SimpleTran: Transferring Pre-Trained Sentence Embeddings for Low Resource Text Classification

Siddhant Garg*, **Rohit Kumar Sharma***, **Yingyu Liang**
Department of Computer Sciences
University of Wisconsin Madison
{sidgarg,rsharma,yliang}@cs.wisc.edu

Abstract

Fine-tuning pre-trained sentence embedding models like BERT has become the default transfer learning approach for several NLP tasks like text classification. We propose an alternative transfer learning approach called SimpleTran which is simple and effective for low resource text classification characterized by small sized datasets. We train a simple sentence embedding model on the target dataset, combine its output embedding with that of the pre-trained model via concatenation or dimension reduction, and finally train a classifier on the combined embedding either by fixing the embedding model weights or training the classifier and the embedding models end-to-end. Keeping embeddings fixed, SimpleTran significantly improves over fine-tuning on small datasets, with better computational efficiency. With end-to-end training, SimpleTran outperforms fine-tuning on small and medium sized datasets with negligible computational overhead. We provide theoretical analysis for our method, identifying conditions under which it has advantages.

1 Introduction

There has been significant progress in pre-training sentence embedding models on large text corpora recently (Devlin et al., 2018; Peters et al., 2018; Howard and Ruder, 2018). Typically these powerful models have a large number of parameters and hence require large-scale training data.

Many real world NLP applications have small or medium-sized datasets from special domains, such as finance, political science, and medicine unlike many popularly studied applications like Machine Translation and Question Answering in the academic research community which have large datasets. A popular trend in NLP recently (Cherry et al., 2019) has been combating the issue of resource scarceness and developing algorithms for low resource settings. In this paper, we consider the task of text classification in a low resource setting which is characterised by the availability of only a small number of training examples.

Fine-tuning (FT) powerful pre-trained models like BERT has recently become the de facto standard for text classification tasks. This entails learning a classifier on the sentence embedding from a pre-trained model while fine-tuning the pre-trained model at the same time. Typically, FT significantly improves the performance on the target data, since pre-training is done on general domain datasets, while the target data stems from special domains with significant semantic differences. FT has been shown to be a simple and effective strategy for several datasets like GLUE (Wang et al., 2018), DBpedia (Lehmann et al., 2015), Sogou News (Wang et al., 2008), etc. which typically have hundreds of thousands of training points.

However, some recent works (Garg et al., 2019) show that FT in a low resource domain may be unstable having a high variance due to lack of enough data to specialize the general semantics learned by the pre-trained model to the target domain. Other works (Sun et al., 2019a; Arase and Tsujii, 2019) have tried to improve upon fine-tuning pre-trained models like BERT when the target datasets are small. Further, Houlsby et al. discuss that fine-tuning all layers of the pre-trained model maybe sub-optimal for small datasets.

The popularity of FT and lack of work on transfer learning methods beyond FT leads us to some natural questions: *Is there an alternative simple efficient method to enhance the specialization of pre-trained models to the target special domains for small-sized datasets? When will it have significant advantages over fine-tuning?*

In this work, we propose a simple and effi-

* Equal contribution by authors

cient method called **SimpleTran** for transferring pre-trained sentence embedding models for low resource datasets from specific domains. First, we train a simple sentence embedding model on the target dataset (which we refer to as the *domain specific model*). We combine the embeddings from this model with those from a pre-trained model using one of three different combination techniques: Concatenation, Canonical Correlation Analysis (CCA), and Kernel Canonical Correlation Analysis (KCCA). Once we have the combined representation, we train a linear classifier on top of it in two different ways: 1) by training only the classifier while fixing the embedding models, or 2) by training the whole network (the classifier plus the two embedding models) end-to-end. The former is simple and efficient. The latter gives more flexibility for transferring at the expense of a marginal computational overhead compared to FT.

We perform experiments on seven small to medium-sized text classification datasets over tasks like sentiment classification, question type classification and subjectivity classification, where we combine domain specific models like Text-CNN with pre-trained models like BERT. Results show that our simple and straightforward method is applicable to different datasets and tasks with several advantages over FT. First, our method with fixed embedding models using concatenation and KCCA can achieve significantly better prediction performance on small datasets and comparable performance on medium-sized datasets. It reduces the run time by $30\%-50\%$, without the large memory GPUs required for FT. Second, our method with end-to-end training outperforms FT, with less than 10% increase in the run time and about 1% increase in memory. We provide theoretical analysis for our combination methods, identifying conditions under which they work.

2 Related Work

Here we focus on the recent progress in pre-trained models that can provide sentence embeddings (explicitly or implicitly) (Peters et al., 2018; Radford et al., 2018). Among these, InferSent (Conneau et al., 2017) is trained on natural language inference data via supervised learning and generalizes well to many different tasks. GenSen (Subramanian et al., 2018) aims at learning a general purpose, fixed-length representation of sentences via multi-task learning, which is useful for transfer and low-resource learning. BERT (Devlin et al., 2018) learns language representations via unsupervised learning using a deep transformer architecture thereby exploiting bidirectional contexts.

The standard practice for transferring these pre-trained models is FT the pre-trained model by doing end-to-end training on both the classifier and the sentence embedding model (Howard and Ruder, 2018; Radford et al., 2018; Peters et al., 2019; Sun et al., 2019b). There exist other more sophisticated transferring methods, but they are typically much more expensive or complicated. For example, Xu et al. does "post-training" the pre-trained model on the target dataset, Houlsby et al. injects specifically designed new adapter layers, and Wang et al. first trains a deep network classifier on the fixed pre-trained embedding and then fine-tunes it. Our focus is to propose alternatives to FT for the low resource setting with similar simplicity and computational efficiency, and study conditions where it has significant advantages.

