# Improving Text Representations with Language Models: Contrastive Learning, Personalization, and Data Augmentataion

by

Ruixue Lian

A dissertation submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

(Electrical and Computer Engineering)

at the

UNIVERSITY OF WISCONSIN–MADISON

2023

Date of final oral examination: 08/01/2023

## ACKNOWLEDGMENTS

My deepest gratitude first and foremost goes to my advisor Dr. William Sethares. He gave me patience, trust, and support to grow, especially giving me the freedom to explore the area and formulate the problems - things that not a lot of Ph.D. students are lucky enough to get from their advisors. I would also like to express my gratitude to my co-advisor, Dr. Junjie Hu. Although we have only been working together for a short period of time, he always asked the right questions that pointed me in the right direction. More importantly, he also asked the tough questions that pushed me forward in the NLP area. I would like to sincerely thank Dr. Dhavan Shah for the many fascinating collaborations and his constant support. I am thankful to Lew Friedland, Mike Wagner, Sijia Yang, Xiaoya Jiang, Yiming Wang, Yini Zhang, Jiyoun Suk, and other professors and students from SMAD and CCCR groups, who have inspired and encouraged me from a host of perspectives different from the engineers and computer scientists.

Bill has introduced me to various real-world problems, and we collaborate with different people across the departments of the university by applying ML/NLP methods. One that impressed me greatly was related to the image denoising of ancient papers, especially LeoCode [1], in which we worked on the papers of Leonardo da Vinci. It was amazing to work with Bill and his Ph.D. advisor Dr. Richard Johnson. We proposed the image denoising method and introduced our techniques to museum conservators and art historians. Although not part of this thesis, this is one of my favorite works.

I would also like to express my appreciation to Robert Nowark, Kangwook Lee, Varun Jog, Po-Ling Loh, John Gubner, and other professors who have taught me fundamental knowledge, especially on machine learning

[1]https://leocode.org/

theory. I want to thank Dr. Irena Knezevic, who has greatly influenced my attitude and efforts toward my work and life. I would like to thank Zhongkai Sun, Elisa Ou, Liang Shang, Rheeya Uppaal, Yuye Jiang, Siyang Chen, Nathan Kalmoe, Leyue Zhang, Ying Xue and other friends for their consistent help in life and for having fun together. Last, I would like to thank my parents for their unresolved love and support.

My Ph.D. journey is going along with the exponential surging of language models, starting from BERT to ChatGPT. I am very excited to work with these cutting-edge models, not only by proposing novel methodologies but also applying them to real-world problems. These things are not likely to happen without them.

# CONTENTS

## ABSTRACT

Text representation learning is a fundamental task in NLP. It aims to capture the semantic, syntactic, and contextual information present in textual data and transform it into a numerical representation that machine learning models can effectively utilize. Effective text representations are crucial for various downstream tasks such as text classification, information retrieval, machine translation, and question answering, etc.

This thesis presents various methodologies to improve sentence-level text representations with language models. Given the fact that subcategories under the same branches are similar and closer to each other, we propose variations of label-aware supervised contrastive loss (LA-SCL) to incorporate and learn the label hierarchy information. Two studies were conducted to improve text representations with personalized features, which can be tailored to specific users by incorporating individual information such as user demographics, or contextual factors. A novel model architecture called personalized transformer memory (PersonalTM) was proposed to effectively involve personalized features given a transformer-based encoder-decoder model on information retrieval tasks. An incremental learning algorithm was proposed to continuously keep the model up-to-date by incorporating streaming personalized data. A study was conducted to perform data augmentation by utilizing GPT-2 to generate text data, specifically focusing on producing utterances that are similar to original data in both semantics and sentiment.

Several practical tasks in Social Media Analysis are also conducted in this thesis: several language models were employed to assess the favorability and hesitancy towards COVID-19 on tweets, analyzing their ideological perspectives. Machine learning models were applied to analyze the dynamics of sexual violence and gender justice discourses across four social media platforms.

## LIST OF TABLES

## LIST OF FIGURES

# 1 INTRODUCTION AND BACKGROUND

## 1.1 Introduction

Text representation learning is a fundamental task in Natural Language Processing (NLP) that aims to capture the rich semantic and contextual information encoded in textual data. It involves transforming raw text into numerical representations that machine learning models can easily process. Effective text representations are crucial for various NLP tasks, such as text classification, information retrieval, machine translation, question answering, etc.

Traditionally, the learning of text representations relied on handcrafted features and linguistic knowledge. These approaches involved engineering features like Word2Vec [25], bag-of-words [156], n-grams [28], and syntactic structures, in order to capture relevant information. However, these methods often suffered from limitations in capturing the complex and nuanced nature of language, and the many contextual dependencies that characterize human language.

In recent years, the advent of language models, particularly transformer-based [124] models like BERT [33] and GPT [11], revolutionized text representation learning. Language models are trained on large-scale corpora and can learn powerful contextual representations of words, sentences, or entire documents. By leveraging self-supervised learning techniques (e.g., masked language modeling (MLM), contrastive learning), these models capture rich linguistic patterns and semantic relationships, enabling them to generate high-quality and contextually-aware text representations.

However, it is essential to understand the limitations of language models in text representation learning. While they excel at capturing context and semantics, language models often struggle with interpretability and generalization to out-of-domain or low-resource scenarios. In this thesis, we explore the advances in text representation learning facilitated by such language models. We delve into the capabilities of language models in capturing contextual information and their impact on downstream NLP tasks.

## 1.2 Text Data Augmentation

Data augmentation is a commonly used technique when the size of a training set is smaller than desired, a frequent issue due to the costs of obtaining labeled data. It aims to increase the diversity of training examples without collecting additional data and to increase the generalization capabilities of a model [39, 75, 5]. On the other hand, self-supervised learning attracts many researchers in the last several years for its good performance on representation learning by leveraging the large amounts of unlabeled data [84, 87]. This chapter first introduces several existing text data augmentation methods. Then we introduce several widely used self-supervised learning techniques in NLP, with a focus on contrastive learning.

### Unlearnable Augmentation

Some popular data augmentation approaches in NLP aim to generate label-invariant text by replacing words with their synonyms. It obtains a 'new' expression while keeping the semantics of the text as similar to the original data as possible. This method relies on the construction of external dictionaries such as WordNet [95], which sorts the synonyms of words according to their similarity [153]. Another approach is to rely on rules to transform text without changing sentence semantics. For example, [111] conducts replacements from expanded to abbreviated form and inversely between a group of words and the corresponding abbreviation. Similarly, dependency trees are used in [26] to generate new sentences.

For instance, "Sally embraced Peter excitedly" could be replaced by "Peter was embraced excitedly by Sally".

Other research creates variety by adding a small amount of noise to the original data without significantly affecting the semantics. Alternatively, the utterances can be augmented by randomly swaping, deleting, inserting, or substituting with strings that are not semantically close to the original data. These methods make the altered text deviate from the original data. They not only expand the amount of training data but can also improve model robustness.

Easy data augmentation (EDA) is a widely used augmentation technique for boosting performance on text classification tasks [133]. It performs synonym replacement, random swap, random insertion, and random deletion on the word level. These techniques can be extended to the sentence level as well. For instance, [90] divides tweets into two halves and then randomly samples and combines the first and second halves of tweets from the same classes. In this way, although the generated data may be ungrammatical and/or nonsensical, they can still preserve emotional semantics.

Instead of modifying discrete words, the aforementioned methods can be applied at the embedding level [129, 94]. For instance, the original words can be replaced with the words closest to them in the vector space by using pretrained word vectors (such as Glove, Word2Vec, and FastText, etc.) Specifically, each word vector may be replaced with one of its k-nearest neighbor vectors or be replaced by a vector which has close cosine similarity.

These methods are easy to implement but limit the scope of replacement. It may also cause the ambiguity or even completely change the sentence semantics if too many replacements occur. Furthermore, these methods are not trainable.

## Augmentation via Machine Translation

Some methods attempt data augmentation by using non-pretrained sequence-to-sequence models using the idea of back translation [140, 155]. For example, it is possible to translate the original document into another language, and then translate back, thus obtaining a new text in the original language (which hopefully preserves a significant amount of the meaning). Thus machine translation methods do not directly replace individual words but rather rewrite the complete sentence in an automated way. For instance, Google's cloud translation API [26] is a popular tool that performs language translation. This method often returns correct grammar and unchanged semantics but the fixed machine translation models may have poor controllability and limited diversity.

## Mixup Augmentation

Mixup augmentation, which interpolates embedding vectors to generate new samples, is a promising method proposed recently [150]. The main idea of Mixup is simple: given two labeled points $(x_i, y_i)$ and $(x_j, y_j)$, where $y$ can be one-hot representation of the label. When using mixup, the labels are no longer represented as one-hot vectors, but rather as continuous probability distributions. New training samples $(x', y')$ can be created by linear interpolating these two points as shown in (1.1) and (1.2), where $\lambda \in [0, 1]$.

$$x' = \lambda x_i + (1 - \lambda) x_j \tag{1.1}$$

$$y' = \lambda y_i + (1 - \lambda) y_j \tag{1.2}$$

Such linear interpolation of the pixels of random image pairs (and their labels) has shown superior performance in enhancing the performance of image classification models [70, 52, 123, 70, 145, 126, 38]. Some research has extended mixup data augmentation to the field of NLP. [49] proposed wordMixup and senMixup: one performs interpolation on each dimension of each of the words in a sentence and another on a pair of sentence embeddings. MixText [17] creates a large amount of augmented training samples by interpolating text in a hidden space. Mixup-Transformer [119] integrates mixup interpolation into transformer-based language model such as BERT.

These methods perform interpolation in a linear manner, which significantly limits the space of the synthetic samples and consequently its regularization effect. To deal with this limitation, [49] proposed Nonlinear Mixup, which utilizes nonlinear interpolation policies for both the input and label pairs. But unlike wordMixup, where all the words share the same scalar mixing policy $\lambda$, non-linear mixup deploys a separate mixing policy for each of the dimensions of each of the words in a given sentence. In addition, [144] proposed SSMix, a novel mixup method where the operation is performed on input text rather than on hidden vectors. It synthesizes a sentence by replacing words in span-level and keeps most discriminative tokens in the mixed text using saliency scores, which measure how each portion of the data (tokens) affects the final prediction. In their experiments, they compute the gradient of classification loss $L$ with respect to the input embedding $e$, and use its magnitude as the saliency. They applied the L2 norm to obtain the magnitude of a gradient vector, which becomes a saliency measure for each token.

These mixup data augmentation methods have often shown superior model performance in low resource settings. This interpolation-based augmentation strategy is domain independent and requires little additional computational cost. It also improves model robustness and is capable of online learning. However, the semantics or syntax may be distorted after mixing. There is also limited diversity for each of the mixup methods.

## Augmentation by Language Models

Pretrained language models, especially the large scale transformer-based models have been widely used in recent years [65]. The training of most pretrained language models (PLMs) is based on self-supervised learning, which utilizes auxiliary tasks to mine information from massive unsupervised data sets, and to learn valuable representations for downstream tasks. For instance, BERT pretraining relies on masked language modeling (MLM) and often achieves superior performance. In the MLM objective, some input tokens are randomly masked by the special token [MASK], and the model attempts to reconstruct the corrupted tokens given their left and right context. This paradigm can also be used for generating a variety of new sequences by filling these masks as proposed by [89]. This method alleviates the problem of ambiguity since MLM considers the entire context. Moreover, to generate sentences that are more compatible with a given set of labels, [137] proposed conditional BERT, which alters BERT segmentation embeddings (The vanilla BERT combines word embeddings, position embeddings, and segmentation embeddings to form the representations) to label embeddings. These methods do not rely on task-specific knowledge and are applicable to any supervised natural language tasks.

Another approach is to exploit a generative pretrained transformer model such as GPT-2, which is a powerful language model pretrained on massive text. It has been widely used to produce coherent sentences and paragraphs. [2] and [72] use GPT-2 to generate sentences by first fine-tuning the model on the original data. Then the generated sentences are filtered by a classifier to encourage the data quality. Similarly, [108] propose label-conditioning GPT-2 to generate augmented data.

## 1.3 Contrastive Learning

The discrete nature of natural language makes data augmentation challenging to generate effective label-preserving transformations for text sequences that can improve model generalization. On the other hand, self-supervised pretraining of transformer-based language model has become the primary method for learning textual representations from large-scale unlabelled corpora. However, pretraining such as MLM at the token level does not explicitly learn semantics at the sequence level. To address these issues, the contrastive self-supervised learning (CSSL) has shown promising results in learning sequence representations in an unsupervised way [61]. The learned sequence representations can be then used on down-stream tasks such as classification or clustering. Multiple previous work [69] has effectively demonstrated that this learning mechanism significantly closes the performance gap between supervised learning and unsupervised learning.

The goal of contrastive learning is to learn an embedding space in which similar samples are close to each other and dissimilar ones are far apart. Fig.1.1 shows a simple framework for contrastive learning [20]. The idea of CSSL is to first create augments $\tilde{x_i}$ and $\tilde{x_j}$ of the original sample $x$. These augments have the same label as the original data. Two augments are considered to be a positive pair if they are created from the same original example and a negative pair otherwise. $f(.)$ is called the encoder and $g(.)$ is called the projector. The object is to distinguish augmented instances that originate from the same sentence from augmented instances that originate from different sentences. CSSL focuses on learning common features between positive pairs and distinguishing the dissimilarity of negative pairs. The objective function of CSSL is shown in Eq. (1.3). Since the first step of contrastive learning is to create augments of the original samples, any of the aforementioned data augmentation methods can be applied.

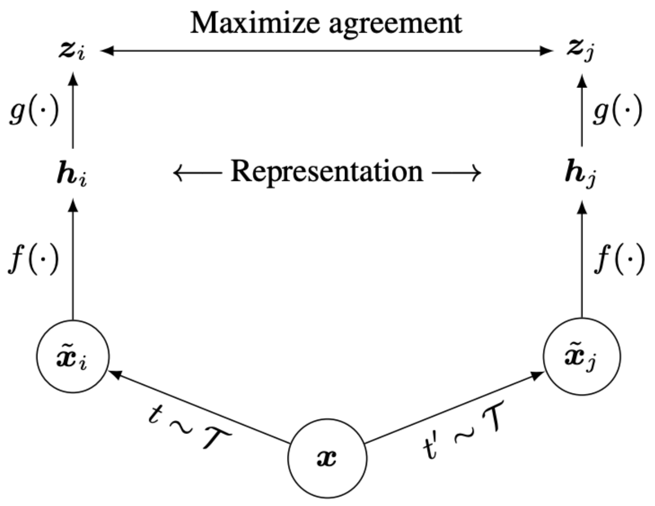


Figure 1.1: A simple framework for contrastive learning

$$l_{i,j} = -\log \frac{\exp(\text{sim}(z_i, z_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(\text{sim}(z_i, z_k)/\tau)} \tag{1.3}$$

Several papers have used contrastive learning in NLP [112]. For instance, [37] proposed CERT, which pretrains BERT using contrastive self-supervised learning at the sentence level. In their experiments, back-translation on original sentences is used to construct positive pairs. CLEAR [139] combines a word-level MLM objective with a sentence-level CL objective to pretrain the language model. They use augmentations including word and span deletion, reording, and substitution. Similarly, DeCLUTR [44] constructs positive augments combining text segments that overlap or are adjacent to the original text segment. Qu et al. [107] proposed CoDA, which synthesizes diverse and informative augmented examples by integrating multiple transformations such as back-translation, c-BERT word replacement, cutoff, mixup, and adversarial training. They further stack different label-preserving transformation in a sequential manner. Moreover, they leverage a contrastive learning objective and add a penalty between augmented samples $x_i$ and other training instances $x_j (j \neq i)$ to better utilize the augmented examples. COCO-LM [93] employs an auxiliary language model ELECTRA, to mask-and-predict tokens in original text sequences. [40] proposed simCSE, a simple contrastive learning framework, which presents an unsupervised approach that predicts the input sentence itself using dropout to create augmented sentences.

## 1.4 Personalization

Improving text representations with personalized features aims to enhance the quality and relevance of generated or represented text by incorporating individualized information such as user demographics, preferences, or contextual factors. Text representations can be tailored to specific users, creating more personalized and engaging content. The personalized information can enhance content recommendations, adapt the language style, extract user-specific information, and even consider emotional factors, with the goal of providing a more tailored and satisfying user experience.

## 1.5 Outline of this Document

The following chapters introduce the innovation of methodologies and applications. Chapter 2 introduces our work of learning label hierarchy with supervised contrastive learning. Chapter 3 and 4 introduce our work on personalization. Chapter 5 presents our work on two augmentation methods. Chapter 6 details some practical work in collaboration with SMAD/CCCR by applying ML/NLP methods to real-world problems. Chapter 7 talks about future directions.

# 2 LEARNING LABEL HIERARCHY WITH SUPERVISED CONTRASTIVE LEARNING

## 2.1 Overview

This chapter proposes modifications of supervised contrastive loss functions to include information about label hierarchies. The label information is also used as class centers; thus, additional terms can be added to the contrastive loss which include differences between instances and centers. These help to improve the representations further. The material in this chapter has been submitted to EMNLP 2023 [83].

## 2.2 Abstract

Supervised contrastive learning (SCL) frameworks traditionally treat each class as independent and thus consider all classes as equally important. It neglects the scenario when a semantic or sentiment label hierarchy exists, which is common in the real world. This paper introduces a family of Label-Aware SCL methods (LA-SCL) that leverage hierarchical and semantic relationships between class labels. Specifically, our LA-SCL methods automatically adjust the distance between instances based on the proximity of their label similarities. It reduces the distance between instances when they have similar high-level semantics or are under the same branch of one category. Another variation further improves intra-cluster compactness and inter-class separation by introducing each instance closer to its corresponding center, which is represented by label features. These modifications allow the algorithm to learn more discriminative feature representations with clearer decision boundaries. Moreover, the learned label embedding matrix can be directly used as a nearest neighbor classifier and applied to linear mapping. Experiments on two datasets show

that the proposed LA-SCL work well on multi-class text classification, outperforming the baseline supervised approaches on full datasets and few-shot cases.

## 2.3 Introduction

Self-supervised contrastive learning (SSCL) aims to learn generalized and discriminative feature representations with unlabeled data, which can be exploited using data augmentation techniques—two augments of the same data point are almost surely in the same class; these form the positive pairs. Different samples, with high likelihood, come from different classes, and hence form the negative pairs [20, 60]. The loss function of the algorithm is designed to draw positive pairs closer and to drive negative pairs apart.

Instance-wise contrastive learning creates an embedding space where instances are well-separated on a uniform sphere [128]. However, it does not consider semantic structures within the data since any two selected instances may be treated as a negative pair. This can be mitigated by supervised contrastive learning (SCL) that constructs positive pairs from the same class and negative pairs from different classes, thereby incorporating semantic information contained in the labels of the data. This narrows the gap between supervised and self-supervised learning [48].

These methods consider each class to be independent and consider all classes to be of equal importance, thus treating the problem as a flat multi-class classification problem without awareness of any relationships among the labels. However, in the real world, it is natural that class labels may relate to each other in complex ways, in particular, they may exist in a hierarchical structure [92, 31, 97, 125, 51].

Within a data hierarchy, different sub-categories under the same branch tend to be more similar than those from different branches, since they will tend to have similar high-level semantics, sentiment, and structure.

For example, “Animal, Corgi” is more similar to “Animal, Beagle” than either is to “Fruits, Apple” and this similarity should be reflected in the embedding representations. Thus the hierarchical structure of the labels and the shared semantics of the class labels suggest that learning methods may be enhanced if the learning mechanism can be made aware of the relationships among the labels. This chapter explores several ways of exploiting these hierarchical relationships between labels; representation learning of SCL with label hierarchical information is still under-explored.

Zeng et al. [146] augment the classification loss by the Cophenetic Correlation Coefficient (CPCC) [117] as a standalone regularizer to maximize the correlation between the label tree structure and class-conditioned representations. Inspired by this work, we propose to augment the SCL method to include hierarchical label information. Since this incorporates information about the relationships among the labels, we call this label-aware SCL, abbreviated LA-SCL. In our methods, the hierarchical label names or brief descriptions of the labels are encoded into embedding vectors, which are used as learnable label representations. The similarity between the label representations is calculated to form the learnable label similarity matrix, which is used to scale the temperature in the contrastive loss function in LA-SCL. The goal is to encourage samples under the same branches to cluster more closely while driving apart samples with labels that are more distant in the hierarchy.

