Subspace representation for text classification with limited training data

Erica K. Shimomoto[1], Edison Marrese-Taylor[1], Hiroya Takamura[1], Ichiro Kobayashi[1,2], Yusuke Miyao[1,3]
National Institute of Advanced Industrial Science and Technology[1]
Ochanomizu University[2], The University of Tokyo[3]
{kidoshimomoto.e, edison.marrese, takamura.hiroya}@aist.go.jp
koba@is.ocha.ac.jp, yusuke@is.s.u-tokyo.ac.jp

Abstract

Using large pre-trained language models (PLM), such as BERT, to tackle natural language processing downstream tasks has become a common practice, where it is often necessary to fine-tune the PLM to achieve the best results. However, these PLMs require large training data to fine-tune the desired task properly. To solve this problem, we propose using subspace-based methods to explore the geometrical structure of embeddings extracted from a PLM, effectively performing text classification when limited training data is available. Our results on the IMDB and Movie Review datasets show that using the appropriate subspace-based method with pre-trained BERT embeddings can outperform fine-tuned BERT when limited training data is available. While our results are limited to sentiment analysis, they showcase a less data-hungry strategy that could be applied in other limited resource contexts.

1 Introduction

With the development of the transformer architecture [1], it has become a common practice to use pre-trained large language models (PLM), such as BERT [2] and GPT3 [3], to tackle downstream tasks in natural language processing.

While there are strategies to use these PLMs without further training, through prompting, and/or training in a parameter-efficient manner by using adapters, fine-tuning these models for the desired task by adding a simple head to the language model is still a standard strategy.

Fine-tuning language models have been shown to lead state-of-the-art results in several downstream tasks in NLP, but struggle to deliver good performance when there is limited training data. Nevertheless, fine-tuning can be tricky as results can highly vary depending on the choice of random seed for hyper-parameter initialization and training data order [4].

In light of this issue, in this paper, we offer a simple alternative to PLM fine-tuning, based on the subspace representation, and demonstrate its effectiveness when there is limited training data in the task of text classification.

Such representation comes from computer vision, based on the empirical evidence that patterns of the same entity (e.g., pictures of the same person) tend to cluster in high-dimensional space [5, 6]. Moreover, it has been shown to hold also for traditional word embeddings of texts, being extensively explored in several NLP downstream tasks, such as word compositionality [7], word polysemy [8], text summarization [9], and text classification [10].

In this setting, each text input is represented as a set of embeddings, which are modeled as low-dimensional linear subspaces in a high-dimensional embedding space. When modeled using the principal components analysis (PCA), the subspace compactly represents the distribution of the features in a set based on the directions of the highest variance. This subspace discards irrelevant information, such as noise, while effectively representing variations, making it a powerful tool when there is limited training data.

We extend these studies to showcase the capabilities of the subspace representation when limited training data is available to fine-tune a PLM. We specifically target the task of sentiment analysis, performing tests on the Movie Review and IMDB datasets. We chose these two datasets as they contain samples from the same domain but significantly differ in size. We compared the subspace-based methods' performance to a standard classifier built on top

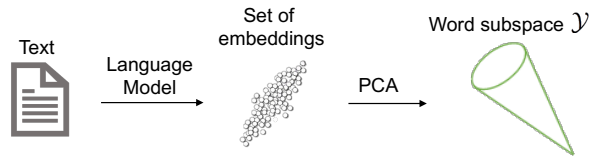

Figure 1: Process of modeling a text as a subspace. Words from the text are extracted and then translated to word vectors by using a word embedding model. Then, the set of word vectors is modeled as a word subspace by using PCA

of BERT, and our results show that the subspace-based methods using pre-trained BERT embeddings can outperform fine-tuned BERT in a low training data regime.

2 Subspace theory overview

In this section, we lay down the foundations of subspace representation and how it is applied to textual data. Then, we discuss the main subspace-based methods used to tackle the text classification task.

2.1 Subspace representation for texts

In this formulation, each text is represented by a set of n embedding vectors organized as the columns in an embedding matrix $\boldsymbol{X} \in \mathbb{R}^{p \times n}$, where p is the size of the embedding vectors. This set of vectors is then modeled as a set of basis vectors spanning a linear subspace. To obtain a consistent subspace representation that can be applied to sets of different numbers of embedding vectors, we apply the principal components analysis (PCA) without data-centering.

