Self-supervised Short Text Modeling through Auxiliary Context Generation

NURENDRA CHOUDHARY, Virginia Tech, USA
CHARU C. AGGARWAL, IBM T.J. Watson Research Center, USA
KARTHIK SUBBIAN, University of Minnesota, USA
CHANDAN K REDDY, Virginia Tech, USA

Short text is ambiguous and often relies predominantly on the domain and context at hand in order to attain semantic relevance. Existing classification models perform poorly on short text due to data sparsity and inadequate context. Auxiliary context, which can often provide sufficient background regarding the domain, is typically available in several application scenarios. While some of the existing works aim to leverage real-world knowledge to enhance short text representations, they fail to place appropriate emphasis on the auxiliary context. Such models do not harness the full potential of the available context in auxiliary sources. To address this challenge, we reformulate short text classification as a dual channel self-supervised learning problem (that leverages auxiliary context) with a generation network and a corresponding prediction model. We propose a self-supervised framework, *Pseudo-Auxiliary Context generation network for Short text Modeling (PACS)*, to comprehensively leverage auxiliary context and is jointly learned with a prediction network in an end-to-end manner. Our PACS model consists of two sub-networks: a Context Generation Network (CGN) that models the auxiliary context's distribution and a Prediction Network (PN) to map the short text features and auxiliary context distribution to the final class label. Our experimental results on diverse datasets demonstrate that PACS outperforms formidable state-of-the-art baselines. We also demonstrate the performance of our model on cold start scenarios (where contextual information is non-existent) during prediction. Furthermore, we perform interpretability and ablation studies to analyze various representational features captured by our model and the individual contribution of its modules to the overall performance of PACS, respectively.

CCS Concepts: • **Computing methodologies → Topic Modeling**; *Lexical semantics*; *Semantic networks*; *Information extraction.*

Additional Key Words and Phrases: Self-attention, short text classification, context learning, self-supervision.

ACM Reference Format:
Nurendra Choudhary, Charu C. Aggarwal, Karthik Subbian, and Chandan K Reddy. 2022. Self-supervised Short Text Modeling through Auxiliary Context Generation. *ACM Trans. Intell. Syst. Technol.* 1, 1, Article 1 (January 2022), 21 pages. https://doi.org/10.1145/3511712

1 INTRODUCTION

Short text classification is a useful but challenging problem in various application settings such as sentiment analysis [40], dialogue systems [19], short-text topic modeling [35], and user intent detection [13]. Unlike paragraphs or documents, short text is ambiguous primarily due to the lack

Authors' addresses: Nurendra Choudhary, nurendra@vt.edu, Virginia Tech, 900 N Glebe Road, Arlington, Virginia, USA, 22203; Charu C. Aggarwal, charu@us.ibm.com, IBM T.J. Watson Research Center, 1101 Route 134 Kitchawan Rd, Yorktown Heights, New York, USA, 10598; Karthik Subbian, subbiank@acm.org, University of Minnesota, 200 Union St SE, Minneapolis, Minnesota, USA, 55455; Chandan K Reddy, reddy@cs.vt.edu, Virginia Tech, 900 N Glebe Road, Arlington, Virginia, USA, 22203.

2157-6904/2022/1-ART1
https://doi.org/10.1145/3511712

of context. The short text derives its context from real-world knowledge bases. A few examples of such cases are given in Table 1. The entity Tim Boyle relates to sports given the context “NFL” and “offseason.” Humans leverage existing knowledge bases to enhance their comprehension of short text[1]. Moreover, they do not focus on the entire knowledge base, but rather focus their attention towards a specific set of entities and their relationships to retrieve the relevant context. Current approaches in short text classification lack such focus mechanisms and employ the entire knowledge base towards downstream tasks. This technique is useful when the downstream task depends on the entire knowledge base, e.g., email communications have limited explicit context of their topics. However, in several real-world applications, we notice adequate availability of auxiliary information that explicitly connects to the short text.

Table 1. Example of short text and its corresponding auxiliary context with the class label. Highlighted words significantly contribute to the final classification.

Short Text	**Auxiliary Context**	**Label**
Tim Boyle ready to take another step	.. the offseason in .. the NFL .. a nose-down offseason ..	Sports (News)
The Lord of the Rings : The Return of the King	.. forces of good and evil fighting .. their quest to ..	Adventure (Movies)
Belkin WaveRest Gel Mouse Pad	.. comfortable with .. smooth, durable surface ..	Office (Reviews)

We discuss some applications in Table 1 where short text appears along with corresponding auxiliary information. For instance, the goal in news classification task is to predict the category given the news headline. The description of the news and their authors serve as the auxiliary information. In the case of movie classification, predicting the genre from the movie title is the objective. The auxiliary information here are the synopsis, cast, and reviews. For the reviews on e-commerce platforms [6], the goal is to predict the product category from the review information. The auxiliary information here is the product type, description, and reviews based on historical purchases of the popular queries. *In this paper, we address this challenge of leveraging the auxiliary context information and enriching the representations of the short text classification model.*

Popular text classification approaches rely on semantic relations between words and their corresponding context. For training a language model, BERT [8] and XLNet [44] adopt a masking and a word-order permutation task, respectively. Although these models are effective in modeling sequences, there is a disconnect between the model training and the prediction phases. In other words, if BERT and XLNet are pre-trained on next sentence prediction, which contains a [SEP] tag to differentiate sentences, the fine-tuning would rely on the [SEP] tag in the data to distinguish between sentences. For short-text classification, the auxiliary context is unavailable during the prediction phase and thus, the sequence connection (through the separator tag [SEP]) between short text and auxiliary context is not present. This *scenario commonly occurs in real-world applications when the auxiliary context is unavailable for classification* (lack reviews when categorizing new Amazon products).

There are two popular approaches for integrating contextual information in short text classification. One of the approaches [5] uses entity linking and conceptualization to link the phrases in the short text to concepts in a knowledge base. However, the performance of the short text classifier is

[1] https://www.scientificamerican.com/article/wired-for-categorization/

challenged by the entity linking task and the availability of a knowledge base of sufficiently high quality. Other approaches [35, 46] extract topic distributions from the short text and combine them with the short-text semantic information to create additional contextual features for classification. *The main drawback of these approaches is that the model cannot adapt (zoom-in or zoom-out) to different topical granularities to create the necessary context. We overcome these challenges by generating necessary contextual information given the short text at different topic granularities and without explicitly linking the entities to knowledge bases.*

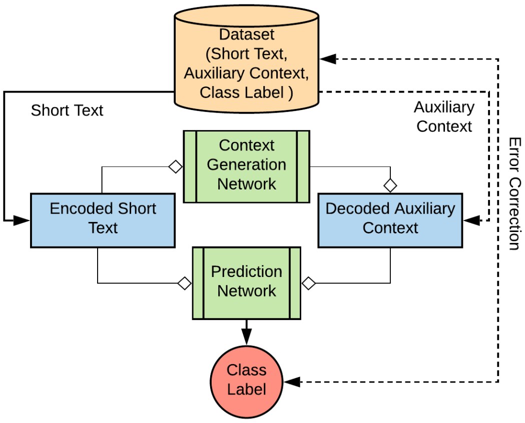


Fig. 1. The proposed self-supervised learning framework that generates auxiliary context for the problem of short text classification. The dotted lines represent the data available during the training but inaccessible during prediction. Error correction is the gradient update to the Context Generation and Prediction Network using back-propagation.