In addition, we also propose to use label representations as the center of the sentence embeddings in LA-SCL. This is inspired by the approach of Li et al. [78] that proposes a ProtoNCE loss, a generalized version of the InfoNCE loss [100]. This learns a representation space by encouraging each instance to become closer to an assigned prototype such as the clustering centroid. In this way, the underlying semantic structure of the data can be encoded. Other related work considers semi-supervised learning and pseudo-labeling [36, 148, 143], which provide other ways to address this issue. Rather than encouraging instances to move closer to the centroid of the clusters, we introduce learnable label representations into the ProtoNCE loss. Since the dimension of these label representations is the same as the linear classifier, we show that it can be applied directly to the downstream classification without further training.

The contributions of this chapter may be summarized as follows:

- LA-SCL augments the SCL to include information about the label hierarchy.
- It improves intra-cluster compactness and inter-cluster separation by reducing the closeness between the instance and its corresponding center, a label representation in our case.
- It learns more discriminative feature representations and contributes to the performance gain on downstream multi-class classification.
- The learned label embedding matrix can be used as a nearest neighbor classifier and directly applied to linearly map.

## 2.4 Background

**Problem Setup** For a supervised multi-class classification task, a labeled dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$ consists of N examples from a joint distribution $P_{\mathcal{X}\mathcal{Y}}$, where $\mathcal{X}$ is the input space of all text sentences, $\mathcal{Y} = \{1, ..., C\}$ is the label space, and C is the number of classes. The goal of representation learning is to use $\mathcal{D}$ to learn a feature encoder $f_\theta : \mathcal{X} \to \mathcal{Z}$ that encodes a text sentence to a semantic sentence embedding in a feature space $\mathcal{Z}$. This allows us to measure the pairwise similarity between two text sentences $x_i, x_j$ by a similarity function $\text{sim}(x_i, x_j)$, which first projects $x_i$ and $x_j$ to $\mathcal{Z}$, i.e., $z_i = f_\theta(x_i)$, and compute a distance between two sentence embeddings in $\mathcal{Z}$. Moreover, learning meaningful embeddings facilitates the learning of a classifier $g_\phi : \mathcal{Z} \to \mathcal{Y}$ that maps learned embeddings to their corresponding labels.

**Supervised Contrastive Learning (SCL)** A major thread of representation learning approaches focuses on supervised contrastive learning [69] that encourage embedding proximity among examples in the same class (as implemented in the exp term in the numerator of Eq. (2.1)) while simultaneously pushing away embeddings from different classes (as implemented in the sum of exp terms in the denominator of Eq. (2.1)). Specifically, for a given example $(x_i, y_i)$, we denote $\mathcal{P}(y_i) = \{x_j | y_j = y_i, (x_j, y_j) \in \mathcal{D}\}$ as the set of sentences in $\mathcal{D}$ having the same label as $x_i$. Thus, the SCL loss is computed on $\mathcal{D}$ as:

$$\begin{aligned} \ell_{\text{SCL}}(x_i, y_i) &= \mathbb{E}_{x_j \sim \mathcal{P}(y_i)} \log \frac{\exp(\frac{\text{sim}(x_i, x_j)}{\tau})}{\sum\limits_{k \notin \mathcal{P}(y_i)} \exp(\frac{\text{sim}(x_i, x_k)}{\tau})} \\ \mathcal{L}_\tau(\mathcal{D}; \theta) &= \mathbb{E}_{(x_i, y_i) \sim \mathcal{D}}\ \ell_\tau(x_i, y_i), \end{aligned} \tag{2.1}$$

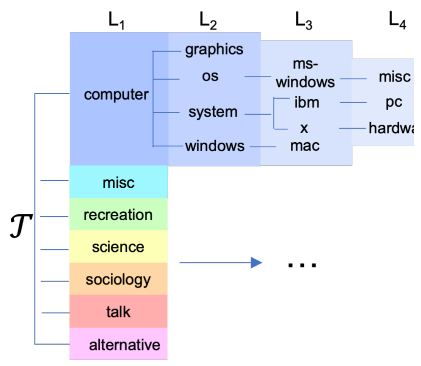
where $|\mathcal{P}(y_i)|$ denotes the size of $\mathcal{P}(y_i)$, and $\tau$ indicates a specific loss function used to compute the empirical expected loss over a dataset $\mathcal{D}$. The fixed hyper-parameter $\tau$ is the temperature that adjusts the embedding similarity of sentence pairs from any two classes.

## 2.5 Method

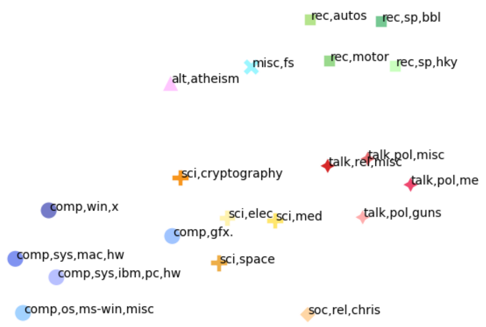
This section describes our proposed label-aware supervised contrastive learning objectives for representation learning.

**Overview:** Our method begins by computing initial label representations from a label hierarchy by a pre-trained language model. These label representations are used to learn a label similarity matrix that captures the label hierarchy information in Section 2.5. In the embedding space, we hypothesize that sentences from the same class or semantically similar classes are closer to each other when compared to sentences from semantically different classes. Subsequently, our method utilizes the label representations and similarity to instantiate three variants of label-aware supervised contrastive learning objectives in Section 2.5.

## Label Hierarchy Representations



(a)



(b)

Figure 2.1: (a) Label hierarchy of the 20NewsGroups dataset. The root node contains 7 classes. Each branch has multiple fine-grained subcategories. (b) t-SNE visualization of hierarchical label embeddings encoded by BERT-base.

A label hierarchy in a labeled dataset refers to a hierarchical tree structure that defines an up-down, coarse-to-fine-grained structure with the corresponding labels being assigned to their corresponding branches. We define a label hierarchical tree as $\mathcal{T}$, with $V$ being the set of intermediate and leaf nodes. Each leaf node $v_c$ is a class label $c \in \mathcal{Y}$, and is associated with a set of examples in the class $c$, i.e., $\mathcal{P}(c)$, where $\mathcal{P}(c) \cap \mathcal{P}(c') = 0$,

$\forall c \neq c'$. Each parent node is the high-level category of its children nodes. The leaf nodes can have different depths in $\mathcal{T}$, which refers to the distance between each leaf node $v_c$ and the root node $v_0$.

Figure 2.1a shows a tree-structured label hierarchy in the 20News-Groups dataset [10], which has 20 classes in 7 high-level categories. Let $L_i$ be the $i$-th layer of $\mathcal{T}$. Sentences can be first categorized into the 7 major classes: "Computer, Recreation, Science, Sociology, Talk, Alternative, Misc" at the $L_1$ level. Moving forward to the subsequent levels, each category can be further split into various fine-grained groups. For example, samples in "Computer" can be splited into four fine-grained groups: "Graphics, OS, System, Windows."

We exploit the hierarchical relationships among the classes and the importance of the labels by leveraging the textual description of the labels. To achieve this, for a given leaf node of class $c \in \mathcal{Y}$, we collect its ancestor nodes all the way to its highest level at $L_1$, and combine their labels into a text sequence. For instance, for a leaf node of "Hardware" at $L_5$, we collect the descriptions of its ancestor labels as "Computer, System, IBM, PC, Hardware", as shown in $\mathcal{T}$ in Figure 2.1a. We then use a template to fill in this combined label sequence to construct a sentence $u_c$, which is further fed into a pre-trained language encoder $f_\theta$ to obtain a label representation denoted as $u_c = f_\theta(u_c)$ for the label $c \in \mathcal{Y}$. As we reuse the language encoder $f_\theta$ for embedding text sentences in $\mathcal{D}$, the label representations are also changing during training. To stabilize the label representations, we propose to reuse the label representations for several iterations, thus updating the labels less frequently than the parameters of the model.

With the label representations for all classes $U = [u_1, \ldots, u_C]$, a pair-wise similarity measurement is applied to compute a label similarity matrix $W \in \mathbb{R}^{C \times C}$, where each entry is the similarity score between a label $c$ and another label $c'$, i.e., $w_{cc'} = \text{sim}(u_c, u_{c'})$. Note that the label embedding matrix $U \in \mathbb{R}^{d \times C}$ can be used as a nearest-neighbor classifier, where it can be directly applied to linearly map an input sentence embedding $x_i \in \mathbb{R}^d$ into the label space $\mathcal{Y}$. Therefore, we can use $U$ as a linear head for the downstream classification without further training.

Figure 2.1b shows the t-SNE [91] visualization of 20 label embeddings on the 20NewsGroup data set extracted by a BERT-base model. Different high-level and lower-level classes are presented by different markers and colors. We find that labels from the same coarse-grained classes are clustered closer to each other. Since clustering properties exist, these label representations will be utilized as additional information to scale the importance of different classes as introduced in the next section.

## Label as Class Centers

The label embeddings are also used as class centers to augment the loss in Eq. (2.1) by adding another loss term $\ell_{ic}$ performing instance-center-wise contrastive as shown in Eq. (2.2). This loss term regards the label sequence $u_c$ constructed for the label c as the center of the sentences from this class. Thus, for each input instance $x_i$, we can construct a positive pair between the instance and its center as $(x_i, u_{y_i})$, and similarly construct negative pairs by comparing the instance $x_i$ with the other label sequences, $(x_i, u_{y_k}), \forall y_k \neq y_i$. Note that this objective is to pull each sentence closer to its center and further apart from the other centers.

$$\ell_{ic}(x_i, y_i) = \log \frac{\exp(\frac{\text{sim}(x_i, u_{y_i})}{\tau})}{\sum_{k \notin \mathbb{P}(i)} \exp(\frac{\text{sim}(x_i, u_{y_k})}{\tau})}. \tag{2.2}$$

## Label Scaling

To introduce the similarity between classes, the key idea is to use the label similarity matrix $W$ to scale the temperature $\tau$ of the updating process by weighting different sample updates according to a measure of their distance or proximity to the labels. Specifically, the negative example pairs in SCL are weighted by their label similarity, performing scaled instance-instance-wise contrastive. The final loss over a dataset $\mathbb{D}$ is the same form as Eq. (2.1) with the individual loss $\ell_\tau$ replaced as follows:

$$\ell_{sii}(x_i, y_i) = \mathbb{E}_{j\sim\mathbb{P}(y_i)} \log \frac{\exp(\frac{\text{sim}(x_i,x_j)}{\tau})}{\sum\limits_{k\notin\mathbb{P}(y_i)} \exp(\frac{\text{sim}(x_i,x_k)}{\tau\cdot s_{ik}})}, \tag{2.3}$$

where we denote $s_{ik} = w_{y_i,y_k}$ as the label similarity between $y_i$ and $y_k$ extracted from the matrix $W$ for notation simplicity.

Eq. (2.3) scales the similarity between negative pairs by applying their label similarity. Consider two samples $x_i$ and $x_k$ from different classes $y_i$ and $y_k$. The similarity $s_{ik}$ tends to be greater if $y_i$ and $y_k$ have the same parent category. Thus, it applies a higher penalty to the negative pairs when they are from different coarse-grained categories, so the learning update tends to push them further apart. Similarly, if applies a lower penalty to the negative pairs when they are from the same coarse-grained categories but in different fine-grained classes. In this way, the label hierarchical information is introduced to assign different penalties, reflecting the similarities and dissimilarities between classes.

Similarly, the temperature in $\ell_{ic}$ can be scaled by the label similarity $s_{ik}$, and thus we can construct a scaled instance-center-wise contrastive loss term $\ell_{sic}$ in Eq. (2.4).

$$\ell_{sic}(x_i, y_i) = \log \frac{\exp(\frac{\text{sim}(x_i,u_i)}{\tau})}{\sum_{k\notin P(i)} \exp(\frac{\text{sim}(x_i,u_k)}{\tau\cdot s_{ik}})}. \tag{2.4}$$

## Label-Aware SCL Variants

Based on the aforementioned methods, we propose the following four label-aware SCL (LA-SCL) family variants, depending on how the label information is incorporated.

**Label-aware Instance-to-Instance (LI)** We call the first variant the *instance-to-instance* that weights the negative example pairs in SCL by their label similarity as shown in Eq. (2.3).

**Label-aware Instance-to-Unweighted-Center (LIUC)** The second variant, which we call the *instance-to-unweighted-center* denoted $\ell_{\text{LIUC}}$, which is the combination of the vanilla SCL loss $\ell_{\text{SCL}}$ and the unweighted $\ell_{ic}$.

$$\ell_{\text{LIUC}} = \ell_{\text{SCL}} + \ell_{ic} \tag{2.5}$$

**Label-aware Instance-to-Center (LIC)** We call the third variation the *instance-to-center* and is denoted as follows:

$$\ell_{\text{LIC}} = \ell_{sii} + \ell_{ic} \tag{2.6}$$

**Label-aware Instance-to-Scaled-Center (LISC)** We call the fourth variation the *instance-to-scaled-center* denoted $\ell_{\text{LISC}}$ as follows:

$$\ell_{\text{LIC}} = \ell_{sii} + \ell_{sic} \tag{2.7}$$

## 2.6 Experimental Settings

| Dataset | train/val/test (original) | train/val/test (LP) | classes ($L_1/L_n$) |
|---|---|---|---|
| 20News | 10,183/1,131/7,532 | 2,263/2,263/7,532 | 7/20 |
| DBPedia | 238,533/2,409/60,794 | 12,048/12,048/60,794 | 9/70 |

Table 2.1: Dataset statistics

**Datasests** 20NewsGroups [10] and DBPedia [3] are used. We leverage the label structures and textual labels originally provided. For DBPedia, we only take the first two levels of the label. The statistics is shown in Table 2.1. We randomly pick up samples from the original training set to construct LP data.

**Sentence Template for Labels** For 20NewsGroups and DBPedia, we use the following sentence templates to fill in labels.

- 20NewsGroups: "It contains {$\text{label}_i$} news."
- DBPedia: "It contains {$\text{label}_i[L_2]$} under {$\text{label}_i[L_1]$} category."

**Implementation Details** We use *bert-base-uncased* provided in huggingface's packages [136] as our backbone models. The averaged word embeddings of the last layer are used as feature representations. For training, we used learning rate 1e-5 with linear scheduler and weight decay 0.1. The model is trained with 20 epochs and validated every 256 steps. The best checkpoints were selected with a patience of 5 according to evaluation metrics. For LP, we use a learning rate of 5e-3 with a weight decay of 0.01. The classifier was trained with 10 epochs and validated after each epoch. The batch size and max sequence length are 32 and 128, respectively, across all the experiments. The temperature $\tau$ is 0.3. During training, we re-encode the label embeddings every 500 steps.

**Evaluation Metrics** We report the following evaluation metrics: (1) classification accuracy on the leaf node called **nodeAcc** (2) classification accuracy on the parent node of the leaf, which is called **midAcc**, (3) classification accuracy on the root node, which is the highest level of each branch and is called **rootAcc**.

## 2.7 Results and Analysis

### Main Results

| Dataset | Objective | direct test | | | linear probe | | |
|---|---|---|---|---|---|---|---|
| | | nodeAcc | midAcc | rootAcc | nodeAcc | midAcc | rootAcc |
| 20NewsGroups | SCL | 54.44 | 61.74 | 69.41 | 65.64 | 72.54 | 78.98 |
| | LI | 61.01 | 67.19 | 73.09 | 67.59 | 74.04 | 79.82 |
| | LIUC | 61.09 | 69.62 | 79.17 | 66.42 | 73.66 | 79.67 |
| | LIC | 69.40 | 75.64 | 81.05 | 68.32 | 75.21 | 80.87 |
| | LISC | 69.45 | 75.90 | 81.08 | 68.47 | 75.33 | 81.07 |
| DBPedia | SCL | 2.42 | – | 38.26 | 96.00 | – | 96.79 |
| | LI | 2.84 | – | 31.25 | 96.14 | – | 96.80 |
| | LIUC | 91.34 | – | 94.65 | 96.00 | – | 96.79 |
| | LIC | 94.85 | – | 96.30 | 96.52 | – | 97.25 |
| | LISC | 95.52 | – | 97.06 | 96.71 | – | 97.35 |

Table 2.2: Classification accuracy (%) in terms of the leaf, mid-layer, and root nodes with models trained on SCL, LI, LIUC, LIC, and LISC on 20NewsGroups and DBPedia datasets.

**LA-SCL beats SCL without further training.** Table 2.2 shows the performance on downstream multi-class classification tasks with different objectives. The experimental results are reported with linear probes (LP) and with direct testing (DT). For LP, we applied a randomly initialized linear layer as a classifier, and it was trained on a small number of labeled samples with the encoder frozen. DT refers to applying the label-representative parameters as the classifier without any further training, as in Section 2.5. Specifically, the test sentence embeddings are multiplied by the parameters representing the label embedding matrix to obtain the probability scores. For SCL, we first used the trained model to extract the label embedding matrix and then applied it to DT. For DBPedia, since there are two layers for each of its labels, there is no midAcc.

From Table 2.2, all four LA-SCL objectives outperform SCL on DT in terms of the accuracy of the leaf node, mid-layer, and root level metrics. Compared to SCL, LI improves the accuracy by effectively penalizing the importance between classes on 20NewsGroups and to a lesser extent on the DBPedia, which is slightly better than chance. This suggests the performance gains are more beneficial on finer-grained hierarchies. Moreover, compared to SCL, the additional contrastive loss between instance and their respective centers introduced by LIUC also induces performance gains, especially on DBPedia. Samples within each cluster move closer and gather around the cluster centers.

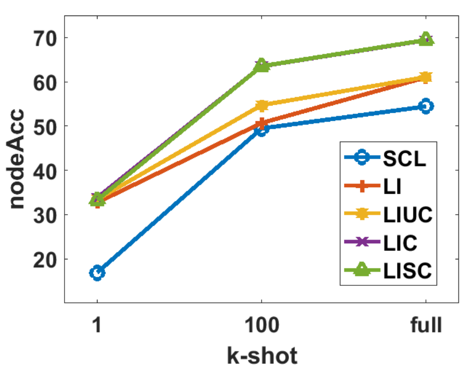
Furthermore, LIC contributes to a significant improvement by generating better label-representative parameters. Besides grouping similar groups together by introducing a weight matrix to the instance-wise contrastive learning loss, LIC drives instances closer to their centers. In contrast, LISC, by weighing only the class centers, provides only a marginal improvement because it only introduces small adjustments to the feature space (when compared to LIC).

In most cases, LP can enhance the performance compared to DT, while maintaining a comparable performance across different objectives. The performance gain introduced by LIC and LISC is substantial enough to narrow the performance gap between DT and LP. In particular, DT performs better than LP on 20NewsGroups, indicating the creation of better feature representations.

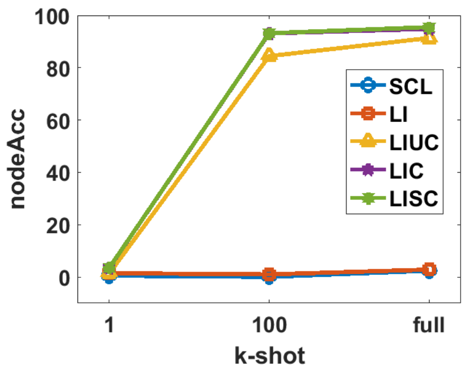
## Few-Shot Cases

**LISC/LIC work well on few-shot cases.** We also conduct experiments with a k-shot setup, with k=1 and k=100. To be specific, we take 1 and 100 sentences from each class to use as a training set. The validation and test sets remain the same. NodeAcc with results on direct testing experiments are shown in Figure 2.2, and the accuracies are summarized in Table 2.6 in Appendix 2.10.

We can observe similar improvements under few-shot cases by applying LA-SCL. Notably, LIC, LIUC, and LISC, which incorporate additional



(a)



(b)

Figure 2.2: NodeAcc of DT with k-shot learning (a) 20NewsGroups (b) DBPedia

contrastive terms between instances and centers, work surprisingly well with 100-shot, especially on the DBPedia dataset, as shown in Figure 2.4b. It significantly improves performance compared to SCL and LI 100-shot achieves comparable performance to the full setup with a huge reduction of the training set size from 0.23M to 7,000. The computational cost is also decreased accordingly.

Incorporating a label similarity matrix to introduce weights between classes has a limited impact on DBPedia, as observed from the marginal improvement achieved by LISC compared to LIC and LIUC. SCL and LI also exhibit a similar pattern. Compared to their performance on 20NewsGroups, in terms of the improvement introduced by LI compared to SCL, and the improvement introduced by LISC, LIC compared to LIUC, a fine-grained hierarchy that captures the intricate relationships between classes would likely be more beneficial.