This representation retains most of the variability of the original embeddings in the set [11, 12], and, consequently, can effectively and compactly represent the context of the corresponding text. Each direction given by the basis vectors of this subspace represents the directions with the highest variance of the text embeddings and can be regarded as the main semantic meanings [9].

Figure 1 shows our approach to model a subspace from a text. To model the word subspace for this set of embeddings, we first need to compute the following autocorrelation matrix, $\boldsymbol{R} = \boldsymbol{X}\boldsymbol{X}^\top$. The orthonormal basis vectors of the m-dimensional subspace $\mathcal{Y}$ are obtained as the eigenvectors with the m largest eigenvalues $\{\lambda_l\}_{l=1}^{m}$ of the matrix $\boldsymbol{R}$. The subspace $\mathcal{Y}$ is then represented by the matrix $\boldsymbol{Y} \in \mathbb{R}^{p \times m}$, which has the corresponding orthonormal basis vectors as its column vectors.

For details on choosing the subspace dimension m, we refer the readers to the Appendix A. In addition, we will refer to the subspaces by their basis vectors' matrices for simplicity.

2.2 Subspace-based methods for text classification

This section discusses the main subspace-based methods used for text classification, specifically for sentiment analysis. We formulate our problem as a single-label classification problem. Given a set of training texts and their respective labels denoted as $\{(\boldsymbol{X}_i, l_i)\}_{i=1}^{d}$, where $\boldsymbol{X}_i \in \mathbb{R}^{p \times n_i}$ are the text's embeddings, $l_i \in (1, 2)$ is its class label, and d is the number of texts in the dataset; we wish to classify an input text $\boldsymbol{X}_{in}$ into one of the classes. In the case of sentiment analysis, we consider only two different classes, i.e., positive and negative sentiment.

2.2.1 MSM

The application of mutual subspace method (MSM) [13] to text classification comes from the observation that a subspace spanned by important words representing a class, i.e., class subspace, can also be derived from a single text or a combination of all texts in the class. Furthermore, a subspace modeled from a text, i.e., text subspace, that also belongs to this class should be almost the same as the class subspace [10].

Given an input text subspace $\boldsymbol{Y}_{in} \in \mathbb{R}^{p \times m_{in}}$ and the c-th class subspace $\boldsymbol{Y}_c \in \mathbb{R}^{p \times m_i}$, we measure their similarity based on their canonical angles [14], computed using the singular value decomposition (SVD) [15].

We first calculate the SVD, $\boldsymbol{Y}_{in}^\top \boldsymbol{Y}_c = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$, where $\boldsymbol{\Sigma} = \text{diag}(\kappa_1, \ldots, \kappa_{m_{in}})$, $\{\kappa_j\}_{j=1}^{m_{in}}$ represents the set of singular values, and $(\kappa_1 \geq \ldots \geq \kappa_{m_{in}})$. The similarity can then be calculated as $S^{in,c}(t) = \frac{1}{t}\sum_{j=1}^{t} \kappa_j^2$, where $1 \leq t \leq m_{in}$, and is equivalent to taking the average of the squared cosine of t canonical angles. Finally, the input text is assigned to the class with the highest similarity.

2.2.2 GSM

The Grassmann Subspace Method (GSM) increases the abstraction level by representing each text subspace $\boldsymbol{Y}$ as a point $\boldsymbol{y}$ on the Grassmann manifold, where each sentiment class can then be modeled by the distribution of the subspaces it contains. Classification is performed by comparing a set of reference points on the Grassmann manifold,

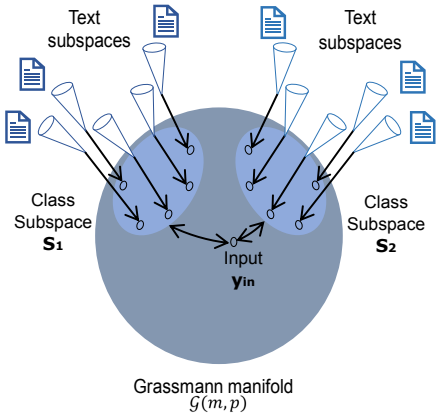

Figure 2: Subspace representation on a Grassmann manifold

modeled as a subspace $\boldsymbol{S}_c$, with an input point $\boldsymbol{y}_{in}$.