In this work, *we reformulate the short text classification problem and design a new self-supervised learning framework with two related tasks, namely, sequence auxiliary context generation task (which acts as our proxy task) and the class prediction task (which acts as the fine-tuning task).* Note that both the related tasks, in a self-supervised learning paradigm, depend only on the auxiliary context that is systematically obtained from the data and do not rely on any additional manual labeling. The sequence generation problem utilizes short text to generate a conditional pseudo-auxiliary context distribution based on statistical inference from the training data distribution. The classification task predicts the final label based on an aggregation of short text features and the auxiliary context distribution. Figure 1 illustrates the proposed learning framework. To achieve this, we propose *Pseudo-Auxiliary Context generation network for Short text modeling (PACS).* The model architecture consists of two mutually trained sub-networks, namely, Context Generation Network (CGN) and Prediction Network (PN). CGN is a sub-network of Self-Attentive Bi-directional Long Short-Term Memory (Bi-LSTM) network which handles the sequence generation sub-problem using an encoder-decoder architecture that encodes short text and decodes to the corresponding auxiliary context. Additionally, to maintain consistency in the domain, we need the auxiliary context to be conditioned upon the short text. The sequential dependence of the decoder on encoder retains the conditionality constraint for the auxiliary context generation. The PN utilizes these features for the final prediction task. *The primary goal of PACS is to learn how to generate auxiliary context in a self-supervised manner for the task of short text classification.* The main contributions of the paper are summarized as follows:

- We reformulate the short text classification problem using self-supervised learning for leveraging auxiliary context through conditional sequence generation and a predictor to map features to the class label.
- Inspired by the human focus mechanism, we develop PACS, a novel self-supervised learning architecture that consists of two sub-networks that are jointly trained on a final prediction task. A context generation network, which is a self-supervised model, first generates sequential auxiliary context distribution from the short text. The predictor network then applies these features to learn a model that maps features to their final class label.
- We perform an extensive set of experiments across several real-world datasets to evaluate the performance of PACS against various state-of-the-art baseline methods. We also analyze the effectiveness of PACS and its sensitivity to dataset size and auxiliary context availability. Additionally, we qualitatively study the model to understand the reasons for its effectiveness.

The rest of this paper is organized as follows: Section 2 discusses the relevant background for the problem. Section 3 reformulates the short text classification problem using self-supervised learning and introduces the architecture of our proposed PACS model. Section 4 describes the real-world datasets, state-of-the-art baselines, and performance metrics used to evaluate the PACS model. We also show the performance along with the model interpretability study of the PACS' architecture. Finally, Section 5 concludes our paper.

2 RELATED WORK

We discuss the background work related to two main sub-areas of research associated to the proposed model: short text classification and attention mechanism in neural models.

2.1 Short Text Classification

One line of research in short text classification relies on explicit feature construction using human-designed sparse features. Cavnar and Trenkle [4] employ n-gram features for text classification and in [29, 32, 36], the authors utilize more complex features such as POS tags and dependency parsing to improve the prediction. Another line of research works aim to leverage knowledge bases for enriching the information of short texts. In [10], Wikipedia is adopted to enrich the information retrieved from short text. Wang et al. [40] map the short text to a set of relevant concepts and leverage Probase to classify the obtained features. Explicit feature modeling generates human-interpretable representations but do not capture contextual semantic information. Another group of research works focus on probabilistic topic identification of short text. Latent Dirichlet Allocation (LDA) [3, 15] leverages word co-occurrence to represent short text as a distribution over its topics and vice-versa. Additionally, Non-negative Matrix Factorization (NMF) has been successfully applied to short text topic modeling [35].

Recently, implicit models have gained popularity due to the proliferation of deep learning algorithms. The models map the original text onto a semantic dense vector in a latent space. Word2Vec [26] and GloVe [31] provide word representations according to their context and co-occurrence, respectively. In [18], the authors utilize a combination of convolutional network to capture semantic features and recurrent network to obtain sequential features for short text classification. Character-level convolutional neural network (CNN) based models [7, 14, 48] provide semantic features from character n-grams using CNN filters for text classification. Bi-LSTM and Self-attention models [8, 20, 22, 44] encode short text and relations between the words for text classification. Topic-model based approaches [21, 43, 47, 50] leverage cross-text word co-occurrences based topic modeling over large documents (or pseudo-documents) to learn word embeddings and further utilize them to solve the problem of data sparsity in short text and improve the classification performance. In [38], the authors utilize auxiliary signals in the area of e-commerce such as purchasing intent to construct graphs of similar short text queries and products. A graph convolution network (GCN) [17] is applied to this graph for improved performance. The graph of similar items limits data sparsity by providing additional information for enhanced classification. Meng et al. [25] utilize weak supervision to generate pseudo-documents for hierarchical text classification. Although implicit models effectively capture syntactic and semantic information, they lack the ability to capture information from knowledge bases. Topic memory networks [46] and short text classification with knowledge powered attention [5] use both short text and encoded knowledge bases to enhance prediction. However, these models do not focus on relevant knowledge bases. Hence, to alleviate this problem, the *proposed PACS model utilizes sample-based auxiliary context to capture features from both the short text and its domain.*

2.2 Attention Mechanism in Neural Models

Attention mechanisms have been proven to be effective in various types of neural models. The mechanism can be broadly grouped into two categories (i) vanilla attention and (ii) self-attention. Bahdanau et al. [2] applied the vanilla attention mechanism to compute relevance score between query and input tokens for machine translation task. Inspired from these works, we employ self-attention to understand the importance of individual words to sentences in PACS during the context generation phase.

Topic memory networks [28, 42, 45, 46, 49] employ pre-trained topic models on external data in order to encode latent topic representations for short text classification. However, such an approach loses out on the domain-relevance by dismantling correspondence between short text and auxiliary context and utilizing a bag of auxiliary context instead of individually aligned samples. Samples with corresponding auxiliary context maintain relevance and remain domain-specific. Additionally, the topic modeling and short text classification architectures are independent and hence, the topic features possess limited relevance to the short text classification task. Also, classifier models that leverage knowledge-bases have shown good performance [12, 23, 30]. Short Text Classification with Knowledge powered Attention (STCKA) [5] model jointly trains the topic modeling and classification task but the scope of the knowledge base is wide. There is no mechanism to consider domain-relevance for validation. We need a framework to model auxiliary context generation conditional on short text. This maintains the relevance of the sample's features and leads to improved final prediction. The context generation needs to be contingent on the final class to learn features significant for the classification task.

3 PACS MODEL ARCHITECTURE

In this section, we introduce the overall self-supervised learning framework for short text classification by leveraging auxiliary context. We describe the overall architecture of **PACS** and its application in the context of short text classification.

3.1 Problem Statement

Let D denote the full dataset out of which D_T and D_V denote the training and validation sets, respectively; i.e., $D = D_T \cup D_V$. Each element $\{st_i, ac_i, y_i\} \in D_T$ and $\{st_j, ac_j, y_j\} \in D_V$ is the set of short text, its auxiliary context, and final class label, respectively. For D_V, $ac_j = \varnothing$ and $y_j \in \{y_1, y_2, ..., y_{|class|}\}$ is the prediction variable. The primary goal of classification is to optimize a model P_θ parameterized by θ, such that:

$$\theta = \arg\min_{\theta} \left(\sum_{i=1}^{|D_T|} -y_i \log\left(P_{\theta i}\right) \right) \tag{1}$$

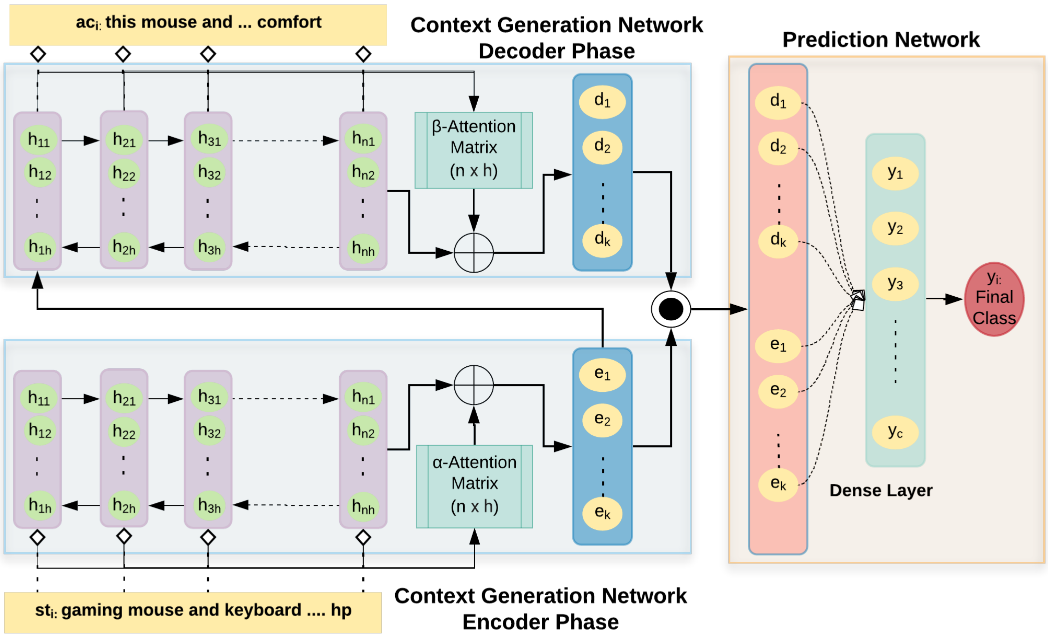


Fig. 2. The overall architecture of the proposed PACS model. During the training phase, the self-supervised Context Generation Network (CGN) encodes the input short text and decodes to the auxiliary context. For validation, the CGN network utilizes the short text to generate a distribution similar to the statistically inferred auxiliary context. The prediction network utilizes the short text encoding and generated auxiliary context distribution for the final class prediction.

