IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 30, 2019, accepted October 22, 2019, date of publication November 8, 2019,
date of current version November 20, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2952360

Cross-Domain Sentiment Classification With Bidirectional Contextualized Transformer Language Models

BATSERGELEN MYAGMAR[1], JIE LI[2], (Senior Member, IEEE), AND SHIGETOMO KIMURA[1], (Member, IEEE)
[1]Department of Computer Science, University of Tsukuba, Tsukuba 305-8577, Japan
[2]Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China

Corresponding author: Jie Li (lijiecs@sjtu.edu.cn)

This work was supported by the JSPS KAKENHI under Grant JP26280027.

ABSTRACT Cross-domain sentiment classification is an important Natural Language Processing (NLP) task that aims at leveraging knowledge obtained from a source domain to train a high-performance learner for sentiment classification on a target domain. Existing transfer learning methods applied on cross-domain sentiment classification mostly focus on inducing a low-dimensional feature representation shared across domains based on pivots and non-pivots, which is still a low-level representation of sequence data. Recently, there have been great progress in the NLP literature in developing high-level representation language models based on Transformer architecture, which are pre-trained on large text corpus and fine-tuned for specific task with an additional layer on top. Among such language models, the bidirectional contextualized Transformer language models of BERT and XLNet have greatly impacted NLP research field. In this paper, we fine-tune BERT and XLNet for the cross-domain sentiment classification. We then explore their transferability in the context of cross-domain sentiment classification through in-depth analysis of two models' performances and update the state-of-the-arts with a significant margin of improvement. Our results show that such bidirectional contextualized language models outperform the previous state-of-the-arts methods for cross-domain sentiment classification while using up to 120 times less data.

INDEX TERMS Transfer learning, cross-domain sentiment classification, pre-trained language model.

I. INTRODUCTION

With the user sentiment and opinion expressions becoming widespread throughout social and e-commerce platforms, correctly understanding these thoughts and views becomes important in facilitating various downstream applications [1]. Sentiment classification, an important task of Natural Language Processing (NLP), aims to identify the emotional tendencies (positive or negative) of given text input [2] and has attracted great research attention in recent years.

Deep neural networks have been successfully applied for diverse machine learning problems, including various NLP tasks, with greatly improved prediction performance metrics. The standard model training for a NLP task had focused on initializing the first layer of a neural network with pretrained word vectors such as word2vec [3] and GloVe [4], and the rest of the network is trained on the task-specific data with convolutional and/or recurrent neural networks. Convolutional neural networks (CNN) [5] are able to learn the local response from the temporal or spatial data but lack the ability to learn sequential correlations. Recurrent Neural Networks (RNN) [6] are used because of their sequence modeling capabilities and dealing with short-term dependencies in a sequence of data, but have trouble when dealing with long-term dependencies. Long Short-Term Memory networks (LSTM) [7], which is a variation of RNN architecture, aims to solve the long-term dependency problem by introducing a memory into the network. RNN-based deep learning architectures has been the standard for various NLP tasks, including sentiment classification. However, these approaches still processed context in one direction only, i.e., create dependencies only on the left or right side of the

The associate editor coordinating the review of this manuscript and approving it for publication was Ah Hwee Tan.

current word. Therefore they cannot capture contexts in both directions at the same time, i.e., consider words on both sides of the current word when capturing dependencies.

Most of these performance improvements in NLP with deep neural networks come only via supervised learning with massive amounts of labeled data. However, in real world applications, there are many scenarios where it is difficult to collect sufficient data for high-performing supervised learning model of a specific task due to factors of scarcity of readily available data or the high expense of data collection. In addition, statistical classifiers assume that both the training and test data come from a common underlying distribution [8], but due to the high variability and sparsity of natural language, oftentimes there is distribution differences in the real world data and the specialized training data [9].

Transfer learning allows us to deal with this scenario by borrowing information from a relevant source domain with abundant labeled data to help improve the prediction performance in the target domain [10]. *Cross-domain sentiment classification* (CDSC) aims at leveraging knowledge obtained from a source domain to train a high-performance learner for sentiment classification on a target domain, e.g., book product review, to help classification in the target domain, e.g., electronics product review, with few or no labeled data. In the literature, transfer learning techniques have been applied to CDSC. Traditional pivot-based CDSC schemes in [11], [12] attempt to infer the correlation between pivot words, i.e., the domain-shared sentiment words, and non-pivot words, i.e., the domain-specific sentiment words, by utilizing multiple pivot prediction tasks. However, these schemes share a major limitation that manual selection of pivots is required.

All of the above discussed schemes need to train a dedicated NLP model from scratch for every new task with its own specialized training data, which could take days and weeks to converge to a stable, high-performance model. Alternatively, substantial work has shown that unsupervised pre-trained language models on large text corpus are beneficial for text classification and other NLP tasks, which can avoid training a new model from scratch. Various approaches are proposed for training general purpose language representation models using an enormous amount of unannotated text, such as ELMo [13] and GPT [14]. Pre-trained models can be fine-tuned on NLP tasks without requiring huge amount of labeled data and have achieved significant improvement over training on task-specific annotated data. More recently, a pre-training technique, Bidirectional Encoder Representations from Transformers (BERT) [15], is proposed and has created state-of-the-art models for a wide variety of NLP tasks,including question answering (SQuAD v1.1), natural language inference, text classification and others. The latest of such pre-trained language models is XLNet [16], a generalized autoregressive pretraining method that enables learning bidirectional contexts by maximizing the expected likelihood over all permutations of the factorization order, and overcomes the limitations of BERT thanks to its autoregressive formulation. Furthermore, XLNet integrates ideas from Transformer-XL [17], the state-of-the-art autoregressive model, into pretraining. In this paper, we fine-tune BERT and XLNet for CDSC and compare them with the current state-of-the-art methods. We also closely study their performances in comparison to each other with various experimental settings.

Our main contributions are summarized as follows:

- This is the first work to explore the usage of Transformer-based bidirectional contextualized language models for CDSC.
- Compare and comprehensively analyze the performance of the two highest performing Transformer language models of XLNet and BERT in the context of CDSC.
- Achieves new state-of-the-arts results with significant improvements over the previous approaches.

II. RELATED WORKS

Over the last decade, many methods have been proposed for cross-domain sentiment classification. Structural Correspondence Learning (SCL) method is proposed by Blitzer *et al.* [11] to learn a joint low-dimensional feature representation for the source and target domains. Similarly, Pan *et al.* [18] propose a Spectral Feature Alignment (SFA) method to align the pivots with the non-pivots to build a bridge between the source and target domains. However, these methods need to manually select the pivots based on criterions such as the frequency in both domains, the mutual information between features and labels on the source domain data, and the mutual information between features and domains [18]. Domain-Adversarial training of Neural Networks (DANN) is proposed by Ganin *et al.* [19] for domain adaptation using a gradient reversal layer to reverse the gradient direction in order to produce representations such that a domain classifier cannot predict the domain of the encoded representation, and at the same time, a sentiment classifier is built on the representation shared by domains to reduce the domain discrepancy and achieves better performance for cross-domain sentiment classification. Proposed approaches by Sun *et al.* [20], and Zellinger *et al.* [21] focus on learning domain invariant features whose distribution is similar in source and target domain. They attempt to minimize the discrepancy between domain-specific latent feature representations. However, all the domain alignment approaches can only reduce, but not remove, the domain discrepancy. Therefore, the target samples distributed near the edge of the clusters, or far from their corresponding class centers are most likely to be misclassified by the hyperplane learned from the source domain [22].