Several prior works for transferring machine learning models exist including that by Daume III which proposes to use $[x, x, 0]$ and $[x, 0, x]$ as the features for a source and target data point respectively where x is a representation of the point, and then train a classifier on top of the union of the source and target data. There are subsequent variants to this like Kim et al.; Kim et al., (Yu and Jiang, 2016) using auxiliary tasks, (Li et al., 2017; Chen et al., 2018) using adversarial training, and (He et al., 2018) using semi-supervision to align the source and target representations and then train on the source labels. A recent work (Arase and Tsujii, 2019) uses phrasal paraphrase relations to improve over BERT FT on small datasets. This however only applies to language understanding tasks which involve para-phrasal relations. Sun et al. show that within-task pre-training can harm the performance of pre-trained models for small datasets. This provides motivation for a transfer learning strategy not involving additional pre-training.

3 Methodology

Let s denote a sentence, and assume that we have a sentence embedding model $v_{1s} = f_1(s)$ which is pre-trained in a source domain and maps s to a vector $v_{1s} \in \mathbb{R}^{d_1}$. Here f_1 is assumed to be a large and powerful model such as BERT. Given a set of labeled training sentences $\mathcal{S} = \{(s_i, y_i)\}_{i=1}^m$ from a target domain, our goal is to use f_1 and $\mathcal{S}$ to learn

a classifier that works well on the target domain. [1]

SimpleTran first trains a sentence embedding model f_2 different from f_1 on $\mathcal{S}$, which is typically much smaller than f_1 and thus can be trained on the small dataset. f_2 can be learned through unsupervised (such as Bag-of-Words) or supervised (such as a text CNN (Kim, 2014) where the last layer is regarded as the sentence embedding) learning techniques. Let $v_{2s} = f_2(s) \in \mathbb{R}^{d_2}$ denote the embedding from this model. Our method combines v_{1s} and v_{2s} to get an adaptive sentence representation $\bar{v}_s$ using one of 3 approaches:

Concatenation: We use $\bar{v}_s^\top = [v_{1s}^\top, v_{2s}^\top]$. We also consider $\bar{v}_s^\top = [v_{1s}^\top, \alpha v_{2s}^\top]$ for some hyper-parameter $\alpha > 0$, to have different emphasis on the two embeddings. Note that previous work (e.g., ELMo (Peters et al., 2018)) uses concatenation of multiple embeddings, but not for transferring pre-trained representations.

CCA: Canonical Correlation Analysis (Hotelling, 1936) learns linear projections Φ_1 and Φ_2 into dimension d to maximize the correlations between the projections $\{\Phi_1 v_{1s_i}\}$ and $\{\Phi_2 v_{2s_i}\}$. Formally, we compute

$$(\Phi_1, \Phi_2) = \max_{P_1, P_2} \frac{\mathbb{E}_i[\langle P_1 v_{1s_i}, P_2 v_{2s_i} \rangle]}{\sqrt{\mathbb{E}_i[\|P_1 v_{1s_i}\|_2^2]}\sqrt{\mathbb{E}_i[\|P_2 v_{2s_i}\|_2^2]}}$$

where $\mathbb{E}_i$ is the average over the training data, d is a parameter satisfying $d \leq \min\{d_1, d_2\}$. To maximize the representation power, we use $d = \min\{d_1, d_2\}$. Then we set $\bar{v}_s^\top = \frac{1}{2}\Phi_1 v_{1s_i} + \frac{1}{2}\Phi_2 v_{2s_i}$.

KCCA: Kernel Canonical Correlation Analysis (Schölkopf et al., 1998) first applies non-linear projections g_1 and g_2 and then CCA on $\{g_1(v_{1s_i})\}_{i=1}^m$ and $\{g_2(v_{2s_i})\}_{i=1}^m$. The technical details can be found in (Schölkopf et al., 1998) or (Hardoon et al., 2004). Again, we set $d = \min\{d_1, d_2\}$ and $\bar{v}_s^\top = \frac{1}{2}g_1(v_{1s_i}) + \frac{1}{2}g_2(v_{2s_i})$.

Finally, our method trains a linear classifier $c(\bar{v}_s)$ on $\bar{v}_s$ using the target dataset $\mathcal{S}$ in two different ways: (i) Training only the classifier c while fixing the weights of the underlying embedding models f_1 and f_2, (ii) End-to-end training the classifier c as well as f_1 and f_2.

Since CCA and KCCA have computationally expensive projections and concatenation is observed to have strong performance in our experiments, we use the end-to-end training method only for concatenation, which we refer to as **ConcatFT** (See Figure 1 for an illustration). Therefore, we have 4 variants of **SimpleTran**: 3 on fixed embedding models (Concat, CCA, and KCCA), and one with end-to-end training (ConcatFT).

[1] Our method is general enough for longer texts, and easily applicable to multiple pre-trained models

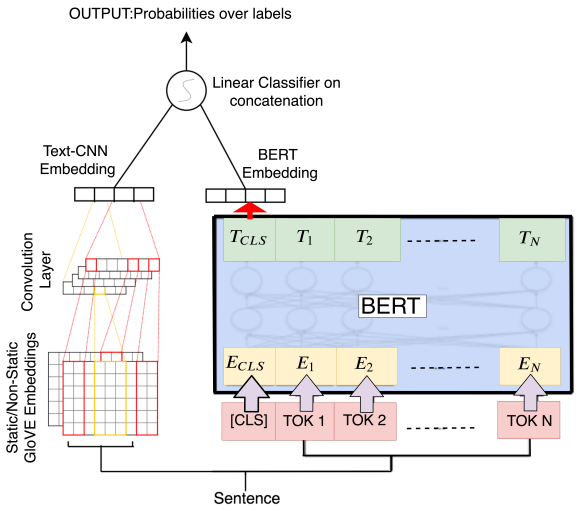

Figure 1: ConcatFT of Text-CNN and BERT

4 Theoretical Analysis

For insights on how our methods affect the information contained in the representations for classification, we analyze them under a theoretical model of the data. We present the theorems here and provide proofs and discussion in the supplementary.