The Grassmann manifold $\mathcal{G}(m,p)$ is defined as a set of m-dimensional linear subspaces of $\mathbb{R}^p$ [16]. This manifold can be embedded in a reproducing kernel Hilbert space by using a Grassmann kernel [17]. In this work, we use the projection kernel, $k_p(\boldsymbol{Y}_1,\boldsymbol{Y}_2) = \frac{1}{m}\sum_{j=1}^{m}\cos^2\theta_j$, which is homologous to the subspace similarity.

Then, a text subspace $\boldsymbol{Y}$ can be represented as a point on the Grassmann manifold with regards to a reference subspace dictionary $\{\boldsymbol{Y}_q\}_{q=1}^{N}$ as:

$$\begin{aligned}\boldsymbol{y} &= k_p(\boldsymbol{Y},\boldsymbol{Y}_q)\\ &= [k_p(\boldsymbol{Y},\boldsymbol{Y}_1),k_p(\boldsymbol{Y},\boldsymbol{Y}_2),...,k_p(\boldsymbol{Y},\boldsymbol{Y}_N)] \in \mathbb{R}^N\end{aligned} \tag{1}$$

Figure 2 shows a conceptual diagram of the subspace representation on a Grassmann manifold. Consider the set of training subspaces $T = \{(\boldsymbol{Y}_i,l_i)\}_{i=1}^{d}$ corresponding to the texts in the training set. We obtain a set of points $\{(\boldsymbol{y},l_i)\}_{i=1}^{d}$ corresponding to each training text subspace by using equation 1 with respect to T. Through the kernel trick using the projection kernel, we now have a set of points on the Grassmann manifold corresponding to each sentiment class. GSM models the sentiment class subspaces $\boldsymbol{S}_c$ based on the set of points corresponding to each class.

We measure the similarity between a sentiment class subspace $\boldsymbol{S}_c$ and an input point $\boldsymbol{y}_{in}$ by $S^{in,c} = \boldsymbol{y}^\top\boldsymbol{P}_c\boldsymbol{y}$, where $\boldsymbol{P}_c = \boldsymbol{S}_c\boldsymbol{S}_c^\top$ is the projection matrix onto the subspace $\boldsymbol{S}_c$.

Although this method is not suitable for tasks with a large number of training samples, due to the kernel matrix calculation, it is very effective to generalize the local distribution of subspaces on the Grassmann manifold when there is limited training data.

2.2.3 GOSM

Word embeddings extracted from models trained based on the co-occurrence of words do not necessarily carry sentiment information, i.e., opposite sentiment words might have very close embedding representations because they often occur in the same context. Based on this observation, previous work [10] has shown that using discriminative versions of the subspace-based methods, such as the Grassmann Orthogonal Subspace Method (GOSM), to be effective in solving sentiment analysis.

GOSM applies the whitening process to the sentiment class subspaces $\boldsymbol{S}_c$ on the Grassmann manifold, maximizing the distance between them, and alleviating the lack of sentiment information, i.e., it increases the distance between opposite sentiment word embeddings.

First, we need to calculate the whitening matrix $\boldsymbol{O}$. We define their projection matrices as $\boldsymbol{P}_c = \boldsymbol{S}_c\boldsymbol{S}_c^\top$, and then calculate the total projection matrix, $\boldsymbol{G} = \sum_{c=1}^{|C|}\boldsymbol{P}_c$, where $|C|$ is the number of classes.

The whitening matrix $\boldsymbol{O}$ can be obtained based on the eigenvalues and eigenvectors of the matrix $\boldsymbol{G} \in \mathbb{R}^{v\times p}$ as $\boldsymbol{O} = \boldsymbol{\Lambda}^{-1/2}\boldsymbol{E}^\top$, where $v = |C|\times m$, $\boldsymbol{\Lambda} \in \mathbb{R}^{v\times v}$ is the diagonal matrix with the j-th highest eigenvalue of the matrix G as the j-th diagonal component, and $\boldsymbol{E} \in \mathbb{R}^{p\times v}$ is a matrix whose j-th column vector is the eigenvector of $\boldsymbol{G}$ corresponding to the j-th highest eigenvalue.

