Less is More: Restricted Representations for Better Interpretability and Generalizability

by

Zhiying Jiang

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2023

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

Supervisor(s):	Jimmy Lin Professor, David Cheriton School of Computer Science, University of Waterloo
Internal Member:	Yaoliang Yu Assistant Professor, David Cheriton School of Computer Science, University of Waterloo
Internal Member:	Wenhu Chen Assistant Professor, David Cheriton School of Computer Science, University of Waterloo
Internal-External Member:	Sirisha Rambhatla Assistant Professor, Department of Management Sciences, Faculty of Engineering, University of Waterloo
External Examiner:	Dale Schuurmans Professor, Department of Computing Science, University of Alberta

Author's Declaration

The thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

The thesis includes contributions from six published papers and public manuscripts. Zhiying Jiang is the first author of first four articles, who is responsible for the idea, implementation, conducting experiments, and paper writing. These four papers are introduced in Chapter 3, 4, 5, 6. Zhiying Jiang is the co-first author of the paper presented in Chapter 7, who is responsible for the implementation, experiments and writing on the part of ranking in low-data regimes. Zhiying Jiang is also the co-first author of the paper presented in Chapter 8, who comes up with the idea and implements the GPT-based classification using information distance independently.

List of publications and co-authors:

- Zhiying Jiang, Yiqin Dai, Ji Xin, Ming Li, Jimmy Lin. Few-Shot Non-Parametric Learning with Deep Latent Variable Model. *In Proceedings of the 36th Conference on Neural Information Processing Systems (NeurIPS) Spotlight, 2022*.
- Zhiying Jiang, Raphael Tang, Ji Xin, Jimmy Lin. How Does BERT Rerank Passages? An Attribution Analysis with Information Bottlenecks. *In Proceedings of the Fourth BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP 2021*.
- Zhiying Jiang, Raphael Tang, Ji Xin, Jimmy Lin. Inserting Information Bottleneck for Attribution in Transformers. *In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP) Findings, 2020*.
- Zhiying Jiang, Matthew Y.R. Yang, Mikhail Tsirlin, Raphael Tang, Yiqin Dai, Jimmy Lin. "Low-Resource" Text Classification: A Parameter-Free Classification Method with Compressors. *In Proceedings of The 61st Annual Meeting of the Association for Computational Linguistics (ACL) Findings, 2023*.
- Rodrigo Nogueira*, Zhiying Jiang*, Ronak Pradeep, Jimmy Lin. Document Ranking with a Pretrained Sequence-to-Sequence Model. *In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP) Findings, 2020*.
- Cynthia Huang*, Yuqing Xie*, Zhiying Jiang*, Jimmy Lin, Ming Li. Approximating Human-Like Few-shot Learning with GPT-based Compression. *In Submission, 2023*.

Abstract

Deep neural networks are prevalent in supervised learning for large amounts of tasks such as image classification [Simonyan and Zisserman, 2014], machine translation [Sutskever et al., 2014] and even scientific discovery [Gilmer et al., 2017]. Their success is often at the sacrifice of interpretability and generalizability. The increasing complexity of models and involvement of the pre-training process make the inexplicability more imminent. The outstanding performance when labeled data are abundant while prone to overfit when labeled data are limited demonstrates the difficulty of deep neural networks' generalizability to different datasets.

This thesis aims to improve interpretability and generalizability by restricting representations. We choose to approach interpretability by focusing on attribution analysis to understand which features contribute to prediction on BERT [Devlin et al., 2019], and to approach generalizability by focusing on effective methods in a low-data regime.

We consider two strategies of restricting representations: (1) adding bottleneck, and (2) introducing compression. Given input x, suppose we want to learn y with the latent representation z (i.e. $x \to z \to y$), adding bottleneck means adding function R such that $L(R(z)) < L(z)$ and introducing compression means adding function R so that $L(R(y)) < L(y)$ where L refers to the number of bits. In other words, the restriction is added either in the middle of the pipeline or at the end of it.

We first introduce how adding information bottleneck can help attribution analysis and apply it to investigate BERT's behavior on text classification in Chapter 3. We then extend this attribution method to analyze passage reranking in Chapter 4, where we conduct a detailed analysis to understand cross-layer and cross-passage behavior.

Adding bottleneck can not only provide insight to understand deep neural networks but can also be used to increase generalizability. In Chapter 5, we demonstrate the equivalence between adding bottleneck and doing neural compression. We then leverage this finding with a framework called Non-Parametric learning by Compression with Latent Variables (NPC-LV), and show how optimizing neural compressors can be used in the non-parametric image classification with few labeled data. To further investigate how compression alone helps non-parametric learning without latent variables (NPC), we carry out experiments with a universal compressor *gzip* on text

classification in Chapter 6. In Chapter 7, we elucidate methods of adopting the perspective of doing compression but without the actual process of compression using T5 [Raffel et al., 2020b]. Using experimental results in passage reranking, we show that our method is highly effective in a low-data regime when only one thousand query-passage pairs are available. In addition to the weakly supervised scenario, we also extend our method to large language models like GPT under almost no supervision — in one-shot and zero-shot settings. The experiments show that without extra parameters or in-context learning, GPT can be used for semantic similarity, text classification, and text ranking and outperform strong baselines, which is presented in Chapter 8.

The thesis proposes to tackle two big challenges in machine learning — "interpretability" and "generalizability" through restricting representation. We provide both theoretical derivation and empirical results to show the effectiveness of using information-theoretic approaches. We not only design new algorithms but also provide numerous insights on why and how "compression" is so important in understanding deep neural networks and improving generalizability.