Theoretical model Assume there exists a "ground-truth" embedding vector v_s^* for each sentence s with label y_s, and a linear classifier $f^*(s) = \langle w^*, v_s^* \rangle$ with a small loss $L(f^*) = \mathbb{E}_s[\ell(f^*(s), y_s)]$ w.r.t. some loss function ℓ (such as cross-entropy), where $\mathbb{E}_s$ denotes the expectation over the true data distribution. The good performance of the concatenation method (see Section 5) suggests that there exists a linear relationship between the embeddings v_{1s}, v_{2s} and v^*. So our theoretical model assumes: $v_{1s} = P_1 v_s^* + \epsilon_1$, and $v_{2s} = P_2 v_s^* + \epsilon_2$ where ϵ_i's are noises independent of v_s^* with variances σ_i^2's. If $P^\top = [P_1^\top, P_2^\top]$ and $\epsilon^\top = [\epsilon_1^\top, \epsilon_2^\top]$, then the concatenation is $\bar{v}_s = [v_{1s}^\top, v_{2s}^\top]^\top = Pv_s^* + \epsilon$. Let $\sigma = \sqrt{\sigma_1^2 + \sigma_2^2}$.

4.1 Concatenation

We have the following theorem about the prediction power of concatenation.

Theorem 1. *Suppose the loss function is λ-Lipschitz for the first parameter, and P has full column rank. Then there exists a linear classifier $\bar{f}$ over $\bar{v}_s$ such that $L(\bar{f}) \leq L(f^*) + \lambda\sigma\|(P^\dagger)^\top w^*\|_2$ where $P^\dagger$ is the pseudo-inverse of P.*

Proof. Let $\bar{f}$ have weight $\bar{w} = (P^\dagger)^\top w^*$. Then

$$\begin{aligned}\langle \bar{w}, \bar{v}_s \rangle &= \langle (P^\dagger)^\top w^*, P v_s^* + \epsilon \rangle \\ &= \langle w^*, P^\dagger P v_s^* \rangle + \langle (P^\dagger)^\top w^*, \epsilon \rangle \\ &= \langle w^*, v_s^* \rangle + \langle (P^\dagger)^\top w^*, \epsilon \rangle.\end{aligned}$$

Then the difference in the losses is

$$\begin{aligned}L(\bar{f}) - L(f^*) &= \mathbb{E}_s[\ell(\bar{f}(s), y_s) - \ell(f^*(s), y_s)] \\ &\leq \lambda \mathbb{E}_s |\bar{f}(s) - f^*(s)| \\ &= \lambda \mathbb{E}_s |\langle (P^\dagger)^\top w^*, \epsilon \rangle| \\ &\leq \lambda \sqrt{\mathbb{E}_s \langle (P^\dagger)^\top w^*, \epsilon \rangle^2} \\ &\leq \lambda \sqrt{\mathbb{E}_s \|(P^\dagger)^\top w^*\|_2^2 \|\epsilon\|_2^2} \\ &= \lambda\sigma \|(P^\dagger)^\top w^*\|_2\end{aligned}$$

where we use the Lipschitz-ness of the loss in the second step, Jensen's inequality in the fourth, and Cauchy-Schwarz inequality in the fifth. □

The assumption of Lipschitz-ness of the loss implies that the loss changes smoothly with the prediction. The assumption on P having full column rank implies that v_{1s}, v_{2s} contain the information of v_s^* and ensures that $P^\dagger$ exists.[2]

Explanation: Suppose P has singular vector decomposition $P = U\Sigma V^\top$, then $\|(P^\dagger)^\top w^*\|_2 = \|(\Sigma^\dagger)^\top V^\top w^*\|_2$. So if the top right singular vectors in V align with w^*, then $\|(P^\dagger)^\top w^*\|_2$ will be small. This means that if P_1 and P_2 together cover the direction w^*, they can capture information important for classification, and then there will be a good linear classifier on the concatenated embeddings (assuming the noise level σ is not too large).

Consider a simple example where v_s^* has 4 dimensions, and $w^* = [1, 1, 0, 0]^\top$, i.e., only the first two dimensions are useful for classification. Suppose $P_1 = \text{diag}(c, 0, 1, 0)$ is a diagonal matrix, so that v_{1s} captures the first dimension with scaling factor $c > 0$ and the third dimension with factor 1, and $P_2 = \text{diag}(0, c, 0, 1)$ so that v_{2s} captures the other two dimensions. Hence we have $(P^\dagger)^\top w^* = [1/c, 1/c, 0, 0]^\top$, and thus $L(\bar{f}) \leq L(f^*) + \sqrt{2}\lambda\frac{\sigma}{c}$. Thus the quality of the classifier is determined by the noise-signal ratio σ/c. If c is small, implying that v_{1s} and v_{2s} contain a large amount of noise, then the loss is large. If c is large, implying that v_{1s} and v_{2s} contain useful information for classification along and very low noise, then the loss is close to that of f^*. Note that $\bar{f}$ can be much better than any classifier that uses only v_{1s} or v_{2s} since the latter only has a part of the features determining the class labels.

4.2 Dimension Reduction

A significant observation in our experiments (see Section 5) is that CCA on the embeddings leads to worse performance (sometimes much worse) than concatenation. To explain this, the following theorem constructs an example, which shows that even when v_{1s} and v_{2s} have good information for classification, CCA can eliminate it and lead to bad performance.

Theorem 2. *Let $\bar{v}_s$ denote the embedding for sentence s obtained by concatenation, and $\tilde{v}_s$ denote that obtained by CCA. There exists a setting of the data and w^*, P, ϵ such that there exists a linear classifier $\bar{f}$ on $\bar{v}_s$ with the same loss as f^*, while CCA achieves the maximum correlation but any classifier on $\tilde{v}_s$ is at best random guessing.*

Proof. Suppose we do CCA to d dimensions. Suppose v_s^* has $d + 2$ dimensions, each being an independent Gaussian. Suppose $w^* = [1, 1, 0, \ldots, 0]^\top$, and the label is $y_s = 1$ if $\langle w^*, v_s^* \rangle \geq 0$ and 0 otherwise. Suppose $\epsilon = 0$, $P_1 = \text{diag}(1, 0, 1, \ldots, 1)$, and $P_2 = \text{diag}(0, 1, 1, \ldots, 1)$.