## Visualization

**LISC improves intra-cluster compactness and inter-class separation.** Figure 2.3 shows the scatter plot of sentence and label embeddings, marked by dots and Xs respectively, and colored by class. We keep the distribution the same as the original training set. Figures 2.3a to 2.3c show the embeddings from bert-base, SCL, and LISC, respectively. In Figure 2.3a, sentences from different classes are mixed and it is hard to find clear decision boundaries. Compared to the sentence embeddings, the label embeddings are grouped in a smaller space around the top left corners of Figure 2.3a.

In Figure 2.3b, after training with SCL, each cluster is treated individually and independently with each other by being pushed apart from one another. Although the feature space becomes more discriminative, classes with similar semantics are not grouped together and are often quite distant from each other. Additionally, the model captures semantic information associated with the labels, enabling it to understand the meaning and context behind each label. However, some labels are positioned close to one another, making it challenging to distinguish instances based solely on their embeddings. For example, observe the overlap between the label embedding representing "talk, religious, misc" and the embedding representing "talk, politics, misc." This proximity poses difficulty in accurately differentiating between the two labels solely based on their embeddings.

Although SCL successfully captures semantic information and separates clusters, it encounters limitations when dealing with classes that share common themes or contexts. It can lead to clusters with similar themes being positioned far apart from each other and label embeddings that overlap one another. These issues can be mitigated by the proposed LISC. As shown in Figure 2.3c, by applying LISC, the clusters belonging to the same high-level classes are brought closer to each other while being separated from clusters of different classes. To illustrate this, consider samples under the high-level class "recreation" depicted in green. Initially, in Figure 2.3b, these sub-categories are widely dispersed. But in Figure 2.3c, the four sub-categories of "recreation" have become grouped close to each other. This shows that penalizing the weights between classes with the label similarity matrix effectively guides the model to bring related sub-categories together. As a result, it facilitates clearer decision boundaries and improves the representation and organization in the embedding space.

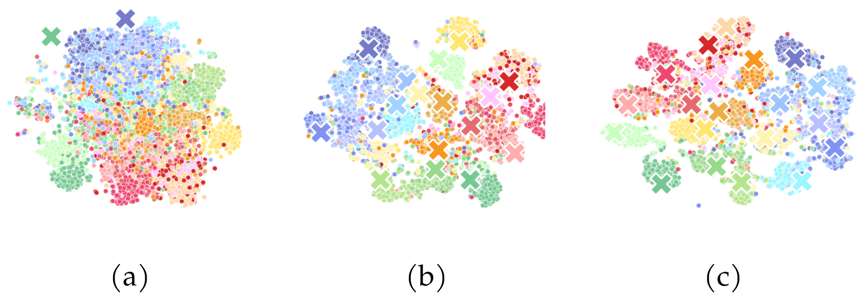


Figure 2.3: t-SNE visualization on 20NewsGroups with (a) bert-base, (b) SCL, (c) LISC. Label representations are marked by appropriately colored "$\times$".

| Method | IntraCluster | InterCluster |
|---|---|---|
| SCL | 14.88 | 16.17 |
| LI | 14.72 | 16.69 |
| LISC | 13.91 | 18.04 |

Table 2.3: Averaged inter- and intra-cluster $L_2$ distances with SCL, LI, and LISC. They measure the compactness and separation of clusters, respectively.

Table 2.3 calculates the intra- and inter-cluster distances to measure the average compactness of the clusters and to measure how far apart the cluster are. We use averaged instance-pairwise and averaged center-pairwise $L_2$ distances to be the intra-cluster and inter-class distances, respectively. Smaller intra-cluster distance implies that data points within a cluster are closer to each other. Meanwhile, the clusters are well-separated with a larger inter-cluster distance. Together, these indicate a tighter clustering with clearer boundaries between clusters. Both LI and LISC reduce the intra-cluster distance and increase the inter-cluster distance.

### Sensitivity to Different Label Hierarchies

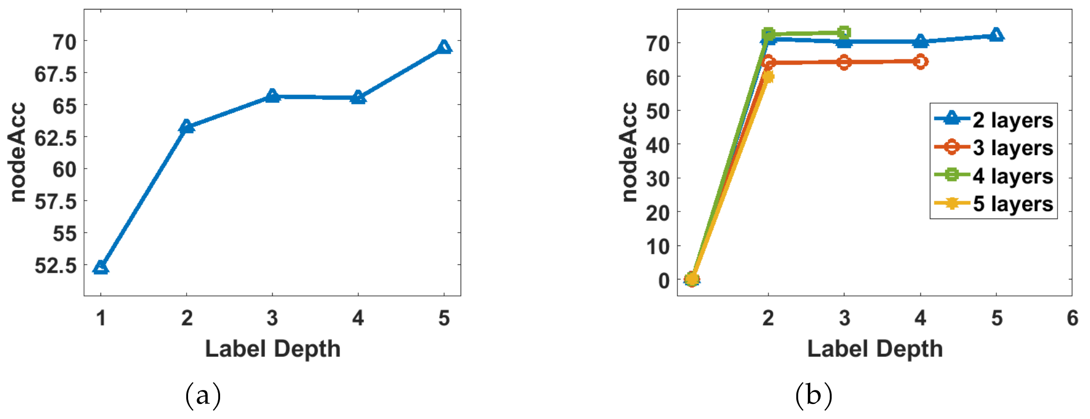


Figure 2.4: Measure the sensitivity to different hierarchies in (a) nodeAcc with different bottom-up label hierarchies ranging from 1-5. (b) nodeAcc on labels grouped by different hierarchies.

**Deeper hierarchical structures work better.** This section assesses how each leaf node label performs under different hierarchical structures. By manipulating the layers of the labels, we can simulate different levels of granularity. To achieve this, we construct different label hierarchies with bottom-up levels ranging from 1-5 on 20NewsGroups. We append an integer at the end of label strings to distinguish them if duplicates exist. The performance is always measured on the leaf nodes.

We observe that the overall performance changes in response to different levels of label granularity, as shown in Figure 2.4a. The model can better distinguish between closely related classes and make more precise predictions by providing more detailed and specific labels. Figure 2.4 grouped the performance based on the hierarchy of leaf nodes with depths ranging from 2-5.

## 2.8 Related Work

**Learning Label Hierarchy** Hierarchical classification is a task involving assigning samples to specific labels (most commonly fine-grained levels) arranged in a structured hierarchy, which is typically represented as a tree or directed acyclic graph, where each node corresponds to a label [106]. Recent work in the NLP field has suggested integrating the label structure into text features by encoding them with a label encoder. For instance, Chen et al. [14] embeds the word and label hierarchies jointly in the hyperbolic space. Zhou et al. [160] proposed a hierarchy-aware global model to extract the label structural information. Zhang et al. [154] designed a label-based attention module to extract information hierarchically from the labels on different levels. Wang et al. [131] proposed a network to embed label hierarchy to text encoder with contrastive learning. Chen et al. [16] proposes a matching network to match labels and text at different abstraction levels. These works propose various network structures aimed at extracting the label hierarchies or merging them with sentence features. Our LA-SCL, in contrast, exploits a small number of known labels and their hierarchical structure in order to improve the learning process.

Ge [42] proposed hierarchical triplet loss, which is useful for finding hard negatives by hierarchically merging sibling branches. A recent work [152] introduces hierarchy preserving loss, applying a hierarchical penalty to contrastive loss with the preservation of hierarchical relationship between labels on images. It used images under the same branch as positive pairs. This approach differs from our LI in constructing penalties from the hierarchical structure and applying it to contrastive loss.

**Contrastive Learning** Self-supervised contrastive learning is a representation learning approach that maximizes agreement between augmented views of the same instance and pushes different instances far apart. Works on text data [113] constructing various augmentations on text level [139, 141, 133, 44], embedding level [133, 49, 119, 123], and via language models [93, 49, 24], etc. SCL effectively learns meaningful representations and improves classification performance by combining supervised and contrastive learning advantages. It was initially introduced in SimCLR [20]. Other following works introduce novel insights to improve the representation learning such as MoCo [55], BYOL [47], and SwAV [13]. SCL has also been applied to NLP tasks such as sentence classification [23], relation extraction [76, 19] and text similarity [147, 40], where it has shown promising results in learning effective representations for text [116, 69, 18].

## 2.9 Conclusion and Limitations

In this chapter, we have proposed the LA-SCL family to include information about the label hierarchy. These introduce additional weights in the SCL loss between negative example pairs by their label similarity matrix constructed from the label features. They bring instances with similar semantics or belonging to the same high-level categories closer to each other. We also propose to use label representations as the center of the corresponding sentence embeddings. This learns a representation space by encouraging each instance to become closer to its centers and thus the underlying hierarchical semantic structures can be encoded. The learned labeled embedding matrix can be directly applied as the nearest classifier. More discriminative feature representations are learned and introduce performance gain on multi-class classification with full and few-shot setups.

Our proposed methods have some limitations, particularly when dealing with highly fine-grained label structures where most of the branches exhibit significant similarities. In this case, it is challenging to distinguish between label embedding similarities. Assigning weights to different classes may not be effective since the similarity scores $w_{cc'}$ are almost identical. This hinders the ability to accurately differentiate between classes and further impacts the performance. Another limitation comes from the common underlying issue of data. Bias can be learned by the model. To mitigate this, debias techniques can be employed to ensure fair and unbiased representation.

## 2.10 Appendix

### LP with Label Embeddings

In the experiments of Section 5, we randomly initialized the parameters of the classifier. An alternative is to use the pretrained label-representative parameters as the linear head, and then to further train on the labeled dataset used in the linear probe. Results on 20NewsGroups are shown in Table 2.4. Comparing their performance to Table 2.2. Further tuning the label embedding matrix on labeled samples with cross-entropy loss impairs the performance with LI and LIUC. It achieves comparable or slightly better performance in terms of LISC and LIC.

| Objective | nodeAcc | midAcc | rootAcc |
|---|---|---|---|
| LI | 67.26 | 73.74 | 78.78 |
| LIUC | 64.42 | 68.08 | 78.45 |
| LIC | 68.99 | 72.90 | 80.75 |
| LISC | 69.15 | 76.00 | 81.40 |

Table 2.4: (%). LP by using label embeddings as an initialized classifier on 20NewsGroups.

| Templates | Objective | directly test | | |
|---|---|---|---|---|
| | | nodeAcc | midAcc | rootAcc |
| 1 | LI | 61.35 | 64.63 | 76.62 |
| | LIUC | 67.66 | 75.31 | 79.93 |
| | LIC | 63.39 | 71.92 | 80.35 |
| | LISC | 67.34 | 75.66 | 79.43 |
| 2 | LI | 66.62 | 73.43 | 78.98 |
| | LIUC | 67.49 | 74.79 | 79.65 |
| | LIC | 65.45 | 73.88 | 80.02 |
| | LISC | 68.35 | 75.11 | 79.61 |
| 3 | LI | 65.43 | 72.29 | 78.52 |
| | LIUC | 67.69 | 74.88 | 80.24 |
| | LIC | 64.70 | 73.25 | 80.20 |
| | LISC | 67.90 | 75.00 | 79.49 |

Table 2.5: Results with different label templates on 20NewsGroups.

### Sensitivity on Different Label Templates

We explore the sensitivity of different label templates on 20NewsGroups as an example. Other than the template used in section 2.6, we also use the following templates

1. This sentence delivers {$\text{label}_i$} news under the category of {$\text{label}_i[L_1]$}
2. Description of {$\text{label}_i$} by generating a sentence from ChatGPT, the prompt given to ChatGPT is “Please generate a sentence to describe {$\text{label}_i$} news.”
3. {$\text{label}_i$}: description of {$\text{label}_i$}

In 2nd template, we use ChatGPT to generate a sentence description for each label. For instance, the description of “recreation,sport,hockey” is “In the latest recreation and sport news, hockey enthusiasts are buzzing

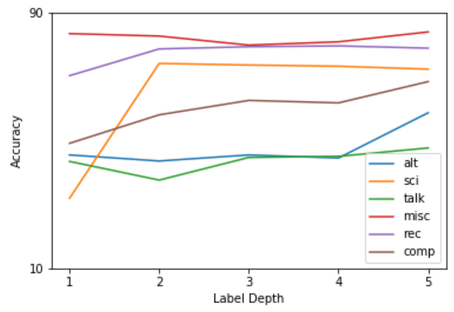


Figure 2.5: The highest level parent node performance with different label hierarchies.

with excitement as teams gear up for an intense season filled with thrilling matches and adrenaline-pumping action on the ice."

## Individual Label Performance over Different Label Hierarchies

This appendix assesses how each parent node label performs with different hierarchical contexts. As shown in Figure 2.5, we measure the performance on 6 parent nodes with models trained with different top-down numbers of label hierarchies. Labels such as "misc" and "recreation" exhibit consistently high performance across all hierarchies. Among them, "misc" only has two levels with one branch and is easy to classify.

On the other hand, there may be labels that show a notable improvement in performance as the hierarchy becomes more granular such as "alternative", "computer", and "talk". Among them, "alternative" only has two levels with one branch. "computer" and "talk" categories benefit from a more fine-grained hierarchy as the performance increase when the hierarchical level goes deeper. These labels benefit from the additional layers in the hierarchy, as the increased specificity enables the model to capture their unique characteristics better.

For "science", it has four leaf nodes corresponding to four sub-categories, all on the second layer in the tree. That is the reason why the performance jumps across layer 1 to 2 and remains stable afterward.

## Comprehensive Few-Shot Cases Results

| Dataset | Objective | directly test | | | linear probe | | |
|---|---|---|---|---|---|---|---|
| | | nodeAcc | midAcc | rootAcc | nodeAcc | midAcc | rootAcc |
| | | | 1-shot | | | | |
| 20NewsGroups | SCL | 16.89 | 22.81 | 42.06 | 58.68 | 66.60 | 74.97 |
| | LI | 32.71 | 41.20 | 56.03 | 58.47 | 65.75 | 74.50 |
| | LIUC | 33.43 | 41.66 | 57.32 | 58.30 | 65.53 | 74 .44 |
| | LIC | 33.82 | 42.11 | 57.47 | 57.79 | 65.52 | 74.08 |
| | LISC | 33.30 | 40.96 | 56.47 | 57.78 | 65.44 | 74.16 |
| DBPedia | SCL | 0.52 | – | 22.95 | 95.50 | – | 95.56 |
| | LI | 1.45 | – | 20.9 | 94.62 | – | 93.69 |
| | LIUC | 1.42 | – | 21.33 | 94.66 | – | 95.66 |
| | LIC | 3.55 | – | 21.11 | 94.25 | – | 95.35 |
| | LISC | 3.58 | – | 20.26 | 94.25 | – | 95.35 |
| | | | 100-shot | | | | |
| 20 newsgroups | SCL | 49.47 | 58.26 | 65.59 | 62.97 | 69.95 | 76.86 |
| | LI | 50.70 | 58.22 | 67.07 | 63.06 | 70.42 | 77.50 |
| | LIUC | 54.73 | 63.09 | 75.05 | 64.23 | 71.38 | 78.09 |
| | LIC | 63.52 | 70.83 | 78.21 | 63.21 | 70.17 | 76.95 |
| | LISC | 63.54 | 70.88 | 78.48 | 64.49 | 72.34 | 78.61 |
| DBpedia | SCL | 0.06 | – | 25.45 | 96.03 | – | 96.69 |
| | LI | 1.00 | – | 23.72 | 96.18 | – | 96.83 |
| | LIUC | 84.45 | – | 88.10 | 95.55 | – | 96.69 |
| | LIC | 93.13 | – | 94.48 | 95.80 | – | 96.61 |
| | LISC | 93.19 | – | 94.63 | 95.78 | – | 96.61 |

Table 2.6: Results on 1-shot and 100-shot in supplement to section 2.7

This section includes the full results in supplement to section 2.7 shown in Table 2.6.

# 3 PERSONALTM: TRANSFORMER MEMORY FOR PERSONALIZED SEARCH

## 3.1 Overview

This chapter introduces methods to perform personalized information retrieval and search by using a large transformer-based generative language model T5. This work first appeared in our paper [115]. We propose a novel model architecture to effectively fuse personalized features with queries, and thus improve the quality and relevance of retrieved documents. We further improve the performance by proposing a hierarchical loss function and applying prefix tuning for lightweight training.

## 3.2 Abstract

The Transformer Memory as a Differentiable Search Index (DSI) [122] has been proposed as a new information retrieval paradigm, which aims to address the limitations of dual-encoder retrieval framework based on the similarity score. The DSI framework outperforms strong baselines by directly generating relevant document identifiers from queries without relying on an explicit index. The memorization power of the DSI framework makes it suitable for personalized retrieval tasks. Therefore, we propose a Personal Transformer Memory (PersonalTM) architecture for personalized text retrieval. PersonalTM incorporates user-specific profiles and contextual user click behaviors, and introduces hierarchical loss in the decoding process to align with the hierarchical assignment of document identifiers. Additionally, PersonalTM employs an adapter architecture to improve the scalability for index updates and reduce computation costs in comparison to the vanilla DSI. Experiments show that PersonalTM outperforms the DSI baseline, BM25, fine-tuned dual-encoder, and other

personalized models in terms of precision at top 1st and 10th positions and Mean Reciprocal Rank (MRR). Specifically, PersonalTM improves p@1 by 58%, 49%, and 12% compared to BM25, Dual-encoder, and DSI, respectively.

## 3.3 Introduction

In recent years, Information Retrieval (IR) has undergone substantial progress owning to the resurgence of deep neural networks, with transformer-based Language Models (LMs) playing a particularly significant role. These models possess the capability of learning natural language representations from vast amounts of data, contributing to the advancement of IR [124, 33, 11, 110, 99, 43, 68]. Since the Dual Encoder (DE) approach [158, 30] is unable to learn the deep interaction between queries and documents, several sequence-to-sequence frameworks have been proposed for IR, which enable the improved generation of documents directly relevant to the queries [120, 29, 122, 66, 163, 7, 58, 130, 73]. For instance, WebGPT proposes a method to answer long-form questions in a text-based web-browsing environment by fine-tuning GPT-3 [98, 11], DSI utilizes a transformer-based encoder-decoder network to generate a ranked list of relevant document id [122]. The goal of DSI is to implicitly learn the relationship between the query document id, and between the document and the document id, by encoding all information into the model parameters. This framework simplifies the system architecture by utilizing a single LM, and leverages the memorization ability of the transformer.

Meanwhile, personalized retrieval is becoming increasingly important, aiming to refine queries and tailor retrieval results to specific preferences of individual users [85, 149, 32, 165, 86, 162]. Numerous studies have been made of personalized retrieval, demonstrating improvement in the precision by incorporating user profiles and contextual history [77, 82].

For example, P-Click [6] reranks the documents based on a user's clicks for a given query, and SLTB [35] outputs personalized ranking list by utilizing diverse clicked-based or topic-based features. RNN-based networks are used in [41, 161] to extract short- and long-term user profiles from personalized historical information and apply it to DE.

In this chapter, we enhance performance for personalized retrieval by incorporating user identification, contextual history, and user click behaviors into the transformer memory framework. Our method involves a novel decoding architecture and applying hierarchical loss aligned with the document id generation. Furthermore, we propose a flexible adaptor strategy to facilitate retraining in the event of an index update. The main contributions are:

- Compared with other personalized retrieval works, we utilize an end-to-end LM to generate relevant document id given the query directly.
- We leverage user identifiers and user-specific contextual data as personalized features into the DSI architecture. Different features are fed to different decoder cross-attention layers to effectively integrate the contextual features without increasing trainable parameters.
- We apply a hierarchical loss function at the decoding stage to further boost the performance aligned with the semantic hierarchical document id assignment.
- We adopt the prefix adapter to learn the interaction between query and contextual data. Model parameters and training time are reduced by 10x and 2x compared to fine-tuning, making it suitable for practical deployment.

## 3.4 Proposed Methods

This chapter proposes a novel approach for personalized retrieval using a personal end-to-end transformer memory (PersonalTM). This section describes a model architecture that integrates the two personalized features into the PersonalTM, with a hierarchical loss at the decoding stage to enhance performance. Additionally, we implement relevant information selection to optimize the utilization of contextual features and incorporate a prefix-adaptor for more agile training.