Acknowledgements

I would like to first thank my advisor Jimmy Lin for all the support along the journey. In addition to providing me with the freedom to explore the field in my preferred direction, Jimmy's guidance and immense knowledge guided me through the years of my study. His tireless commitment to making sure that I was on the right track has been instrumental and shows genuine care for the long-term growth and success of students, which I really appreciate.

I want to express my appreciation to my other committee members: Yaoliang, Wenhu, Sirisha and Dale for committing your time to read my lengthy thesis and to provide invaluable advice.

I would also like to thank my undergraduate advisor Heng Ji, who brought me into research in the first place. By the time I was trying a little bit of everything and Heng showed me the excitement and fun of doing research, which leads me to decide on my Ph.D.

I want to thank my collaborators and friends who provided insightful advice and mental support. Thank you Prof. Ming Li for all the time spent discussing Kolmogorov complexity and the theory we are excited about. Thanks Rodrigo for all the mentorship and papers collaborating together. Thanks Ralph and Jayden for being my collaborators on most of my papers and for the off-topic chitchat. Thanks Yuqing and Cynthia for staying up together for a paper. I also want to extend my appreciation to Wei, Peng, Xueguang, Crystina, Minghan, Leo and all other DSG lab mates. My gratefulness is also extended to my undergrad lab mates Feifei, Boliang, Lifu, Spencer, Xiaoman, and TT. I also want to thank my friend Orianna for the time, fun and mental support and for making me more extroverted (a little bit). I'd like to thank Ying for inputting a placid atmosphere about music and games throughout my undergrad to my Ph.D. (some emoji here). I also want to thank Naughty Dog, Nintendo, and Riot for the fun they bring to me and thank Tim Horton, Papa John's, and T&T for making me not starve.

My appreciation for my parents is beyond words. I cannot enumerate all the things they have done for me. They have provided me with a happy childhood. Their trust in me and supportive attitude let me explore anything I like. I wanted to go abroad for my undergrad study because I would love to learn architecture but the freedom in choosing my major is limited in my hometown. They don't hesitate to satisfy my self-willed and sudden request. The list of my willful decisions goes on and on but my parents always support me. Even when I was making decisions that they didn't understand, they trust, support, and love me as usual.

Finally, I want to express my deepest gratitude to my wife Yiqin. Many people acknowledge their wives' "quiet" company. That sounds strange to me as you are not quiet at all. You have been my counselor, collaborator, and co-founder all along my Ph.D. During the flash 3.5 years, we wrote papers together, did rebuttal together, built two web apps together, and developed weird theories about wacky phenomena in human society together. I am so fortunate that I haven't experienced the loneliness in a typical Ph.D journey as you know almost every detail of my research and we share similar aesthetics of research. Starting from day one, we are each other's best friend, soul mate, and therapist. I cannot express how lucky I feel to meet you, to have you as my lifelong partner, and to continue our "irrational exuberance" journey with you. I'm sorry that I kept you waiting so long but *our* adventure formally begins.

Dedication

To Yiqin — my lifelong partner, cofounder, and collaborator, and to my parents.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The start of the deep learning boom can be traced back to 2012, when Krizhevsky et al. [2012] utilize GPUs to accelerate deep convolutional neural networks [LeCun et al., 1995] and won ImageNet [Deng et al., 2009] contest by a large margin. Subsequently, researchers have come to recognize the potency of augmenting the number of layers and parameters [He et al., 2016; Simonyan and Zisserman, 2014]. Combining hundreds of millions of training parameters with tens of millions of labeled data, deep neural networks achieve state-of-the-art results on numerous tasks [Gilmer et al., 2017; Simonyan and Zisserman, 2014; Sutskever et al., 2014]. The landscape of deep learning is further shifted by pretrained models. By the time BERT [Devlin et al., 2019] came out, it improved the General Language Understanding Evaluation (GLUE) [Wang et al., 2018b], a widely used NLP benchmark covering various NLP tasks like natural language inference and semantic similarity, from 70.0 to 80.5 [1]. Similarly, ViT [Dosovitskiy et al., 2020] also achieves the state of art on image classification tasks on seven datasets. Adopting systems embedded with deep neural networks in the real world is promising, isn't it?

[1] https://gluebenchmark.com/leaderboard

1.1 Challenges

In this thesis, we discuss two major challenges when attempting to adopt deep neural networks to the real world: interpretability and generalizability.

Interpretability: Both non-pretrained deep neural networks and pretrained ones are black box models [Shwartz-Ziv and Tishby, 2017] — feeding input to models yields output, while the underlying mechanisms that facilitate this operation remain undisclosed. This highlights a concern of untrustworthiness when black-box models are applied to real-world scenarios [Ribeiro et al., 2016]. Especially in more life-critical domains like medical science, it is hard for people to trust the decision made by a system that they don't understand. This problem becomes even more fatal when the slight manipulation of images [Szegedy et al., 2013] that humans can barely notice or even the change of one pixel [Su et al., 2019] can fool the neural networks to make different predictions. Until we can find the universal theory of deep neural networks, we may assume different deep neural networks have different mechanisms. Without understanding the mechanism of every single deep neural network, how can we provide trustworthy prediction in a universal way? We focus on attribution analysis, which answers the question of which features are most important to prediction.