Transfer learning has been successfully applied in computer vision where lower network layers are trained on high-resource supervised datasets like ImageNet to learn generic features [5], and are then fine-tuned on target tasks, leading to impressive results for image classification and object detection [23], [24]. Following the successful practice of pre-trained models for computer vision tasks, high-level

contextualized language models pre-trained on unlabeled large text corpus and fine-tuned for a given specific task have recently been proposed in NLP with great results. Howard and Ruder [25] proposed ULMFiT, the first to propose fine-tuning with pre-trained language model, showcasing the effectiveness of discriminative fine-tuning, and gradual unfreezing for retaining prior knowledge and circumventing catastrophic forgetting during fine-tuning. There are two existing strategies for applying pre-trained language representations to downstream tasks: feature-based and fine-tuning. The feature-based approach, such as ELMo proposed by Peters *et al.* [13], uses tasks-specific architectures that include the pre-trained representations as additional features. Many fine-tuning approaches, such as the Generative Pre-trained Transformer (OpenAI GPT) proposed by Radford *et al.* [14] and the Bidirectional Encoder Representations from Transformers (BERT) proposed by Devlin *et al.* [15] introduce minimal task-specific parameters, and are trained on the downstream tasks by simply fine-tuning the pre-trained parameters. Among the unsupervised pre-training methods for language models in the literature, the two most successful pretraining objectives are autoregressive (AR) language modeling that seeks to estimate the probability distribution of a text corpus with an autoregressive model [13], [14], and autoencoding (AE) language modeling that aims to reconstruct the original data from corrupted input [15]. Yang *et al.* [16] proposed the XLNet, a combination of AR and AE language modeling where it can capture dependencies beyond the input sequence limit and process bidirectional contexts at the same time .

III. PROBLEM DESCRIPTION AND NOTATIONS

We use the most common notations for transfer learning as defined in [18] applied on cross-domain sentiment classification. Transfer learning comprises of two main concepts: a domain and a task. A domain $\mathcal{D}$ consists of a feature space $\mathcal{X}$, and a marginal probability distribution $P(X)$ over the feature space, where $X = \{x_1, \ldots, x_n\} \in \mathcal{X}$. For a binary bag-of-words representation of an input text document, the feature space $\mathcal{X}$ would be the set of all possible binary term vectors, x_i is the i-th term vector corresponding to input and X is the random variable associated with sampling input documents. Given a domain $\mathcal{D} = \{\mathcal{X}, P(X)\}$, a task $\mathcal{T}$ defines a label space $\mathcal{Y}$ and a conditional probability distribution $P(Y|X)$ that is learned from the training data pairs of $x_i \in X$ and $y_i \in \mathcal{Y}$. In the context of binary sentiment classification task, $\mathcal{Y}$ is the set of all possible labels $\{1, 0\}$ representing *positive* and *negative* sentiments, y_i has value of either 1 or 0, and Y is the random variable associated with input document's label.

Given a source domain $\mathcal{D}_S$ with its task $\mathcal{T}_S$ and a target domain $\mathcal{D}_T$ with its task $\mathcal{T}_T$, the objective of cross-domain sentiment classification is to learn the conditional probability distribution $P_T(Y_T|X_T)$ in $\mathcal{D}_T$ by utilizing the knowledge learned from $\mathcal{D}_S$ and $\mathcal{T}_S$, where $\mathcal{D}_S \neq \mathcal{D}_T$ and sufficient labeled training data are available in $\mathcal{D}_S$. Typically either only few or no labeled data are available in the target domain $\mathcal{D}_T$. In cross-domain sentiment classification, the marginal probability distributions in source and target domains are different $P_S(X_S) \neq P_T(X_T)$, i.e. the text documents in these domains discuss different topics. The task of the cross-domain sentiment classification is to learn a robust classifier $P_S(Y_S|X_S)$ trained on labeled data in the source domain to predict the polarity of unlabeled examples from the target domain using the learned classifier $P_S(Y_T|X_T)$, where $Y_T = Y_S$, i.e., both domains have the same label space.

IV. BIDIRECTIONAL TRANSFORMER LANGUAGE MODELS

A. TRANSFORMER

Before the introduction of Transformers, previous state-of-the-art sequence modeling approaches in NLP relied mostly on recurrent neural networks (RNN), such as Long Short-Term Memory (LSTM) [7] and gated RNN [26]. However, the recurrent models' inherent sequential nature stymies parallelization during training and limits its ability to contextualize longer input sequences. Attention mechanisms have become an integral part of compelling sequence modeling and transduction models in various tasks, allowing modeling of dependencies without regard to their distance in the input or output sequences [27].

The Transformer [28] is first introduced to improve the speed of training models for neural machine translations using the attention mechanism. Its architecture reduces sequential computation with multiple self-attention heads. In order to compute a representation of an input sequence, self-attention mechanism associates different positions of the sequence. Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions. The original Transformer has encoder-decoder structure, with the encoder mapping an input sequence to a sequence of continuous representations, which is used by the decoder to generate an output sequence one element at a time. Each of the encoder and the decoder consists of 6 identical layers, with each containing two sub-layers of 8 parallel self-attention heads and a fully connected feed-forward neural network.

The input representation to the first encoder layer is a concatenation of WordPiece embeddings [29] and positional embeddings generated from the input sequence. An attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key. Specifically, given an embedded vector x for an input sequence, we create a Query, Key, and Value vector for each input embedding token by multiplying the embedding by three learned matrices W^Q, W^K, W^V respectively. For parallel computation, we stack the Query, Key and Value vectors into matrices Q, K, V. Then the self-attention function is given by:

$$Attention(x) = Attention(Q, K, V) = softmax(\frac{QK^\top}{\sqrt{d_k}})V, \quad (1)$$

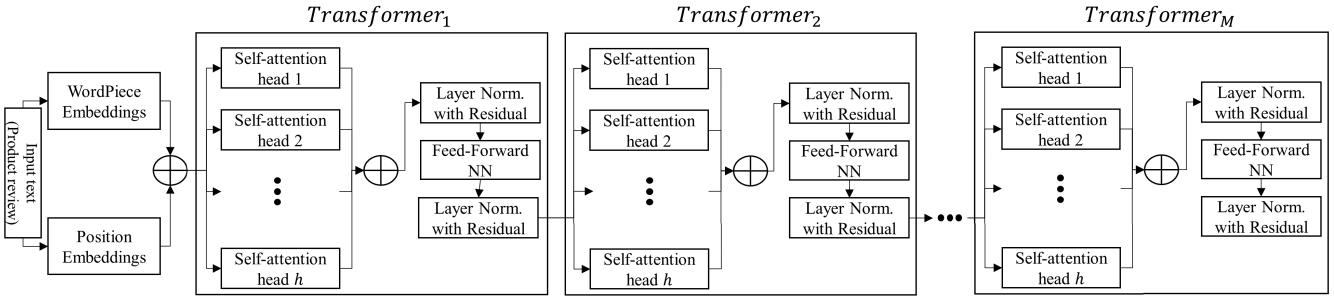

FIGURE 1. The encoder layers of the standard transformer architecture. Each encoder layer's output is passed as the input to the next layer.

where d_k is the dimension of queries and keys. The Transformer performs such self-attention function in parallel with multiple attention heads by projecting the queries, keys and values h times with different, learned linear projections to d_k, d_k and d_v dimensions, respectively. Attention function is performed in parallel on each of these projected versions of queries, keys and values, resulting d_v-dimensional output values.

$$\begin{aligned} MultiHead(x) &= MultiHead(Q, K, V) \\ &= Concat(head_1, \ldots, head_h)W^O, \end{aligned} \quad (2)$$

where $head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$, $Concat$ is the concatenation function, the projections are parameter matrices $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{model} \times d_v}$ and $W^O \in \mathbb{R}^{hd_v \times d_{model}}$ with $d_{model} = d_k h$.