### PersonalTM model architecture integrating personalized features

Given a query, we utilize a transformer-based encoder-decoder network (such as T5) as the base structure to generate relevant document ids with two types of additional personalized information: 1) user profiles such as user identifiers; 2) personal context such as user browsing histories in cookies. The model is trained to learn the relationships between the query and document id, document and document id, as well as the query and the personalized information.

As shown in Figure 3.1, let $q$ be the query, $p$ be static user identifier, $d$ be target document, and $h$ be personal context feature. We concatenate $p$ to $q$ into one sequence $[p; q]$ so that $p$ is attended to $q$ by self-attention layers in the encoder, so the model can learn the connection between this specific user and his preferred $d$. $p$ is a synthetic user identifier constructed for each user by randomly selecting and concatenating $k$ tokens from a BERT tokenizer dictionary [96]. We use $k = 4$. $h$ is user-clicked documents.

Instead of concatenating $h$ to $q$, we design a decoding structure so that they interact in the embedding space as shown in Figure 3.2 for the following reasons: 1) it is not effective to concatenate $h$ with $q$ due to input length limitations; 2) when there is more than one $h$, the concatenation leads to information loss after truncation. To better use $h$ since relevant contextual information would be more useful, we propose two similarity measurement modules, which are introduced in Section 3.4.

Let $f$ be the encoder, we extract $f([p; q])$ and $f(h)$ as shown in Figure 3.2, and input them to the higher and lower decoder layers via cross-attention, separately. Each feature representation is a vector with dimension $n \times l \times d$, where $n$ is the batch size, $l$ is the max length of input sequence, $d$ is the feature dimension. Specifically, $f(h)$ is input to the first decoder layer and $f([p; q])$ is input to the remaining decoding layers. This allows $h$ to be integrated without adding additional layers or increasing the number of trainable parameters of the model. Because information about the past vanishes in left-to-right language models [127], the information injected in the upper layers would have larger weights than that in the lower levels, reflecting the different importance of $[p; q]$ and $h$.

Multiple works have shown the effectiveness of the fusion of different types of features at decoder layers via cross-attention [15, 21, 27, 22]. We also experiment with another decoding structure that involves a fusion layer to merge $f([p; q])$ and $f(h)$, and feed this fused information to each decoder cross-attention layer. Specifically, we extract $f([p; q])$ and $f(h)$, and concatenate these embeddings by $[f([p; q]);f(h)]$. An MLP layer consisting of two linear layers is used to project $[f([p; q]);f(h)]$ to the original dimension aligning the decoder input. This way, query and personalized information are fused, and are input into every decoder cross-attention layer.

Transformer-based LMs have achieved state-of-the-art performance on many tasks. Still, their overall capacity or trainable parameters are prohibitively large for retraining, particularly for IR's frequent index updates. Previous work has proposed alternative lightweight model tuning approaches such as prefix tuning [79, 56, 54, 142]. In addition to the adaptor's superior performance on few-shot cases and its lower computation

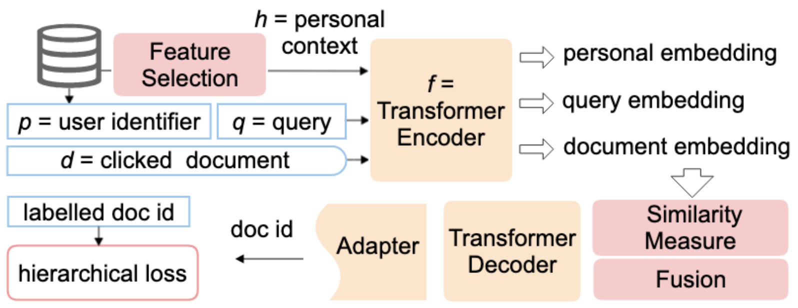


Figure 3.1: The training workflow of PersonalTM model architecture, integrating personalized features.

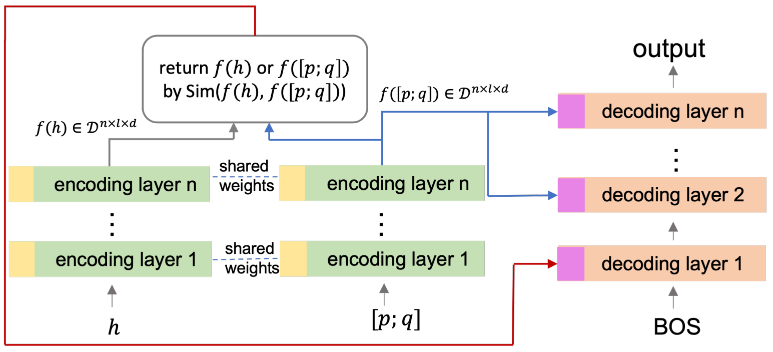


Figure 3.2: An overview of our proposed encoder-decoder architecture. A similarity measurement is used to select relevant contextual personalized information. $f([p;q])$ and $f(h)$ are fed into higher and lower decoder layers. Prefix-adaptor is applied and their parameters are marked in yellow and pink.

and storage requirements, we demonstrate that it is a promising method for personalization. By injecting additional continuous learnable parameters into each transformer layer, we achieve the updated model using the prefix adaptor. The training speed is thus doubled since only the newly injected parameters are updated and the original LM is frozen during training. The keys and values of the attention head in each self-attention and cross-attention layer are prepended with randomly initialized parameters. As shown in Figure 3.2, the newly injected prefix parameters are marked yellow and pink.

## Hierarchical loss

Through semantic clustering, we assign each document a unique document id, similar to [122]. For instance, *any country music lyrics* and *country lyrics tabs chords for country music fan* should have closer or even the same high-level clusters because they have similar meanings. However, decoding the document id by digit has the problem of accumulated errors. To leverage the semantic hierarchy of the document id and penalize more severe semantic retrieval errors, we apply an additional hierarchical loss based on the document id hierarchy at the decoding steps. Therefore, prediction error on the first several digits in the document id (top-level clusters) will have a higher impact than lower digits (lower-level clusters) because a such error will end up with documents having different topics with the query.

The overall loss function we use is shown in Eq. (3.2), where the first term $l_0$ refers to cross-entropy loss as shown in Eq. (3.1)

$$l_0 = \text{cross-entropy}(\text{logits}, \text{labels}) \tag{3.1}$$

$$l = l_0 + w_i \cdot l_0; \sum_i w_i = 1, w_1 > w_2 > \ldots \geqslant w_{n-1} \geqslant w_n \tag{3.2}$$

The second term in Eq. (3.2) refers to the hierarchical loss. Specifically, we multiply each digit of the original loss with a scalar $w_i$, by which different penalties are applied to different positions in the document id, modulating their importance accordingly. We apply higher weights to higher-level positions as shown in Eq. (3.2).

### Relevance denoise for personal context

Intuitively, only relevant personal context is useful because it implicitly increases the weights on tokens that imply user's preference. In contrast, irrelevant context introduces noise and thus can lead the model to incorrect predictions. For instance, given an incoming query *python running environment*. Among this user's browsed history, some of them (e.g. *python programming*) are relevant, while some (e.g. *what do pythons eat* or *country music*) are irrelevant to this query. Our experiment has demonstrated that relying purely on a model to do denoise is inefficient. Therefore, we adopt the following two mechanisms to retain as much useful personalized information for the predictions.

The first mechanism is feature selection for $h$ on the sequence level among this user's browsing cookies, as shown in Figure 3.1. We apply a lightweight algorithm to select relevant personal context among user-browsed history. The algorithm selects the relevant documents according to the ratio of how many tokens in the document overlapped in the incoming query. The user-clicked documents in the session with this ratio greater than a certain value are considered personal context kept. We use the clicked documents in the latest session as the default context.

The second mechanism is similarity measurement between $f([p; q])$ and $f(h)$ in the latent space, as shown in Figure 3.2. A cosine similarity score $s$ is calculated between $f([p; q])$ and $f(h)$. If $s$ is greater than a certain threshold, $h$ is kept and integrated into the decoding stage. Otherwise, $h$ is omitted.

**Algorithm 1** Relevant personal context selection

```
1: for row in testset do
2:     u_history = history[row["user_id"]], query = row["query"]
3:     for history in u_history do
4:         ratio = Similarity(history, query)
5:         relevant_history = list()
6:         if ratio >= threshold then
7:             relevant_history.append((ratio, history))
8:         end if
9:     end for
10:    return relevant_history[argmax(ratio)] if relevant_history is not
   None else return u_history[t-1]
11: end for
```

## 3.5 Experiments

### Datasets

We use AOL4PS [50] dataset in our experiments. It contains query, the corresponding clicked documents, and timestamps indicating the timeline of each user's search history. This dataset has a range of 12 weeks. We select relevant personal context from the first 9 weeks, which is considered historical data. The samples in the last 3 weeks are divided into training and test sets, which are 218,559 and 53,357, respectively. The number of distinct queries is 382,222 in the total dataset. Among them, 19,957 queries in the test set do not appear in the training set, and they are considered zero-shot.

### Experimental Setup

The pretrained T5 (t5-base) is our backbone model [136]. We use AdamW as an optimizer with a learning rate of {2e-5, 5e-5} for the training, and adopt batch size = 128 in all settings. In our experiments with prefix adaptor, we use a prefix length of 5. The hidden states dimension of the MLP layer is 512, and the dropout is 0.1 in all settings. To construct the hierarchical document ids, k-means ($k = 10$) clustering provided by `fast-kmeans` [1] is applied recursively over all document embeddings generated from the pre-trained BERT (bert-base-uncased) model. All documents are clustered into 10 clusters. The algorithm is applied recursively if the cluster size exceeds 100. Clusters with sizes smaller than 100 documents are assigned an arbitrary unique number from 0-99 as the cluster id. The next level's cluster id is appended to the current level. The average document id length is 6. The ratio and $s$ for feature selection are 0.6 and 0.8, respectively. For the weights in Function (3.2), $w_1$ is 3/6, $w_2$ is 2/6, $w_3$ is 1/6.

## Experimental Results and Analysis

Experiment results between PersonalTM and several baseline methods are presented in Table 3.1. We use p@k and Mean Reciprocal Rank (MRR) as our evaluation metrics, where p@k measures the accuracy of the top $k$ documents for the incoming query, MRR calculates the mean of the inverse of the ranks at which the first relevant document is retrieved for the incoming query. Both of them are in %. Consistent with the observations in [122], DSI significantly outperforms BM25 and finetuned DE (encoder in Finetuned DE is the same as the encoder of finetuned DSI). With our proposed hierarchical loss function, p@1 increases by 0.66% compared to the one with cross-entropy loss only (3 - 4 in Table 3.1). It is because hierarchical loss grants higher penalties for the error of higher-level digits, it improves the precision. To further investigate how the hierarchical loss contributes to the performance, we find that hierarchical loss improves the precision at every document id digit, especially at the first three digits, with an average p@1 increment of 4.83%.

With an optimized hierarchical loss function in the decoding stage, the p@1 increases by 0.66% compared to the single cross-entropy loss between

[1]https://pypi.org/project/fast-pytorch-kmeans/

| # | Method | p@1 | p@10 | MRR |
|---|---|---|---|---|
| 1 | BM25 | 21.61 | 32.87 | 25.46 |
| 2 | Finetuned DE | 30.58 | 42.13 | 34.25 |
| 3 | DSI | 67.42 | 75.84 | 70.58 |
| 4 | DSI + HieLoss | 68.08 | 77.33 | 71.52 |
| 5 | DSI + UserIdentifier | 69.16 | 77.60 | 72.19 |
| 6 | DSI + HieLoss + UserIdentifier | 70.06 | 78.47 | 73.04 |
| 7 | PersonalTM ($h$ in latest session) | 71.20 | 80.10 | 74.25 |
| 8 | PersonalTM ($h$ relevant to $q$) | 79.60 | 87.47 | 82.51 |

Table 3.1: Results of baseline methods and PersonalTM. Note all baselines and treatments are finetuned with the same training dataset.

predictions and labels (#3 vs #4 in Table 3.1). To further investigate how the hierarchical loss contributes to the performance gain, we plot the curves of p@1 at different document id clusters with and without hierarchical loss in Figure 3.3. This precision curve demonstrates hierarchical loss improves the performance at every document id cluster, especially on the first three digits, increasing 4.83% at p@1 on average. This occurs because the hierarchical loss imposes higher penalties for errors in these higher-level positions.

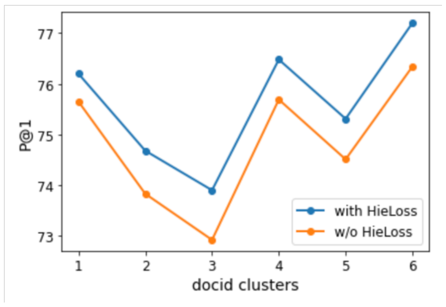


Figure 3.3: P@1 at different document id positions with and without hierarchical loss, respectively. The overall performance improves with hierarchical loss, especially at the first several positions by adding higher penalties to them.

By comparing lines 3 (no user identifier p) and 5 (with user identifier p) in Table 3.1, p@1 is improved by 1.74%. This indicates p can effectively help the model remember user's preference, particularly for predictions of unseen queries from existing users.

Comparison of lines 6 and 7 in Table 3.1 demonstrates the strength of the proposed model structure involving personal context h. By only using h from the latest session, p@1 can be improved by 1.14%. By comparing line 8 with lines 6 and 7, we can conclude that after applying the similarity selection to denoise personalized context, p@1 is significantly improved by 9.54% and 8.40% respectively. We also compare the performance with other personalization methods proposed in previous work in Table 3.2. PersonalTM has a comparable or better performance compared to other personalization methods.

| Method | p@1 | Method | p@1 | Method | p@1 |
|---|---|---|---|---|---|
| P-Click [6] | 59.56 | GRADP [161] | 77.06 | SLTB [35] | 71.09 |
| HRNN [41] | 76.53 | PersonalTM | 79.60 | | |

Table 3.2: Comparison with other personalization methods. precision numbers are averaged reported values in [50].

The results of the prefix adaptor are shown in Table 3.3. By comparing lines 1 - 2, the model parameters and training time of the adaptor are reduced by 10x and 2x compared to fine-tuning, respectively. It significantly reduces cost and improves agility for index updates while achieving comparable performance.

The performance of relevant personal context similarity measurements is shown in lines 3.1 and 4.1 in Table 3.3. Comparing 3.1 and 3.2 as well as 4.1 and 4.2, we find that our proposed PersonalTM beats the naive method that simply uses the latest clicked document by 1.09% and 1.62% for p@1, respectively. This occurs because our model structure explicitly learns the mapping between the query and the personal context. Besides, the

| # | Method | Imple | p@1 | p@10 | MRR | para |
|---|---|---|---|---|---|---|
| 1 | w/o h | finetune | 70.06 | 78.47 | 73.04 | 222.9 |
| 2 | w/o h | prefix | 69.90 | 78.25 | 72.87 | 29.5 |
| | Using latest clicked document | | | | | |
| 3.1 | [p; q; h] | prefix | 68.92 | 77.83 | 72.10 | 29.5 |
| 3.2 | PersonalTM | prefix | 70.01 | 78.53 | 73.01 | 29.5 |
| | Relevant personal context similarity measurement | | | | | |
| 4.1 | [p; q; h] | prefix | 75.26 | 85.37 | 79.28 | 29.5 |
| 4.2 | PersonalTM | prefix | 76.89 | 86.06 | 80.20 | 29.5 |
| 4.3 | PersonalTM (f) | prefix | 79.05 | 86.98 | 82.10 | 29.7 |

Table 3.3: Results with prefix adaptor.

proposed PersonalTM forces the decoder to digest this information without dilution by the long forwarding network. The result of fusion performance is shown in line 4.3 of Table 3.3. It improves p@1 by 2.16% compared to PersonalTM without fusion. Note fusion increases the computation time by an additional 30%, especially during the decoding process. Therefore, there is trade-offs between the computation cost, model size or storage space, and performance.

| # | Method | p@1 | p@10 | MRR |
|---|---|---|---|---|
| a | BM25 (with relevant history) | 20.97 | 31.03 | 24.53 |
| b | DualEncoder (T5) | 6.66 | 17.81 | 9.90 |
| c | DSI | 8.14 | 16.76 | 10.71 |
| d | DSI + HieLoss | 11.45 | 23.39 | 15.14 |
| e | DSI + HieLoss + UserIdentifier | 15.17 | 28.20 | 19.21 |
| f | PersonalTM (prefix adaptor) | 37.19 | 56.94 | 43.76 |

Table 3.4: Evaluation on Zero-shot cases.

Furthermore, we evaluate the model performance on the zero-shot set, whose queries are never seen in the past. The results are shown in Table

3.4. We found that DSI's zero-shot performance is not ideal, which aligns with the observation in [122]. But by comparing line f with the rest, we achieve significant improvement by using PersonalTM and the relevant personal context.

## 3.6 Ablation Study

In our proposed PersonalTM, there is a similarity measurement module, which measures the semantic similarity between the incoming query and contextual personalized information at the feature representation level. By using this module, only personalized features that are semantically similar to the incoming query are retained and further utilized. We investigate the performance of this measurement conducting the same experiments with the module removed. All other settings remain the same. Results and comparison are shown in Table 3.5.

By comparing #1.1 and #1.2, #2.1 and #2.2 in Table 3.5, the precision values are improved with the similarity measurement with both the latest session and relevant contextual personalized features. In #1.1, by using the latest session contextual personalized information and without the similarity measurement, the performance is even worse than the experiment using query information only. It is because most of the latest session information is not pertaining to the incoming queries, and thus much noise is introduced by these irrelevant features. Therefore, the similarity measurement module effectively selects useful information and filters out irrelevant ones.

## 3.7 Case Study

In this section, we demonstrate examples where our proposed methods succeed or fail as shown in Table 3.6. We are mainly focused on DSI,

| # | Method | p@1 | p@10 | MRR |
|---|---|---|---|---|
| Latest session contextual personalized features | | | | |
| 1.1 | PersonalTM (w/o similarity measurement) | 69.24 | 77.92 | 72.33 |
| 1.2 | PersonalTM | 70.01 | 78.53 | 73.01 |
| Relevant contextual personalized features | | | | |
| 2.1 | PersonalTM (w/o similarity measurement) | 75.92 | 85.77 | 79.71 |
| 2.2 | PersonalTM | 76.89 | 86.06 | 80.20 |

Table 3.5: Ablation study with / without similarity measurement module, which effectively selects useful information.

| | | |
|---|---|---|
| 1 | Query | boatsforsale |
| | Clicked doc | 36165 (new and used boats for sale boats for sale co UK) |
| | Wrong predictions | 395100 (boats com new and used boats for sale everything boats) |
| 2 | Query | optician programs in seattle |
| | Clicked doc | 81189 (home seattle central college) |
| | Wrong predictions | 81255 (home port of seattle) |
| 3 | Query | lyrics country music |
| | Clicked doc | 11112 (www any country music lyrics com) |
| | Wrong predictions | 355800 (country lyrics tabs chords for country music fans) |
| 4 | Query | ivillage horoscopes |
| | Clicked doc | 10769 (www I village co uk) |
| | Wrong predictions | 300600 (astrology com horoscopes tarot psychic readings) |

Table 3.6: Examples where our proposed methods succeed or fail. #1 shows that all of the methods fail easily while there exist similar documents. #2 shows when hierarchical loss improves performance. #3 and #4 show when user identifier and contextual personalized information improve performance, respectively. The number string (e.g., 36165) indicates document id.

DSI with hierarchical loss, DSI with user identifier, and PersonalTM for comparisons.

In #1 of Table 3.6, given the query "*boatsforsale*", the relevant clicked document is "*new and used boats for sale boats for sale co UK*". While all of the above-mentioned four methods fail by making a prediction of "*boats com new and used boats for sale everything boats*". By comparing these two documents, we found that they are quite similar both in semantics and sentence structure. It is hard to make the correct predictions while there exist quite similar documents.

In #2, given the query "*optician programs in Seattle*", the clicked relevant document is "*home Seattle central college*" with corresponding document id `81189`, while DSI makes a wrong prediction of "*home port of Seattle*" with document id `81255`. By comparing these two document ids, we noticed that they have the same id at the first two positions although their semantics are different, thus causing confusion. This can be fixed by introducing hierarchical loss since it applies higher penalties to the first several positions of the document ids. In other words, it helps the model be on the right track at the very beginning of the decoding procedure.

In #3, we present an example of how the user identifier improves the performance by introducing additional useful information. Given the query "*lyrics country music*", different people are interested in different topics according to their interests. For instance, some people are interested in "*www any country music lyrics com*", while others are interested in "*country lyrics tabs chords for country music fans*". Making the prediction based on popularity without additional user information may not satisfy all users. User attributes such as user identifier is a good candidate to mitigate this issue. It is especially helpful in zero-shot cases, in which the query has not been seen during training.