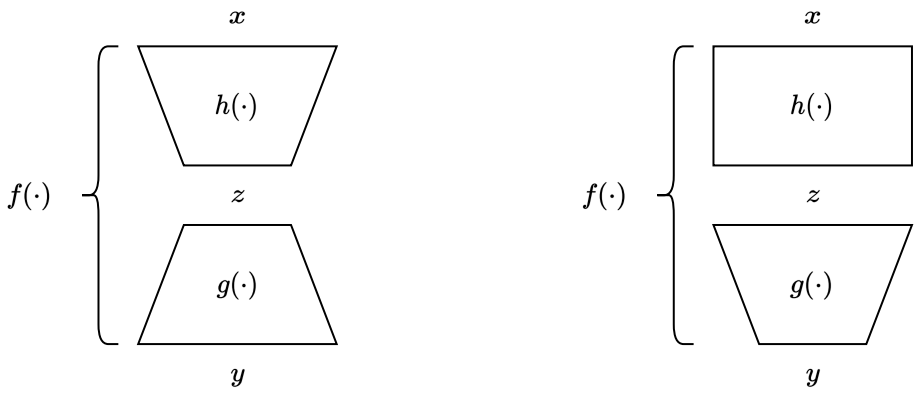
Generalizability: The success that deep neural networks bring in always requires huge amounts of labeled data [Marcus, 2018]. Theoretically, the sample complexity theory derived from VC dimension [Vapnik, 1999] has set a loose bound on the number of labeled data that are required to achieve a satisfactory result and it increases with the model complexity. Although the theory is developed for binary classification models, previous works [Zhang et al., 2021b] have also found it to hold for non-pretrained deep neural networks. However, labeled datasets are luxurious as they take a long time to construct. For example, it takes several years to construct ImageNet [Deng et al., 2009] and it requires a similar time span for constructing Penn Tree Bank [Marcus et al., 1993]. Fortunately, there are abundant unlabeled data — no matter it is the resources freely accessible online or the datasets that are waiting to be labeled by the system. Pretrained models like BERT [Devlin et al., 2019] or GPT [Radford et al., 2018b] utilize the power of self-supervised learning with unlabeled datasets and enable the downstream tasks to be fine-tuned using only a few thousand labeled data to achieve the satisfactory result. Promising as this direction is, it still faces generalizability problems when fine-tuning out-of-distributed (OOD) datasets. For example, Yu

et al. [2021] have found that when the data distribution in the fine-tuning datasets is substantially different from any pretrained dataset, the inductive bias hinders fine-tuning and makes the model less pliable. In order to make as minimum assumption for the dataset as possible, we focus on *Non-Supported Learning* (NSL) where we are only given limited labeled data without the help of labeled support sets in meta-learning-based few-shot learning.

1.2 Restricted Representations

The principle of parsimony has historically been well-known in various domains across ages. For example, Williams of Ockham in the thirteenth century phrased that "Entities shall not be multiplied beyond necessity" in the domain of philosophy and theology; a similar gist is expressed by Albert Einstein, paraphrased later by Roger Sessions as "Everything should be made as simple as possible, but no simpler" [Robinson, 2018] in the twentieth century in the context of physics.

In machine learning, the principle of parsimony also emerges in different areas: (1) Model Selection. "Ockham's razor" is often used as the heuristic to choose a simpler model given various models have the same ability to fit the data, which is formalized by "Minimum Description Length (MDL)" [Rissanen, 1989]. (2) Data Modeling. Generative models like Auto-Encoders [Rumelhart et al., 1985] use latent variables with low dimensions to capture the features that are essential to reconstruct the input but not more. (3) Representation Learning. PCA [Hotelling, 1933] and latent representation of VAE [Rolinek et al., 2019] share the similarity of reducing the dimensionality of the representation while keeping the expressiveness. The representations then can be used for downstream tasks like classification. (4) Intelligence emergence hypothesis. Ma et al. [2022] propose a theoretical framework for explaining the emergence of intelligence and the principle of parsimony is one of the two principles. It states that identifying low-dimensional structures of the external world and organizing them in a structured way is the main objective of intelligent systems' learning. From the above examples, we hypothesize the "restricted representation" is a crucial part of the principle of parsimony. By "restricted", we illustrate in Figure 1.1 below to demonstrate its intuitive meaning. Given any input x, we want to learn $f(x)$. Now suppose we have an intermediate state $z = h(x)$ and $g(z) = f(x)$ (i.e., $f = g \circ h$), the "restriction" can be viewed as another function R that either makes $L(R(f(x))) < L(f(x))$ or $L(R(h(x))) < L(h(x))$, where



(a) Restrict Information by Inserting Bottleneck (b) Restrict Information by Compression

Figure 1.1: Two Ways to Restrict Information. Figure 1.1a is used in Chapters 3 to 5; Figure 1.1b is used in Chapters 6 to 8.

L is the bit length. Figure 1.1a shows restricting $h(x)$, the intermediate part, which can be viewed as inserting a bottleneck while Figure 1.1b shows restricting $f(x)$, which can be viewed as data compression.

In this thesis, we investigate how "restricted representation" help leverage interpretability and generalizability. We mainly focus on two ways of restricting representations: (1) adding bottleneck and (2) doing compression. For interpretability, we analyze the attribution in document classification and passage re-ranking. For generalizability, we specifically investigate effective methods in a labeled-data-scarce environment.