Let the linear classifier $\bar{f}$ have weight $[1, 0, \mathbf{0}, 0, 1, \mathbf{0}]^\top$ where $\mathbf{0}$ is the zero vector of d dimensions. Clearly, $\bar{f}(s) = f^*(s)$ for any s, so it has the same loss as f^*.

For CCA, since the coordinates of v_s^* are independent Gaussians, v_{1s} and v_{2s} only have correlation in the last d dimensions. Solving the CCA optimization, the projection matrices for both embeddings are the same $\phi = \text{diag}(0, 0, 1, \ldots, 1)$ which achieves the maximum correlation. Then the CCA embedding is $\tilde{v}_s = [0, 0, (v_s^*)_{3:(d+2)}]$ where $(v_s^*)_{3:(d+2)}$ are the last d dimensions of v_s^*, which contains no information about the label. Therefore, any classifier on $\tilde{v}_s$ is at best random guessing. □

Explanation: Intuitively, v_{1s} and v_{2s} have some common information and each has a set of special

[2] Dropping the full-rank assumption leads to a more involved and non-intuitive analysis

information about the correct class labels. If the two sets of special information are uncorrelated, then they will be eliminated by CCA. Now, if the common information is irrelevant in determining the labels, then the best any classifier can do on the CCA embeddings is just random guessing. This is a fundamental drawback of this unsupervised technique, clearly demonstrated by the extreme example in the theorem. In practice, the common information can contain some relevant information for the classification task, thus making CCA embeddings worse than concatenation but better than random guessing. KCCA can be viewed as CCA on a nonlinear transformation of v_{1s} and v_{2s} where the special information gets mixed non-linearly and cannot be separated out and eliminated by CCA. This explains why the poor performance of CCA is not observed for KCCA in our experiments.

Empirical Verification of Theorem 2 An important insight from the analysis is that when the two sets of embeddings have special information that is not shared with each other but is important for classification, then CCA will eliminate such information and have bad prediction performance. Let $r_{2s} = v_{2s} - \Phi_2^\top \Phi_2 v_{2s}$ be the residue vector for the projection Φ_2 learned by CCA for the special domain, and similarly define r_{1s}. Then the analysis suggests that the residues r_{1s} and r_{2s} contain information important for prediction. We conduct experiments for BERT+CNN-non-static on Amazon reviews, and find that a classifier on the concatenation of r_{1s} and r_{2s} has accuracy 96.3%. This is much better than 81.3% on the combined embeddings via CCA. These observations provide positive support for our analysis.

5 Experiments

5.1 Datasets

We evaluate our method on different text classification tasks including sentiment classification, question type classification, subjectivity classification, etc. We consider 3 small datasets: derived from Amazon, IMDB and Yelp reviews; and 4 medium-sized datasets: movie reviews (MR), opinion polarity (MPQA), question type classification (TREC) and subjectivity classification (SUBJ). The Amazon, Yelp and IMDB review datasets capture sentiment information from different target domains which is very different from the general text corpora of the pre-trained models and have been used in recent related work (Sarma et al., 2018). The dataset statistics are summarized in Table 1.

- *Amazon*: A Amazon product reviews dataset with 'Positive' / 'Negative' reviews [3].
- *IMDB*: A dataset of movie reviews on IMDB with 'Positive' or 'Negative' reviews [3].
- *Yelp*: A dataset of restaurant reviews from Yelp with 'Positive' or 'Negative' reviews[3].
- *MR*: A dataset of movie reviews based on sentiment polarity and subjective rating (Pang and Lee, 2005)[4].
- *MPQA*: An unbalanced dataset (70% negative examples) for opinion polarity detection (Wiebe and Wilson, 2005)[5].
- *TREC*: A question type classification dataset with 6 classes for questions about a person, location, numeric information, etc. (Li and Roth, 2002)[6].
- *SUBJ*: A dataset for classifying a sentence as subjective or objective (Pang and Lee, 2004).

Dataset	c	N	Test
Amazon (Sarma et al., 2018)	2	1000	100
IMDB (Sarma et al., 2018)	2	1000	100
Yelp (Sarma et al., 2018)	2	1000	100
MR (Pang and Lee, 2005)	2	10662	1067
MPQA (Wiebe and Wilson, 2005)	2	10606	1060
TREC (Li and Roth, 2002)	6	5952	500
SUBJ (Pang and Lee, 2004)	2	10000	1000

Table 1: Dataset statistics. *c*: Number of classes, *N*: Dataset size, *Test*: Test set size (if no standard test set, we use a random train/dev/test split of 80/10/10 $\%$)

5.2 Models for Evaluation

We choose 2 domain specific models: A Bag-of-Words model that averages word vectors in the sentence to get its embedding; and a Text-CNN (Kim, 2014) with 3 approaches to initialize the word embeddings: (i) randomly initialized which we refer to as CNN-rand (ii) initialized with GloVe vectors and made non-trainable which we refer to as

[3] https://archive.ics.uci.edu/ml/datasets/Sentiment+Labelled+Sentences

[4] https://www.cs.cornell.edu/people/pabo/movie-review-data/

[5] http://mpqa.cs.pitt.edu/

[6] http://cogcomp.org/Data/QA/QC/

	Amazon				Yelp				IMDB			
	BERT-FT: 94.00		**Adapter:** 94.25		**BERT-FT:** 91.67		**Adapter:** 93.50		**BERT-FT:** 92.33		**Adapter:** 90.50	
	BOW	**CNN-R**	**CNN-S**	**CNN-NS**	**BOW**	**CNN-R**	**CNN-S**	**CNN-NS**	**BOW**	**CNN-R**	**CNN-S**	**CNN-NS**
Default	79.2	91.1	94.7	95.9	81.3	92.7	95.2	95.8	89.3	93.2	96.6	96.8
ConcatFT	-	**94.0**	**95.7**	**96.8**	-	**96.2**	**97.2**	**98.3**	-	**97.0**	**98.3**	**98.4**
Concat	89.6	93.2	95.3	96.4	89.0	96.5	97.1	98.3	89.3	96.2	98.1	98.3
KCCA	89.1	91.5	94.3	95.8	88.5	91.5	91.9	96.2	88.3	94.1	97.9	97.2
CCA	50.9	79.1	83.6	81.3	50.3	71.5	67.8	69.4	51.0	80.8	83.3	85.0

