks.
  - Chunks are then encoded into vector representations
  - stored in vector database.
- Retrieval
  - encode the query
  - then computes the similarity scores between the query vector and the vector of chunks within the indexed corpus.
  - prioritizes and retrieves the top K chunks
  - used as the expanded context in prompt.
- Generation.
  - The posed query and selected documents are synthesized into a coherent prompt
  - LLM formulate a response using the new prompt.

Why Fine-Tuning for Sentence Embedding?

- **Pre-trained LLMs are Token-Centric**:
  - LLMs are pre-trained primarily on token-level tasks (e.g., masked word prediction).
  - May not capture sentence-level semantics without further training.
- **Sentence-Level Meaning Needs Contextualization**:
  - To understand entire sentences, models need to encode the relationships between words in a sentence.
  - Fine-tuning helps the model to "aggregate" context across tokens to form meaningful sentence-level representations.
- **Improves Semantic Tasks**:
  - Tasks like sentence similarity, paraphrase detection, and semantic search require embeddings that represent entire sentences' meanings.
  - Fine-tuning adjusts the model parameters to enhance sentence-level coherence and semantic similarity.
- **Aligns Model with Target Task**:
  - Adapt to specific downstream tasks (embeddings become task-relevant)
  - e.g., in sentence similarity tasks, fine-tuning on paired data helps the model distinguish between similar and dissimilar sentences.

Task dependent fine-tuning

- Each task requires a different notion of similarity.
    - `Apple launches the new iPad`
    - `NVIDIA is gearing up for the next GPU generation`
- Are those two sentences similar?
    - In STS: not similar
    - In classification tasks: classify the sentences to news categories such as Sports, Technology, Politics