1.2.1 For Interpretability

Deep neural networks are exceptional at various tasks but those black-box models always remain inexplicable [Kovaleva et al., 2019]. The lack of interpretability is even severe for pretrained models in NLP (e.g., BERT). One important question to ask in order to understand the prediction result is: Which hidden features contribute most to the prediction? Attribution analysis that uses attribution scores is used to resolve this problem. The definition of attribution scores is as follows:

Given an m-dimensional instance $\mathbf{x} \in \mathbb{R}^m$, the numerical score $s(x_i, \mathbf{y})$ denotes the relevance between the i-th feature of $\mathbf{x}$ and the prediction $\mathbf{y}$.

Gradient-based methods [Fong and Vedaldi, 2017; Selvaraju et al., 2017; Smilkov et al., 2017] and probing methods [Clark et al., 2019; Liu et al., 2019] were widely adopted but gradient-based methods fail to capture all the information associated with the correct prediction [Li et al., 2016]; probing methods fail to capture the inner mechanism like how information flows through the network [Guan et al., 2019].

We approach the attribution analysis from an information-theoretic perspective where we insert an information bottleneck to restrict the information flowing through the network so that only the most important information is kept. How much information is still kept about the input features indicates the attribution score. The intuition is then applied to attribution analysis for text classification fine-tuned with BERT, and both qualitative and quantitative analyses are carried out to show that our method is superior to other attribution analysis methods. This work is explained in detail in Chapter 3.

We push the attribution analysis further for the passage-reranking task that combines BM25 [Robertson and Jones, 1976] and BERT [Nogueira and Cho, 2019]. Based on the attribution map obtained from inserting bottleneck, we then can carry out a more detailed analysis to identify both the similarity and differences between BM25 and BERT with regard to how they rank passages; to analyze the role that special tokens play in ranking; to test the robustness of the top-ranked passage and to testify whether BERT has positional bias. We cover this part in detail in Chapter 4.

1.2.2 For Generalizability

Generalizability refers to models' ability to adapt to unseen data from the same data distribution as the training set. Deep neural networks with huge amounts of training parameters are expressive enough to memorize random labels [Zhang et al., 2017] achieving training error $e_{\text{train}} = 0$ while test error e_{test} is huge. With scarce labeled data, those deep learning models are even easier to overfit [Wang et al., 2020b].

In this thesis, we specifically evaluate generalizability under the scenario where both the number of labeled data and training parameters are reduced. Specifically, we investigate "generalizat-

bility" in a *Non-Supported Learning (NSL)* scenario:
Given any target dataset $\mathbf{D} = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n)$ belonging to c classes. For each class, we have k labeled samples. The remaining $n - ck$ unlabeled samples need to be classified into c classes without the need for labeled support sets.

Adding a bottleneck in the supervised scenario can help us better understand deep neural networks' prediction while adding a bottleneck in an unsupervised scenario can help to learn in a non-parametric way.

We derive negative evidence lower bound (nELBO) from the perspective of adding bottleneck in an unsupervised scenario and then combining a distance metric based on Kolmogorov complexity [Kolmogorov, 1963] to show how nELBO can be used to do classification in a parameter-free way. Townsend et al. [2019b] derive nELBO from the "bits-back" argument [Hinton and Van Camp, 1993], which demonstrates optimizing latent variable models in an unsupervised manner equals optimizing a neural compressor. This connection brings up the question of whether using neural compression, in general, can enhance the non-parametric learning with traditional compressors [Cebrián et al., 2005; Chen et al., 1999, 2004; Keogh et al., 2004b]. We show that using the actual compression with compressor-based distance metric achieves outstanding image classification in the few-shot scenario. We introduce this work in Chapter 5.

We push this line further by restricting representation by compression only with absolutely *no* training, even without unsupervised training. We use a simple compressor *gzip* together with a compressor-based distance metric and outperform pretrained models like BERT on five out-of-distributed datasets and achieve a competitive result with the family of non-pretrained deep neural networks on six in-distributed datasets for text classification. This line of work is introduced in Chapter 6.

We also show that with the perspective of compression, we can utilize language models more effectively without doing the actual compression. We take advantage of the probabilistic distribution captured by pretrained generative models like T5 [Raffel et al., 2020b] without the cost of applying coding schemes or discretization. Our experiments in passage reranking illustrate the effectiveness of this method when labeled data are scarce. We introduce this work in Chapter 7.

We finally extend the above method to a more general scenario where GPT-based methods are used under (almost) no supervision. We demonstrate how can we use this method in various

NLP tasks including semantic similarity, zero-shot passage reranking and classification without parameters or doing actual compression. This work is introduced in Chapter 8.

1.3 Contribution

The main contributions of this thesis are summarized as follows:

For interpretability,

- We improved the interpretability of pre-trained models by generating a more trustworthy attribution map by inserting information bottleneck. Quantitatively, it improves other attribution maps like Integrated Gradient (IG) by up to 150%. This model-agnostic attribution method can be applied to almost all neural networks. We make code publicly available at https://github.com/bazingagin/IBA.
- We analyzed the passage reranking at a token-wise level which demonstrated the granularity that an attribution map can provide. We facilitate the understanding of passage reranking with BERT by showing BERT's strength in capturing contextual information, its robustness to token removal, and its positional bias towards the start of the passages.

For generalizability,

- We unified previous compressor-based methods into a learning framework with replaceable modules called *Non-Parametric learning by Compression* (NPC). Based on NPC, we proposed a new framework called *Non-Parametric learning by Compression with Latent Variables* (NPC-LV) which leverages the power of neural networks to get a classifier without any supervised training. Code is publicly available at https://github.com/bazingagin/npc_lv.
- We proved the effectiveness of NPC-LV by showing its few-shot image classification performance. Specifically, our method can surpass supervised learning by up to 59% and can even outperform semi-supervised learning without any label involved in training.