Let X_p represent a probability distribution over parameters p. Under the assumption that $X_{st_i} \sim X_{st_j}$, we estimate parameter set θ as a strong predictor for D_V. In PACS, we leverage $ac_i \in D_T$ to learn a robust θ for prediction when ac_j is not available.

$$\lambda = \underset{\lambda}{\arg\max} \left(\sum_{i=1}^{|D_T|} P\left(ac_{\lambda i} | st_i\right) \right) \ni \forall i = 1 \longrightarrow |D_T|, X_{\lambda i} \sim X_{ac_i} \tag{2}$$

$$\theta = \underset{\theta}{\arg\min} \left(\sum_{i=1}^{|D_T|} -y_i \log\left(X_{st_i, ac_{\lambda i}}\right) \right) \tag{3}$$

where $\lambda \in \mathbb{R}^k$ (k is the number of parameters equivalent to weights in a neural network) is the set of sequence generator parameters estimated by maximizing the probability of generated auxiliary context $ac_{\lambda i}$ given short text st_i over the training set $i = 1$ to $|D_T|$ while maintaining similarity between distributions $X_{\lambda i}$ and X_{ac_i}. $\theta \in \mathbb{R}^{|class| \times 2k}$ is the set of classifier parameters that minimize the cross-entropy between target class y_i and combined features of short text and the generated auxiliary context X_{st_i, ac_λ}.

3.2 PACS Model Architecture

Figure 2 illustrates the overall architecture of our framework. The model consists of two modules corresponding to the self-supervised learning pipelines - Context Generation Network (CGN) and Prediction Network (PN). CGN learns a sequence generation function to map short text to its corresponding auxiliary context and PN learns a prediction model to classify a concatenation of encoded short text and auxiliary context to its sparse categories.

3.2.1 Context Generation Network (CGN). The aim of CGN is to generate the auxiliary context for a given short text sequence. We model the problem as a sequence-generation task, where, the input short text $\{st_1, st_2, ..., st_{|D|}\} \in ST$ generates the corresponding output auxiliary context $\{ac_1, ac_2, ..., ac_{|D|}\} \in AC$. To achieve this goal, we use a self-supervised model based on self-attentive Bi-LSTM networks. More formally, we optimize for parameters of the generator model $f_\lambda : X_{st} \longrightarrow X_\theta \ni X_\theta \sim X_{ac}$, where X_{st}, X_θ, X_{ac} is the distribution of short text, generated and original auxiliary context distribution, respectively.

The embedding layer converts the sparse one-hot ST into a dense k-dimensional representation $\{st_{i1}, st_{i2}, ..., st_{ik} \in st_i\}$ (k is decided empirically). The embedding layer has multiple variants - Word2Vec [26], BERT [8], and XLNet [44]. Word2Vec are pre-trained semantic vectors that use word's context from a large corpus. BERT and XLNet are language models trained on large corpora with word-masking and word-permutation, respectively. For BERT and XLNet, we extract the last layer for word embeddings.

The Bidirectional Long Short Term Memory (Bi-LSTM) layer encodes the sequence with a forward h_{ft} and backward h_{bt} LSTM [37]. Back-propagation through time (BPTT) [41] simultaneously updates the hidden states of the forward (f_t) and backward (b_t) pass. The weights provide sequential information of the sentences. We adopt Rectified Linear Unit (ReLU) as our activation function for non-linearity and faster convergence.

$$h_{f_t} = LSTM(w_{f_t}, h_{f_{t-1}}) \tag{4}$$

$$h_{b_t} = LSTM(w_{b_t}, h_{b_{t-1}}) \tag{5}$$

$$o_t = max\{0, W[f_t, b_t]x_t + b_t\} \tag{6}$$

where h_{f_t} and h_{b_t} are the hidden LSTM units of the forward and backward pass, respectively and o_t is the ReLU activation unit for the combined weights. To capture long-term sentence dependencies, we employ the *self-attention* network. It computes attention weights that denote the significance of relation between words in the sentence. Given maximum sequence length of input is n, the weight matrix $\alpha \in \mathbb{R}^{n \times h}$ for the dot-product attention over h hidden units and the final sentence encoding e_t scaled with the attention weights (α_{ij}) is given by:

$$\alpha_{ij} = \frac{o_i o_j^T}{\sqrt{2h}}; \quad e_t = \sum_j \frac{exp(\alpha_{tj})}{\sum_t exp(\alpha_{tj})} o_{tj} \tag{7}$$

The attention weights (α_{ij}) indicate the significance of the interaction between encoded sequential outputs o_i and o_j in the latent space to the final encoded output at the timestep e_t. The encoding e_t initiates the decoder model. The decoder for the auxiliary context is similarly modeled with attention matrix $\beta \in \mathbb{R}^{n \times h}$ and sequence decoding d_t given by the following equations.

$$\beta_{ij} = \frac{o_i o_j^T}{\sqrt{2h}}; \quad d_t = \sum_j \frac{exp(\beta_{tj})}{\sum_t exp(\beta_{tj})} o_{tj} \tag{8}$$

However, this problem encounters an *information bias.* The short text contains little evidence to support a full reconstruction of the auxiliary context (X_{ac}). However, for our primary problem of classification, we merely expect additional information from the auxiliary context's space to support the short text. Hence, CGN will only predict a distribution $X_\theta \sim X_{ac}$ for the additional information. To this end, we employ KL-divergence as our loss function L_{KLD}.

$$L_{KLD}(X_\theta, X_{ac}) = X_\theta \log\left(\frac{X_\theta}{X_{ac}}\right) \tag{9}$$

where X_θ and X_{ac} are the distributions that need to be compared. Unlike other prominent loss functions designed to evaluate point-wise categorical predictions, KL-divergence measures the similarity between prediction distributions and the ground truth. The significance of X_θ to y is successfully demonstrated in our experimental results presented in Section 4.3.

3.2.2 Prediction Network (PN). In PN, we utilize a dense network to predict the final class label y from features $x_i \in x$ ($\forall$ i) extracted from a concatenation of input text e_i and the auxiliary context's predicted distribution d_i.

$$x_i = e_i \odot d_i \tag{10}$$

The final probability for each category $y_i \in y$ is given by:

$$P(y_i|x_i) = \frac{exp\left(\sum_{j=1}^{2k} w_{ij}x_{ij} + b_i\right)}{\sum_{i=1}^{c} exp\left(\sum_{j=1}^{2k} w_{ij}x_{ij} + b_i\right)} \tag{11}$$

where c is the number of classes, k is the number of output units in the short text encoder and auxiliary context decoder, $w, x \in \mathbb{R}^{c\times 2k}$, $y \in \mathbb{R}^c$ and $x_i = \{x_{i1}, x_{i2}, ..., x_{i2k}\}$. We utilize cross-entropy as the loss because the prediction class labels are categorical.

$$L_{CE}(\hat{y}, y) = -\sum_{i=1}^{c} \hat{y}_i \log(y_i) \tag{12}$$

Algorithm 1: PACS algorithm

```
Input: Training data $D_T$;
Output: Generator model $f_\lambda$, Predictor $P_\theta$;
Randomly initialize trainable parameters $\lambda$, $\theta$;
Initialize loss $l_{cgn}, l_{pn} = 0$;
for $\{st_i, ac_i, y_i\} \in D_T$; till convergence of $\lambda$, $\theta$ do
    $e_i$ = Encode($st_i$) via Eq. (7);
    $d_i$ = Decode($e_i$) via Eq. (8);
    $l_{cgn} = l_{cgn} + L_{KLD}(d_i, ac_i)$ from Eq. (9);
    $x_i = e_i \odot d_i$;
    $p_i$ = Dense($x_i$) via Eq. (11);
    $l_{pn} = l_{pn} + L_{CE}(y_i, p_i)$ from Eq. (12);
    # Update $\lambda$ and $\theta$ with back-propagation;
    $\lambda \leftarrow \lambda - \Delta_\lambda l_{cgn}$
    $\theta \leftarrow \theta - \Delta_\theta l_{pn}$
end
return $f_\lambda, P_\theta$
```

3.2.3 PACS algorithm. Algorithm 1 provides a high-level pseudo-code of the proposed PACS model. The goal is to estimate the generator model f_λ and predictor P_θ given input training set D_T. Lines 4 and 5 encode the short text and generate auxiliary context distribution, respectively. The losses for CGN (l_{cgn}) and Prediction Network (l_{pn}) are calculated from lines 6 to 9. Based on the losses, λ and θ are updated in line 11 through back-propagation. The updated λ and θ form the parameters for our generator model f and predictor model P, respectively. The semantic representations in PACS are empirically set to 300 dimensions (k=300). We adopt ReLU as our activation unit to introduce non-linearity and Dropout (p=0.5) to avoid over-fitting on the training set.