Each Transformer layer consists of two sub-layers. The first sub-layer is the multi-head attention and its normalized output is fed to the second sub-layer of fully connected feed forward network. The activation function for the feed forward networks is ReLU. Formally, the hidden states of Transformer with M number of Transformer layers are calculated as follows:

$$Tr_m(x) = norm(Att(x) + FFN(att(x))), \quad (3)$$

where

$$\begin{aligned} Att(x) &= norm(x + MultiHead(x)) \\ FFN(x) &= max(0, xW_1 + b_1)W_2 + b_2, \end{aligned}$$

where $norm$ is the normalization function with linear connection following [30], FFN is fully connected feed forward network, W_1 and W_2 are the weights of the first and second fully connected networks with b_1, b_2 as bias values, and $m \in M$. These fully connected networks have separate weight parameters for each encoder layer. Each encoder layer passes its output as an input to the next encoder layer, with the final encoder layer producing the final encoded representation for fine-tuning. Fig. 1 shows the architecture of the Transformer's layers. In the original Transformer [28], the layer size M is 6 and the multi-head h is 8.

B. BERT

BERT, which stands for Bidirectional Encoder Representations from Transformers, is built upon recent works in pre-training contextual representations such as ELMo [13], and ULMFiT [25], but these models are either unidirectional or shallowly bidirectional, meaning contextualized representation of a word only considers the words to its left or to its right. BERT, on the other hand, has deeply bidirectional contextualization that combines the representations of both left-context and right-context models. Its model architecture is a multi-layer bidirectional Transformer encoder based on the original Transformer model proposed in [28]. The BERT model retains only the encoder part of the original model, without any decoder. It has 12 identical encoder layers, with each having two sub-layers of 12 parallel attention head and also a fully connected feed-forward network.

For pre-training, unlike ELMo [13] and OpenAI GPT [14] that use left-to-right or right-to-left language models, BERT uses two unsupervised prediction tasks. First is next sentence prediction task, where two sentences (*A*,*B*) are selected from the text corpus and a classifier is trained to predict whether *B* actually follows *A*. 50% of the time *B* is the actual next sentence that follows *A*, and 50% of the time it is a random sentence from the corpus. The second task is the Masked Language Model task, where they mask some percentage of the input tokens at random, and then predict only those masked tokens. Specifically, given a text sequence $x = [x_1, \ldots, x_T]$, BERT first constructs a corrupted version $\hat{x}$ by randomly setting a 15% of tokens in x to a special symbol [MASK]. If denote the masked tokens as $\bar{x}$, then the training objective is to reconstruct $\bar{x}$ from $\hat{x}$:

$$\begin{aligned} \max_\theta \log p_\theta(\bar{x}|\hat{x}) &\approx \sum_{t=1}^{T} m_t \log p_\theta(x_t|\hat{x}) \\ &= \sum_{t=1}^{T} m_t \log \frac{\exp(H_\theta(\hat{x})_t^\top e(x_t))}{\sum_{x'} \exp(H_\theta(\hat{x})_t^\top e(x'))}, \end{aligned} \quad (4)$$

where $m_t = 1$ indicates token x_t is masked, $e(x)$ denotes the embedding of x and H_θ is a Transformer that maps a length-T text sequence x into a sequence of hidden vectors $H_\theta(x) = [H_\theta(x)_1, H_\theta(x)_2, \ldots, H_\theta(x)_T]$. Note that the $\approx$ sign in (4) indicates that when calculating $p_\theta(\bar{x}|\hat{x})$, BERT makes an independence assumption that all masked tokens $\bar{x}$ are separately constructed. The biggest advantage of this training objective is it allows the model simultaneous access to the

contextual information on both sides of a token. BERT is the first fine-tuning based representation model that achieves state-of-the-art performance on a large suite of sentence-level and token-level tasks, outperforming many systems with task-specific architectures and advances the state-of-the-art for eleven NLP tasks [15].

C. XLNET

BERT has achieved strong performances across multiple tasks but it had the following major flaws:

- The original Transformer architecture can capture context within the specified maximum input sequence length. If a document is longer than the specified length, it would be divided into segments with each of them being processed by the model independently from scratch without any connection between them.
- BERT is trained to predict tokens replaced with the [MASK] symbol. However, this [MASK] token never appears in downstream tasks, which creates a discrepancy between pre-training and fine-tuning.
- BERT makes predictions for the masked tokens with assumption that there is no dependencies between these masked tokens, which is bit over-simplification and can cause reduced number of dependencies that BERT can learn at once.

XLNet [16] solves BERT's first flaw of input length context constraint with the architecture of Transformer-XL [17], which itself is a modification upon the original Transformer [28]. Transformer-XL introduces *Recurrence Mechanism* and *Relative Positional Encoding* to the Transformer architecture to capture long-term dependencies for documents that are longer than the maximum allowed input length. With *Recurrence Mechanism*, the hidden state sequence computed for the previous segment is fixed and cached to be reused as an extended context when the model processes the next new segment. Although the gradient still remains within a segment, this additional input allows the network to exploit information in the history, leading to an ability of modeling longer-term dependency and avoiding context fragmentation. *Relative Positional Encoding* encodes position of a context in relative distance from the current token at each attention module, as opposed to encoding position statically only at the beginning like in BERT. This is done so to accommodate the Recurrence Mechanism and avoid having tokens from different segments having the same positional encoding.

Despite its ability to capture long-term dependencies, Transformer-XL still only holds unidirectional context, i.e., predicts the current token based on the given sequential context on its left or its right side only. XLNet solves the issue of unidirectional context, without using [MASK] symbol as in BERT, by introducing a language modeling objective called *Permutation language modeling* that predicts a current token based on the given preceding context just like traditional language model. However, instead of predicting tokens in sequential order, tokens are predicted following a random permutation order. One problem with this objective is the computational high expense and slow convergence if we to go through every permutation. Hence to reduce the optimization difficulty, only the last tokens in a factorization order is chosen for training. Formally, let Z_T be the set of all possible permutations of the length-T index sequence $[1, 2, \ldots, T]$ with z_t and $z_{<t}$ denoting the t-th element and the first $t-1$ elements of a permutation $z \in Z_T$. To choose the tokens in a factorization order, z is split into a non-target subsequence $z_{\leq c}$ and a target subsequence $z_{>c}$, where c is the cutting point. Then the permutation language modeling objective is to maximize the log-likelihood of the target subsequence conditioned on the non-target subsequence as follows:

$$\begin{aligned}\max_{\theta} \mathbb{E}_{z\sim Z_t}[\log p_\theta(x_{z_{>c}}|x_{z_{\leq c}})] \\ = \mathbb{E}_{z\sim Z_t}[\sum_{t=c+1}^{|z|} \log p_\theta(x_{z_t}|x_{z_{<t}})] \\ = \mathbb{E}_{z\sim Z_t}[\sum_{t=c+1}^{|z|} \log \frac{exp(e(x)^\top g_\theta(x_{z_{<t}}, z_t))}{\sum_{x'} \exp(e(x')^\top g_\theta(x_{z_{<t}}, z_t))}, \quad (5)\end{aligned}$$

where $e(x)$ denotes the embedding of x input sequence, $g_\theta(x_{z_{<t}}, z_t)$ denotes a new type of representations which additionally take the target position z_t as input. To compute $g_\theta(x_{z_{<t}}, z_t)$, XLNet introduces a scheme called *Two-Stream Self-Attention* that uses two sets of hidden representations:

- The content stream $h_\theta(x_{z_{\leq t}})$, or abbreviated as h_{z_t}, is same as the hidden states in the original Transformer. This representation encodes both the context and x_{z_t}.
- The query stream $g_\theta(x_{z_{<t}}, z_t)$, or abbreviated as g_{z_t}, only has the contextual information x_{z_t} and the position z_t, without any knowledge of the content x_{z_t}.

The language model is trained to predict each token in the sentence using only the query stream. The content stream is used as input to the query stream. During fine-tuning, the query stream is thrown away and the input data is represented with the content stream. Formally, for each self-attention layer $m = 1, 2, \ldots, M$, the two streams of representations are updated with shared set of parameters as follows:

$$\begin{aligned} g_{z_t}^m &\leftarrow Attention(Q = g_{z_t}^{m-1}, KV = h_{z_{<t}}^{m-1}; \theta), \\ h_{z_t}^m &\leftarrow Attention(Q = h_{z_t}^{m-1}, KV = h_{z_{\leq t}}^{m-1}; \theta), \end{aligned}$$

where Q, K, V denote the query, key, value in an attention operation. The update rule of the content stream is same as the original Transformer self-attention. The query representation in the last layer $g_{z_t}^M$ is used to compute (5).

D. FINE-TUNING FOR CDSC

Given a source domain $\mathcal{D}_S$ with its task $\mathcal{T}_S$ and a target domain $\mathcal{D}_T$ with its task $\mathcal{T}_T$, the objective of CDSC is to learn the conditional probability distribution $P_S(Y_S|X_S)$ in $\mathcal{D}_S$ and then apply the learned distribution model on the target source domain $\mathcal{D}_T$ and with its task $\mathcal{T}_T$, where $\mathcal{D}_S \neq \mathcal{D}_T$. We have

sufficient labeled training data available in $\mathcal{D}_S$ but no labeled data in the target domain $\mathcal{D}_T$.

We shall fine-tune the pre-trained Transformer models, BERT and XLNet, with a labeled sentiment data from a selected source domain and measure its performance in predicting the sentiment polarity of other domain's sentiment data. To measure and compare the effectiveness of BERT and XLNet for cross-domain sentiment classification, on top of the pre-trained models we will only add one fully connected feed-forward network that consists of two linear transformations with GELU activation [31] in between. Given source domain labeled data X_S, we calculate the probability distributions of input sequences using a softmax activation function.

$$\begin{aligned} f(x_i) &= GELU(Tr_M(x_i)W_1 + b_1)W_2 + b_2 \\ p(y_i|x_i) &= \frac{e^{f(x_i)}}{\sum_j e^{f(x_j)}}, \end{aligned} \tag{6}$$

where $Tr_M(x_i)$ is the output from the last Transformer layer M of either BERT or XLNet for the input sequence $x_i \in X_S$. W_1 and W_2 are the weights of the first and second linear transformations with b_1, b_2 as bias values. The cost function to minimize is the cross-entropy loss as follows:

$$\mathcal{L} = -\sum_{i=1}^{N} (y_i \log p(y_i|x_i) + (1 - y_i) \log(1 - p(y_i|x_i)), \tag{7}$$

where N is the total number of samples in the current batch, y_i is the given label of the input sequence (1 for positive review and 0 for negative review) and $p(y_i|x_i)$ is the probability of the input sequence being positive.

After fine-tune training, we apply the learned models on the target domain and predict the sentiment binary values using softmax function $p(y_i|x_i)$ with the trained parameters where $x_i \in X_T$.

V. EXPERIMENTATION

A. DATASET

Our experiments are conducted on the Amazon reviews dataset [11] that has been widely used in the literature for cross-domain sentiment classification. The dataset contains reviews from five product types (i.e. domains): Books, DVD, Electronics, Kitchen and Video. There are 6000 labeled review data for each domain with 3000 positive reviews (higher than 3 stars) and 3000 negative reviews (lower than 3 stars). Following the convention in [18], we construct 20 cross-domain sentiment classification tasks. We fine-tune on the pre-trained BERT-Large [32] and XLNet-Large [33] language models with differing number of labeled data from the selected source domain and test the trained models on the other domain data.

B. PRE-TRAINING

For our experiment we use the latest pre-trained cased BERT-Large model, refered to simply as BERT henceforth, with new pre-processing technique called Whole Word Masking where all of the tokens corresponding to a word are masked at once, instead of masking those tokens belonging to a word individually. It has 24 Transformer layers with 4096 hidden dimensions, 16 attention heads and a total of 340M parameters. For the pre-training, BERT uses the concatenation of BookCorpus (800M words) [34] and English Wikipedia (2,500M words) as pre-training data. BERT is pre-trained with batch size of 256 sequences with each sequence containing maximum of 512 tokens for 1,000,000 steps, which is approximately 40 epochs over the 3.3 billions word corpus.

For pre-training data, in addition to the BookCorpus and English Wikipedia datasets, cased XLNet-Large model, refered to simply as XLNet henceforth, uses Giga5 (16GB text) [35], ClueWeb 2012-B [36] and Common Crawl [37] as part of its pre-training data. ClueWeb2012-B and Common Crawl articles are filtered out and after tokenization wih SentencePiece [38], the total pre-training data for XLNet amounts to 32.89B subword pieces, which is an order of magnitude greater than the pre-training data used for BERT. XLNet's architecture has, similar to BERT, 24 Transformer layers with 4096 hidden dimensions and 16 attention heads. XLNet is pre-trained with batch size of 2048 and sequence length of 512 for 500,000 steps.

C. IMPLEMENTATION DETAILS

For fine-tune training of the language models, the hidden dimensions of the fully connected networks following the last layer of the Transformers is 1024. for cross-domain sentiment classification. The dropout probability is kept at 0.1. For the input, the maximum sequence length is set to 256 with batch size of 32. The learning rate is 2e-5 and optimization is done with Adam optimizer. Training and testing of TensorFlow implementations of BERT [32] and XLNet [33] are performed separately on a single Google Cloud TPU v2 and the total experiment time was over 400 hours for each TPU.

For comparison with other state-of-the-arts CDSC methods, the BERT and XLNet models are trained on 6000 labeled data from a source domain for 3000 steps and evaluate the prediction accuracy on all 6000 data of the remaining domains.

In addition, to show BERT and XLNet's effectiveness in low resource transfer learning scenarios, we train the models on different amount of source domain labeled data and test each trained model on all of the other domains. We compare the runtimes of these two language models in the same configuration scenarios with varying number of steps for the training phase and also with different number of samples for the testing phase.