Classification under this framework is very similar to GSM; however, we compute a whitening transformation matrix during the training stage. In the classification stage, we first apply this whitening transformation to all reference subspaces $\boldsymbol{S}_c$ and input point $\boldsymbol{y}_{in}$ following:

$$\begin{aligned}\boldsymbol{S}_c^o &= \boldsymbol{O}\boldsymbol{S}_c,\\ \boldsymbol{y}_{in}^o &= \boldsymbol{O}\boldsymbol{y}_{in}.\end{aligned} \tag{2}$$

Then, classification follows as described in GSM.

3 Experiments and Results

In this section, we describe the experiments performed to assess the effectiveness of the discussed subspace-based methods in low training data regimes and compare them against a transformer-based language model using full fine-tuning. More specifically, we compare against BERT [2], using the HuggingFace library [18] implementation and the "bert-base-uncased" pre-trained model.

In our experiments, we used two different datasets:

- **Movie review (MR) dataset v2.0**[1], proposed by [19], containing 2000 reviews extracted from the IMDB website, with 1000 positive sentiment review, and 1000 negative sentiment reviews.
- **IMDB dataset**[2], proposed by [20], containing 50000 reviews also extracted from the IMDB website, where half of the reviews express positive sentiment, and half express negative sentiment.

We specifically chose these two datasets because they contain samples from the same domain but significantly differ in size. For both datasets, we used the standard train-test sets.

Classification using subspaces followed the methods explained in Section 2.2. When using BERT, we added a classification head that consists of a linear layer on top of the pooled output. We will refer to this method as "standard" as this is one of the standard ways to perform classification when using BERT[3].

While there are several ways to extract embeddings from BERT, in our experiments, we considered the following: Given a text t with n tokens[4], we extracted n embeddings. For each token, we considered the sum of the token's representation taken from the last four layers in BERT. Therefore, each text can be represented as a matrix $\boldsymbol{X} \in \mathbb{R}^{p \times n}$, where p is the size of BERT's hidden layers (i.e., 768).

We first compared their performances on the larger IMDB dataset. Due to this dataset's size, we did not test the Grassmann variations of the subspace-based methods.

Results are shown in Table 1. When using the pre-trained BERT, we can see that although the standard classifier can perform relatively well, MSM could outperform it. However, once we fine-tune BERT, the standard classifier achieves the best performance, although the difference with MSM is smaller than with pre-trained BERT. These results indicate that the subspace-based methods can be helpful when fine-tuning is not feasible.

We then assessed their performance in the smaller MR dataset. Results are shown in Table 2. We can see that without fine-tuning, the standard classifier performs poorly. On the other hand, MSM shows a performance consistent with the IMDB dataset. Furthermore, we can see results further improve with GSM and GOSM.

Table 1: Results on the IMDB dataset, comparing the MSM with standard classifier when using pre-trained BERT and fine-tuned BERT on the IMDB training set

Fine-tuning	Classifier	Acc.	Rec.	Prec.	f1
None	standard	71.71	84.61	67.25	74.94
	MSM	**77.46**	**73.25**	**80.53**	**76.72**
IMDB	standard	**92.60**	**92.23**	**92.93**	**92.58**
	MSM	92.00	90.65	93.38	91.99

Table 2: Results on the Movie Review dataset, comparing the subspace-based methods with the standard classifier when using pre-trained BERT and fine-tuned BERT on the MR and IMDB training set

Fine-tuning	Classifier	Acc.	Rec.	Prec.	f1
None	standard	55.53	83.37	53.80	65.04
	MSM	77.44	74.87	79.01	76.88
	GSM	86.24	86.24	86.39	86.31
	GOSM	**91.14**	**91.14**	**91.20**	**91.17**
MR	standard	85.64	89.58	83.63	86.27
IMDB	standard	**88.14**	**90.89**	**86.25**	**88.44**

In addition, we can observe that when we fine-tune BERT on the MR dataset, the standard classifier's performance significantly improves; However, we can see it achieves about the same performance as GSM with pre-trained BERT embeddings. Fine-tuning BERT on IMDB helps improve its performance on the MR dataset, but it is still far below the performance it achieved for IMDB.

4 Conclusions

We presented a subspace-based alternative for fine-tuning large transformer language models when there is limited training data for text classification. When modeled using PCA, the subspace representation can compactly represent the distribution of the features in a set, making it a powerful tool in low training data regimes. Our results on the IMDB and Movie Review datasets show that using the appropriate subspace-based method with pre-trained BERT embeddings can outperform fine-tuned BERT when limited training data is available. Although limited, our results show that subspace-based methods can offer an effective alternative for fine-tuning in text classification.