4 EXPERIMENTAL SETUP

In this section, we describe the datasets in our experiments and then provide baselines along with implementation details.

4.1 Dataset Description

To analyze the performance of the proposed PACS model and compare it against other state-of-the-art text classification approaches, we conducted comprehensive experiments using several real-world datasets. Table 2 summarizes additional details including some basic statistics of the datasets.

Table 2. Dataset Statistics (the average sentence length, maximum sentence length, number of classes, and the number of training/validation samples in each of the datasets). ST and AC columns represent the number of Short Text and Auxiliary Context samples, respectively.

Dataset	Avg Len		Max Len		No. of	# samples	
	ST	AC	ST	AC	classes	Train	Val
Amazon	9	70	52	4881	43	912,000	608,000
HuffPost	9	20	44	243	41	120,551	80,341
RT	3	126	25	2473	12	17,886	11,924
ArXiv	8	150	33	558	41	24,600	16,400

- **Amazon Reviews[2]:** The dataset contains 142.5 million amazon reviews and metadata of 43 product categories from May 1996 to July 2014. In our experiments, we utilize the "product title" as the short text to predict the "product category". During training, we employ the corresponding "review body" as the auxiliary context. We randomly sample a uniform number of data points from each class for our experiments. We also study the effect of this sampling and the scalability of our model in Section 4.2.
- **HuffPost News[3]:** The dataset consists of 200K news headlines and metadata of 41 categories from 2012 to 2018. We utilize the "headline" as the short text to predict the "category". For training, we leverage the corresponding "short_description" as the auxiliary context.
- **Rotten Tomatoes (RT) Movies[4]:** It contains 30K movies and metadata of 12 genres from 1914 to 2018. The experiments use the "Title" as the short text to predict "Genre". The training employs the corresponding "Description" as the auxiliary context.

[2] https://s3.amazonaws.com/amazon-reviews-pds/tsv/index.txt

[3] https://www.kaggle.com/rmisra/news-category-dataset

[4] https://www.kaggle.com/ayushkalla1/rotten-tomatoes-movie-database

- **ArXiv Papers**[5]**:** This dataset contains 41K research papers from arXiv labeled with 42 tags/categories. In the experiments, we utilize "title" as the short text to predict the category "term" and generate the corresponding auxiliary context "summary".

To study the performance of PACS on varying number of classes, we consider n-class versions of our datasets. A n-class version of the dataset only utilizes the top n classes that have the most number of samples for training and testing phases of PACS and the baselines. We did not include other text classification benchmark datasets (such as the GLUE [39]) because they lack auxiliary context. In such cases, our model will mirror the performance of its embedding layer (Word2Vec, BERT or XLNET) as there is no enrichment of features from the auxiliary context. In addition to this, although the datasets contain an auxiliary context, they are not considered ($AC = \varnothing$) in the evaluation procedures of our experiments. This simulates the real-world problem where the context is available for learning but unavailable for classification of new samples.

4.2 Performance Comparison

We perform several experiments to compare our model against other state-of-the-art methods. Additionally, we compare the performance of PACS by varying the number of classes, data points, and the availability of auxiliary context.

4.2.1 Comparison with baselines. We compare PACS with several state-of-the-art baseline models for short text classification. In the training phase, all the models have access to both the short text and auxiliary context. For the validation phase, the models exclusively use the short text for class prediction. The baseline models used for comparison in our experiments are as follows:

- *Latent Dirichlet Allocation* (**LDA**)[6] [3]: In our experiment, we learn topic models T as a distribution over the combination of short text and auxiliary context ($T_\theta \sim ST \cup AC$). Simultaneously, the final topic matrix allows us to extract word vectors as a distribution over the topic models.
- *Topic Memory Networks* (**TMN**) [46]: The model is a combination of three major units: neural topic model to infer latent topics, topic memory mechanism to extract features from latent topics and a final prediction model to map features to a class.
- *Short Text Classification with Knowledge powered Attention* (**STCKA**)[7] [5]: The model learns the Concept to Short Text (C-ST) and Concept to Concept (C-CS) attention from auxiliary context. This conceptual knowledge is applied to predict short text's classes.
- *Bi-directional Transformers* (**BERT**) [8]: The model utilizes transformers to capture the co-dependence of different sentence units as attention weights. BERT achieves this through training a language model by masking certain inputs. We adopt the large pre-trained BERT model and fine-tune it on our datasets. The fine-tuning inputs are a concatenation of short text, a separator label *[SEP]* and the auxiliary context. Also, we adopt two variations of fine-tuning; one is trained only on short text (BERT-ST) and other has additional access to auxiliary context (BERT-STAC).
- *Auto-regressive Transformers-XL* (**XLNet**) [44]: Unlike BERT, XLNet learns the language model through maximizing likelihood over all permutations for input masks. We adopt the large pre-trained XLNet model and fine-tune it on our datasets. Similar to the variants of BERT mentioned above, we use two variations based on training phase: XLNet-ST and XLNet-STAC.

4.2.2 Implementation details. LDA provides vectors for words in the dataset. The sentence vector is an average over the word vectors. The sentence vector trains the dense classifier to optimize for

[5] https://www.kaggle.com/neelshah18/arxivdataset

[6] LDA models the topic distribution that we utilize as the text vectors of our baselines.

[7] Due to unavailability of original code, we implemented STCKA based on the original paper and fine-tuned the hyper-parameters to obtain the best possible result.

Table 3. Hyper-parameter values in our experiments for PACS and the baselines.

Models	**Dropout Rate**	**Dense Units**	**Max Seq Length**	**# Model Parameters**
LDA	NA	128	NA	38.4K
TMN	0.5	64	100	8.98 M
STCKA	0.5	64	100	1.45 M
BERT	0.2	64	100	110.3 M
XLNet	0.2	64	100	146.8 M
PACS	0.5	64	100	5.48 M

the final class label. The remaining models comprise of end-to-end frameworks that output the class labels through intermediate feature extraction. To maintain fair comparison, we consistently set embedding dimensions ($k = 300$) and the number of hidden units ($h = 128$) to be constant across models. We empirically tune the other hyper-parameters, within our computational capacity, for best results. For sequential models involving LSTMs, we use back propagation through time (BPTT) to speed up the training process. We train all the networks including PACS with mini-batch Adam Optimizer [16] with a batch size of 64. We use gensim [34] for the baseline LDA model. For BERT and XLNet, we use their keras implementations [8]. The developers for these modules provide a precise mechanism for integrating classification. For TMN, we adopt the code provided by the authors [46]. For STCKA and PACS, we build on the layers in Tensorflow 2.0 [1] to model the architectures. We utilize K-fold cross validation with train, validation, and test ratio of 75:10:15 in our experimental setup. For training PACS, we use an NVIDIA P40 with 12GB of VRAM. We implemented PACS with keras Bi-LSTM and Attention layers in congruence with its Merge layer for joint training. PACS is trained with a contrastive learning procedure with one negative sample for each positive sample for the different classes. Contrastive learning ensures better discriminative power in the classification procedure as the model is able to differentiate between positive and negative samples for each class. This also reduces the class bias by maintaining a constant ratio of positive and negative samples for each class. The final hyper-parameters for PACS and our baselines are summarized in Table 3. Note that PACS requires considerably less number of parameters compared to XLNet and BERT, thus decreasing its reliance on availability of computational resources (high GPU memory)[9].