D. PERFORMANCE COMPARISON

The baseline methods included in the comparison are following:

- **DAmSDA** [19]: an adversarial network based domain adaptation method that utilizes representations encoded in a 30,000-dimensional feature vector.

TABLE 1. The CDSC accuracy of the state-of-the-arts methods on the Amazon reviews dataset.

Source	Target	DAmSDA	CNN-aux	AMN	HATN	HANP	**BERT**	**XLNet**
Books	DVD	86.12%	84.42%	85.62%	87.07%	88.12%	92.49%	**95.10%**
	Electronics	79.02%	80.63%	80.55%	85.75%	85.81%	93.13%	**95.92%**
	Kitchen	81.05%	83.38%	81.88%	87.03%	88.91%	94.08%	**96.54%**
	Video	84.98%	84.43%	87.25%	87.80%	89.21%	91.75%	**94.54%**
DVD	Books	85.17%	83.07%	84.53%	87.78%	89.18%	93.67%	**95.68%**
	Electronics	76.17%	80.35%	80.42%	86.32%	86.87%	93.25%	**95.17%**
	Kitchen	82.60%	81.68%	81.67%	87.47%	88.54%	94.15%	**96.42%**
	Video	83.80%	85.87%	87.40%	89.12%	91.25%	93.88%	**95.82%**
Electronics	Books	79.92%	77.38%	77.52%	84.03%	85.67%	91.83%	**93.56%**
	DVD	82.63%	79.07%	80.53%	84.32%	85.29%	89.93%	**91.99%**
	Kitchen	85.80%	87.15%	87.83%	90.08%	91.08%	95.37%	**96.79%**
	Video	81.70%	78.78%	82.12%	84.18%	85.96%	89.33%	**91.79%**
Kitchen	Books	80.55%	78.47%	79.05%	84.88%	85.04%	91.74%	**95.29%**
	DVD	82.18%	79.07%	79.50%	84.72%	86.47%	90.34%	**94.44%**
	Electronics	88.00%	86.73%	86.68%	89.33%	90.43%	94.82%	**96.46%**
	Video	81.47%	78.82%	82.15%	84.85%	85.93%	89.82%	**94.31%**
Video	Books	83.00%	81.48%	83.50%	87.10%	88.94%	93.05%	**95.31%**
	DVD	85.90%	85.25%	86.88%	87.90%	88.54%	93.32%	**95.60%**
	Electronics	77.67%	82.32%	79.68%	85.98%	86.11%	92.87%	**95.71%**
	Kitchen	79.52%	81.28%	80.98%	86.45%	87.21%	93.35%	**96.11%**
Average		82.36%	81.98%	82.79%	86.61%	87.76%	92.61%	**95.13%**

- **CNN-aux** [12]: a CNN model based on the approach proposed by Kim *et al.* [39]. It jointly trains the cross-domain sentence embedding and the sentiment classifier.
- **AMN** [40]: an adversarial network based method that learns domain-shared representations based on memory networks and adversarial training.
- **HATN** [41]: an attention network with hierarchical positional encoding that focuses on both the word and sentence level sentiments.
- **HANP** [42]: a hierarchical attention network than can obtain both domain independent and domain specific features at the same time by adding prior knowledge.
- **BERT**: the proposed fine-tuned auto-encoding bidirectional contextualized language model pre-trained on Masked language modeling and the Next sentence prediction tasks. Its architecture is based on the standard Transformer model.
- **XLNet**: the proposed fine-tuned auto-regressive bidirectional contextualized language model pre-trained on Permutation language modeling task. Its architecture is based on Transformer-XL model and has two-stream self-attention mechanism.

We use classification accuracy as our performance metrics, which is defined as follows:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}.$$

Table 1 shows the classification accuracy of various state-of-the-arts methods in comparison to the bidirectional contextualized language models on the cross-domain sentiment classification task. For BERT and XLNet, we report the mean accuracy rate from 10 separate runs using all of the 6000 labeled data available in the source domain. It can be observed that the bidirectional contextualized Transformer language models of BERT and XLNet greatly outperforms the previous state-of-the-arts methods. BERT outperforms previous state-of-the-arts methods by at least 2% accuracy. However, XLNet produces results that further improves the CDSC accuracy by 2.5% in comparison to BERT. XLNet is the only method where all of the prediction accuracy rates are well above above 90%.

The most interesting results are observed in Fig. 2 and Table 2. For BERT and XLNet, we report the mean bootstrapped results from predicting four target domain data with 95% confidence interval from 40 observations where source domain labeled data are selected randomly with replacement. BERT outperforms the previous SOTA methods using around 300 samples or around 20 times less data. XLNet outperforms previous state-of-the-arts methods after fine-tuning only with 50 source domain training samples, i.e., around 120 times less data than the previous SOTA methods. These results proves that pre-trained Transformer language models are very adaptive at capturing context with only few samples and are highly suitable for transfer learning. Also it can be observed that XLNet is much more efficient at capturing contextualized