1) http://www.cs.cornell.edu/people/pabo/movie-review-data/
2) https://ai.stanford.edu/ amaas/data/sentiment/
3) This architecture follows the *BertForSequenceClassification* implementation of the *Huggingface* Transformer library.
4) According to BERT's tokenizer, including the CLS token.

Acknowledgments

This paper is based on results obtained from the project JPNP20006, commissioned by the New Energy and Industrial Technology Development Organization (NEDO).

References

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In **Advances in neural information processing systems**, pages 5998–6008, 2017.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In **Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)**, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. **Advances in neural information processing systems**, 33:1877–1901, 2020.

[4] Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. **arXiv preprint arXiv:2002.06305**, 2020.

[5] Satosi Watanabe and Nikhil Pakvasa. Subspace method of pattern recognition. In **Proc. 1st. IJCPR**, pages 25–32, 1973.

[6] Taizo Iijima, Hiroshi Genchi, and Ken-ichi Mori. A theory of character recognition by pattern matching method. In **Learning systems and intelligent robots**, pages 437–450. Springer, 1974.

[7] Hongyu Gong, Suma Bhat, and Pramod Viswanath. Geometry of compositionality. In **Thirty-First AAAI Conference on Artificial Intelligence**, 2017.

[8] Jiaqi Mu, Suma Pallathadka Bhat, and Pramod Viswanath. Geometry of polysemy. In **5th International Conference on Learning Representations, ICLR 2017**, 2019.

[9] Hongyu Gong, Tarek Sakakini, Suma Bhat, and JinJun Xiong. Document similarity for texts of varying lengths via hidden topics. In **Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)**, pages 2341–2351, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[10] Erica K Shimomoto, François Portet, and Kazuhiro Fukui. Text classification based on the word subspace representation. **Pattern Analysis and Applications**, 24(3):1075–1093, 2021.

[11] Jiaqi Mu, Suma Bhat, and Pramod Viswanath. Representing sentences as low-rank subspaces. In **Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)**, pages 629–634, 2017.

[12] Erica K Shimomoto, Lincon S Souza, Bernardo B Gatto, and Kazuhiro Fukui. Text classification based on word subspace with term-frequency. In **2018 International Joint Conference on Neural Networks (IJCNN)**, pages 1–8. IEEE, 2018.

[13] Kazuhiro Fukui and Atsuto Maki. Difference subspace and its generalization for subspace-based methods. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, 37(11):2164–2177, 2015.

[14] Françoise Chatelin. **Eigenvalues of Matrices: Revised Edition**. SIAM, 2012.

[15] Kazuhiro Fukui and Osamu Yamaguchi. Face recognition using multi-viewpoint patterns for robot vision. **Robotics Research, The Eleventh International Symposium, ISRR**, pages 192–201, 2005.

[16] Yasuko Chikuse. Statistics on special manifolds. **Springer, Lecture. Notes in Statistics**, 174, 2013.

[17] Jihun Hamm and Daniel D Lee. Grassmann discriminant analysis: a unifying view on subspace-based learning. In **Proceedings of the 25th international conference on Machine learning**, pages 376–383. ACM, 2008.

[18] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Transformers: State-of-the-art natural language processing. In **Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations**, pages 38–45, 2020.

[19] Bo Pang and Lillian Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In **Proceedings of the 42nd annual meeting on Association for Computational Linguistics**, page 271. Association for Computational Linguistics, 2004.

[20] Andrew Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In **Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies**, pages 142–150, 2011.

A Subspace dimension

In general, the dimension m of each subspace is empirically determined. For sentences, about four dimensions should suffice to retain most of the sentence embeddings' variance [11]. However, for texts or sets of texts, more dimensions will likely be necessary. The amount of variance retained by the basis vectors of the subspace can be determined by using the cumulative contribution rate $\mu(m)$. Considering that we want to keep a minimum of μ_{min} of the text variance, we can determine m by ensuring that $\mu(m)_d \geq \mu_{min}$, where:

$$\mu(m)_d = \frac{\sum_{l=1}^{m}(\lambda_l)}{\sum_{l=1}^{p}(\lambda_l)}. \tag{3}$$