4.2.3 Performance comparison results. We validate the models' predictions using two evaluation metrics: Accuracy (estimate total prediction) and class-weighted F1-score (estimate the harmonic mean between precision and recall). The metrics are computed for each class and the reported results are averages weighted by the number of samples in each class. Table 4 depicts the results of our experiments. We observe that our model outperforms the current state-of-the-art by $\approx 8\%$ in Accuracy and $\approx 7.5\%$ in F1 score, for fewer number of classes; and $\approx 200\%$ in Accuracy and $\approx 45\%$ in F1-score for higher number of classes. We conjecture that these improvements are primarily due to the additional contextual features in the auxiliary text distribution that are generated by our CGN module in PACS.

4.2.4 Sensitivity to dataset attributes. In these experiments, we analyze the reliability of PACS on different dataset attributes. For our study, we vary the datasets in the number of data points and availability of auxiliary data.

[8] https://github.com/CyberZHG/keras-bert, https://github.com/CyberZHG/keras-xlnet
[9] Code for the PACS model is shared on https://github.com/Akirato/PACS

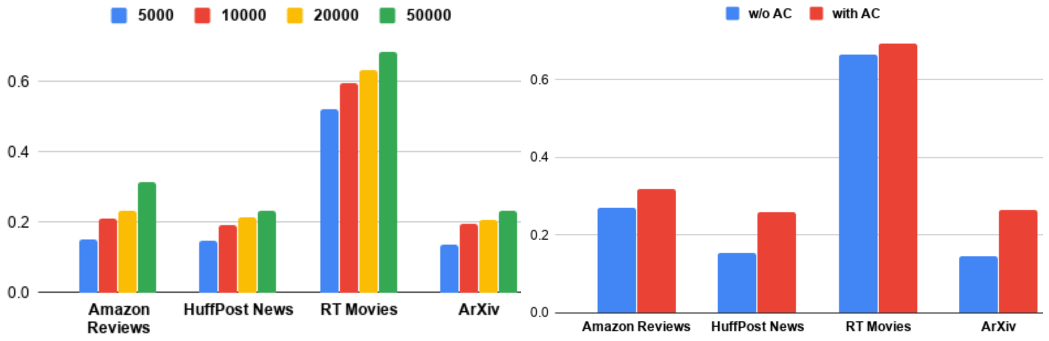
Table 4. Performance comparison of the proposed PACS model with several baseline methods across various datasets and evaluation metrics; (a) Accuracy and (b) F-score. ST and STAC report the values when the models are trained without and with additional auxiliary context, respectively. PACS (ours) reports the results for PACS with BERT as the primary embedding layer. The *Improv* row compares the performance improvement of PACS (ours) relative to XLNet (STAC).

(a) Accuracy measures

Dataset	**Amazon**			**Huffpost**			**RT**			**ArXiv**		
# Classes	3	10	43	3	10	41	3	5	12	3	10	41
LDA	43.4	30.4	4.91	42.7	24.3	3.2	40.4	37.2	27.9	43.7	25.0	4.1
TMN	55.9	37.5	5.2	43.1	24.9	5.6	48.8	50.7	44.9	44.3	26.1	5.7
STCKA	73.7	53.3	7.1	65.2	35.4	7.8	70.1	62.7	43.1	65.5	36.3	8.9
BERT (ST)	72.4	53.0	7.3	64.8	30.1	7.0	71.2	59.1	39.5	65.7	31.1	7.7
BERT (STAC)	74.3	54.7	7.3	66.4	36.1	8.2	71.8	63.9	44.6	66.9	37.2	8.2
XLNet (ST)	73.9	52.1	4.6	61.4	41.1	4.6	69.0	62.4	42.1	61.5	41.5	4.7
XLNet (STAC)	76.5	55.3	7.9	65.3	44.7	7.9	72.3	64.6	45.3	65.6	45.7	8.8
PACS (ours)	**85.4**	**59.7**	**31.2**	**74.6**	**51.3**	**24.7**	**81.9**	**75.4**	**68.8**	**75.0**	**51.9**	**25.8**
Improv(%)	11.6	8.0	294.9	14.2	14.8	212.7	13.3	16.7	51.9	14.3	13.6	193.2

(b) F-score measures

Dataset	**Amazon**			**Huffpost**			**RT**			**ArXiv**		
# Classes	3	10	43	3	10	41	3	5	12	3	10	41
LDA	.41	.29	.05	.40	.23	.03	.38	.35	.26	.42	.24	.05
TMN	.52	.36	.05	.41	.24	.05	.46	.49	.42	.42	.24	.07
STCKA	.69	.51	.07	.63	.33	.08	.65	.59	.41	.65	.34	.09
BERT (ST)	.67	.51	.07	.61	.28	.07	.67	.56	.37	.62	.30	.09
BERT (STAC)	.71	.51	.07	.64	.34	.08	.67	.61	.43	.65	.35	.10
XLNet (ST)	.71	.50	.04	.57	.38	.04	.66	.60	.40	.59	.39	.06
XLNet (STAC)	.72	.53	.08	.61	.43	.07	.69	.62	.44	.61	.45	.08
PACS (ours)	**.81**	**.57**	**.30**	**.69**	**.49**	**.23**	**.79**	**.70**	**.64**	**.70**	**.51**	**.24**
Improv(%)	12.5	7.5	275	13.1	14.0	228.6	14.5	12.9	45.5	14.8	13.3	200



(a)

(b)

Fig. 3. Sensitivity to dataset attributes (best viewed in color). (a) The graph depicts the shift in F-score with varying dataset sizes: 5000, 10000, 20000, and 50000 samples. (b) The graph shows the change in F-score with and without Auxiliary context.

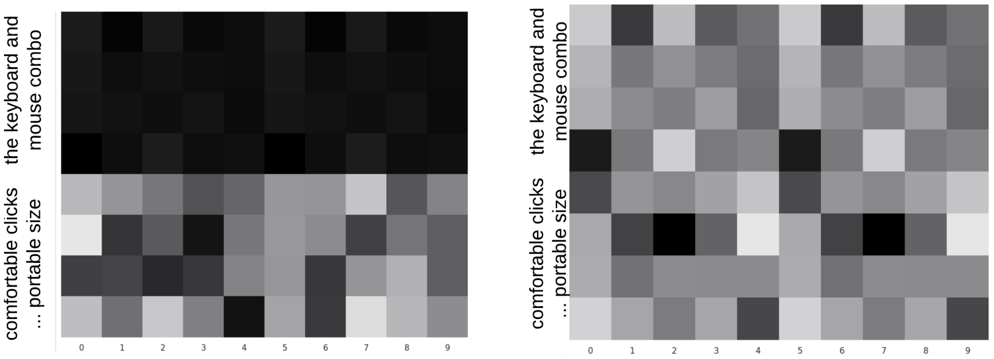
- *Number of data points*: We randomly sample subsets of various size from the datasets and measure the performance metrics and computational time.
- *Auxiliary context*: We remove the auxiliary context and only utilize the short text to analyze the change in performance.

Figure 3 shows the performance variation according to the dataset attributes. In this experiment, we notice a significant increase of 25%-106% in Accuracy as the number of data points increases. This indicates better generalizability of the auxiliary context generated by the CGN module with an increase in data points. Also, absence of auxiliary context hinders the model's performance and decreases Accuracy by 2%-56%. These results clearly illustrate the utility of auxiliary context generation. This conclusion is further supported by the qualitative evidence presented in Section 4.3.3.

4.3 Model Interpretability

In these set of experiments, we qualitatively analyze the influence of generated auxiliary context to the overall prediction and study the effect of different sentence segments to the generation process. This helps us in understanding PACS' attention mechanism and better interpret its underlying activations.

4.3.1 Significance of auxiliary context. Auxiliary context provides additional information to improve performance of the prediction network. We understand the context's exact contribution through the weights learnt by the dense classifier. Figure 4 shows a heat map of the classifier weights. We notice that weights in the case of XLNet-STAC are significantly focused on the short text. This indicates a lack of attention towards the auxiliary context. This is resolved in PACS that uniformly captures features from both short text and auxiliary context.



(a) (b)

Fig. 4. Significance of auxiliary context. The top and bottom 4 rows represent the weights of short text and auxiliary context, respectively. (a) and (b) present the activation of PN's hidden units for 10 classes in Amazon Reviews with XLNet-STAC and PACS (XLNet), respectively. Darker cells represent more relative weightage. The weights for short text look paler in PACS because of normalization.