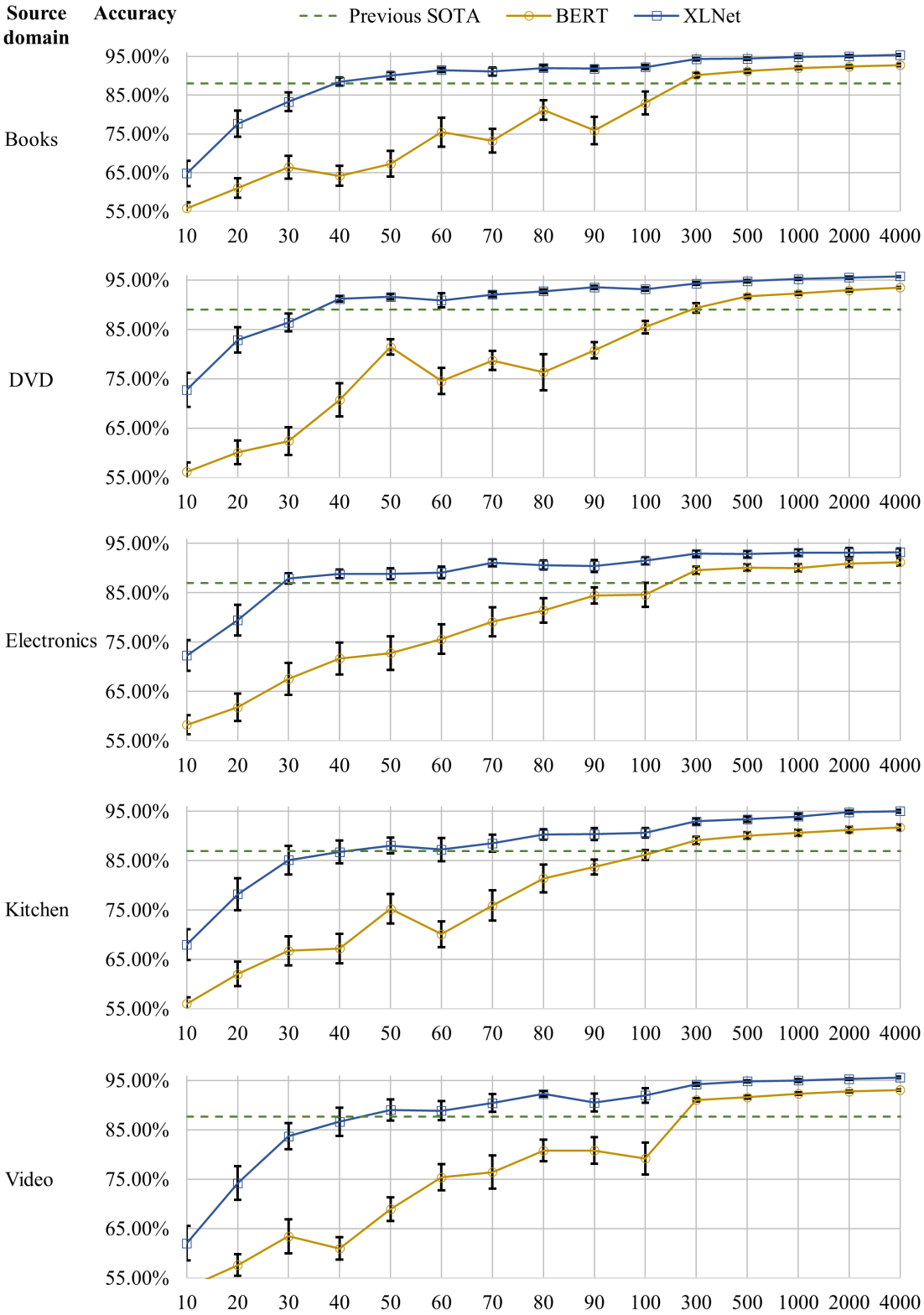

FIGURE 2. CDSC accuracy of BERT and XLNet on Amazon review dataset, fine-tuned on different amounts of labeled data from the source domain and tested on all available labeled data of the target domain.

representations than BERT that it can fine-tune its pre-trained parameters to very quickly pivot towards capturing sentiment polarity in the given sequences. This higher efficiency performance is due to the combination of different pre-training objective function, ability to capture dependencies longer than the sequence length and the larger

TABLE 2. CDSC accuracy rates of BERT and XLNet on Amazon review dataset with the corresponding margins of error.

Previous SOTA | BERT | XLNet

Source domain training samples amount	Source domain: Books	DVD	Electronics	Kitchen	Video
10	55.85% (1.50%)	56.15% (1.91%)	58.27% (1.95%)	56.04% (1.33%)	52.95% (0.95%)
	64.81% (3.25%)	72.74% (3.48%)	72.30% (3.13%)	68.00% (3.09%)	62.03% (3.47%)
20	61.08% (2.51%)	60.09% (2.39%)	61.83% (2.76%)	62.06% (2.45%)	57.61% (2.18%)
	77.65% (3.39%)	82.85% (2.54%)	79.45% (3.11%)	78.21% (3.22%)	74.22% (3.38%)
30	66.41% (2.94%)	62.37% (2.80%)	67.57% (3.25%)	66.75% (2.94%)	63.43% (3.47%)
	83.30% (2.42%)	86.38% (1.79%)	87.90% (1.08%)	85.12% (2.91%)	83.70% (2.63%)
40	64.17% (2.56%)	70.72% (3.38%)	71.71% (3.24%)	67.23% (2.98%)	60.99% (2.31%)
	88.49% (1.04%)	91.14% (0.55%)	88.80% (0.89%)	86.75% (2.32%)	86.64% (2.85%)
50	67.29% (3.28%)	81.41% (1.54%)	72.78% (3.41%)	75.26% (3.00%)	68.97% (2.40%)
	90.02% (0.93%)	91.60% (0.57%)	88.86% (1.10%)	88.08% (1.58%)	89.02% (2.14%)
60	75.48% (3.73%)	74.54% (2.62%)	75.61% (2.97%)	70.11% (2.60%)	75.38% (2.64%)
	91.44% (0.54%)	90.88% (1.44%)	89.08% (1.17%)	87.26% (2.37%)	88.88% (1.96%)
70	73.24% (3.06%)	78.69% (1.95%)	79.13% (2.94%)	75.93% (3.04%)	76.44% (3.35%)
	91.07% (1.08%)	92.04% (0.50%)	91.07% (0.73%)	88.50% (1.72%)	90.43% (1.81%)
80	81.16% (2.54%)	76.35% (3.65%)	81.43% (2.50%)	81.41% (2.84%)	80.79% (2.18%)
	92.01% (0.65%)	92.71% (0.40%)	90.60% (0.92%)	90.30% (1.03%)	92.29% (0.55%)
90	75.88% (3.55%)	80.74% (1.65%)	84.47% (1.62%)	83.74% (1.52%)	80.80% (2.69%)
	91.85% (0.56%)	93.52% (0.35%)	90.42% (1.21%)	90.38% (1.23%)	90.52% (1.82%)
100	82.98% (2.97%)	85.43% (1.24%)	84.57% (2.42%)	86.15% (1.02%)	79.20% (3.22%)
	92.24% (0.55%)	93.15% (0.34%)	91.48% (0.76%)	90.64% (0.93%)	91.97% (1.49%)
300	90.17% (0.61%)	89.30% (0.97%)	89.54% (0.77%)	89.10% (0.75%)	91.06% (0.33%)
	94.28% (0.28%)	94.28% (0.29%)	92.90% (0.63%)	92.98% (0.58%)	94.20% (0.30%)
500	91.18% (0.43%)	91.66% (0.33%)	90.09% (0.67%)	90.09% (0.65%)	91.60% (0.30%)
	94.41% (0.33%)	94.79% (0.25%)	92.81% (0.69%)	93.40% (0.54%)	94.84% (0.22%)
1000	92.02% (0.34%)	92.26% (0.30%)	90.03% (0.79%)	90.64% (0.65%)	92.31% (0.25%)
	94.83% (0.27%)	95.19% (0.20%)	93.14% (0.69%)	93.95% (0.48%)	95.00% (0.28%)
2000	92.37% (0.33%)	92.91% (0.24%)	90.93% (0.74%)	91.27% (0.57%)	92.77% (0.16%)
	95.05% (0.27%)	95.48% (0.22%)	93.13% (0.90%)	94.84% (0.29%)	95.31% (0.16%)
4000	92.73% (0.32%)	93.47% (0.12%)	91.19% (0.77%)	91.76% (0.58%)	93.05% (0.11%)
	95.35% (0.24%)	95.76% (0.15%)	93.20% (0.76%)	95.04% (0.30%)	95.61% (0.13%)
6000	92.86% (0.32%)	93.73% (0.13%)	91.62% (0.74%)	91.68% (0.61%)	93.15% (0.11%)
	95.52% (0.26%)	95.77% (0.17%)	93.53% (0.68%)	95.13% (0.29%)	95.68% (0.11%)
	88.01%	88.96%	87.00%	86.97%	87.70%

pre-training datasets. Table 2 shows the CDSC accuracy rates with the corresponding margins of error.

In Table 3 and In Table 4, we compare the runtimes of the two models during fine-tune training and testing. The reported results are the mean duration times from 10 separate runs for each training step size and test data size. The test data are identical for both models and are randomly selected with replacement. We can see that XLNet

TABLE 3. Comparison of running time during fine-tuning.

Training steps	BERT	XLNet
300	458	499 (+9%)
1000	627	713 (+14%)
3000	1140	1355 (+19%)
9000	2605	3259 (+25%)
30000	7735	9908 (+28%)

TABLE 4. Comparison of running time during testing.