Other than that, personalized contextual information is also helpful. For instance, in #4, given the query "*village horoscopes*", the model always makes the prediction of "*astrology com horoscopes tarot psychic readings*" because this document has overlapped tokens with the query. However, this document is not what the user expected. By looking into users' historically clicked documents in session, we can find relevant contextual information such as "*village co UK*". This relevant personalized contextual feature is helpful by making links between query and relevant documents, and thus makes the predictions more accurate and improves the performance.

## 3.8 Conclusions and Limitations

In this work, we propose a transformer memory framework for personalized retrieval, integrating two essential types of personalized information into the model. We propose a novel decoder architecture that effectively leverages personalized context without increasing trainable parameters, and apply a hierarchical loss function to optimize the performance. Our framework also incorporates the prefix adaptor mechanism to facilitate the learning of the interaction between the query and contextual personalized features. In the future, we will further enhance the design by incorporating incremental learning and enabling dynamic updates to overcome the limitations of frequent index updates and zero-shot retrieval.

There are also some limitations to these methods. For instance, if there is a new incoming document, we have to re-assign a unique id to it by re-applying the hierarchical clustering method. And the models have to be re-trained with the new incoming documents again. In order to mitigate this, future work such as incremental learning could be used to update the models dynamically. Besides, we will utilize more contextual personalized features such as browsing histories from short- and long-term sessions. Furthermore, we found that zero-shot performance is not ideal due to the model's memorization capacity and index size limitations. In order to mitigate this, other further work are needed to improve model generalization ability.

# 4 INCREMENTAL USER EMBEDDING MODELING FOR PERSONALIZED TEXT CLASSIFICATION

## 4.1 Overview

This chapter introduces a method to consistently update user profiles given a stream of user personalized data. This work first appeared in [82]. We propose to fuse past and incoming user data with additional self-attention layers and propose a training method to incrementally update the model with the incoming data.

## 4.2 Abstract

Individual user profiles and interaction histories play a significant role in providing customized experiences in real-world applications such as chatbots, social media, retail, and education. Adaptive user representation learning by utilizing user personalized information has become increasingly challenging due to ever-growing history data. In this work, we propose an incremental user embedding modeling approach, in which embeddings of user's recent interaction histories are dynamically integrated into accumulated history vectors via a transformer encoder. This modeling paradigm allows us to create generalized user representations in a consecutive manner and also alleviate the challenges of data management. We demonstrate the effectiveness of this approach by applying it to a personalized multi-class classification task based on the Reddit dataset, and achieve 9% and 30% relative improvement on prediction accuracy over a baseline system for two experimental settings through appropriate comment history encoding and task modeling.

## 4.3 Introduction

Given that archived user records such as user-specific demographic attributes and daily utterances can reveal users' personalization, interests and attitudes towards different topics, it allows us to infer user preferences and hence provide personalized user experiences based on the records collected by social media platforms or conversational agent systems. These context signals always reflect consistent personalities, which can provide meaningful information and thus can be leveraged to build coherent models to facilitate a variety of application needs better [77, 157]. Thus, it becomes imperative to develop adaptive user representations by incorporating personalized features from different modalities for downstream tasks such as user preference prediction [53, 12], personalized dialogue generation [138, 151, 104], and personalized recommendations [45, 67, 159].

Most previous work which proposes to generate personalized representations assumes the availability of a users' entire history. For instance, Wu et al. built a personalized response generation system consists of a mechanism of splitting memories to capture personalization: one for user profile and the other for user-generated information such as comment histories [138]. Grbovic et al. proposed a method capturing long-term and short-term user interests to build customized embeddings, which can be applied to real-time personalized recommendation [45]. However, it is impracticable to access the entire historical data in some applications due to various constraints. To address this issue, Bui et al. proposed a way to utilize existing personalization techniques in the Federated Learning setting [12].

On the other hand, user embedding modeling in batch mode always train models from scratch by aggregating all available data. It is not only time-consuming but may also result in outdated models [88]. Besides, it cannot capture the gradually-changing user interests and hence cannot reflect the recency of user data. To alleviate these issues, incremental learning has been proposed to update models in an iterative manner, which allows the model to learn new information as soon as it is available and thus leads to up-to-date models. It is not only capable of lifelong learning with restricted resources, but also captures the adaptive representation, and reduces the cost of data storage and maintenance [8, 64, 134].

Based on those findings, we propose an incremental user embedding modeling method to generate adaptive user representation by leveraging personalized information. We demonstrate the effectiveness of this approach on a multi-class text classification task on the Reddit dataset following [12] as this dataset could mimic real-world scenarios such as data collected by chatbots. One example sample is shown in Table 4.1. The objective of this text classification task is to predict which class (i.e., subreddit) a given comment was posted by utilizing user profiles and comment histories, where user profiles refer to static user-specific key-value pairs.

| |
|---|
| **Incoming comment:** Watch funny anime. |
| **Subreddit:** Anime |
| **Author:** HaraJiang, **Time:** 2019-07-01 00:28:49 |
| **Comment histories:** |
| Victory Royale is really interesting. |
| Warframe is available for free, it is a good game. |
| I see a water dragon. |

Table 4.1: A constructed example from Reddit dataset

## 4.4 Proposed Methods

This study focuses on building adaptive user representations in an incremental manner. In this section, we first introduce several batch learning methods to investigate model performance by utilizing personal information. Then we propose an incremental embedding learning method, which can continuously integrate newly available data into existing trained models and is effective in capturing the evolving change of user interests.

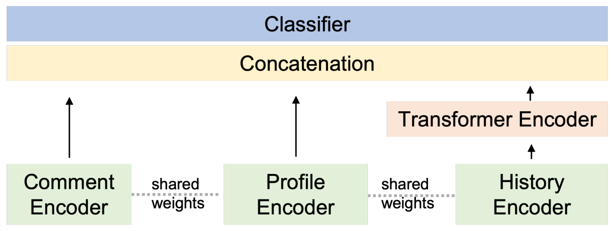


Figure 4.1: Architecture of the proposed model

## Batch Learning Methods

We fine-tune a BERT model [34] for multi-class classification by adding a linear classifier layer on top of it. As shown in Fig. 1, the input is a concatenation of different types of feature representations, and the output is the predicted subreddit indicating where the given comment was posted. The model is trained and evaluated using three different input feature representations.

1. $[q]$: incoming comment embedding $q$ only.
2. $[q, U_p]$: concatenation of incoming comment embedding $q$ and static user profile representation $U_p$.
3. $[q, U_p, U_h]$: concatenation of incoming comment embedding $q$, user profile representation $U_p$, and user history representation $U_h$.

We also explore two ways to represent user history representation $U_h$.

1. **Mean-pooling**: use mean-pooling of all history embeddings as $U_h$.

2. **Self-attention**: add an upper-layer transformer over all history embeddings and then use the mean-pooling of transformer output as $U_h$.

## Incremental User Embedding Learning

The core idea of incremental embedding learning method is to dynamically integrate recent comment history embeddings into the accumulated history vectors via an upper-layer transformer encoder. As shown in algorithm 1, history embedding tensor $A$ and accumulated history embedding tensor $B$ are initialized first. Both $A$ and $B$ are in the shape of $n \times T \times d$, where $n$ is the number of users, $T$ is the time-span of interaction history, and $d$ is the dimension of sequence embedding. $t_0$ is a timestamp chosen as history, and samples prior to $t_0$ are used for initialization. We then construct the dataset in the following form: $(i, t, y)$, where $t$ is a timestamp such that $t \in [t_0, T]$ and $y$ is the ground truth label of an incoming comment at $t$. This dataset was fed into the model in chronological order since $B$ is updated gradually.

For an incoming comment at $t$, the objective is to predict its associated subreddit by leveraging the accumulated history embedding in $B$. Meanwhile, $B$ is updated by integrating multiple latest comment history embeddings into it. Specifically, we get recent $n_0$ history embeddings and the accumulated history embedding from $A$ and $B$, respectively. Those vectors are input into the upper-layer transformer encoder and get attended with each other. U_profile represents the personalized user profile incrementally constructed from comment histories. It is actually the mean-pooling of the transformer encoder output, and is used to update $B$. By using this momentum update, the impact of long-term history interactions is gradually decayed. Here, $\alpha$ is the momentum parameter, $\alpha \in [0, 1]$.

With the proposed incremental user embedding learning method, accumulated user embeddings can be continuously updated with recent activities. It can adaptively capture the evolving change of user interests.

**Algorithm 2** Incremental user embedding learning

```
1: Initialization:
2: Let $i$ be user id, $i \in [0, n]$
3: Let $j$ be history index, $j \in [0, T]$
4: $A[i][j]$ = encoder(history$[i][j]$)
5: $B[i][j]$ = mean($A[i][:j]$)
6: for batch in dataset do
7:     q = encoder(history$[i][t]$)
8:     history = $A[i][t - n_0 : t - 1]$
9:     past_activity = $B[i][t - n_0 - 1]$
10:    U_profile = mean(transformer([history, past_activity]))
11:    $B[i][t-1]$ = $\alpha$ * U_profile + $(1 - \alpha)$ * $B[i][t-1]$
12:    loss, logits = classifier([q, $B[i][t-1]$])
13: end for
```

## 4.5 Experiments

### Datasets

In this section, we describe how three datasets were built from the Reddit comment dataset[1] [4]. The dataset used to pre-train BERT model is referred to as "pre-training set". The other two datasets used to investigate batch learning methods and incremental embedding learning are called "subset1" and "subset2", respectively.

The "pre-training set" consists of comments from the top 256 subreddits in 2019/07 by following the procedure described in [12] . Each comment corresponds to one subreddit indicating where it was posted. There are around 5M comments distributed in 256 classes in total, and this dataset is imbalanced.

[1]https://files.pushshift.io/reddit/comments/

The "subset1" was randomly selected from the “pre-training set" by balancing the subreddit distribution. User profiles (e.g. Username and created time associated with each comment) are also kept. In addition, we pick up 5 comment histories that are most recent to each incoming comment and we list them in the reverse chronological order. There are around 126K incoming comments in total. We split them into training, validation, and test sets by 50%, 25%, and 25%.

The “subset2" was built by picking up 40 histories for each incoming comment. Besides, history's associated subreddits are also kept, while two user-related attributes are thrown away. The other settings remain the same as subset1. After customizing the sequential dataset as mentioned in section 2.2, the subset2 distribution looks similar to the original imbalanced one.

## Experimental Setup

All three datasets were preprocessed by standard text normalization steps, such as removing urls, trimming extra spaces, and filtering non-ASCII words and characters. A BERT model with shared weights is used as encoder, and its output of the first token ([CLS] token) is used as sequence embedding.

To conduct personalized prediction on subset1, the BERT model is fine-tuned together with other hyperparameters. User profiles are represented as key-value pairs, which are concatenated into the input sequence before encoding.

In incremental embedding modeling on subset2, we first use masked language model (MLM) to pre-train BERT with and without history's associated subreddit, respectively. To include subreddits of history data as input, we first expand tokenizer vocabulary. Then we prepend subreddits in textual format in front of history sequences. After pretraining, we extract history embeddings from frozen BERT with different strategies depending on whether we prepend their associated subreddits or not. We weight the cross-entropy loss by the inverse frequency of each category to deal with the imbalanced dataset.

We use the bert-base-uncased model provided in huggingface's transformer package [136]. For the upper-layer transformer, we use 4 layers of transformer encoder, and each layer with 8 attention heads. AdamW is used as our optimizer with an initial learning rate of 2e-5 and we use a linear decay learning rate scheduler across all model training. BERT max sequence length is 128 in all settings. The dropout probability is always 0.1. For pre-training, we use batch size of 16 and 5 epochs. For all other experiments, we use batch size of {64, 512, 1024}. The maximum number of iteration is set as 15 with an early stop if no improvement over accuracy on validation set. $\alpha$ is 0.1 for incremental update. It takes no more than 40 mins to perform incremental modeling with 8 NVIDIA V100 GPUs.

## Experimental Results and Analysis

Table 4.2 summarizes the classification accuracy of batch learning methods described in section 2.1. We observed that: (1) the network incorporates user profile or comment histories can boost the model performance. The classification accuracy is improved by 7% relatively by incorporating user profiles, and 41% relatively with the mean-pooling of history embeddings as additional feature; (2) the attention between posting histories contributes 62% relative improvement to accuracy. It shows that self-attention between histories would enhance personalized features and reinforce users' preference and interests. Although the model with self-attention over all history embedding performs better compared to using history embedding mean-pooling, it is still a trade-off between these two methods in terms of data storage capacity and computational resources since model with self-attention has more complicated structure, leading to more hyperparameters.

| Model | Feature | Acc(%) | Rel.Imp.(%) |
|---|---|---|---|
| BERT | $[q]$ | 17.95 | 0 |
| BERT | $[q, U_p]$ | 19.23 | 7 |
| Mean-pooling | $[q, U_p, U_h]$ | 25.37 | 41 |
| Self-attention | $[q, U_p, U_h]$ | 29.02 | 62 |

Table 4.2: Batch learning method performance.

The results of the mean-pooling method are regarded as the lower bound. Similarly, the results with upper-layer transformer encoder are regarded as an upper bound since attention behaviors over entire histories enhance personalized semantic features. The goal of incremental embedding modeling is to asymptotically approximate the upper bound method and get their performance as close as possible. Moreover, the incremental modeling would capture the recency of user data and reflect the evolving change of user interests over time while these cannot be achieved by self-attention method. We do not include static $U_p$ in incremental update since it does not play a significant role as results shown in Table 4.2.

Table 4.3 shows the incremental modeling results. It cannot be directly compared with Table 4.2 since it is performed on sequential dataset instead. Histories are encoded without and with considering their associated subreddits in the 2nd and 3rd column respectively. From 2nd column, our incremental method lies in between the lower and upper bounds as we expected. It improves the lower bound mean-pooling method by 9% relatively, and is 14% relatively below the upper bound self-attention approach in terms of accuracy.

In the 3rd column of Table 4.3, history's associated subreddits are considered as part of history attributes and are prepended in the texual format in front of histories before encoding. In this way, both the prior information and the semantics of history subreddits are incorporated into the sequence embedding. From the results, the accuracy of proposed incremental modeling is 30% relatively higher than the mean-pooling method, and is only 2% behind the upper bound. It significantly closes the gap between the incremental modeling and the self-attention method compared to 2nd column. This is because the attention behaviors introduced by the upper-layer transformer reinforce the personalised features, especially the prior information, which provides the most direct information in classification.

By comparing the incremental modeling performances with two ways of history encoding, around 59% relative improvement in accuracy can be achieved. To further investigate the performance gain introduced by prior, we predict the incoming comment by the majority count of its history's subreddits as a direct comparison. The incremental modeling can improve the majority count by 38% in accuracy. These analyses indicate that the incremental modeling effectively utilizes semantics from both utterances and history prior.

| Model | Acc (%) w/o subreddit | Acc (%) w subreddit |
|---|---|---|
| Majority count | - | 36.06 |
| Mean-pooling | 28.79 | 38.27 |
| Self-attention | 36.72 | 50.80 |
| **Incremental** | **31.44** | **49.94** |

Table 4.3: Incremental embedding modeling performance. Histories are encoded without and with their associated subreddits in 2nd and 3rd column, respectively.

## 4.6 Conclusions and Future Work

In this work, we proposed a method to generate personalized adaptive user representations by utilizing user profiles and user interaction histories in a consecutive manner. Specifically, we proposed an incremental user embedding modeling paradigm, which can dynamically integrate most recent user activities into the accumulated history embedding vectors. We show that a better performance can be achieved on downstream predictions with proper history encoding. Besides, we show that this approach not only captures the recency of user data and reflects the evolving change of user interests, but also keeps the model up-to-date and improves data storage efficiency.

Since the old accumulated history vectors do not update in a timely manner with the updates of the model, we have to introduce other methods such as regularization to improve the performance in the future. Additionally, we would like to extend the proposed incremental modeling to other applications such as personalized intent classification and personalized recommendation.

# 5 PREDICTING HEALTH-RELATED QUALITY OF LIFE CHANGE FROM PATIENT LANGUAGE IN THYROID CANCER STUDIES USING NATURAL LANGUAGE PROCESSING

## 5.1 Overview

This chapter introduces two data augmentation methods given limited clinical patient utterances to improve the prediction of health-related scores. The first method leverages GPT-2 [109] to generate synthetic patient utterances with either positive or negative sentiments. The second method rearranges and permutates the text in two consecutive transcripts. This work first appeared in [81]. This work is in collaboration with MD. David Schneider.

## 5.2 Abstract

Background: Patient-reported outcomes (PRO) allow clinicians to measure health-related quality of life (HRQOL) and understand patients' treatment priorities, but obtaining PRO requires surveys which are not part of routine care. We aimed to develop a preliminary natural language processing (NLP) pipeline to extract HRQOL trajectory based on deep learning models using patient language. Materials and methods: Our data consisted of transcribed interviews of 100 patients undergoing surgical intervention for low-risk thyroid cancer, paired with HRQOL assessments completed during the same visits. Our outcome measure was HRQOL trajectory measured by the SF-12 physical and mental component scores (PCS and MCS), and average THYCA-QoL score. We constructed an NLP pipeline based on BERT, a modern deep language model that captures context semantics, to predict HRQOL trajectory as measured by the above endpoints. We compared this to baseline models using logistic regression and support vector machines trained on bag-of-words representations of transcripts obtained using Linguistic Inquiry and Word Count (LIWC). Finally, given the modest dataset size, we implemented two data augmentation methods to improve performance: first by generating synthetic samples via GPT-2, and second by changing the representation of available data via sequence-by-sequence pairing, which is a novel approach. Results: A BERT-based deep learning model, with GPT-2 synthetic sample augmentation, demonstrated an area under the curve (AUC) of 76.3% in the classification of HRQOL accuracy as measured by PCS, compared to the baseline logistic regression and bag-of-words model, which had an AUC of 59.9%. The sequence-by-sequence pairing method for augmentation had an AUC of 71.2% when used with the BERT model. Conclusions: NLP methods show promise in extracting PRO from unstructured narrative data, and in the future may aid in assessing and forecasting patient's HRQOL in response to medical treatments. Our experiments with optimization methods suggest larger amounts of novel data would further improve performance of the classification model.

## 5.3 Introduction

Patient-reported outcomes (PRO) allow clinicians, researchers, and caregivers to assess patient well-being and health directly from a patient without relying on interpretation by a clinician. Health related quality of life (HRQOL) is an outcome measured by PRO instruments such as filling out an additional questionnaire. However, although PRO is important in making patient-centered treatment decisions, it is still an imperfect measure of HRQOL: while surveys capture HRQOL at a given moment, they do not reflect the entirety or trajectory of patient experience and do not allow patients to indicate domains with the greatest impact on their lives. To overcome this, patient's spoken language or conversation, which provide rich information in a more natural way, have been increasingly used. Narrative data is available from patient communications and patient portals associated with the electronic health record (EHR). Since written and spoken language provide a window into a person's psychosocial well-being, they have the potential to capture information about the function and quality of life, and other subtleties not included in traditional surveys. Given narrative data, deep language models can be used to extract meaningful semantics. These models can be used to measure a patient's psychosocial well-being from readily available sources of narrative data, such as patient interviews.

Thyroid cancer ranks among the most common malignancies in the United States. While PRO are themselves subject to the limitations above, nonetheless, they are the best available numeric representation of HRQOL, which is essential in thyroid cancer. Extraction of HRQOL outcomes after surgery can help personalize treatment since most patients are expected to experience a decrease in HRQOL just after diagnosis or surgery, but longer-term trajectories are often of greater clinical interest [103]. Patients at risk for long-term decrement in HRQOL may need active referral to resources such as counseling and support groups. If we are able to identify these patients preoperatively, the patient may choose less invasive treatment options. On the other hand, no tools exist to extract HRQOL from patient language thus far. For these reasons, we use thyroid cancer as a model in demonstrating a role for NLP techniques in extracting HRQOL information as measured by common PRO instruments. Given a unique, novel dataset from a randomized controlled trial, we adapted deep learning models to predict HRQOL directly from patient language.

## 5.4 Related Work

Dictionary-based bag-of-words text analysis tools such as the Linguistic Inquiry and Word Count (LIWC) program classify words by several different categories according to psychological, categorical, and language constructs [105]. Though LIWC is limited by the accuracy and comprehensiveness of the dictionary they are based upon, it has been used in many studies evaluating indicators of psychosocial well-being [46] and has also served as a comparison point for deep learning methods [121, 63, 9].