4.3.2 Qualitative significance of auxiliary context. In this experiment, we qualitatively analyze the top-ranked words in the auxiliary context that aid overall prediction improvement. For each sample short text, we analyze the attention weights and pick words with maximum weights towards the final prediction (shown in Table 5). We observe that auxiliary context supports the prediction of

Table 5. Qualitative results showing the significance of Auxiliary Context. Blue color indicates correct label and Red represents incorrect prediction. ST and AC refer to short text and auxiliary context, respectively. From the last two columns, we observe that the predicted label is incorrect for short text and is correct when we utilize both short text and auxiliary context.

	Short Text	**Significant Words in Auxiliary Context**	**ST Label**	**ST + AC Label**
News	Hugh Grant Marries For The First Time At Age 57	actor, longtime, girlfriend, ceremony	POLITICS	ENTERTAINMENT
	What You Need To Know About Cambridge Analytica	blowing, whistle, admissions, probe	BUSINESS	POLITICS
Movies	The Masked Saint	wrestler, vigilante, fighting, crisis	DRAMA	ACTION
	Rivers and Tides: Andy Goldsworthy Working With Time	documentarian, art, stonewall, sculptures	DRAMA	ART & FOREIGN
	Every Which Way But Loose	boxer, fight, stopover	COMEDY	ACTION
Reviews	Alien Frontiers: Factions	player, factions, expansion, packs	Video DVD	TOYS
	Gaming Computer Mouse Pad	edge, mousepad, cushion, wide	ELECTRONICS	OFFICE PRODUCTS
	Travelon Rfid Blocking Purse Organizer	bag, zippered, compartment, handles	APPAREL	LUGGAGE
ArXiv	Bias-Driven Revision of Logical Domain Theories	proof, theory, probabilities,information	cs.AI	stat.ML
	Sequence to Sequence – Video to Text	dynamics, frames, image, generation, words	cs.CL	cs.CV
	Online Learning via Sequential Complexities	statistical,prediction, sequential, minimax	cs.CL	cs.AI

short text by identifying significant words that add semantic relevance to the ambiguous short text. As we observe from the Table 5, this enables estimation of a better prediction model.

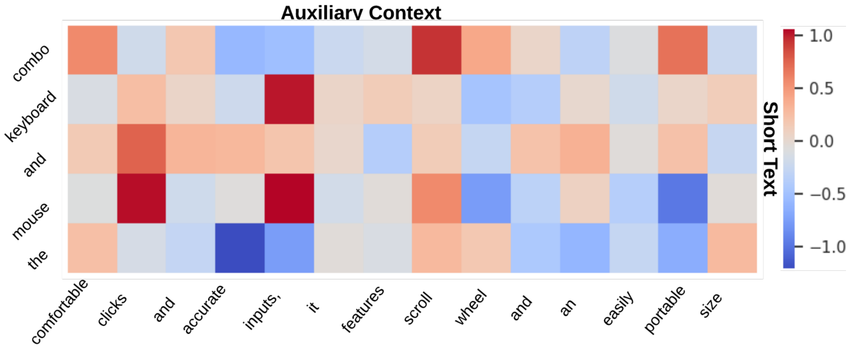


Fig. 5. Attention weights of sequence generation from short text to auxiliary context (best viewed in color). Figure illustrates activations for an example from the Amazon product catalog.

4.3.3 Attention weights for prediction. We analyze the segments of text that help in the sequence encoding and decoding in CGN. We run PACS on a sample test case and analyze the attention weights given to each word embedding during the sequence generation from short text to auxiliary context.

Figure 5 displays the attention weights for a sample sequence. We can observe that semantically-rich words (e.g., mouse, keyboard) receive more attention (higher weights) than stop words (e.g., and, the) for final prediction. Stop words do not provide discriminative features, and thus, have limited utility in the problems of classification. PACS enriches the feature set by ignoring stop words and focusing on semantically significant words through its attention modules. This improves discriminative classification and thus, we observe a corresponding increase in performance.

4.4 Ablation Study

We study the importance of various embedding layers and the presence of self-attention mechanism, and Bi-LSTMs in this section.

4.4.1 Embedding layers study. In this experiment, we replace the embedding layer in PACS and study the difference in performance.

- *Word2Vec (W2V)*: We train a skip-gram model [26] on the dataset and extract the word-level features for CGN encoder and decoder.
- *Pre-trained models*: We extract the features from the last layer of GPT, BERT, BART, and XLNet language models and utilize it in CGN.

Table 6. Significance of the embedding layer. Experiment to show the importance of the embedding layer. The embedding layers tested are W2V, GPT, BERT, BART, and XLNet.

Metrics	**Accuracy**			**F-score**		
# Classes	3	10	43	3	10	43
PACS (W2V)	85.4	59.7	31.24	.811	.553	.300
PACS (GPT)	85.5	59.8	31.21	.813	.553	.302
PACS (BERT)	85.4	59.8	31.23	.812	**.556**	.301
PACS (BART)	85.6	**59.9**	31.20	.811	**.556**	.301
PACS (XLNet)	**85.7**	59.6	**31.70**	**.814**	.555	**.305**

Table 6 provides the results for variants of PACS on the Amazon Reviews dataset. These results demonstrate that variation in embedding layer leads to insignificant change in performance. Hence, PACS does not rely on the quality of representations, but only leverages the layer for a dense dimensionality reduction from a one-hot sparse semantic space.

4.4.2 Self-attention and Bi-LSTM study. In this experiment, we study the individual contribution of the self-attention mechanism and Bi-LSTM component to the overall architecture of PACS. We test two variants of PACS, namely, one without self-attention (w/o SA) and one without Bi-LSTM (w/o BL). We test these variants against PACS on each of the datasets for short text classification. We perform the ablation by blocking weight updates to the ablated component during the training phase. Hence, the number of parameters and dimensions remain intact and fairly comparable. Additionally, we also train other model variants without any auxiliary context (w/o AC) and without any short text information (w/o ST) to analyze the significance of auxiliary context and short text, respectively, to the overall performance of the model.

Table 7 shows the results of the ablation study. In this experiment, we can observe that SA, Bi-LSTM, AC, and ST contribute 8%-30%, 2%-9%, 13%-46% and 10%-32% to the overall performance accuracy, respectively. Self-Attention mechanism captures the contribtuion of the inter-token

Table 7. Ablation study results on (a) Accuracy and (b) F-score measures. Performance comparison of the contributions from different components of the model such as Self Attention (SA), Bi-LSTM (BL), Auxiliary Context (AC), and Short Text (ST) to the overall performance. PACS (pipe) is the pipeline version of PACS that independently learns the short-text and auxiliary context features.

(a) Accuracy measures

Dataset	**Amazon**			**HuffPost**			**RT Movie**			**ArXiv**		
# Classes	3	10	43	3	10	41	3	5	12	3	10	41
(w/o SA)	60.6	42.9	24.3	57.9	35.7	16.7	65.3	59.7	47.3	59.1	35.8	17.1
(w/o BL)	78.2	54.8	29.5	68.4	46.7	22.2	74.6	69.5	62.9	68.7	47.1	22.4
(w/o AC)	46.5	30.1	17.6	44.3	26.6	12.7	48.4	45.4	36.8	45.4	27.5	13.8
(w/o ST)	53.6	36.5	21	51.2	31.2	14.8	56.9	52.6	42.1	52.3	31.7	15.5
PACS (pipe)	80.6	54.0	27.6	68.1	45.9	21.9	75.8	71.7	57.3	60.4	45.8	22.6
PACS (ours)	**85.7**	**59.6**	**31.7**	**74.3**	**51.3**	**24.4**	**81.9**	**75.0**	**68.8**	**74.6**	**52.5**	**25.2**

(b) F-score measures

Dataset	**Amazon**			**HuffPost**			**RT Movie**			**ArXiv**		
# Classes	3	10	43	3	10	41	3	5	12	3	10	41
(w/o SA)	.59	.41	.23	.55	.34	.16	.60	.59	.47	.60	.39	.17
(w/o BL)	.76	.44	.26	.63	.46	.20	.60	.66	.55	.69	.50	.32
(w/o AC)	.42	.25	.15	.42	.22	.11	.39	.37	.36	.48	.26	.23
(w/o ST)	.51	.33	.19	.49	.28	.14	.50	.48	.42	.54	.33	.21
PACS (pipe)	.61	.51	.27	.65	.47	.20	.71	.60	.59	.61	.43	.28
PACS (ours)	**.81**	**.56**	**.31**	**.69**	**.49**	**.23**	**.79**	**.70**	**.64**	**.71**	**.55**	**.34**

dependence to the overall representation, whereas, Bi-LSTMs model the sequential/positional information of the vectors. The lack of Bi-LSTM does not cause significant changes except in high number of classes. Also, we observe that auxiliary context plays the most prominent role in enhancing the prediction followed by SA. Thus, we conclude that auxiliary context generation and inter-token relation play a more vital role than sequential information.