Table 2: Test accuracy for Amazon, Yelp and IMDB datasets. BOW, CNN-R, CNN-S and CNN-NS refers to Bag of Words, text CNN with random initialised, static and non-static word embeddings. Best results in boldface.

CNN-static and (iii) initialized with GloVe vectors and trainable (CNN-non-static). We use 12 layer BERT (Devlin et al., 2018) base uncased model as the pre-trained model. We also experiment with other pre-trained models like GenSen and InferSent and present these results in the Appendix.

5.3 Results on Small Datasets

On the 3 small datasets, we evaluate variants of **SimpleTran** in Table 2. The key observations from these results are as follows:

- ConcatFT always gets the best performance. Furthermore, even Concat and KCCA always improves over the used baselines (the domain specific and fine-tuned BERT). This demonstrates its effectiveness in transferring the knowledge from the general domain while exploiting the target domain. The CCA variant shows inferior performance, worse than the baselines. Our analysis in Section 4 shows that this is because CCA can potentially remove useful information and capture nuisance or noise. This is also further verified empirically at the end of this section.

- Concat is simpler and computationally cheaper than KCCA and achieves better results, hence is the recommended method over KCCA.

- Our method gets better results using better domain specific models (using CNN instead of BOW). This is because better specific models capture more domain specific information useful for classification.

The Concat, CCA and KCCA variants of our method require less computational resources than BERT-FT. The total time of our method is the time taken to train the text-CNN, extract BERT embeddings, apply a combination (concatenation, CCA, or KCCA), and train a classifier on the combined embedding. For the Amazon dataset, Concat requires about 125 seconds, reducing around 30% of the 180 seconds for FT BERT. Additionally, our approach has small memory requirements as it can be computed on a CPU in contrast to BERT-FT which requires, at minimum, a 12GB memory GPU. The total time of ConcatFT is 195 seconds, which is less than a 9% increase over FT. It also has a negligible 1.04% increase in memory (the number of parameters increases from 109,483,778 to 110,630,332 due to the text-CNN).

5.4 Results on Medium-sized Datasets

We use the CNN-non-static model and omit KCCA due to its inefficient non-linear computations and summarize the results in Table 3. Again, ConcatFT achieves the best performance on all the datasets improving the performance of BERT-FT. This improvement comes at small computational overhead. On the MR dataset, ConcatFT requires 610 seconds which is about 9% increase over the 560 seconds of BERT-FT, and recall that it only has about 1% increase in memory. Concat can achieve comparable test accuracy on all the tasks while being much more computationally efficient. On the MR dataset, it requires 290 seconds, reducing about 50% of the 560 seconds for BERT-FT.

	MR	**MPQA**	**SUBJ**	**TREC**
CNN-non-static	80.93	88.38	89.25	92.98
BERT No-FT	83.26	87.44	95.96	88.06
Adapter BERT	85.55	90.40	97.40	96.55
BERT FT	86.22	90.47	96.95	96.40
Concat	85.60	90.06	95.92	96.64
CCA	85.41	77.22	94.55	84.28
ConcatFT	**87.15**	**91.19**	**97.60**	**97.06**

Table 3: Test accuracy for medium-sized datasets. Best results on the datasets are highlighted in boldface.

The Adapter approach (Houlsby et al., 2019) injects new adapter modules into the pre-trained BERT model, freezes the weights of BERT and trains the adapter module on the target data. Therefore,

Size	CNN	No-FT	FT	Concat	CCA	ConcatFT
100	76.23	75.69	76.85	73.28	55.78	**78.76**
200	74.95	77.31	79.41	77.74	61.73	**80.02**
500	78.39	78.75	80.38	80.10	58.87	**80.96**
2000	78.30	80.78	82.75	80.89	79.53	**83.94**
4000	79.29	81.96	83.88	83.85	83.37	**84.82**
6000	80.14	82.80	84.72	85.08	85.28	**86.06**
8528	80.93	83.26	86.22	85.60	85.41	**87.15**

Table 4: Test accuracy for the MR dataset with different training dataset sizes. FT and No-FT refers to BERT fine-tuned and not fine-tuned respectively.

our method with fixed embedding models (Concat, CCA and KCCA) can be directly compared with this since neither fine-tunes the BERT parameters. Interestingly, the Concat variant of our method can outperform the Adapter approach for small datasets and perform comparably on medium sized datasets having 2 clear advantages over the latter:

- We do not need to open the BERT model and access its parameters to introduce intermediate layers and hence our method is modular for a large range of pre-trained models.
- Concat introduces roughly only 1% extra parameters as compared to the $3-4\%$ of the latter thereby being more parameter efficient. Concat-FT which performs end-to-end fine-tuning over the BERT model beats the performance of the Adapter approach for all the datasets.

5.5 Effect of Dataset Size

To study the effect of the dataset size on the performance, we vary the training data size in MR dataset via random sub-sampling and then use our method. From Table 4, we observe that ConcatFT gets the best results across all training data sizes, significantly improving over BERT-FT. Concat gets performance comparable to BERT-FT on a wide range of dataset sizes, from 500 points on. The performance of CCA improves with more training data as more data leads to less noise and thus less nuisance information in the obtained embeddings (Ref Sec 4).