In contrast, deep learning models such as BERT attempt to model the underlying structure of language and construct sequence representations using a word's surrounding context. In the social sciences, BERT has been successfully used for the tasks of opinion and sentiment analysis. Previous work using NLP for similar applications in the medical field largely draws from data available in the EHR [101, 102]. These methods prove capable of more accurate predictions from the complexity of human language.

While many approaches train models using the written documentation of doctors and other clinicians [71, 114], which are likely to include similarity in structure and industry-specific language, we analyze speech from transcripts of patients, who have little to no medical background and are likely to speak in a more freeform manner than would be typically used by medical professionals. The goal of this study is to assess and further improve language model performance on prediction of HRQOL trajectory from patient language. The task described herein hence differs from traditional NLP analysis of medical text. Furthermore, few previous studies examine the trajectories of clinical outcomes such as HRQOL, rather than values at a single point in time. We implemented and evaluated a method for classifying narrative data into positive and negative HRQOL trajectory, which we defined as the sign of the difference between HRQOL at two different time points. Such tools to analyze patient language may extend to data sources from outside the medical encounter, including mobile device applications and social media.

## 5.5 Dataset Description

The dataset used in this study is a rare example of sets of prospectively collected and transcribed patient interviews paired with numeric scores on several validated Likert-like PRO measures for HRQOL. It is from a prospective randomized trial assessing surgical interventions for low-risk thyroid cancer. Each patient was evaluated at up to five set time-points: preoperative (6 weeks to 24 hours before operation), and 2 weeks, 6 weeks, 6 months, and 1 year postoperative. At each visit, the patient participated in a semi-structured interview and also completed each of the following validated HRQOL assessment surveys, including both general and disease-specific assessments of HRQOL including the 12-item short form survey (SF-12), and the THYCA-QoL.

## 5.6 Proposed Methods

Figure 5.1 gives an overview of the experiments described in this work. After basic pre-processing steps, a baseline model based on logistic regression using frequencies of LIWC categories was performed. Then BERT was fine-tuned on classification and compared to the LIWC-based model. Finally, two data augmentation techniques further improve the performance of BERT on down-stream tasks.

The primary outcome was the patient's HRQOL trajectory measured by each of validated HRQOL surveys. As stated above, we defined the HRQOL trajectory to be the direction of the difference in the HRQOL score (PCS, MCS, or average THYCA) between two time points. Of up to five transcripts from pre-defined time points at which the survey packet was administered, two trajectories were computed: the 1-2 trajectory (the

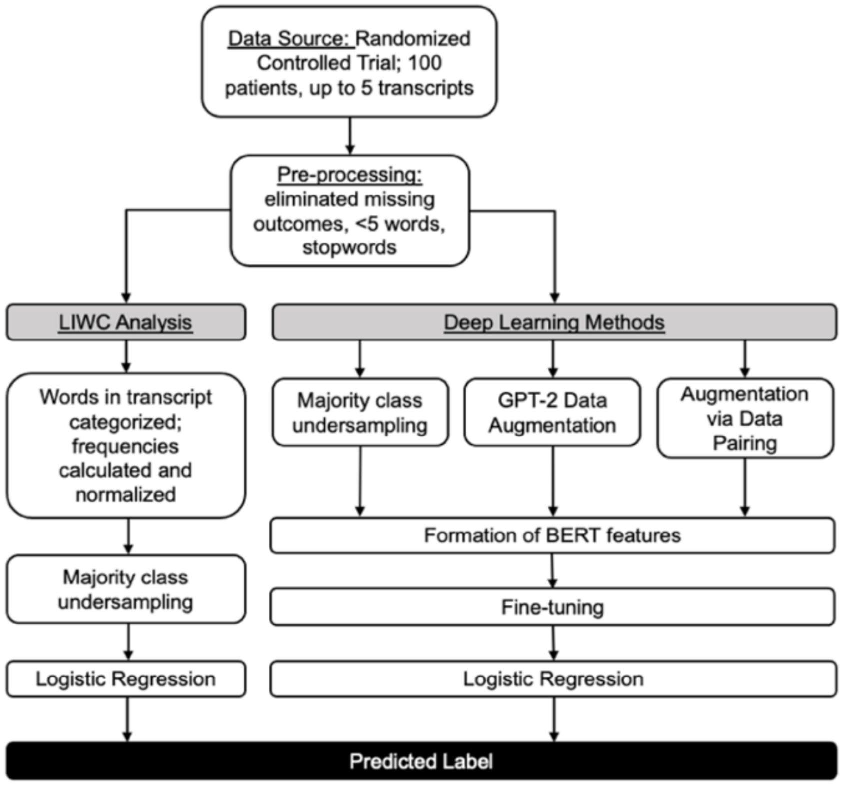

Data Source: Randomized Controlled Trial; 100 patients, up to 5 transcripts
Pre-processing: eliminated missing outcomes, <5 words, stopwords
LIWC Analysis
Deep Learning Methods
Words in transcript categorized; frequencies calculated and normalized
Majority class undersampling
GPT-2 Data Augmentation
Augmentation via Data Pairing
Formation of BERT features
Majority class undersampling
Fine-tuning
Logistic Regression
Logistic Regression
Predicted Label

Figure 5.1: Data analysis workflow

direction of HRQOL change between transcript 1 and transcript 2) and the 2-last trajectory (between transcript 2 and the last available transcript). Patients' utterance in transcript 1 was used for the classification of 1-2 trajectory label, and utterances in the last available transcript was used for the classification of 2-last trajectory label. Sequences that had a positive HRQOL trajectory were labeled as class 1, and those had a negative trajectory were labeled by class 0. The problem thus becames one of binary classification.

Since LIWC is a widely used computational tool meant to ascertain the linguistic characteristics of text, we use this as a benchmark to extract sequence representations for our analysis. LIWC operates by comparing every word in a given text to an internal dictionary of 76 categories, and then assigns the word to the appropriate categories. LIWC only considers its lexical contents.

LIWC categories include grammatical parts of speech (e.g., articles, pronouns, prepositions), psychological processes (e.g., affect, cognition, sensory and perceptual processing), and personal concerns (e.g., body states, death, and religion). For each text, LIWC produces an output specifying the percentage of words contained in that text that fall into each linguistic category of interest. For example, a text containing 15 first-person pronouns within a total of 1000 words would receive a score of 1.5 for the first-person pronouns category. For each patient, all transcripts were concatenated and frequencies were calculated for 9 LIWC categories of interest: positive emotion, negative emotion, feeling, biological processes, body, health, past-focused, present-focused, and future-focused (Table 5.1). The frequencies were then normalized, and then we applied logistic regression with the LIWC features to create a baseline model. We use under-sampling to deal with the imbalanced training set distribution.

| LIWC categories | Examples |
|---|---|
| Positive emotions | Love, nice, sweet |
| Negative emotions | Hurt, ugly, nasty |
| Feeling | Feels, touch |
| Biologic processes | Eat, blood, pain |
| Body | Cheek, hands, spit |
| Health | Clinic, flu, pill |
| Past-focused | Ago, did, talked |
| Present-focused | Today, is, now |
| Future-focused | May, will, soon |

Table 5.1: Linguistic features of patient interview transcripts

## Analysis with BERT

A large amount of previous work has demonstrated the improvements in text classification within biomedical and clinical domains by using modern language models [74, 57, 80, 1]. In our experiments, a logistic regression layer was added on top of BERT. It was then trained together with BERT for sequence classification. The first token of the BERT output ([CLS]), an array with dimension 768, was used as the sequence embedding. To deal with the imbalanced class distribution, we also under-sampled the majority classes. Three BERT models were fine-tuned separately with respect to each outcome.

## Training data augmentation

**Augmentation via GPT-2** Inspired by previous work which used GPT-2 to synthesize sequences for training set augmentation [72, 2], we use GPT-2 to generate representative sequences, or synthetic sequences similar in nature to those found in the original dataset in order to augment the training data. The main idea is shown in Algorithm 1.

**Algorithm 3** Augmentation via GPT-2

```
Input:
Let $D = (x_i, y_i)_{i=1}^{n}$ be original data
Do:
Train a classifier $h$ from $D$
Fine-tune language model $G$ by $D$ and obtain $G_{tuned}$
Generate $D^* = (x_i^{'}, y_i^{'})_{i=1}^{N}$ using $G_{tuned}$
Filter $D^*$ by $h$ to obtain $D_{generated}$
return $D_{generated}$
```

We first fine-tuned GPT-2 $G$ on the original dataset $D$, and the fine-tuned model $G_{tuned}$ was used for sequence generation. For fine-tuning, each input utterance in the model was represented as a concatenation of the textual class label, the sequence itself, and an end-of-text marker in the following form: Label: seq <endoftext>. The class label was either “positive” or “negative” (representing the HRQOL trajectory). Two separate GPT-2 models were trained on the data from each class. After training, the fine-tuned models were used to synthesize new sequences. The textual labels ("positive" or "negative") were provided as prompt into one of the fine-tuned GPT-2 models corresponding to their training class. Then a sequence was generated and extended until it reached the special end-of-text marker, thus simulating additional data from that class. In this way, an arbitrary number of label-invariant sequences can be generated.

Next, we fine-tuned BERT on the original dataset and it was used as a classifier $h$ to evaluate the quality of the generated sequences and obtain $G_{\text{generated}}$. Specifically, each generated sequence was input into the trained BERT model and output the predicted label along with a probability score, which is actually softmax of the output. Generated sequences were discarded if their predicted labels did not match the initial prompt or if the probability scores were less than a threshold value. After augmenting the training set by 2, 3, and 4 times its original size, we combined the original training set with the generated set for training to make full use of the available data. Then the models were evaluated on the test set, which is a subset of the original dataset that is maintained separately.

**Augmentation via sequence pairing** Up to this point, only sequences from transcript 1 were used to classify 1-2 trajectory, and only sequences from the last transcript were used to classify 2-last trajectory. Incorporating sequences from transcript 2 into the model, as well as the temporal relationship between transcript 1 and transcript 2, would make full use of available data. Similarly, we wished to incorporate utterances from transcript 2 in the prediction of 2-last trajectory.

Inspired by [90], which augmented the dataset by concatenating random two halves of the sequences with the same polarity, we propose a novel method of pairing sequences between the different transcripts from the same patient. Specifically, we paired sequences from transcript 1 or the last transcript (text $A$) with sequences from transcript 2, 3 or 4 (text $B$). Each pair shared a single label indicating the trajectory. After creating the sequence pairs between text $A$ and text $B$, the training set was augmented by randomly selecting from these pairs for each patient. Assuming an equal number of sequences $n$ for each transcript, the dataset could be augmented to at most $n^2$ pairs using this technique.

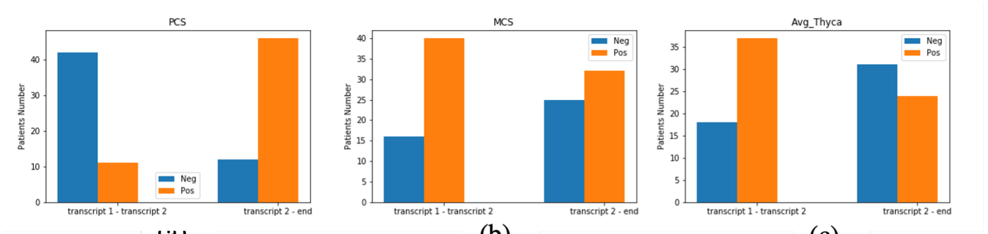
Then sequence pairs were input into BERT separated by the special [SEP] token in the following form: [CLS]sequence1[SEP]sequence 2[SEP]. The separator between two sequences indicated the end of sequence 1 and the start of sequence 2. During training, each pair was iteratively truncated to fit the maximum length requirement.

## 5.7 Experiments and Results

### Experimental setup

In this study we selected the Physical Score (PCS) and Mental Score (MCS) of the SF-12 and the average of the score on THYCA-QoL (Average THYCA) as primary outcome measures. Each transcript consists of alternating utterances between the patient and the interviewer. After removing all interviewer utterances, as well as all utterances under five words long, the remaining lines in the cleaned transcript were used as a sequence to be input into the models. Missing data were considered missing at random. Patients missing a particular HRQOL outcome score were excluded from the corresponding analysis. The distribution of each label on patient level after preprocessing is shown in Figure 5.2. Each patient has up to 5 transcripts, and each transcript has 250 utterances. From Figure 5.2, there are more negative samples in preoperative phase and more positive ones afterwards, as we expected.

After balancing the dataset using random undersampling of the majority class, the number of sequences corresponding to PCS, MCS, and Average THYCA labels were 18074, 14450, and 16056, respectively. For each experiment, 80% of the dataset was used for training, and 20% held out as a test set.



(a) (b) (c)

Figure 5.2: Distribution of HRQOL trajectories after preprocessing

To fine-tune BERT on classification, 5-fold cross validation was used to find the optimal hyperparameter values. The model bert-base-uncased provided in the package of huggingface was used [135]. We use batch size 16 and maximum sequence length of 256 (sentences with more than 256 tokens were truncated). During the optimization procedure, a dropout probability of 0.1 was used. AdamW was used as an optimizer, we used epoch$\in \{3, 4, 5\}$ and the learning rate was set to 2e-5 with a linear decay scheduler.

To fine-tune GPT-2, we use DistilGPT2, which is a smaller and faster version of GPT-2, also provided in huggingface's package [135]. We use 3 epochs, learning rate 3e-5, maximum sequence length 1024, and batch size 16. The output sequences were filtered using top-k sampling with k=50, and selected using top-p sampling with $p \geqslant 0.8$. We synthesized 60000 samples for each class. The generated sequences were pre-processed in the manner described in the preprocessing section. While using BERT as a filtering, generated sequences were discarded if their predicted labels did not match the initial prompt, or if the associated probability scores were less than 0.9.

## Experimental results

**LIWC** After 500 epochs of training, the accuracy of sequence classification using logistic regression on the test set with respect to PCS, MCS, and Average THYCA were 0.58, 0.55, 0.55, respectively. While this model had some classification power, it was only slightly better than chance. This may have been due to the low dimensionality and sparsity of the LIWC features, which failed to capture enough features correlated with the labels, a disadvantage inherent in using a fixed set of human engnineered feature categories. Next, we turned to modern deep learning methods to improve upon these results.

**BERT** Results in predicting HRQOL trajectories by fine-tuning BERT are shown in Table 5.2 and Figure 5.3. For PCS, MCS, and average THYCA, the sequence classification accuracies were 0.64, 0.64, and 0.61, respectively; the AUC were 0.69, 0.69, and 0.65. For the performance at the patient level, when the label was determined using a majority vote of the classification of all sequences belonging to the patient, the accuracy for PCS, MCS, and Avg_Thyca were 0.77, 0.62 and 0.61, respectively. Thus when fine-tuning BERT on the down-stream classification problem, it outperformed the baseline model with LIWC features by improving model generalization capabilities.

**Training set augmentation via GPT-2** The next set of experiments augment the training set by 2, 3, and 4 times its original size and then evaluates on the same test set. The results are shown in Table 5.3. From the results, the model performance improved with the increase of training set size.

| | PCS | MCS | Avg_Thyca |
|---|---|---|---|
| Accuracy | 0.64 | 0.64 | 0.61 |
| F1 | 0.62 | 0.64 | 0.59 |
| Precision | 0.65 | 0.65 | 0.62 |
| Recall | 0.58 | 0.64 | 0.56 |
| AUC | 0.69 | 0.69 | 0.65 |

Table 5.2: Performance of fine-tuning BERT on the original dataset

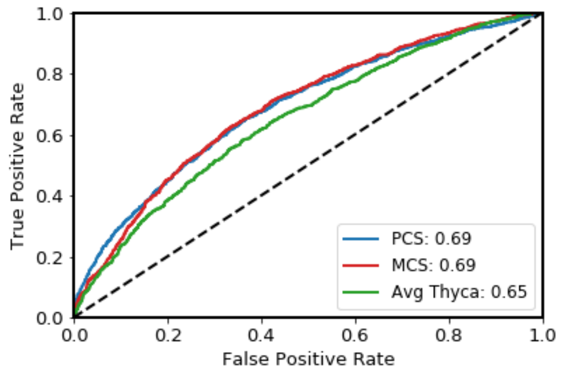


Figure 5.3: ROC curves of PCS, MCS, and Avg_Thyca by fine-tuning BERT

When the training set was augmented to 4 times that of the original dataset, classification accuracy and AUC were 0.71 and 0.78, respectively. These improve the accuracy by around 11% in comparison to the model trained on the original dataset. In addition, the accuracy at the patient level determined by a majority vote of sequences classification was 0.78 when training set was 4 times the size of the original. Thus, we find that the model performance can be significantly improved by augmenting the training set using GPT-2 generated sequences.

**Training set augmentation via utterance pairing** Similarly, the training set was augmented to 2, 3, and 4 times its original size and the model was evaluated on the same test set. The results are shown in Table 5.4.

| | Training set size (As multiple of original) | | | |
|---|---|---|---|---|
| | 1 | 2 | 3 | 4 |
| Accuracy | 0.64 | 0.68 | 0.70 | 0.71 |
| F1 | 0.62 | 0.67 | 0.68 | 0.69 |
| Precision | 0.65 | 0.70 | 0.72 | 0.73 |
| Recall | 0.58 | 0.63 | 0.65 | 0.66 |
| AUC | 0.69 | 0.74 | 0.77 | 0.78 |

Table 5.3: Performance with training set augmentated by GPT-2

From the results, with the augmentation of the training set to 4 times of its original size, the classification accuracy can be improved by around 7%. The accuracy at patient level was 0.68.

| | Training set size (As multiple of original) | | | |
|---|---|---|---|---|
| | 1 | 2 | 3 | 4 |
| Accuracy | 0.62 | 0.65 | 0.65 | 0.68 |
| F1 | 0.61 | 0.63 | 0.64 | 0.67 |
| Precision | 0.63 | 0.66 | 0.67 | 0.68 |
| Recall | 0.59 | 0.60 | 0.60 | 0.67 |
| AUC | 0.66 | 0.70 | 0.71 | 0.73 |

Table 5.4: Performance with training set augmented by pairing utterances

In summary, we have classified the trajectory of HRQOL measurements by using patient utterances. We first showed that compared to using the logistic regression model with LIWC features, fine-tune BERT on down-stream classification can boost the performance since a better representation can be generated. Then we proposed two training data augmentation methods to further improve the performance. Specifically, we augment the training set by using GPT-2 to generate label-invariant sequences, and by randomly pairing utterances from two transcripts for each patient.

Classifying HRQOL trajectory is of great clinical relevance for assessing and forecasting a cancer patient's response to treatment. Since exact HRQOL values represent one point in time, they are of less interest than HRQOL trajectory. Moreover, this suggests that narrative data analysis may provide more detailed or nuanced HRQOL information than survey results alone. Based on these findings, we propose using our methodology for the extraction of PRO using patient narrative data.

## 5.8 Conclusions

NLP models that extract HRQOL from narrative data such as interview transcripts can allow for tailoring of treatment options and identification of patients at risk for worse outcomes. We demonstrated the effectiveness of leveraging modern deep language models to predict the HRQOL trajectories. We also proposed two training data augmentation methods to further improve the performance. Future work will validate the methods in this paper on transcriptions of unstructured patient-physician conversations taking place in a typical office visit.

# 6 PRATICAL TASKS IN SOCIAL MEDIA ANALYSIS

## 6.1 Overview

This chapter shows how modern language models such as those discussed in the previous chapters can be applied to real-world problems in communication analysis. In Section 6.2, we applied language models to analyze favorability and hesitancy about the COVID-19 vaccine by analyzing Twitter posts. This work first appeared in [62] and [59]. In section 6.3, we analyze the public discourse on MeToo across four platforms: Facebook, Twitter, Instagram, and Reddit. This work first appeared in [118].

## 6.2 Ideological Differences about COVID-19 Vaccine Favorability and Hesitancy

Vaccine hesitancy has been a growing public health issue, but during COVID-19, understanding vaccine hesitancy and promoting vaccine favorability takes on a troubling immediacy. With the growing political polarization on scientific issues, the COVID-19 vaccine-related sentiment has recently become divided across ideological lines. This study aims to understand how vaccine favorability and specific vaccine-related concerns including possible side effects, distrust in medical professionals, and conspiratorial beliefs concerning COVID-19 vaccines were articulated and transmitted by Twitter users from opposing ideological camps and with different follower scopes. Using a combination of computational approaches, including supervised machine-learning and structural topic modeling, we examined tweets surrounding COVID-19 vaccination.