4.4.3 Pipeline vs Joint-learning study. We conduct this experiment to analyze the difference between separately (pipeline) and jointly learning the features for short text and the corresponding auxiliary context. For this, we independently learn the features of short text with XLNet and train the CGN network to generate auxiliary context. We concatenate these independently learnt short text and auxiliary context features and input them to train a dense prediction network that outputs the final class label. The experiment is conducted on the same cross-validation split as the main PACS model.

From the results in Table 7, we observe that the joint-learning model is able to significantly outperform the pipeline model. The reason for these improvements is that the direct connection between the CGN and PN module which allows the network to capture the class signal and backpropagate it to optimize task-specific auxiliary context generation and feature extraction. Thus, we can conclude that jointly learning CGN is better for enhancing the features of short text.

4.4.4 Loss function study. We conduct this experiment to analyze the difference in PACS' performance based on different loss functions. For this study, we consider the standard loss functions for classification; cross entropy [9, 24], InfoNCE [27], and hinge loss [11], and compare it to our choice

Table 8. Analysis for the significance of the loss function. Performance comparison using different loss functions. The loss functions tested are cross entropy, InfoNCE, hinge loss, and KL-divergence.

Metrics	**Accuracy**			**F-score**		
# Classes	3	10	43	3	10	43
PACS (Cross Entropy)	75.0	48.9	20.8	.707	.445	.195
PACS (InfoNCE)	82.2	55.9	28.0	.778	.519	.269
PACS (Hinge Loss)	78.6	52.3	24.3	.741	.483	0.232
PACS (KL-divergence)	**85.7**	**59.6**	**31.7**	**.814**	**.555**	**.305**

- the KL-divergence loss. The experiment is conducted on the same cross-validation split as the main PACS model.

From the results in Table 8, we observe that the KL-divergence loss outperforms the other loss functions in the comparative study. Thus, we can conclude that KL-divergence loss is better for generating pseudo-auxiliary context and improving classification performance.

5 CONCLUSION

We reformulated the short text classification as a self-supervised learning problem that leverages the sample's auxiliary context through conditional sequence generation. Furthermore, a predictor network then utilizes the short text encoding and the generated sequence as features for the final class prediction. We developed Pseudo-Auxiliary Context generation network for Short text modeling (PACS) to employ the reformulation and comprehensively leverage the auxiliary context for the problem of short text classification. The network consists of two sub-modules: context generation network and prediction network. PACS jointly trains the sub-modules in an end-to-end self-supervised learning framework to exclusively capture features relevant to the class prediction. We evaluated PACS through comparative studies against state-of-the-art baselines on benchmark datasets. Our experiments indicate that PACS outperforms several baselines on short text classification using popular metrics such as accuracy and F-score. Additionally, we also performed qualitative interpretability analysis to identify the function of inner mechanisms for some sample cases. Through the trained weights, we observed that the positive contribution of auxiliary context increases with the number of classes. The attention weights demonstrate the effectiveness of context generation. Furthermore, we also performed an ablation study to comprehend the contribution of individual networks to the overall architecture.

Acknowledgments

This work was supported in part by the US National Science Foundation grants IIS-1838730 and Amazon AWS credits.

REFERENCES

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. TensorFlow: A System for Large-Scale Machine Learning. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation* (Savannah, GA, USA) *(OSDI'16)*. USENIX Association, USA, 265–283.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1409.0473

[3] David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research* 3, Jan (2003), 993–1022.

[4] William B. Cavnar and John M. Trenkle. 1994. N-Gram-Based Text Categorization. In *In Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, Vol. 161175. Citeseer, 161–175.

[5] Jindong Chen, Yizhou Hu, Jingping Liu, Yanghua Xiao, and Haiyun Jiang. 2019. Deep short text classification with knowledge powered attention. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 6252–6259.

[6] Nurendra Choudhary, Nikhil Rao, Sumeet Katariya, Karthik Subbian, and Chandan K. Reddy. 2022. ANTHEM: Attentive Hyperbolic Entity Model for Product Search. In *WSDM '22: The Fifteenth ACM International Conference on Web Search and Data Mining, Phoenix, AZ, USA, February 21-25, 2022* (Phoenix, AZ, USA) *(WSDM '22)*. Association for Computing Machinery, New York, NY, USA.

[7] Nurendra Choudhary, Rajat Singh, Ishita Bindlish, and Manish Shrivastava. 2018. Neural network architecture for credibility assessment of textual claims. *arXiv preprint arXiv:1803.10547* (2018).

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, Jill Burstein, Christy Doran, and Thamar Solorio (Eds.). Association for Computational Linguistics, 4171–4186. https://doi.org/10.18653/v1/n19-1423

[9] Abhirup Dikshit and Biswajeet Pradhan. 2021. Interpretable and explainable AI (XAI) model for spatial drought prediction. *Science of The Total Environment* 801 (2021), 149797. https://doi.org/10.1016/j.scitotenv.2021.149797

[10] Evgeniy Gabrilovich, Shaul Markovitch, et al. 2007. Computing semantic relatedness using wikipedia-based explicit semantic analysis.. In *IJcAI*, Vol. 7. 1606–1611.

[11] Claudio Gentile and Manfred KK Warmuth. 1998. Linear hinge loss and average margin. *Advances in neural information processing systems* 11 (1998), 225–231.

[12] Allen Ginsberg, Sholom M Weiss, and Peter Politakis. 1988. Automatic knowledge base refinement for classification systems. *Artificial Intelligence* 35, 2 (1988), 197–.

[13] Jian Hu, Gang Wang, Fred Lochovsky, Jian-tao Sun, and Zheng Chen. 2009. Understanding user's query intent with wikipedia. In *Proceedings of the 18th international conference on World wide web*. 471–480.

[14] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. 2333–2338.

[15] Sergey Ioffe. 2006. Probabilistic linear discriminant analysis. In *European Conference on Computer Vision*. Springer, 531–542.

[16] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1412.6980

[17] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations (ICLR)*.

[18] Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *Twenty-ninth AAAI conference*.

[19] Ji Young Lee and Franck Dernoncourt. 2016. Sequential Short-Text Classification with Recurrent and Convolutional Neural Networks. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, Kevin Knight, Ani Nenkova, and Owen Rambow (Eds.). The Association for Computational Linguistics, 515–520. https://doi.org/10.18653/v1/n16-1062

[20] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Online, 7871–7880. https://doi.org/10.18653/v1/2020.acl-main.703

[21] Chenliang Li, Yu Duan, Haoran Wang, Zhiqian Zhang, Aixin Sun, and Zongyang Ma. 2017. Enhancing Topic Modeling for Short Texts with Auxiliary Word Embeddings. *ACM Trans. Inf. Syst.* 36, 2, Article 11 (aug 2017), 30 pages. https://doi.org/10.1145/3091108

[22] Zhouhan Lin, Minwei Feng, Cícero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A Structured Self-Attentive Sentence Embedding. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net. https://openreview.net/forum?id=BJC_jUqxe

[23] Robert Mac Gregor. 1991. The evolving technology of classification-based knowledge representation systems. In *Principles of semantic networks*. Elsevier, 385–400.

[24] Shie Mannor, Dori Peleg, and Reuven Rubinstein. 2005. The Cross Entropy Method for Classification. In *Proceedings of the 22nd International Conference on Machine Learning* (Bonn, Germany) *(ICML '05)*. Association for Computing Machinery, New York, NY, USA, 561–568. https://doi.org/10.1145/1102351.1102422

[25] Yu Meng, Jiaming Shen, Chao Zhang, and Jiawei Han. 2019. Weakly-supervised hierarchical text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 6826–6833.

[26] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.

[27] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748* (2018).

[28] Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, and Rabab Ward. 2016. Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 24, 4 (2016), 694–707.

[29] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*. Association for Computational Linguistics, 79–86.

[30] Mykola Pechenizkiy, Seppo Puuronen, and Alexey Tsymbal. 2003. Feature extraction for classification in knowledge discovery systems. In *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*. Springer, 526–532.