- We demonstrated how we can use negative evidence lower bound (nELBO) as an estimated compressed length for classification under NPC-LV by connecting the equivalence between training a generative model and obtaining a better neural compressor. We further illustrate the correlation between compression rate and classification accuracy which may indicate this framework can be further improved with the development of generative models.

- We showed that without any neural networks, NPC with a traditional compressor like *gzip* has the potential to achieve the state of the art performance on four low-resource language text classification tasks, outperforming pre-trained models like BERT, mBERT, and even KinyaBERT. We also showed that our method can achieve competitive results to deep neural networks on five out of seven in-distributed (English) datasets with *zero* training parameters. The advantage of our method is more obvious under few-shot settings, surpassing BERT by up to 193.7%. We make code publicly available at https://github.com/bazingagin/npc_gzip.

- We further illustrated that equipped with the perspectives of compression, we can better utilize language models, especially a pre-trained model like T5 in passages reranking. We showed that our novel method outperforms all previous state of the art models in zero-shot settings and is extremely effective in low-data regimes, much more effective than encoder-only models like BERT.

- We show how GPT models can be used with almost no supervision, and no extra parameters for various NLP tasks like semantic similarity, zero-shot passage reranking, and zero-shot as well as one-shot text classification. It outperforms fine-tuning and in-context learning in almost all scenarios when labeled data are extremely limited, which may indicate that compression-based methods leverage the prior knowledge better than in-context learning and fine-tuning in cases where there is an exceedingly restricted pool of annotated data.

Chapter 2

Background

In this chapter, we briefly introduce the background knowledge needed to understand the thesis. Specifically, we will cover data compression, information bottleneck, Kolmogorov complexity, and information distance derived from it.

2.1 Information Theory

The field of information theory quantifies information in an algebraic way, drawing the connection between information and uncertainty [Shannon, 1948].

Entropy: Suppose we have a random variable X, whose finite alphabet is V. Given the probability distribution $p(x) = \text{Prob}(X = x), x \in V$, the entropy of X is defined as:

$$H(X) \triangleq -\mathbb{E} \log p(X), \tag{2.1}$$

$H(X)$ measures the average information/uncertainty of a random variable X.

Joint Entropy: Extending entropy to two random variables renders the joint entropy:

$$H(X, Y) \triangleq -\mathbb{E} \log p(X, Y) \tag{2.2}$$

$H(X, Y)$ represents the average information we get from observing X and Y.

Conditional Entropy: Similarly, if we replace the joint distribution with a conditional distribution $p(X|Y)$, we will have the conditional entropy:

$$H(X|Y) \triangleq -\mathbb{E} \log p(X|Y) \tag{2.3}$$

$H(X|Y)$ describes how much information still left in X given the information of Y.

Mutual Information: Given two random variables X, Y, one quantity that's especially interesting is the mutual information $\mathrm{I}(X;Y)$.

$$\mathrm{I}(X;Y) = H(X) - H(X|Y) = H(Y) - H(Y|X). \tag{2.4}$$

It describes how much information gained of X due to knowing Y (i.e., how much reduction of the uncertainty does knowing Y create).

The most fundamental building blocks of information theory is given above, and based on which we can now introduce "information bottleneck" and "data compression".

2.1.1 Information Bottleneck

Information bottleneck was first proposed by Slonim and Tishby [1999] as a clustering algorithm. It is Tishby and Zaslavsky [2015]; Tishby et al. [2000] and Shwartz-Ziv and Tishby [2017] that connects the information bottleneck method to explaining deep neural networks. The theoretical framework of information bottleneck hypothesizes that deep neural networks all go through two phases: the fitting phase and the compression phase. Essentially, Shwartz-Ziv and Tishby [2017] argue that as the training proceeds, deep neural networks first encode all the information for prediction and then discard that irrelevant information and only keep the most relevant representation.

The essential idea of information bottleneck is to *only* keep the information that's relevant to the prediction. Concretely, the goal of the information bottleneck is to find a representation Z that minimizes the mutual information between Z and input X and at the same time maximizes that mutual information between Z and output Y:

$$\mathrm{I}(Y;Z) - \beta \cdot \mathrm{I}(X;Z), \tag{2.5}$$

where β controls the trade-off between the restriction and reconstruction. In other words, the larger the β is, the narrower the bottleneck is.

While it's still unclear whether this principle holds for all deep neural networks[1], we do not rely on the correctness of this principle for our interpretability method. Instead of explaining the mechanisms of how deep neural networks learn, we just use information bottleneck as inspiration for the loss function of representation learning and utilize the representation for the attribution map.

2.1.2 Data Compression

In a compression scenario, suppose we have a sender *Alice* and a receiver *Bob*. *Alice* wants to send a message that contains a sequence of symbols $\mathbf{x} = (x_1, x_2, ..., x_n)$ to *Bob*. The ultimate goal of the lossless compressor is to compress $\mathbf{x}$ into the minimum amount of bits $\mathbf{x}'$ that can later be decompressed back to $\mathbf{x}$ by *Bob*. To achieve the shortest compressed length, shorter codes are assigned to symbols with higher probability. According to Shannon's Source Coding Theorem [Shannon, 1948], this length of bits is no shorter than the entropy of the sequence, whose definition is $H(\mathbf{x}) \triangleq \mathbb{E}[-\log p_{\text{data}}(\mathbf{x})]$, where $p_{\text{data}}(\mathbf{x})$ represents the probability distribution of each symbol in the sequence. However, the "true" probabilistic distribution $p_{\text{data}}(\mathbf{x})$ is unknown to us. We can only access samples and approximate it with $p_\theta(\mathbf{x})$. That is:

$$\mathbb{E}[-\log p_\theta(\mathbf{x})] \geq H(\mathbf{x}) \triangleq \mathbb{E}[-\log p_{\text{data}}(\mathbf{x})]. \tag{2.6}$$

Given an entropy coding scheme, the better $p_\theta(\mathbf{x})$ approximates $p_{\text{data}}(\mathbf{x})$, the closer we can get to the minimum code length. Besides, in order to convert the message into a sequence of bits, a coding scheme is needed.