Test data size	BERT	XLNet
20	85	77 (-9%)
50	87	78 (-11%)
100	84	79 (-6%)
200	85	78 (-8%)
500	86	79 (-8%)
1000	89	81 (-10%)
2000	91	83 (-9%)
4000	99	86 (-13%)
6000	110	90 (-18%)

is more efficient than BERT during testing, on average around 10% less time spent on testing. However, XLNet has shown to be much more resource-hungry when it comes to training. In our case where the main SOTA results are reported from 3000 training steps, XLNet is almost 20% slower than BERT. XLNet's runtime is higher than BERT in training due to its segment recurrence mechanism for capturing context dependencies in documents longer than the maximum input sequence length. However, during testing, this segment recurrence mechanism actually decreases the runtime for XLNet to be less than BERT's because the representations from the previous segments can be reused instead of being computed from scratch as in the case of the standard Transformer.

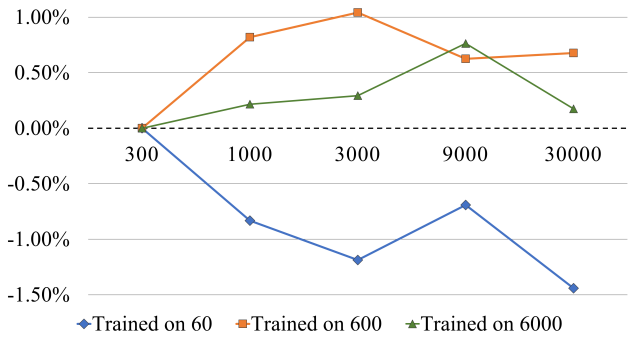

FIGURE 3. BERT's accuracy rate fluctuation over longer fine-tune training steps.

In Fig. 3 and 4, we evaluate the effect of different number of training steps (300, 1000, 3000, 9000, 30000) on the CDSC accuracy rate. BERT and XLNet models are fine-tuned on varying amounts of labeled data (60, 600, 6000) from a source domain ('Books') and tested on all 6000 data of a target domain ('Video'). The results are the mean accuracy rate change over 10 separate runs for each step size and fine-tune training data size. We observe that in general and at least in the context of CDSC, there is a noticeable trade-off

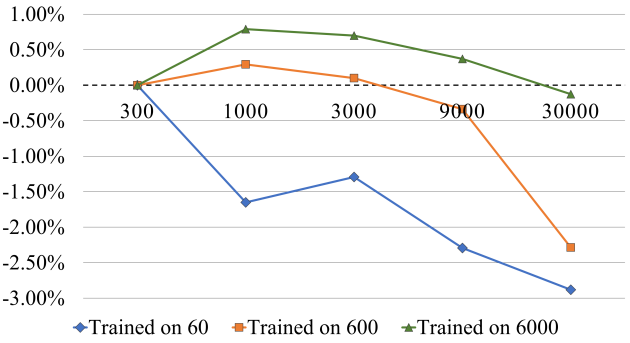

FIGURE 4. XLNet's accuracy rate fluctuation over longer fine-tune training steps.

between amount of training data and training step size. For both models fine-tuned with only few labeled data, e.g., 60, the accuracy rate drops off immediately when trained for longer than the baseline 300 steps, meaning it overfits the source domain. For XLNet, there is recognizable decrease in performance after 1000 training steps for all models. We believe that XLNet captures the necessary contextual dependencies earlier in the training steps, when compared to BERT, and longer it trains, the parameters more overfit the source domain. Therefore even though XLNet runs slower than BERT, it learns more quickly with fewer training steps.

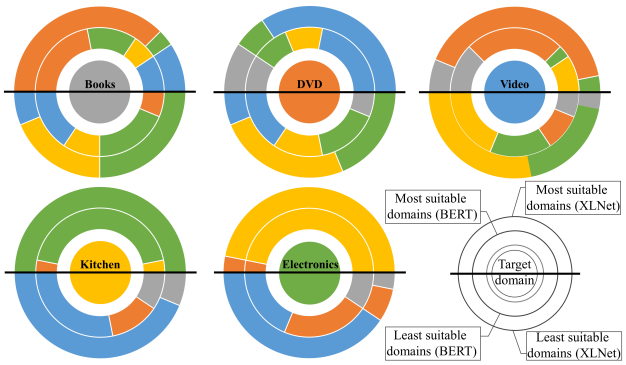

FIGURE 5. BERT (inner ring) and XLNet's (outer ring) domain *transferability* of a given domain (inner circle).

In Fig. 5, we show the *transferability*, i.e., ease of transfer learning, among the five domains from 640 separate experimental observations for each domain. Upper half of a doughnut graph indicates for a given target domain (inner circle), which domains were the most suitable source domain. Contrarily, the lower half indicates which domains were the least suitable. Suitability is measured with CDSC accuracy rates, i.e., higher the rate, more suitable the source domain was for the given target. Inner ring displays the results obtained with BERT and outer ring contains the results of XLNet. For example: for 'Books' domain, 'DVD' is the most suitable source and 'Electronics' is the least suitable. 'Kitchen' and 'Electronics' have high level of *transferability*, due to their contents being the most similar. For 'Electronics',

fine-tuned BERT indicates 'Video' and 'DVD' are equally least suitable, but XLNet overwhelmingly points towards 'Video' as the least suitable domain.

VI. CONCLUSION AND FUTURE WORK

In this paper, we apply the bidirectional contextualized Transformer language models of BERT and XLNet on cross-domain sentiment classification task. Due to their unsupervised pre-training tasks utilizing large unlabeled datasets and their self-attention Transformer mechanisms, BERT and XLNet both greatly outperforms the previous state-of-the-arts methods for CDSC task. When compared closely, XLNet outperforms BERT on all CDSC tasks. XLNet is is very efficient in capturing context and achieves state-of-the-arts results with only using 50 fine-tune training samples, i.e., around 120 times fewer data than the previous high-performing CDSC methods trained on. XLNet's better prediction accuracy is mostly due to its novel pre-training objective, ability to capture long-term dependencies, and larger pre-training dataset. XLNet is more resource-hungry than BERT, but learns contextual data much quicker than BERT with fewer fine-tuning steps. For future, it is interesting to explore how, or whether, BERT's performance improves if pre-trained on the similar amount of data as in XLNet. Also making both models lighter and more resource-efficient can be an interesting area to explore.

REFERENCES

[1] H. Zhou, M. Huang, T. Zhang, X. Zhu, and B. Liu, "Emotional chatting machine: Emotional conversation generation with internal and external memory," in *Proc. 32nd AAAI*, 2018, pp. 1–9.

[2] B. Liu, "Sentiment analysis and opinion mining," *Synth. Lect. Hum. Lang. Technol.*, vol. 5, no. 1, pp. 1–167, 2012.

[3] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 3111–3119.

[4] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proc. EMNLP*, 2014, pp. 1532–1543.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[6] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proc. EMNLP*, 2013, pp. 1631–1642.

[7] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[8] Q. Li, "Literature survey: Domain adaptation algorithms for natural language processing," Dept. Comput. Sci., Graduate Center, City Univ. New York, New York, NY, USA, Tech. Rep., 2012, pp. 8–10.

[9] Y. Goldberg, "Neural network methods for natural language processing," *Synth. Lect. Hum. Lang. Technol.*, vol. 10, no. 1, pp. 1–309, 2017.