[31] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 1532–1543.

[32] Matt Post and Shane Bergsma. 2013. Explicit and implicit syntactic features for text classification. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. 866–872.

[33] Yanru Qu, Han Cai, Kan Ren, Weinan Zhang, Yong Yu, Ying Wen, and Jun Wang. 2016. Product-based neural networks for user response prediction. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, 1149–1154.

[34] Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. ELRA, Valletta, Malta, 45–50.

[35] Tian Shi, Kyeongpil Kang, Jaegul Choo, and Chandan K Reddy. 2018. Short-text topic modeling via non-negative matrix factorization enriched with local word-context correlations. In *Proceedings of the 2018 World Wide Web Conference*. 1105–1114.

[36] Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning*. Association for Computational Linguistics, 1201–1211.

[37] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. 2012. LSTM neural networks for language modeling. In *Thirteenth annual conference of the international speech communication association*.

[38] Kshitij Tayal, Nikhil Rao, Saurabh Agarwal, Xiaowei Jia, Karthik Subbian, and Vipin Kumar. 2020. Regularized Graph Convolutional Networks for Short Text Classification. In *Proceedings of the 28th International Conference on Computational Linguistics: Industry Track*. International Committee on Computational Linguistics, Online, 236–242. https://doi.org/10.18653/v1/2020.coling-industry.22

[39] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In the Proceedings of ICLR.

[40] Fang Wang, Zhongyuan Wang, Zhoujun Li, and Ji-Rong Wen. 2014. Concept-based short text classification and ranking. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*. 1069–1078.

[41] Paul J Werbos. 1990. Backpropagation through time: what it does and how to do it. *Proc. IEEE* 78, 10 (1990), 1550–1560.

[42] Jason Weston, Sumit Chopra, and Antoine Bordes. 2015. Memory Networks. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1410.3916

[43] Yi Yang, Hongan Wang, Jiaqi Zhu, Yunkun Wu, Kailong Jiang, Wenli Guo, and Wandong Shi. 2020. Dataless Short Text Classification Based on Biterm Topic Model and Word Embeddings. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, Christian Bessiere (Ed.). International Joint Conferences on Artificial Intelligence Organization, 3969–3975. https://doi.org/10.24963/ijcai.2020/549 Main track.

[44] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*. 5754–5764.

[45] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*. 1480–1489.

[46] Jichuan Zeng, Jing Li, Yan Song, Cuiyun Gao, Michael R. Lyu, and Irwin King. 2018. Topic Memory Networks for Short Text Classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (Eds.). Association for Computational Linguistics, 3120–3131. https://doi.org/10.18653/v1/d18-1351

[47] Lu Zhang, Jiandong Ding, Yi Xu, Yingyao Liu, and Shuigeng Zhou. 2021. Weakly-supervised Text Classification Based on Keyword Graph. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. 2803–2813.

[48] Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*. 649–657.

[49] Xiao Zhou, Cecilia Mascolo, and Zhongxiang Zhao. 2019. Topic-enhanced memory networks for personalised point-of-interest recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 3018–3028.

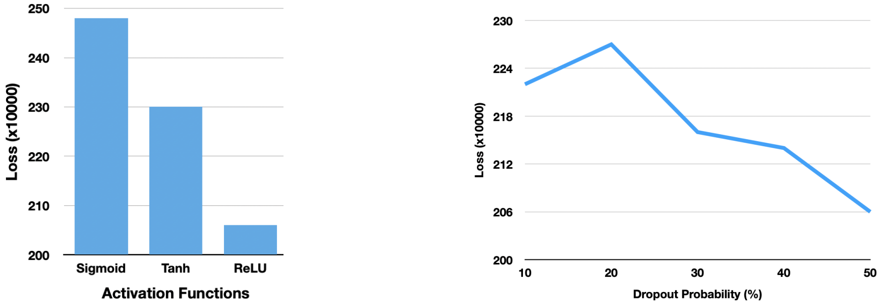
[50] Yuan Zuo, Congrui Li, Hao Lin, and Junjie Wu. 2021. Topic Modeling of Short Texts: A Pseudo-Document View with Word Embedding Enhancement. *IEEE Transactions on Knowledge and Data Engineering* (2021), 1–1. https://doi.org/10.1109/TKDE.2021.3073195

A HYPER-PARAMETER TUNING

To support reproducibility, this section discusses the sensitivity of PACS on the initial hyper-parameters. Furthermore, we provide the details of hyper-parameters utilized in our final experimental setup. We vary the following hyper-parameters and analyze its impact on the model performance and loss.

A.1 Activation Function

We evaluate three activation functions - Sigmoid, Tanh and ReLU. Qu et al. [33] show that ReLU and Tanh are better than sigmoid for deep models. Figure 6a presents a similar finding for our PACS model. ReLU performs slightly better than Tanh, possibly because it induces sparsity for least significant values.



(a) Activation Function vs Loss

(b) Dropout vs Loss

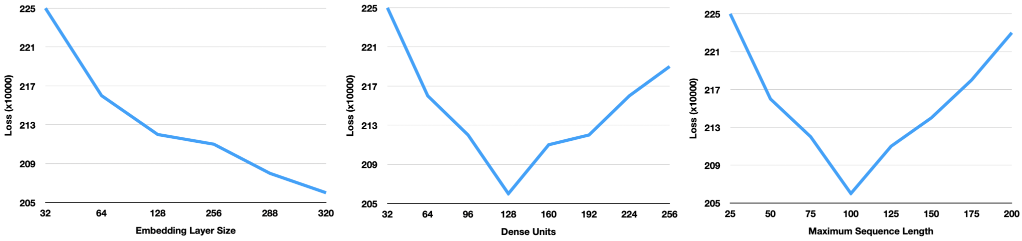
Fig. 6. Sensitivity of PACS to hyper-parameters; (a) Activation function and (b) Dropout probability.

A.2 Dropout Values

Figure 6b demonstrates PACS' loss on five dropout probabilities: 0.1, 0.2, 0.3, 0.4 and 0.5. We observe that dropout does not significantly effect the model's loss. Hence, we choose 0.5 because it reduces the number of parameter updates.

A.3 Embedding Size

Figure 7a demonstrates that increasing the embedding improves model's convergence loss. The reason is that additional parameters capture better features for word's reconstruction. Also, it reveals that decreasing k may decrease the loss further. However, due to computational restrictions, we utilize the maximum possible $k = 1000$. We believe, given higher memory, the performance can improve for higher values of k.



(a) Embedding Layer Size vs Loss (b) Dense Units vs Loss (c) Max. Sequence Length vs Loss

Fig. 7. Sensitivity of PACS to hyper-parameters; (a) Embedding Layer size, (b) Dense units, and (c) Maximum Sequence Length.

A.4 Dense Units

Figure 7b displays that increasing the dense units from 32 to 64 decreases the model's convergence loss. However, increasing the number further to 128 and 256 results in an increase in the convergence loss. This is because the increase in number of parameters requires more epochs to optimize for the minimum. However, computational restraints inhibit our ability to spend more epochs.

A.5 Maximum Sequence Length

Figure 7c depicts that increasing the sequence length from 25 to 100 improves the model's convergence loss. However, increasing it further captures redundant information and leads to a decrease in performance. Hence, we set it to 100 in our experimental setup. Based on the above empirical studies, the final hyper-parameters in our experimental setup are given in Table 3.

B TRAINING PHASE

Figure 8 illustrates the training and validation loss over epochs during the training phase. We observe that the model converges in $\approx$ 40 epochs. The loss function is categorical cross entropy.

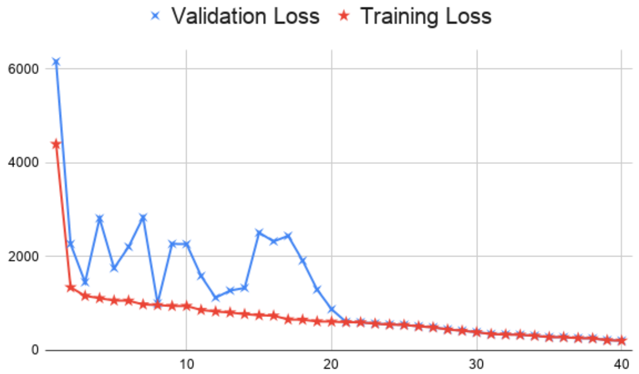


Fig. 8. Epochs vs Training Loss and Validation Loss.