One of the most famous coding schemes is Huffman Coding [Huffman, 1952]. The main idea is to build a tree according to the probability of each symbol and encode symbols one by another. As each symbol is assigned a variable-length code (i.e., a binary string), the number of bits required to code a message is always an integer. It's proved that Huffman coding is the optimal *symbol* coding with an overhead of one bit per symbol [Cover, 1999]. However, the

[1] For controversial perspective and discussion, see: https://openreview.net/forum?id=ry_WPG-A-

accumulation of one bit per symbol can still be large. In order to push the coding length closer to the entropy, *stream* coding schemes like Arithmetic Coding (AC) [Witten et al., 1987b] and Asymmetric Numeral Systems (ANS) [Duda, 2009] are proposed. Stream coding schemes, as the name suggests, treat the whole message as a sequence and encode them together. It is able to shrink the code length of the whole sequence with additional two bits of overhead. Both AC and ANS share a similar gist of converting sequences into numeral systems. We will introduce ANS as it is a relatively new and the state of the art coding scheme.

ANS: The essence of ANS is to encode one or more data points into a single natural number, called state $s \in \mathbb{N}$. Depending on different vocabularies and manipulations, there are different variations of ANS (details in Duda et al. [2015]). We introduce one of them — rANS (range ANS), which is the variant we use in this thesis. The notation we use here is unconventional in order to be consistent with the main part of the thesis.

Let's notate our state at timestamp t as $s_t \in \mathbb{N}$, and notate our symbol/message at t as x_t, $x_t \in V$, where $V = \{0, 1\}$ is the vocabulary set. We have two simple methods to encode a binary sequence into a natural number bit by bit — $s_t = 2s_{t-1} + x_t$ or $s_t = s_{t-1} + 2^m x_t$. The former means appending information to the least significant position while the latter is adding information to the most significant position. It's obvious that encoding a new symbol into the most significant position requires remembering m while encoding in the least significant position only needs the previous state s_{t-1} and new information x_t. It's also easy to decode: depending on whether the current state s_t is even or odd, we not only know if the last encoded symbol x_t is 0 or 1, but we can also decode the state following $s_{t-1} = \frac{s_t}{2}$ or $s_{t-1} = \frac{s_t - 1}{2}$.

The above example illustrates encoding and decoding methods when there are two elements in the vocabulary with uniform distribution $p(0) = p(1) = \frac{1}{2}$. In this case, it's optimal to scale up s_t to two for both 0 or 1 as we essentially only spend 1 bit per encoded symbol. However, when the probability is not uniformly distributed, the entropy is smaller, and scaling up by 2 for both symbols will not be optimal anymore. rANS generalizes the process to any discrete probability distribution and any size of the vocabulary.

Intuitively, scaling up by a smaller factor for a more probable symbol and scaling up by a larger factor for a less probable symbol will provide us with a more efficient representation. Concretely, we have a sequence of messages $\mathbf{x} = (x_1, x_2, x_3, ..., x_n)$, and a vocabulary $V =$

$\{v_1, v_2, v_3, ..., v_k\}$, with size k, $x_i \in V$. We also have probability mass distribution of V: $P = \{p_{v_1}, p_{v_2}, p_{v_3}, ..., p_{v_k}\}$. Correspondingly, let's define frequency counts $F = \{f_{v_1}, f_{v_2}, f_{v_3}, ..., f_{v_k}\}$, $f_{v_i} = p_{v_i} \times M$ where $M = \sum_{i=1}^{k} f_{v_i}$. M can be viewed as a multiplier, demonstrating the precision of ANS, which is a predefined variable in the implementation. We can also get cumulative frequency counts from F as follows $B = \{b_{v_1}, b_{v_2}, b_{v_3}, ..., b_{v_k}\}$ where $b_{v_i} = \sum_{j=1}^{i-1} f_{v_j}$. Now we get everything we need to define the encoding function G:

$$
\begin{aligned}
&s_t = G(s_{t-1}, x_t), \\
&G(s_{t-1}, x_t) = \left\lfloor \frac{s_{t-1}}{f_{x_t}} \right\rfloor \times M + b_{x_t} + s_{t-1} \bmod f_{x_t}.
\end{aligned}
\tag{2.7}
$$

The procedure can be interpreted as follows: We have various M-sized blocks partitioning natural number $\mathbb{N}$. Encoding can be viewed as finding the exact location of the natural number that represents the state, by first finding the corresponding block ($\left\lfloor \frac{s_{t-1}}{f_{x_t}} \right\rfloor \times M$) followed by finding the sub-range representing that symbol within M (b_{x_t}) and finding the exact location within that sub-range ($s_{t-1} \bmod f_{x_t}$). The decoding function $H(s_t)$ is the reverse of the encoding:

$$
\begin{aligned}
&s_{t-1}, x_t = H(s_t), \\
&x_t = \text{argmax}\ \{b_{x_t} < (s_t \bmod M)\}, \\
&s_{t-1} = f_{x_t} \left\lfloor \frac{s_t}{M} \right\rfloor + s_t \bmod M - b_{x_t}.
\end{aligned}
\tag{2.8}
$$