We used Synthesio to retrieve a 1% random sample of tweets containing a broad range of COVID-19-related keywords between 1 March and 30 June 2020. A collection of vaccine-related keywords were used to extract a vaccine-related dataset, resulting in a sample of 349,979 tweets. The coding schemes for variables of interest pertain to general vaccination favorability, side effects, distrust in scientists, and conspiracy theories. We classified tweets along these variables of interests by manually labeling a subset of tweets, which were used to train a ML classifier. Specifically, two human coders encoded vaccine favorability on a binary basis for each of the following variables: vaccine favorability, vaccine unfavorability, side effects, distrust, and conspiracy. They labeled 5,000 randomly selected tweets and continued coding until the balance between the two classes was roughly reached.

We then fine-tuned BERT on the downstream classification problem. To deal with the unbalanced dataset, we use under-sampling, which randomly removes samples from the majority classes. After fine-tuning, the models were used to automatically label the remaining tweets. The performance in terms of each variable is shown in Table 6.1. Samples labeled by both human and BERT are combined for further analysis.

| | Accuracy | AUC | F1 |
|---|---|---|---|
| Vaccine favorability | 0.71 | 0.79 | 0.70 |
| Vaccine unfavorability | 0.75 | 0.82 | 0.74 |
| Side effects | 0.92 | 0.95 | 0.95 |
| Distrust | 0.84 | 0.89 | 0.85 |
| Conspiracy | 0.86 | 0.90 | 0.81 |

Table 6.1: BERT performance on the testset

While the raw results of this study may appear modest, with classification accuracies in the range of 70-90%, the presentation in [62] has already been cited 46 times, indicating that the work is both timely and of significant interest to researchers in the application domain.

## 6.3 Public Discourse Before and After #MeToo across Four Platforms

In digital spaces, sharing personal stories and sexually traumatic experiences have created a network of acknowledgment, raising the consciousness about gender violence and inequality. The expanded capacity to personal expressions and social sharing has catalyzed different types of politicization: it has a mobilizing potential for social change, while inviting increased contention. Focusing on public discourses on #MeToo and feminism over three years across Twitter, Facebook, Instagram, and Reddit, we employ supervised machine learning to classify the different strands of discourses and use time-series modeling to formally model their interrelations.

Twitter, Facebook, Instagram, and Reddit data were collected using Synthesio (www.synthesio.com), which is a social listening platform that collects online conversations across various media platforms and sources. Synthesio allows us to collect Twitter data from Twitter Decahose Stream, available public facing (non-regional) content from Facebook and Instagram, and Reddit data from Firehose of all major subreddits. We collected #MeToo, sexual violence, and related discourses on these platforms over a three-year time span (December 1, 2016 – April 31, 2020), using search strings and relevant hashtags such as "#MeToo", "sexual assault," and "sexual abuse", etc. Given the focus of the study, we retained English content only, resulting in a total number of 26,965,357 social media posts (Twitter 14,894,578, Facebook 1,803,220, Instagram 2,186,127, and Reddit 8,081,432).

Then we used a supervised machine learning technique to classify social media posts into key variables of the study. Two trained undergraduate students coded the following variables, all of which obtained the satisfactory level of intercoder reliability: (1) networked acknowledgment,

(2) activism, (3) Feminism contention. Using the 11,000 social media posts with human labels, we trained a deep neural network on the subset of human-coded data; then this model was used to label the remaining unlabeled utterances. Specifically, we fine tune the pre-trained BERT, which is one of the widely established transformer-based language models, on the down-stream binary classification and let the trained model be a classifier to automatically annotate the remaining samples.

In our experiments, we first employed commonly used text preprocessing methods such as removing URL, RT, , extra space, and non-ASCII words and characters to process the sequences. To fine-tune BERT on the down-stream classification, a linear classifier was added on top, and the hyperparameters were tuned together to better facilitate the predictions. The BERT output of the first token was used as a sequence representation, where each sequence embedding is of dimension 768. For each discourse hyperparameter, a grid-search with 5-fold cross-validation was used on the training set to find optimal values. Besides, given the imbalanced nature of the dataset (i.e., distribution is extremely skewed), we applied an under-sampling approach. The performance of our final variables is shown in Table 6.2.

| | Accuracy | Macro F1 | Precision | Recall | AUC |
|---|---|---|---|---|---|
| Networked acknowledgement | 0.78 | 0.78 | 0.78 | 0.78 | 0.86 |
| General feminism criticism | 0.85 | 0.86 | 0.84 | 0.87 | 0.93 |
| Feminism contention | 0.85 | 0.86 | 0.84 | 0.87 | 0.93 |

Table 6.2: Supervised machine learning results for three discourses

# 7 FUTURE WORK

## 7.1 Apply LA-SCL to Real-World Social Media Analysis

While doing human annotations, people leverage codebooks, which contain keywords or rules to assign sentiment labels. Given that, we can create a description of each class from this information, thus applying our proposed LA-SCL described in Chapter 2. By applying this, the requirement of labeled samples is reduced while maintaining the performance on the downstream classification problem.

## 7.2 Whether Contrastive Learning can Improve Reinforcement Learning for Human Feedback (RLHF)

When applying RLHF, the choice of representations can influence how well the model learns from human feedback. By using contrastive learning to pretrain the model, we provide it with a stronger initial understanding of the data's underlying structure. This can lead to more effective learning during RLHF, as the model has better-starting representations to build upon. High-quality representations obtained through contrastive learning can potentially enhance the model's ability to generalize from human feedback to unseen scenarios.

## 7.3 Investigate Whether the Discriminative Representations could Improve Generative Language Model Performance

Given the exploding of very large-scale generative language models (e.g., ChatGPT), it is natural to think about how the emergence of aligned or separated representations correlates with the generative prompt-based prediction ability. On the other hand, Zimmermann et al. [164] proved that feedforward models trained with objectives belonging to the commonly used InfoNCE family learn to invert the underlying generative model of the observed data implicitly. It is important to bridge the correlation between the discriminative representations and the generative ability of LLMs.

## 7.4 Apply LLMs to Social Media Analysis

Regarding real-world scenarios, it is not only costly to obtain human-annotated data but also ends up with imbalanced or long-tailed distributions. These challenges can be mitigated by leveraging large language models (LLMs) such as ChatGPT, as a lot of work has shown its superior ability on few-shot or even zero-shot learning. Work such as Chain-of-Thought (CoT) [132] has significantly improved the ability of large language models to perform complex reasoning.

Inspired by these investigations, we can apply LLMs such as ChatGPT to our social media analysis in collaboration with CCCR and SMAD. Particularly, we can first show a few human-annotated samples to ChatGPT by combing the text and its corresponding sentiment information. Then we can ask the LLMs to make predictions on massive unlabeled samples by giving them proper prompts. This will largely reduce the labor work on human annotation.

## REFERENCES

[1] Alsentzer, Emily, John R Murphy, Willie Boag, Wei-Hung Weng, Di Jin, Tristan Naumann, and Matthew McDermott. 2019. Publicly available clinical bert embeddings. *arXiv preprint arXiv:1904.03323*.

[2] Anaby-Tavor, Ateret, Boaz Carmeli, Esther Goldbraich, Amir Kantor, George Kour, Segev Shlomov, Naama Tepper, and Naama Zwerdling. 2020. Do not have enough data? deep learning to the rescue! In *Proceedings of the aaai conference on artificial intelligence*, vol. 34, 7383–7390.

[3] Auer, Sören, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. Dbpedia: A nucleus for a web of open data. In *The semantic web: 6th international semantic web conference, 2nd asian semantic web conference, iswc 2007+ aswc 2007, busan, korea, november 11-15, 2007. proceedings*, 722–735. Springer.

[4] Baumgartner, Jason. 2019. Reddit comments dumps.

[5] Bayer, Markus, Marc-André Kaufhold, and Christian Reuter. 2021. A survey on data augmentation for text classification. *arXiv preprint arXiv:2107.03158*.

[6] Bennett, Paul N, Ryen W White, Wei Chu, Susan T Dumais, Peter Bailey, Fedor Borisyuk, and Xiaoyuan Cui. 2012. Modeling the impact of short-and long-term behavior on search personalization. In *Proceedings of the 35th international acm sigir conference on research and development in information retrieval*, 185–194.

[7] Bevilacqua, Michele, Giuseppe Ottaviano, Patrick Lewis, Wen-tau Yih, Sebastian Riedel, and Fabio Petroni. 2022. Autoregressive search engines: Generating substrings as document identifiers. *arXiv preprint arXiv:2204.10628*.

[8] Bielak, Piotr, Kamil Tagowski, Maciej Falkiewicz, Tomasz Kajdanowicz, and Nitesh V Chawla. 2021. Fildne: A framework for incremental learning of dynamic networks embeddings. *Knowledge-Based Systems* 107453.

[9] Biggiogera, Jacopo, George Boateng, Peter Hilpert, Matthew Vowels, Guy Bodenmann, Mona Neysari, Fridtjof Nussbeck, and Tobias Kowatsch. 2021. Bert meets liwc: Exploring state-of-the-art language models for predicting communication behavior in couples' conflict interactions. *arXiv preprint arXiv:2106.01536*.

[10] Brennan, M. D. n.d. The 20 newsgroups dataset. Available online.

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

[12] Bui, Duc, Kshitiz Malik, Jack Goetz, Honglei Liu, Seungwhan Moon, Anuj Kumar, and Kang G Shin. 2019. Federated user representation learning. *arXiv preprint arXiv:1909.12535*.

[13] Caron, Mathilde, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. 2020. Unsupervised learning of visual features by contrasting cluster assignments. *Advances in neural information processing systems* 33:9912–9924.

[14] Chen, Boli, Xin Huang, Lin Xiao, Zixin Cai, and Liping Jing. 2020. Hyperbolic interaction model for hierarchical multi-label classification. In *Proceedings of the aaai conference on artificial intelligence*, vol. 34, 7496–7503.

[15] Chen, Chun-Fu Richard, Quanfu Fan, and Rameswar Panda. 2021. Crossvit: Cross-attention multi-scale vision transformer for image classification. In *Proceedings of the ieee/cvf international conference on computer vision*, 357–366.

[16] Chen, Haibin, Qianli Ma, Zhenxi Lin, and Jiangyue Yan. 2021. Hierarchy-aware label semantics matching network for hierarchical text classification. In *Proceedings of the 59th annual meeting of the association for computational linguistics and the 11th international joint conference on natural language processing (volume 1: Long papers)*, 4370–4379.

[17] Chen, Jiaao, Zichao Yang, and Diyi Yang. 2020. Mixtext: Linguistically-informed interpolation of hidden space for semi-supervised text classification. *arXiv preprint arXiv:2004.12239*.

[18] Chen, Junfan, Richong Zhang, Yongyi Mao, and Jie Xu. 2022. Contrastnet: A contrastive learning framework for few-shot text classification. In *Proceedings of the aaai conference on artificial intelligence*, vol. 36, 10492–10500.

[19] Chen, Tao, Haizhou Shi, Siliang Tang, Zhigang Chen, Fei Wu, and Yueting Zhuang. 2021. Cil: Contrastive instance learning framework for distantly supervised relation extraction. *arXiv preprint arXiv:2106.10855*.

[20] Chen, Ting, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, 1597–1607. PMLR.

[21] Chen, Yinpeng, Xiyang Dai, Dongdong Chen, Mengchen Liu, Xiaoyi Dong, Lu Yuan, and Zicheng Liu. 2022. Mobile-former: Bridging

mobilenet and transformer. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, 5270–5279.

[22] Chen, Zhe, Yuchen Duan, Wenhai Wang, Junjun He, Tong Lu, Jifeng Dai, and Yu Qiao. 2022. Vision transformer adapter for dense predictions. *arXiv preprint arXiv:2205.08534*.

[23] Chi, Jianfeng, William Shand, Yaodong Yu, Kai-Wei Chang, Han Zhao, and Yuan Tian. 2022. Conditional supervised contrastive learning for fair text classification. *arXiv preprint arXiv:2205.11485*.

[24] Chuang, Yung-Sung, Rumen Dangovski, Hongyin Luo, Yang Zhang, Shiyu Chang, Marin Soljačić, Shang-Wen Li, Wen-tau Yih, Yoon Kim, and James Glass. 2022. Diffcse: Difference-based contrastive learning for sentence embeddings. *arXiv preprint arXiv:2204.10298*.

[25] Church, Kenneth Ward. 2017. Word2vec. *Natural Language Engineering* 23(1):155–162.

[26] Coulombe, Claude. 2018. Text data augmentation made simple by leveraging nlp cloud apis. *arXiv preprint arXiv:1812.04718*.

[27] Cui, Yutao, Cheng Jiang, Limin Wang, and Gangshan Wu. 2022. Mixformer: End-to-end tracking with iterative mixed attention. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, 13608–13618.

[28] Damashek, Marc. 1995. Gauging similarity with n-grams: Language-independent categorization of text. *Science* 267(5199): 843–848.

[29] De Cao, Nicola, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. 2020. Autoregressive entity retrieval. *arXiv preprint arXiv:2010.00904*.

[30] Dehghani, Mostafa, Hamed Zamani, Aliaksei Severyn, Jaap Kamps, and W Bruce Croft. 2017. Neural ranking models with weak supervision. In *Proceedings of the 40th international acm sigir conference on research and development in information retrieval*, 65–74.

[31] Demszky, Dorottya, Dana Movshovitz-Attias, Jeongwoo Ko, Alan Cowen, Gaurav Nemade, and Sujith Ravi. 2020. Goemotions: A dataset of fine-grained emotions. *arXiv preprint arXiv:2005.00547*.

[32] Deng, Yang, Yaliang Li, Wenxuan Zhang, Bolin Ding, and Wai Lam. 2022. Toward personalized answer generation in e-commerce via multi-perspective preference modeling. *ACM Transactions on Information Systems (TOIS)* 40(4):1–28.

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

[34] ———. 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*, 4171–4186.

[35] Dou, Zhicheng, Ruihua Song, and Ji-Rong Wen. 2007. A large-scale evaluation and analysis of personalized search strategies. In *Proceedings of the 16th international conference on world wide web*, 581–590.

[36] Dwibedi, Debidatta, Yusuf Aytar, Jonathan Tompson, Pierre Sermanet, and Andrew Zisserman. 2021. With a little help from my friends: Nearest-neighbor contrastive learning of visual representations. In *Proceedings of the ieee/cvf international conference on computer vision*, 9588–9597.

[37] Fang, Hongchao, Sicheng Wang, Meng Zhou, Jiayuan Ding, and Pengtao Xie. 2020. Cert: Contrastive self-supervised learning for language understanding. *arXiv preprint arXiv:2005.12766*.

[38] Faramarzi, Mojtaba, Mohammad Amini, Akilesh Badrinaaraayanan, Vikas Verma, and Sarath Chandar. 2020. Patchup: A regularization technique for convolutional neural networks. *arXiv preprint arXiv:2006.07794*.

[39] Feng, Steven Y, Varun Gangal, Jason Wei, Sarath Chandar, Soroush Vosoughi, Teruko Mitamura, and Eduard Hovy. 2021. A survey of data augmentation approaches for nlp. *arXiv preprint arXiv:2105.03075*.

[40] Gao, Tianyu, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. *arXiv preprint arXiv:2104.08821*.

[41] Ge, Songwei, Zhicheng Dou, Zhengbao Jiang, Jian-Yun Nie, and Ji-Rong Wen. 2018. Personalizing search results using hierarchical rnn with query-aware attention. In *Proceedings of the 27th acm international conference on information and knowledge management*, 347–356.

[42] Ge, Weifeng. 2018. Deep metric learning with hierarchical triplet loss. In *Proceedings of the european conference on computer vision (eccv)*, 269–285.

[43] Gillick, Daniel, Alessandro Presta, and Gaurav Singh Tomar. 2018. End-to-end retrieval in continuous space. *arXiv preprint arXiv:1811.08008*.

[44] Giorgi, John M, Osvald Nitski, Gary D Bader, and Bo Wang. 2020. Declutr: Deep contrastive learning for unsupervised textual representations. *arXiv preprint arXiv:2006.03659*.

[45] Grbovic, Mihajlo, and Haibin Cheng. 2018. Real-time personalization using embeddings for search ranking at airbnb. In *Proceedings of the 24th acm sigkdd international conference on knowledge discovery & data mining*, 311–320.

[46] Greaves, Felix, Daniel Ramirez-Cano, Christopher Millett, Ara Darzi, and Liam Donaldson. 2013. Use of sentiment analysis for capturing patient experience from free-text comments posted online. *Journal of medical Internet research* 15(11):e239.

[47] Grill, Jean-Bastien, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. 2020. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems* 33:21271–21284.

[48] Gunel, Beliz, Jingfei Du, Alexis Conneau, and Ves Stoyanov. 2020. Supervised contrastive learning for pre-trained language model fine-tuning. *arXiv preprint arXiv:2011.01403*.

[49] Guo, Hongyu, Yongyi Mao, and Richong Zhang. 2019. Augmenting data with mixup for sentence classification: An empirical study. *arXiv preprint arXiv:1905.08941*.

[50] Guo, Qian, Wei Chen, and Huaiyu Wan. 2021. Aol4ps: A large-scale data set for personalized search. *Data Intelligence* 3(4):548–567.

[51] Han, Xu, Pengfei Yu, Zhiyuan Liu, Maosong Sun, and Peng Li. 2018. Hierarchical relation extraction with coarse-to-fine grained attention. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, 2236–2245.

[52] Harris, Ethan, Antonia Marcu, Matthew Painter, Mahesan Niranjan, Adam Prügel-Bennett, and Jonathon Hare. 2020. Fmix: Enhancing mixed sample data augmentation. *arXiv preprint arXiv:2002.12047*.

[53] Hasan, Fatema, Kevin S Xu, James R Foulds, and Shimei Pan. 2021. Learning user embeddings from temporal social media data: A survey. *arXiv preprint arXiv:2105.07996*.

[54] He, Junxian, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2021. Towards a unified view of parameter-efficient transfer learning. *arXiv preprint arXiv:2110.04366*.

[55] He, Kaiming, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, 9729–9738.

[56] Houlsby, Neil, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, 2790–2799. PMLR.

[57] Huang, Kexin, Jaan Altosaar, and Rajesh Ranganath. 2019. Clinicalbert: Modeling clinical notes and predicting hospital readmission. *arXiv preprint arXiv:1904.05342*.

[58] Hui, Kai, Honglei Zhuang, Tao Chen, Zhen Qin, Jing Lu, Dara Bahri, Ji Ma, Jai Prakash Gupta, Cicero Nogueira dos Santos, Yi Tay, et al. 2022. Ed2lm: Encoder-decoder to language model for faster document re-ranking inference. *arXiv preprint arXiv:2204.11458*.

[59] Hwang, Juwon, Min-Hsin Su, Xiaoya Jiang, Ruixue Lian, Arina Tveleneva, and Dhavan Shah. 2022. Vaccine discourse during the

onset of the covid-19 pandemic: Topical structure and source patterns informing efforts to combat vaccine hesitancy. *Plos one* 17(7): e0271394.

[60] Jaiswal, Ashish, Ashwin Ramesh Babu, Mohammad Zaki Zadeh, Debapriya Banerjee, and Fillia Makedon. 2020. A survey on contrastive self-supervised learning. *Technologies* 9(1):2.

[61] ———. 2021. A survey on contrastive self-supervised learning. *Technologies* 9(1):2.

[62] Jiang, Xiaoya, Min-Hsin Su, Juwon Hwang, Ruixue Lian, Markus Brauer, Sunghak Kim, and Dhavan Shah. 2021. Polarization over vaccination: Ideological differences in twitter expression about covid-19 vaccine favorability and specific hesitancy concerns. *Social Media+ Society* 7(3):20563051211048413.

[63] Jiang, Zheng Ping, Sarah Ita Levitan, Jonathan Zomick, and Julia Hirschberg. 2020. Detection of mental health from reddit via deep contextualized representations. In *Proceedings of the 11th international workshop on health text mining and information analysis*, 147–156.

[64] Kabbach, Alexandre, Kristina Gulordava, and Aurélie Herbelot. 2019. Towards incremental learning of word embeddings using context informativeness. In *Proceedings of the 57th annual meeting of the association for computational linguistics: Student research workshop*, 162–168. Association for Computational Linguistics.

[65] Kalyan, Katikapalli Subramanyam, Ajit Rajasekharan, and Sivanesan Sangeetha. 2021. Ammus: A survey of transformer-based pretrained models in natural language processing. *arXiv preprint arXiv:2108.05542*.