We present a qualitative analysis through examples which SimpleTran is able to correctly classify but the pre-trained and domain specific model fail to classify correctly independently in the Appendix.

6 Conclusion

We proposed a simple method for transferring a pre-trained sentence embedding model for text classification tasks in a low resource setting. We experimentally show that our method can transfer the knowledge from the pre-trained model and leverage that in the target domain, leading to substantial improvement over baselines on small and medium-sized datasets. We also provided theoretical analysis identifying the success conditions of the method and explaining the experimental results.

References

Yuki Arase and Jun'ichi Tsujii. 2019. Transfer fine-tuning: A BERT case study. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5393–5404, Hong Kong, China. Association for Computational Linguistics.

Xilun Chen, Yu Sun, Ben Athiwaratkun, Claire Cardie, and Kilian Weinberger. 2018. Adversarial deep averaging networks for cross-lingual sentiment classification. *Transactions of the Association for Computational Linguistics*, 6:557–570.

Colin Cherry, Greg Durrett, George Foster, Reza Haffari, Shahram Khadivi, Nanyun Peng, Xiang Ren, and Swabha Swayamdipta, editors. 2019. *Proceedings of the 2nd Workshop on Deep Learning Approaches for Low-Resource NLP (DeepLo 2019)*. Association for Computational Linguistics, Hong Kong, China.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680.

Hal Daume III. 2007. Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 256–263.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Siddhant Garg, Thuy Vu, and Alessandro Moschitti. 2019. Tanda: Transfer and adapt pre-trained transformer models for answer sentence selection.

David R Hardoon, Sandor Szedmak, and John Shawe-Taylor. 2004. Canonical correlation analysis: An overview with application to learning methods. *Neural computation*, 16(12):2639–2664.

Ruidan He, Wee Sun Lee, Hwee Tou Ng, and Daniel Dahlmeier. 2018. Adaptive semi-supervised learning for cross-domain sentiment classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3467–3476.

H Hotelling. 1936. Relations between two sets of variates. *Biometrika*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1746–1751.

Young-Bum Kim, Karl Stratos, and Dongchan Kim. 2017. Domain attention with an ensemble of experts. In *Proceedings of the 55th ACL (Volume 1: Long Papers)*, pages 643–653.

Young-Bum Kim, Karl Stratos, and Ruhi Sarikaya. 2016. Frustratingly easy neural domain adaptation. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 387–396.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. 2015. DBpedia - a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web Journal*, 6(2).

Xin Li and Dan Roth. 2002. Learning question classifiers. In *Proceedings of the 19th International Conference on Computational Linguistics - Volume 1*, COLING '02, pages 1–7, Stroudsburg, PA, USA. Association for Computational Linguistics.

Zheng Li, Yun Zhang, Ying Wei, Yuxiang Wu, and Qiang Yang. 2017. End-to-end adversarial memory network for cross-domain sentiment classification. In *IJCAI*, pages 2237–2243.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity. In *Proceedings of ACL*, pages 271–278.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of ACL*, pages 115–124.

Matthew Peters, Sebastian Ruder, and Noah A Smith. 2019. To tune or not to tune? adapting pretrained representations to diverse tasks. *arXiv preprint arXiv:1903.05987*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of NAACL-HLT*, pages 2227–2237.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Prathusha K Sarma, Yingyu Liang, and Bill Sethares. 2018. Domain adapted word embeddings for improved sentiment classification. In *Proceedings of the 56th ACL*, pages 37–42.

Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. 1998. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319.

Sandeep Subramanian, Adam Trischler, Yoshua Bengio, and Christopher J Pal. 2018. Learning general purpose distributed sentence representations via large scale multi-task learning. *arXiv preprint arXiv:1804.00079*.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019a. How to fine-tune BERT for text classification? *CoRR*, abs/1905.05583.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019b. How to fine-tune bert for text classification? *arXiv preprint arXiv:1905.05583*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP*, Brussels, Belgium. Association for Computational Linguistics.

Canhui Wang, Min Zhang, Shaoping Ma, and Liyun Ru. 2008. Automatic online news issue construction in web environment. In *Proceedings of the 17th WWW*, page 457466, New York, NY, USA. Association for Computing Machinery.

Ran Wang, Haibo Su, Chunye Wang, Kailin Ji, and Jupeng Ding. 2019. To tune or not to tune? how about the best of both worlds? *ArXiv*.

Janyce Wiebe and Theresa Wilson. 2005. Annotating expressions of opinions and emotions in language. *Language Resources and Evaluation*, 39(2):165–210.

Hu Xu, Bing Liu, Lei Shu, and S Yu Philip. 2019. Bert post-training for review reading comprehension and aspect-based sentiment analysis. In *Proceedings of the 2019 NAACL: HLT*, pages 2324–2335.

Jianfei Yu and Jing Jiang. 2016. Learning sentence embeddings with auxiliary tasks for cross-domain sentiment classification. In *Proceedings of EMNLP 2016*.

A Appendix

A.1 Qualitative Analysis

We present some qualitative examples from the Amazon, IMDB and Yelp datasets on which BERT and CNN-non-static are unable to provide the correct class predictions, while Concat or KCCA can successfully provide the correct class predictions in Table 5. We observe that these are either short sentences or ones where the content is tied to the specific reviewing context as well as the involved structure to be parsed with general knowledge. Such input sentences thus require combining both the general semantics of BERT and the domain specific semantics of CNN-non-static to predict the correct class labels.

Correctly classified by KCCA
However-the ringtones are not the best, and neither are the games.
This is cool because most cases are just open there allowing the screen to get all scratched up.
Correctly classified by Concatenation
TNot nearly as good looking as the amazon picture makes it look .
Magical Help .

(a) Amazon

Correctly classified by KCCA
I would have casted her in that role after ready the script .
Predictable , but not a bad watch .
Correctly classified by Concatenation
I would have casted her in that role after ready the script .
Predictable , but not a bad watch .