[10] X. Wan, "Co-training for cross-lingual sentiment classification," in *Proc. Joint Conf. 47th Annu. Meeting ACL 4th Int. Joint Conf. Natural Lang. Process. (AFNLP)*, 2009, pp. 235–243.

[11] J. Blitzer, M. Dredze, and F. Pereira, "Domain adaptation for sentiment classification," in *Proc. ACL*, 2007, pp. 440–447.

[12] J. Yu and J. Jiang, "Learning sentence embeddings with auxiliary tasks for cross-domain sentiment classification," in *Proc. EMNLP*, 2016, pp. 236–246.

[13] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," 2018, *arXiv:1802.05365*. [Online]. Available: https://arxiv.org/abs/1802.05365

[14] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. (2018). *Improving Language Understanding by Generative Pre-Training*. [Online]. Available: https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf

[15] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2018, *arXiv:1810.04805*. [Online]. Available: https://arxiv.org/abs/1810.04805

[16] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V. Le, "XLNet: Generalized autoregressive pretraining for language understanding," 2019, *arXiv:1906.08237*. [Online]. Available: https://arxiv.org/abs/1906.08237

[17] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov, "Transformer-xl: Attentive language models beyond a fixed-length context," 2019, *arXiv:1901.02860*. [Online]. Available: https://arxiv.org/abs/1901.02860

[18] S. J. Pan, X. Ni, J.-T. Sun, Q. Yang, and Z. Chen, "Cross-domain sentiment classification via spectral feature alignment," in *Proc. 19th Int. Conf. WWW*, 2010, pp. 751–760.

[19] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *J. Mach. Learn. Res.*, vol. 17, no. 59, pp. 1–35, 2016.

[20] B. Sun, J. Feng, and K. Saenko, "Return of frustratingly easy domain adaptation," in *Proc. 30th AAAI*, 2016, pp. 1–8.

[21] W. Zellinger, T. Grubinger, E. Lughofer, T. Natschlger, and S. Saminger-Platz, "Central moment discrepancy (CMD) for domain-invariant representation learning," 2017, *arXiv:1702.08811*. [Online]. Available: https://arxiv.org/abs/1702.08811

[22] C. Chen, Z. Chen, B. Jiang, and X. Jin, "Joint domain alignment and discriminative feature learning for unsupervised deep domain adaptation," in *Proc. 33rd AAAI*, 2019, pp. 3296–3303.

[23] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, "DeCAF: A deep convolutional activation feature for generic visual recognition," in *Proc. ICML*, 2014, pp. 1–9.

[24] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "CNN features off-the-shelf: An astounding baseline for recognition," in *Proc. IEEE Conf. CVPR Workshops*, Jun. 2014, pp. 806–813.

[25] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," 2018, *arXiv:1801.06146*. [Online]. Available: https://arxiv.org/abs/1801.06146

[26] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," 2014, *arXiv:1412.3555*. [Online]. Available: https://arxiv.org/abs/1412.3555

[27] Y. Kim, C. Denton, L. Hoang, and A. M. Rush, "Structured attention networks," in *Proc. ICLR*, 2017, pp. 1–21.

[28] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.

[29] Y. Wu *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation," 2016, *arXiv:1609.08144*. [Online]. Available: https://arxiv.org/abs/1609.08144

[30] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," 2016, *arXiv:1607.06450*. [Online]. Available: https://arxiv.org/abs/1607.06450

[31] D. Hendrycks and K. Gimpel, "Gaussian error linear units (GELUs)," 2016, *arXiv:1606.08415*. [Online]. Available: https://arxiv.org/abs/1606.08415

[32] *TensorFlow Code and Pre-Trained Models for BERT*. Accessed: Jun. 2019. [Online]. Available: https://github.com/google-research/bert

[33] *XLNet: Generalized Autoregressive Pretraining for Language Understanding*. Accessed: Jun. 2019. [Online]. Available: https://github.com/zihangdai/xlnet

[34] Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler, "Aligning books and movies: Towards story-like visual explanations by watching movies and reading books," in *Proc. ICCV*, Dec. 2015, pp. 19–27.

[35] R. Parker, D. Graff, J. Kong, K. Chen, and K. Maeda, "English gigaword fifth edition," Linguistic Data Consortium, Philadelphia, PA, USA, Tech. Rep., 2011.

[36] J. Callan, M. Hoy, C. Yoo, and L. Zhao. (2009). *The ClueWeb09 Dataset*. [Online]. Available: https://lemurproject.org/clueweb09.php/

[37] Common Crawl. *Common Crawl Corpus*. Accessed: Jun. 2019. [Online]. Available: https://commoncrawl.org/the-data/

[38] T. Kudo and J. Richardson, "SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing," 2018, *arXiv:1808.06226*. [Online]. Available: https://arxiv.org/abs/1808.06226
[39] Y. Kim, "Convolutional neural networks for sentence classification," 2014, *arXiv:1408.5882*. [Online]. Available: https://arxiv.org/abs/1408.5882
[40] Z. Li, Y. Zhang, Y. Wei, Y. Wu, and Q. Yang, "End-to-end adversarial memory network for cross-domain sentiment classification," in *Proc. IJCAI*, 2017, pp. 2237–2243.
[41] Z. Li, Y. Wei, Y. Zhang, and Q. Yang, "Hierarchical attention transfer network for cross-domain sentiment classification," in *Proc. 32nd AAAI*, 2018, pp. 1–8.
[42] T. Manshu and W. Bing, "Adding prior knowledge in hierarchical attention neural network for cross domain sentiment classification," *IEEE Access*, vol. 7, pp. 32578–32588, 2019.

BATSERGELEN MYAGMAR received the B.S. degree in computer science from the Ramapo College of New Jersey, in 2006, and the M.S. degree in computer engineering from the University of Florida, in 2010. He is currently pursuing the Ph.D. degree with the Department of Computer Science, University of Tsukuba, Japan. His research interests include machine learning, transfer learning, domain adaptation, and natural language processing.

JIE LI received the B.Eng. degree in computer science from Zhejiang University, Hangzhou, China, the M.Eng. degree in electronic engineering and communication systems from the China Academy of Posts and Telecommunications, Beijing, China, and the Dr.Eng. degree from the University of Electro-Communications, Tokyo, Japan. He is currently with Department of Computer Science and Engineering, Shanghai Jiaotong University, Shanghai, China, where he is also a Chair Professor. He was a full Professor with the Department of Computer Science, University of Tsukuba, Japan. He was a Visiting Professor with Yale University, USA, Inria Sophia Antipolis, and Inria Grenoble-Rhone-Alpes, France. His current research interests include big data, cloud computing, machine learning and networking, network security, OS, and modeling and performance evaluation of information systems. He has also served on the program committees for several international conferences. He is the Co-Chair of the IEEE Technical Community on Big Data and the IEEE Big Data Community, and the Founding Chair of the IEEE ComSoc Technical Committee on Big Data. He serves as an Associate Editor for many IEEE journals and TRANSACTIONS.

SHIGETOMO KIMURA received the B.Eng., M.Eng., and Dr.Info.Sc. degrees from Tohoku University, Japan. He is an Associate Professor with the Faculty of Engineering, Information and Systems, University of Tsukuba. His research interests include network protocols and performance evaluation of communication systems. He is a member of the ACM, IPSJ, IEICE, and JSSST.