We first find the precise location within the sub-range using the inverse function of cumulative counts (argmax $\{b_{x_t} < (s_t \bmod M)\}$). With x_t we can reverse steps in Equation (2.7) to get the previous state. As we can see, ANS decodes in the reverse order of encoding (i.e., last in first out), which makes it compatible with the bits-back argument. From the encoding function, we know that:

$$
\frac{s_t}{s_{t-1}} \approx \frac{M}{f_{x_t}} = \frac{1}{p_{x_t}}. \tag{2.9}
$$

Encoding a sequence of symbols $\mathbf{x}$ results in:

$$
s_n \approx \frac{s_0}{p_{x_1} p_{x_2} ... p_{x_n}}. \tag{2.10}
$$

Thus, the total coding length is:

$$\log s_n \approx \log s_0 + \sum_{i=1}^{n} \log \frac{1}{p_{x_i}}, \tag{2.11}$$

where s_0 refers to the initial state. Dividing by n we will get the average coding length that approximates the entropy of the data.

Discretization: ANS is defined for symbols in a finite alphabet; bits-back coding works for discrete latent variables. However, continuous latent variables have proven to be powerful in many latent variable models. In order to use those latent variable models for lossless compression, discretizing continuous variables into discrete ones is a necessary step. Townsend et al. [2019b] shows, based on MacKay [2003] using bits-back coding, continuous latent variables can be discretized to arbitrary precision without affecting the compression rate. Suppose a probability density function p is approximated using a number of "buckets" of equal width $\sigma\mathbf{z}$. For any given bucket j, we can know its probability mass $p(\mathbf{z}^{(j)})\sigma\mathbf{z}$ where $\mathbf{z}^{(j)}$ is some point in the bucket j. Let's notate the discrete distribution as P and Q for both prior and posterior distribution. Then for any given bucket j, $P(j) \approx p(\mathbf{z}^{(j)})\sigma\mathbf{z}$. The expected message length with a discretized latent variable is:

$$-\mathbb{E}_{Q(j|\mathbf{x})} \log \frac{p(\mathbf{x}|\mathbf{z}^{(j)})p(\mathbf{z}^{(j)})\sigma\mathbf{z}}{q(\mathbf{z}^{(j)}|\mathbf{x})\sigma\mathbf{z}}. \tag{2.12}$$

The width of buckets $\sigma\mathbf{z}$ is canceled. Therefore, as long as the bins for inference models match the generative models, continuous latent variables can be discretized up to arbitrary precision.

In this thesis, we only consider basic discretization techniques like dividing continuous distribution into bins with *equal width* or *equal mass*. We discretize the prior (top layer) with equal mass and all subsequent latent layers with equal width. As Equation (2.12) shows, ideally we want the discretization to align between inference models and generative models. However, discretization of $\mathbf{z}_i \sim p_\theta(\mathbf{z}_i|\mathbf{z}_{i+1})$ relying on $\mathbf{z}_i \sim q_\phi(\mathbf{z}_i|\mathbf{z}_{i-1})$ is not possible without sampling. In the compression stage, when decoding $\mathbf{z}_i$, $\mathbf{z}_{i+1}$ is not available and so is $p_\theta(\mathbf{z}_i|\mathbf{z}_{i+1})$. In the decompression stage, similarly, $q_\phi(\mathbf{z}_i|\mathbf{z}_{i-1})$ is not available for $p_\theta(\mathbf{z}_i|\mathbf{z}_{i+1})$ to match with. Therefore, we need to sample from the training dataset beforehand to get unbiased estimates of the statistics of the marginal distribution [Kingma et al., 2019a]. This process only needs to be done once and can be saved for future use.

2.2 Algorithmic Information Theory

Given a simple binary string, $s = \overbrace{00...0}^{\text{n}}$, one way to view s is to see it as a list of symbols drawn from vocabulary $\Sigma = \{0, 1\}$. Suppose the probability of drawing 0 or 1 is uniformly distributed, according to information theory, we know that n bits are required to encode s, as s is just $\frac{1}{2^n}$ of all possible combinations given Σ. If we only focus on this single s, we can see a specific pattern: s consists of n "0"s. This pattern can be expressed in a simple program U: "for i=1 to n, print(0)". Now suppose *Alice* wants to transmit s to *Bob*, instead of using n bits, *Alice* can use only $\log n$ bits to encode n and an extra $O(1)$ bit as the reference of the program, assuming *Bob* also has the program. In this way, s can be compressed significantly.

Focusing on "single" objects like a single string without the notion of a probability distribution is what differentiates Information theory from Algorithmic Information Theory. We introduce two fundamental concepts in algorithmic information theory: (1) Kolmogorov complexity and (2) information distance.

2.2.1 Kolmogorov Complexity

Kolmogorov complexity $K(x)$ [Kolmogorov, 1963] is used to describe the length of the shortest binary program that can produce x on a universal computer. It has been proved that the expected length of the shortest binary program is approximately equal to entropy. Besides, any programming language can be used to describe x, and according to invariance theorem [Li et al., 2008], any chosen language is as efficient as the optimal language with constant overhead. Although the invariance theorem demonstrates that $K(x)$ is language independent, $K(x)$ is incomputable as it takes infinitely long to find the *shortest* computer program. Nevertheless, we can still approximate $K(x)$ using compressors, as intuitively, $K(x)$ can be viewed as the length of x after being maximally compressed.