(b) IMDB

Correctly classified by KCCA
The lighting is just dark enough to set the mood .
I went to Bachi Burger on a friend's recommendation and was not disappointed .
dont go here .
I found this place by accident and I could not be happier .
Correctly classified by Concatenation
The lighting is just dark enough to set the mood .
I went to Bachi Burger on a friend's recommendation and was not disappointed .
dont go here .
I found this place by accident and I could not be happier .

(c) Yelp

Table 5: Sentences from Amazon, IMDB, Yelp datasets where KCCA and concatenation of BERT and CNN-non-static embeddings succeeds while they individually give wrong predictions.

A.2 Training Details, Hyper-parameters

We train domain specific embeddings on the training data and extract the embeddings. We combine these with the embeddings from the pre-trained models and train a regularized logistic regression classifier on top. The classifier is learned on the training data, while using the dev data for hyper-parameter tuning of the regression weights. The classifier can be trained by fixing the weights of the underlying embedding models or training the whole network end-to-end. The performance is tested on the test set. For concatenation, we also tune the hyper-parameter α over the validation data on medium-sized datasets while fixing it to 1 on small datasets. We use test accuracy as the performance metric and report all results averaged over 10 experiments unless mentioned otherwise. The experiments are performed on an NVIDIA Tesla V100 16 GB GPU.

Text-CNN For all datasets, we use filter windows of sizes 3, 4, 5 with 100 feature maps each and a dropout rate of 0.5 trained with $L2$ constraint loss to obtain 384 dimensional sentence embeddings.

BERT We use BERT[7] in two ways: (i) Fine-tuned end-to-end on the train and validation splits of the datasets. BERT fine-tuning results were reported after fine-tuning over the dataset for 20 epochs with early stopping by choosing the best performing model on the validation data. (ii) A classifier learned naïvely on pre-trained BERT model embeddings of 768 dimensions.

InferSent We use the pre-trained InferSent model to obtain 4096 dimensional sentence embeddings using the implementation provided in the SentEval[8] repository. For all the experiments, InferSent v1 was used.

GenSen Similar to the InferSent model, we use the GenSen model implemented in the SentEval repository to obtain 4096 dimensional sentence embeddings.

We build upon code of the SentEval toolkit and use the sentence embeddings obtained from each of the models: BERT, InferSent and GenSen to perform text classification. A batch size of 128 is used and the classifier is trained for 2 epochs on the training data. Further experimental details are presented below:

[7] https://github.com/google-research/bert

[8] https://github.com/facebookresearch/SentEval

				BOW	CNN-rand	CNN-static	CNN-non-static
Amazon	**Default**			79.20 ± 2.31	91.10 ± 1.64	94.70 ± 0.64	95.90 ± 0.70
	BERT	94.00 ± 0.02	ConcatFT	-	94.05 ± 0.23	95.70 ± 0.50	***96.75 ± 0.76***
			Concat	89.59 ± 1.22	93.20 ± 0.98	95.30 ± 0.46	**96.40 ± 1.11**
			KCCA	89.12 ± 0.47	91.50 ± 1.63	94.30 ± 0.46	95.80 ± 0.40
			CCA	50.91 ± 1.12	79.10 ± 2.51	83.60 ± 1.69	81.30 ± 3.16
	GenSen	82.55 ± 0.82	Concat	82.82 ± 0.97	92.80 ± 1.25	94.10 ± 0.70	95.00 ± 1.0
			KCCA	79.21 ± 2.28	91.30 ± 1.42	94.80 ± 0.75	**95.90 ± 0.30**
			CCA	52.80 ± 0.74	80.60 ± 4.87	83.00 ± 2.45	84.95 ± 1.45
	InferSent	85.29 ± 1.61	Concat	51.89 ± 0.62	90.30 ± 1.48	94.70 ± 1.10	95.90 ± 0.70
			KCCA	52.29 ± 0.74	91.70 ± 1.49	95.00 ± 0.00	**96.00 ± 0.00**
			CCA	53.10 ± 0.82	61.10 ± 3.47	65.50 ± 3.69	71.40 ± 3.04
Yelp	**Default**			81.3 ± 2.72	92.71± 0.46	95.25 ± 0.39	95.83 ± 0.14
	BERT	91.67 ± 0.00	ConcatFT	-	96.23 ± 1.04	97.23 ± 0.70	***98.34 ± 0.62***
			Concat	89.03 ± 0.70	96.50 ± 1.33	97.10 ± 0.70	**98.30 ± 0.78**
			KCCA	88.51 ± 1.22	91.54 ± 4.63	91.91 ±1.13	96.2 ± 0.87
			CCA	50.27 ± 1.33	71.53 ± 2.46	67.83 ± 3.07	69.4 ± 3.35
	GenSen	86.75 ± 0.79	Concat	85.94 ± 1.04	94.24 ± 0.53	95.77 ± 0.36	**96.03 ± 0.23**
			KCCA	83.35 ± 1.79	92.58 ± 0.31	95.41 ± 0.45	95.06 ± 0.56
			CCA	57.14 ± 0.84	84.27 ± 1.68	86.94 ± 1.62	87.27± 1.81
	InferSent	85.7 ± 1.12	Concat	50.83 ± 0.42	91.94 ± 0.46	96.10 ± 1.30	**97.00 ± 0.77**
			KCCA	50.80 ± 0.65	91.13 ± 1.63	95.45 ± 0.23	95.57 ± 0.55
			CCA	55.91 ± 1.23	60.80 ± 2.22	54.70 ± 1.34	59.50 ± 1.85
IMDB	**Default**			89.30 ± 1.00	93.25 ± 0.38	96.62 ± 0.46	96.76 ± 0.26
	BERT	92.33 ± 0.00	ConcatFT	-	97.07 ± 0.95	98.31 ± 0.83	***98.42 ± 0.78***
			Concat	89.27 ± 0.97	96.20 ± 2.18	98.10 ± 0.94	**98.30 ± 1.35**
			KCCA	88.29 ± 0.65	94.10 ± 1.87	97.90 ± 0.30	97.20 ± 0.40
			CCA	51.03 ± 1.20	80.80 ± 2.75	83.30 ± 4.47	84.97 ± 1.44
	GenSen	86.41 ± 0.66	Concat	86.86 ± 0.62	95.63 ± 0.47	97.22 ± 0.27	**97.42 ± 0.31**
			KCCA	84.72 ± 0.93	93.23 ± 0.38	96.19 ± 0.21	96.60 ± 0.37
			CCA	51.48 ± 1.02	86.28 ± 1.76	87.30 ± 2.12	87.47 ± 2.17
	InferSent	84.3 ± 0.63	Concat	50.36 ± 0.62	92.30 ± 1.26	97.90 ± 1.37	97.10 ± 1.22
			KCCA	50.09 ± 0.68	92.40 ± 1.11	97.62 ±0.48	**98.20 ± 1.40**
			CCA	52.56 ± 1.15	54.50 ± 4.92	54.20 ± 5.15	61.00 ± 4.64