[66] Kang, Taegwan, Hwanhee Lee, Byeongjin Choe, and Kyomin Jung. 2021. Entangled bidirectional encoder to autoregressive decoder for sequential recommendation. In *Proceedings of the 44th international acm sigir conference on research and development in information retrieval*, 1657–1661.

[67] Kang, Wang-Cheng, and Julian McAuley. 2018. Self-attentive sequential recommendation. In *2018 ieee international conference on data mining (icdm)*, 197–206. IEEE.

[68] Karpukhin, Vladimir, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*.

[69] Khosla, Prannay, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. 2020. Supervised contrastive learning. *arXiv preprint arXiv:2004.11362*.

[70] Kim, Jang-Hyun, Wonho Choo, and Hyun Oh Song. 2020. Puzzle mix: Exploiting saliency and local statistics for optimal mixup. In *International conference on machine learning*, 5275–5285. PMLR.

[71] Krishna, Kundan, Amy Pavel, Benjamin Schloss, Jeffrey P Bigham, and Zachary C Lipton. 2021. Extracting structured data from physician-patient conversations by predicting noteworthy utterances. In *Explainable ai in healthcare and medicine*, 155–169. Springer.

[72] Kumar, Varun, Ashutosh Choudhary, and Eunah Cho. 2020. Data augmentation using pre-trained transformer models. *arXiv preprint arXiv:2003.02245*.

[73] Lee, Hyunji, Sohee Yang, Hanseok Oh, and Minjoon Seo. 2022. Generative retrieval for long sequences. *arXiv preprint arXiv:2204.13596*.

[74] Lee, Jinhyuk, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2020. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics* 36(4):1234–1240.

[75] Li, Bohan, Yutai Hou, and Wanxiang Che. 2021. Data augmentation approaches in natural language processing: A survey. *arXiv preprint arXiv:2110.01852*.

[76] Li, Dongyang, Taolin Zhang, Nan Hu, Chengyu Wang, and Xiaofeng He. 2022. Hiclre: A hierarchical contrastive learning framework for distantly supervised relation extraction. *arXiv preprint arXiv:2202.13352*.

[77] Li, Jiwei, Michel Galley, Chris Brockett, Georgios P Spithourakis, Jianfeng Gao, and Bill Dolan. 2016. A persona-based neural conversation model. *arXiv preprint arXiv:1603.06155*.

[78] Li, Junnan, Pan Zhou, Caiming Xiong, and Steven CH Hoi. 2020. Prototypical contrastive learning of unsupervised representations. *arXiv preprint arXiv:2005.04966*.

[79] Li, Xiang Lisa, and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*.

[80] Li, Yikuan, Shishir Rao, Jose Roberto Ayala Solares, Abdelaali Hassaine, Rema Ramakrishnan, Dexter Canoy, Yajie Zhu, Kazem Rahimi, and Gholamreza Salimi-Khorshidi. 2020. Behrt: transformer for electronic health records. *Scientific reports* 10(1):1–12.

[81] Lian, Ruixue, Vivian Hsiao, Juwon Hwang, Yue Ou, Sarah E Robbins, Nadine P Connor, Cameron L Macdonald, Rebecca S Sippel,

William A Sethares, and David F Schneider. 2023. Predicting health-related quality of life change using natural language processing in thyroid cancer. *Intelligence-Based Medicine* 7:100097.

[82] Lian, Ruixue, Che-Wei Huang, Yuqing Tang, Qilong Gu, Chengyuan Ma, and Chenlei Guo. 2022. Incremental user embedding modeling for personalized text classification. In *Icassp 2022-2022 ieee international conference on acoustics, speech and signal processing* (*icassp*), 7832–7836. IEEE.

[83] Lian, Ruixue, William Sethares, and Junjie Hu. 2023. Learning label hierarchy with supervised contrastive learning. *Preprint*.

[84] Liu, Hong, Jeff Z HaoChen, Adrien Gaidon, and Tengyu Ma. 2021. Self-supervised learning is more robust to dataset imbalance. *arXiv preprint arXiv:2110.05025*.

[85] Liu, Jingjing, Chang Liu, and Nicholas J Belkin. 2020. Personalization in text information retrieval: A survey. *Journal of the Association for Information Science and Technology* 71(3):349–369.

[86] Liu, Jiongnan, Zhicheng Dou, Qiannan Zhu, and Ji-Rong Wen. 2022. A category-aware multi-interest model for personalized product search. In *Proceedings of the acm web conference 2022*, 360–368.

[87] Liu, Xiao, Fanjin Zhang, Zhenyu Hou, Li Mian, Zhaoyu Wang, Jing Zhang, and Jie Tang. 2021. Self-supervised learning: Generative or contrastive. *IEEE Transactions on Knowledge and Data Engineering*.

[88] Losing, Viktor, Barbara Hammer, and Heiko Wersing. 2018. Incremental on-line learning: A review and comparison of state of the art algorithms. *Neurocomputing* 275:1261–1274.

[89] Lowell, David, Brian E Howard, Zachary C Lipton, and Byron C Wallace. 2020. Unsupervised data augmentation with naive augmentation and without unlabeled data. *arXiv preprint arXiv:2010.11966*.

[90] Luque, Franco M. 2019. Atalaya at tass 2019: Data augmentation and robust embeddings for sentiment analysis. *arXiv preprint arXiv:1909.11241*.

[91] Van der Maaten, Laurens, and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research* 9(11).

[92] Małkiński, Mikołaj, and Jacek Mańdziuk. 2022. Multi-label contrastive learning for abstract visual reasoning. *IEEE Transactions on Neural Networks and Learning Systems*.

[93] Meng, Yu, Chenyan Xiong, Payal Bajaj, Saurabh Tiwary, Paul Bennett, Jiawei Han, and Xia Song. 2021. Coco-lm: Correcting and contrasting text sequences for language model pretraining. *arXiv preprint arXiv:2102.08473*.

[94] Mikolov, Tomas, 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.

[95] Miller, George A. 1995. Wordnet: a lexical database for english. *Communications of the ACM* 38(11):39–41.

[96] Mireshghallah, Fatemehsadat, Vaishnavi Shrivastava, Milad Shokouhi, Taylor Berg-Kirkpatrick, Robert Sim, and Dimitrios Dimitriadis. 2021. Useridentifier: implicit user representations for simple and effective personalized sentiment analysis. *arXiv preprint arXiv:2110.00135*.

[97] Murdock, Calvin, Zhen Li, Howard Zhou, and Tom Duerig. 2016. Blockout: Dynamic model selection for hierarchical deep networks. In *Proceedings of the ieee conference on computer vision and pattern recognition*, 2583–2591.

[98] Nakano, Reiichiro, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. 2021. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*.

[99] Ni, Jianmo, Gustavo Hernández Ábrego, Noah Constant, Ji Ma, Keith B Hall, Daniel Cer, and Yinfei Yang. 2021. Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models. *arXiv preprint arXiv:2108.08877*.

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

[101] Pakhomov, Serguei, Nilay Shah, Penny Hanson, Saranya Balasubramaniam, and Steven A Smith. 2008. Automatic quality of life prediction using electronic medical records. In *Amia annual symposium proceedings*, vol. 2008, 545. American Medical Informatics Association.

[102] Pakhomov, Serguei VS, Nilay D Shah, Holly K Van Houten, Penny L Hanson, and Steven A Smith. 2011. The role of the electronic medical record in the assessment of health related quality of life. In *Amia annual symposium proceedings*, vol. 2011, 1080. American Medical Informatics Association.

[103] Park, Jin-Hee, Yong Sik Jung, Ji Young Kim, Yujung Jo, and Sun Hyoung Bae. 2020. Trajectories of health-related quality of life in breast cancer patients. *Supportive Care in Cancer* 28(7):3381–3389.

[104] Pei, Jiahuan, Pengjie Ren, and Maarten de Rijke. 2021. A cooperative memory network for personalized task-oriented dialogue systems with incomplete user profiles. In *Proceedings of the web conference 2021*, 1552–1561.

[105] Pennebaker, James W, Roger J Booth, Ryan L Boyd, and Martha E Francis. 2015. Linguistic inquiry an d word count: Liwc2015 operator's manual. *Retrieved June* 20:2017.

[106] Pulijala, Ashwin, and Susan Gauch. 2004. Hierarchical text classification. In *International conference on cybernetics and information technologies, systems and applications: Citsa*, vol. 1, 257–262.

[107] Qu, Yanru, Dinghan Shen, Yelong Shen, Sandra Sajeev, Jiawei Han, and Weizhu Chen. 2020. Coda: Contrast-enhanced and diversity-promoting data augmentation for natural language understanding. *arXiv preprint arXiv:2010.08670*.

[108] Quteineh, Husam, Spyridon Samothrakis, and Richard Sutcliffe. 2020. Textual data augmentation for efficient active learning on tiny datasets. Association for Computational Linguistics.

[109] Radford, Alec, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog* 1(8):9.

[110] Raffel, Colin, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.* 21(140):1–67.

[111] Regina, Mehdi, Maxime Meyer, and Sébastien Goutal. 2020. Text data augmentation: Towards better detection of spear-phishing emails. *arXiv preprint arXiv:2007.02033*.

[112] Rethmeier, Nils, and Isabelle Augenstein. 2021. A primer on contrastive pretraining in language processing: Methods, lessons learned and perspectives. *arXiv preprint arXiv:2102.12982*.

[113] ———. 2023. A primer on contrastive pretraining in language processing: Methods, lessons learned, and perspectives. *ACM Computing Surveys* 55(10):1–17.

[114] Reyes-Ortiz, José A, Beatriz A González-Beltrán, and Lizbeth Gallardo-López. 2015. Clinical decision support systems: a survey of nlp-based approaches from unstructured data. In *2015 26th international workshop on database and expert systems applications (dexa)*, 163–167. IEEE.

[115] Ruixue Lian, Clint Solomon Gustavo Aguilar Pragaash Ponnusamy Jialong Han Chengyuan Ma Chenlei Guo, Sixing Lu. 2023. Personaltm: Transformer memory for personalized search. In *Proceedings of the 46th international acm sigir conference on research and development in information retrieval*.

[116] Sedghamiz, Hooman, Shivam Raval, Enrico Santus, Tuka Alhanai, and Mohammad Ghassemi. 2021. Supcl-seq: Supervised contrastive learning for downstream optimized sequence representations. *arXiv preprint arXiv:2109.07424*.

[117] Sokal, Robert R., and F. James Rohlf. 1962. The comparison of dendrograms by objective methods. *Taxon* 11(2):33–40.

[118] Suk, Jiyoun, Yini Zhang, Zhiying Yue, Rui Wang, Xinxia Dong, Dongdong Yang, and Ruixue Lian. 2023. When the personal be-

comes political: Unpacking the dynamics of sexual violence and gender justice discourses across four social media platforms. *Communication Research* 00936502231154146.

[119] Sun, Lichao, Congying Xia, Wenpeng Yin, Tingting Liang, Philip S Yu, and Lifang He. 2020. Mixup-transformer: Dynamic data augmentation for nlp tasks. *arXiv preprint arXiv:2010.02394*.

[120] Sutskever, Ilya, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. *Advances in neural information processing systems* 27.

[121] Tanana, Michael J, Christina S Soma, Patty B Kuo, Nicolas M Bertagnolli, Aaron Dembe, Brian T Pace, Vivek Srikumar, David C Atkins, and Zac E Imel. 2021. How do you feel? using natural language processing to automatically rate emotion in psychotherapy. *Behavior Research Methods* 1–14.

[122] Tay, Yi, Vinh Q Tran, Mostafa Dehghani, Jianmo Ni, Dara Bahri, Harsh Mehta, Zhen Qin, Kai Hui, Zhe Zhao, Jai Gupta, et al. 2022. Transformer memory as a differentiable search index. *arXiv preprint arXiv:2202.06991*.

[123] Uddin, AFM, Mst Monira, Wheemyung Shin, TaeChoong Chung, Sung-Ho Bae, et al. 2020. Saliencymix: A saliency guided data augmentation strategy for better regularization. *arXiv preprint arXiv:2006.01791*.

[124] Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30.

[125] Verma, Nakul, Dhruv Mahajan, Sundararajan Sellamanickam, and Vinod Nair. 2012. Learning hierarchical similarity metrics. In *2012 ieee conference on computer vision and pattern recognition*, 2280–2287. IEEE.

[126] Verma, Vikas, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. 2019. Manifold mixup: Better representations by interpolating hidden states. In *International conference on machine learning*, 6438–6447. PMLR.

[127] Voita, Elena, Rico Sennrich, and Ivan Titov. 2019. The bottom-up evolution of representations in the transformer: A study with machine translation and language modeling objectives. *arXiv preprint arXiv:1909.01380*.

[128] Wang, Tongzhou, and Phillip Isola. 2020. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *International conference on machine learning*, 9929–9939. PMLR.

[129] Wang, William Yang, and Diyi Yang. 2015. That's so annoying!!!: A lexical and frame-semantic embedding based data augmentation approach to automatic categorization of annoying behaviors using# petpeeve tweets. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, 2557–2563.

[130] Wang, Yujing, Yingyan Hou, Haonan Wang, Ziming Miao, Shibin Wu, Hao Sun, Qi Chen, Yuqing Xia, Chengmin Chi, Guoshuai Zhao, et al. 2022. A neural corpus indexer for document retrieval. *arXiv preprint arXiv:2206.02743*.

[131] Wang, Zihan, Peiyi Wang, Lianzhe Huang, Xin Sun, and Houfeng Wang. 2022. Incorporating hierarchy into text encoder: a contrastive

learning approach for hierarchical text classification. *arXiv preprint arXiv:2203.03825*.

[132] Wei, Jason, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems* 35:24824–24837.

[133] Wei, Jason, and Kai Zou. 2019. Eda: Easy data augmentation techniques for boosting performance on text classification tasks. *arXiv preprint arXiv:1901.11196*.

[134] Wei, Kun, Cheng Deng, Xu Yang, and Maosen Li. 2020. Incremental embedding learning via zero-shot translation. *arXiv preprint arXiv:2012.15497*.

[135] Wolf, Thomas, Julien Chaumond, Lysandre Debut, Victor Sanh, Clement Delangue, Anthony Moi, Pierric Cistac, Morgan Funtowicz, Joe Davison, Sam Shleifer, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: System demonstrations*, 38–45.

[136] Wolf, Thomas, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

[137] Wu, Xing, Shangwen Lv, Liangjun Zang, Jizhong Han, and Songlin Hu. 2019. Conditional bert contextual augmentation. In *International conference on computational science*, 84–95. Springer.

[138] Wu, Yuwei, Xuezhe Ma, and Diyi Yang. 2021. Personalized response generation via generative split memory network. In *Proceedings of the 2021 conference of the north american chapter of the association for computational linguistics: Human language technologies*, 1956–1970.

[139] Wu, Zhuofeng, Sinong Wang, Jiatao Gu, Madian Khabsa, Fei Sun, and Hao Ma. 2020. Clear: Contrastive learning for sentence representation. *arXiv preprint arXiv:2012.15466*.

[140] Xie, Qizhe, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V Le. 2019. Unsupervised data augmentation for consistency training. *arXiv preprint arXiv:1904.12848*.

[141] Xie, Qizhe, Zihang Dai, Eduard Hovy, Thang Luong, and Quoc Le. 2020. Unsupervised data augmentation for consistency training. *Advances in neural information processing systems* 33:6256–6268.

[142] Xie, Tianbao, Chen Henry Wu, Peng Shi, Ruiqi Zhong, Torsten Scholak, Michihiro Yasunaga, Chien-Sheng Wu, Ming Zhong, Pengcheng Yin, Sida I Wang, et al. 2022. Unifiedskg: Unifying and multi-tasking structured knowledge grounding with text-to-text language models. *arXiv preprint arXiv:2201.05966*.

[143] Yang, Fan, Kai Wu, Shuyi Zhang, Guannan Jiang, Yong Liu, Feng Zheng, Wei Zhang, Chengjie Wang, and Long Zeng. 2022. Class-aware contrastive semi-supervised learning. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, 14421–14430.

[144] Yoon, Soyoung, Gyuwan Kim, and Kyumin Park. 2021. Ssmix: Saliency-based span mixup for text classification. *arXiv preprint arXiv:2106.08062*.

[145] Yun, Sangdoo, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. 2019. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the ieee/cvf international conference on computer vision*, 6023–6032.

[146] Zeng, Siqi, Remi Tachet des Combes, and Han Zhao. 2023. Learning structured representations by embedding class hierarchy. In *The eleventh international conference on learning representations*.

[147] Zhang, Dejiao, Shang-Wen Li, Wei Xiao, Henghui Zhu, Ramesh Nallapati, Andrew O Arnold, and Bing Xiang. 2021. Pairwise supervised contrastive learning of sentence representations. *arXiv preprint arXiv:2109.05424*.

[148] Zhang, Dejiao, Feng Nan, Xiaokai Wei, Shangwen Li, Henghui Zhu, Kathleen McKeown, Ramesh Nallapati, Andrew Arnold, and Bing Xiang. 2021. Supporting clustering with contrastive learning. *arXiv preprint arXiv:2103.12953*.

[149] Zhang, Han, Songlin Wang, Kang Zhang, Zhiling Tang, Yunjiang Jiang, Yun Xiao, Weipeng Yan, and Wen-Yun Yang. 2020. Towards personalized and semantic retrieval: An end-to-end solution for e-commerce search via embedding learning. In *Proceedings of the 43rd international acm sigir conference on research and development in information retrieval*, 2407–2416.

[150] Zhang, Hongyi, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. 2017. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*.

[151] Zhang, Saizheng, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Personalizing dialogue agents: I have a dog, do you have pets too? *arXiv preprint arXiv:1801.07243*.

[152] Zhang, Shu, Ran Xu, Caiming Xiong, and Chetan Ramaiah. 2022. Use all the labels: A hierarchical multi-label contrastive learning framework. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, 16660–16669.

[153] Zhang, Xiang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *Advances in neural information processing systems* 28:649–657.

[154] Zhang, Xinyi, Jiahao Xu, Charlie Soh, and Lihui Chen. 2022. La-hcn: label-based attention for hierarchical multi-label text classification neural network. *Expert Systems with Applications* 187:115922.

[155] Zhang, Yi, Tao Ge, and Xu Sun. 2020. Parallel data augmentation for formality style transfer. *arXiv preprint arXiv:2005.07522*.

[156] Zhang, Yin, Rong Jin, and Zhi-Hua Zhou. 2010. Understanding bag-of-words model: a statistical framework. *International journal of machine learning and cybernetics* 1:43–52.

[157] Zhang, Yizhe, Xiang Gao, Sungjin Lee, Chris Brockett, Michel Galley, Jianfeng Gao, and Bill Dolan. 2019. Consistent dialogue generation with self-supervised feature learning. *arXiv preprint arXiv:1903.05759*.

[158] Zhou, Giulio, and Jacob Devlin. 2021. Multi-vector attention models for deep re-ranking. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, 5452–5456.

[159] Zhou, Guorui, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. 2018. Deep interest evolution network for click-through rate prediction. `1809.03672`.

[160] Zhou, Jie, Chunping Ma, Dingkun Long, Guangwei Xu, Ning Ding, Haoyu Zhang, Pengjun Xie, and Gongshen Liu. 2020. Hierarchy-aware global model for hierarchical text classification. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, 1106–1117.

[161] Zhou, Y, Z Dou, S Ge, and JR Wen. 2019. Dynamic personalized search based on rnn with attention mechanism. *Chinese Journal of Computer* 42:812–826.

[162] Zhou, Yujia, Zhicheng Dou, and Ji-Rong Wen. 2020. Encoding history with context-aware representation learning for personalized search. In *Proceedings of the 43rd international acm sigir conference on research and development in information retrieval*, 1111–1120.

[163] Zhuang, Shengyao, Houxing Ren, Linjun Shou, Jian Pei, Ming Gong, Guido Zuccon, and Daxin Jiang. 2022. Bridging the gap between indexing and retrieval for differentiable search index with query generation. *arXiv preprint arXiv:2206.10128*.

[164] Zimmermann, Roland S, Yash Sharma, Steffen Schneider, Matthias Bethge, and Wieland Brendel. 2021. Contrastive learning inverts the data generating process. In *International conference on machine learning*, 12979–12990. PMLR.

[165] Zuo, Simiao, Qingyu Yin, Haoming Jiang, Shaohui Xi, Bing Yin, Chao Zhang, and Tuo Zhao. 2022. Context-aware query rewriting for improving users' search experience on e-commerce websites. *arXiv preprint arXiv:2209.07584*.