Table 6: Test accuracy (± std dev) for Amazon, Yelp and IMDB review datasets. Default values are performance of the domain specific models. Default values for BERT, Gensen and InferSent correspond to fine-tuning these models. Concat-FT refers to Concat Fine-Tuning and gets the best results (in boldface and italic). Concat, CCA, KCCA correspond to classifier trained on the combined embeddings, with fixed embedding model weights and the best results from these are in boldface.

- **Concat:** The hyper-parameter α determines the weight corresponding to the domain specific embeddings. We tune the value of α via grid search in the range $[0.002, 500]$ in multiplicative steps of 10 over the validation data.

- **CCA:** The regularization parameter for canonical correlation analysis is tuned via grid search in $[0.00001, 10]$ in multiplicative steps of 10 over the validation data.

- **KCCA**: We use a Gaussian kernel with a regularized KCCA implementation where the Gaussian sigma and the regularization parameter are tuned via grid search in $[0.05, 10]$ and $[0.00001, 10]$ respectively in multiplicative steps of 10 over the validation data.

- **ConcatFT**: We first train the domain specific model independently on the training data of the target dataset. We combine these embeddings with those obtained from BERT by concatenation. We train a simple linear classifier along with both the Text-CNN and BERT models end-to-end. The end-to-end training is done over the training data for 20 epochs with early stopping by choosing the best performing model on the validation data. As the models can update the embeddings while learning the classifier, we do not use the α parameter for the concatenation here.

A.3 Results and Error Bounds

We present a comprehensive results along with error bounds on small datasets (Amazon, IMDB and

	MR	MPQA	SUBJ	TREC
CNN-non-static	80.93 ± 0.16	88.38 ± 0.28	89.25 ± 0.08	92.98 ± 0.89
BERT No Fine-tuning	83.26 ± 0.67	87.44 ± 1.37	95.96 ± 0.27	88.06 ± 1.90
BERT Fine-tuning	86.22 ± 0.85	90.47 ± 1.04	96.95 ± 0.14	96.40 ± 0.67
Concat	85.60 ± 0.95	90.06 ± 0.48	95.92± 0.26	96.64 ± 1.07
CCA	85.41 ± 1.18	77.22 ± 1.82	94.55 ± 0.44	84.28 ± 2.96
ConcatFT	**87.15 ± 0.70**	**91.19 ± 0.84**	**97.60 ± 0.23**	**97.06 ± 0.48**

Table 7: Test accuracy (± std dev) for four medium-sized datasets. Best results on the datasets are highlighted in boldface. The domain specific embedding model used is CNN-non-static, and the pre-trained model used is BERT.

	Amazon	Yelp	IMDB	MR	MPQA	SUBJ	TREC
BERT-FT	94.00 ± 0.02	91.67 ± 0.00	92.33 ± 0.00	86.22 ± 0.95	90.47 ± 1.04	96.95 ± 0.14	96.40 ± 0.67
Adapter	94.25 ± 0.96	93.50 ± 1.00	90.50 ± 0.58	85.55 ± 0.38	90.40 ± 0.14	97.40 ± 0.26	96.55 ± 0.30
Concat	96.40 ± 1.11	98.30 ± 0.78	98.30 ± 1.35	85.60 ± 0.95	90.06 ± 0.48	95.92 ± 0.26	96.64 ± 1.07
ConcatFT	**96.75 ± 0.76**	**98.34 ± 0.62**	**98.42 ± 0.78**	**87.15 ± 0.70**	**91.19 ± 0.84**	**97.60 ± 0.23**	**97.06 ± 0.48**

Table 8: Comparing SimpleTran's test accuracy (± std dev) with Adapter on small and medium-sized datasets. Best results are shown in boldface. SimpleTran's ConcatFT results outperform Adapter results for all datasets.

Yelp reviews) in Table 6, where we evaluate **SimpleTran** using three popularly used pre-trained sentence embedding models, namely BERT, GenSen and InferSent. We present the error bounds on the results for medium sized datasets in Table 7.

A.4 Comparison with Adapter Modules

We compare our method with the Adapter approach of (Houlsby et al., 2019) in Table 8. Recall that the Adapter approach injects new adapter modules into the pre-trained BERT model, freezes the weights of BERT and trains the weights of the adapter module on the target data for transferring. Therefore, our method with fixed embedding models (Concat, CCA and KCCA) can be directly compared with the adapter module approach since neither of them fine-tunes the BERT model parameters. Interestingly, the Concat variant of our method can outperform the Adapter module approach for the small datasets and perform comparably on medium sized datasets having 2 clear advantages over the latter. (i) We do not need to open the BERT model and access its parameters to introduce intermediate layers and hence our method is modular for a large range of pre-trained models. (ii) Concat introduces roughly only 1% extra parameters as compared to the $3-4\%$ of the latter thereby being more parameter efficient. The Concat-FT variant of our method which performs end-to-end fine-tuning over the BERT model beats the performance of the Adapter approach for all the datasets.