2.2.2 Information Distance

With the basic idea of Kolmogorov complexity, we can define an information distance:

$$E(x,y) = \max\{K(x|y), K(y|x)\} = K(xy) - \min\{K(x), K(y)\}, \tag{2.13}$$

where $K(x|y)$ means the length of the binary program that on input y outputs x, $K(xy)$ denotes the length of the shortest program computing x and y without telling which one is which (i.e., no separator encoded between x and y).

The idea behind this measurement, on a high level, is that the similarity between two objects indicates the existence of a simple program that can convert one to another. The simpler the converting program is, the more similar the objects are. For example, the negative of an image is very similar to the original one as the transformation can be simply described as "inverting the color of the image".

Theorem 1. *The function $E(x,y)$ is an admissible distance and a metric. It is minimal in the sense that for every admissible distance D, we have $E(x,y) \leq D(x,y) + O(1)$.*

Intuitively, *admissible distance* refers to distances that are meaningful (e.g., excluding metrics like $D(x,y) = 0.3$ for any $x \neq y$) and computable (formal definition is in Section 5.5). Combining those definitions, we can see Theorem 1 means $E(x,y)$ is *universal* in a way that is optimal and can discover all effective similarities between two objects.

In order to compare the similarity, the relative distance is preferred. Li et al. [2001a] propose a normalized version of $E(x,y)$ called *Normalized Information Distance* (NID).

Definition 1 (NID). *NID is a function: $\Omega \times \Omega \to [0,1]$, where Ω is a non-empty set, defined as:*

$$\mathit{NID}(x,y) = \frac{\max\{K(x|y), K(y|x)\}}{\max\{K(x), K(y)\}}. \tag{2.14}$$

Equation (2.14) can be interpreted as follows: Given two sequences x, y, $K(y) \geq K(x)$:

$$\text{NID}(x,y) = \frac{K(y) - I(x:y)}{K(y)} = 1 - \frac{I(x:y)}{K(y)}, \tag{2.15}$$

where $I(x:y) = K(y) - K(y|x)$ means the *mutual algorithmic information*. $\frac{I(x:y)}{K(y)}$ means the shared information (in bits) per bit of information contained in the most informative sequence, and Equation (2.15) here is a specific case of Equation (2.14). Theoretically, NID is a desirable

distance metric as it satisfies the metric (in)equalities up to additive precision $O(1/K(\cdot))$ where $K(\cdot)$ is the maximum complexities of objects involved in (in)equalities (proof shown in Li et al. [2004]).

Chapter 3

Restrict Representations by Inserting Information Bottlenecks for Attribution in Text Classification

In this chapter, we discuss one way to restrict representations — inserting an information bottleneck. The inserted bottleneck helps us to improve interpretability by learning representations that can generate a veracity attribution map. Specifically, we use our method to interpret BERT on text classification tasks.

Pretrained transformers achieve the state of the art across tasks in natural language processing [Devlin et al., 2019], motivating researchers to investigate their inner mechanisms. One common direction is to understand what features are important for prediction. We apply information bottlenecks to analyze the attribution of each feature for prediction on a black-box model. We use BERT as an example and evaluate our approach both quantitatively and qualitatively. We show the effectiveness of our method in terms of attribution and the ability to provide insight into how information flows through layers by both quantitative and qualitative analysis. The work is based on our paper:

- Zhiying Jiang, Raphael Tang, Ji Xin, Jimmy Lin. Inserting Information Bottleneck for Attribution in Transformers. *In Proceedings of the 2020 Conference on Empirical Methods*

in Natural Language Processing (EMNLP) Findings, 2020.

3.1 Overview

The urge to interpret deep neural networks is increasingly prominent, with the success of these black-box models remaining vastly inexplicable both theoretically and empirically. Within natural language processing (NLP), this desire is particularly true for the pretrained transformer, which has witnessed an influx of literature on interpretability analysis. Such papers include visualizing transformer attention mechanisms [Kovaleva et al., 2019], probing the geometry of transformer representations [Hewitt and Manning, 2019], and explaining the span predictions of question answering models [van Aken et al., 2019]. While useful these methods are, they are often limited to specific types of architectures. For example, visualizing attention can only interpret those models that incorporate attention mechanisms, and numbers of previous works [Alvarez-Melis and Jaakkola, 2018; Jain and Wallace, 2019] have shown that attention cannot faithfully interpret predictions made by neural networks.

To interpret deep neural networks regardless of the architectures, we focus on model-agnostic attribution methods, which only require the input, the output, and the model, regardless of what type of model it is. The essence of the attribution method is to answer, "Which hidden features contribute the most toward a prediction?" To resolve this question, a number of methods [Selvaraju et al., 2017; Smilkov et al., 2017] generate attribution scores for features, which provide a human-understandable "explanation" of how a particular prediction is made at the instance level. Specifically, given an instance, these methods assign a numerical score for each hidden feature denoting its relevance to the prediction.

Previous papers have demonstrated that gradient-based methods fail to capture all the information associated with the correct prediction [Li et al., 2016]. To address this weakness, Schulz et al. [2020] insert information bottlenecks [Tishby et al., 2000] for attribution, attaining both stronger empirical performance and a theoretical upper bound on the information used. Additionally, mutual information is unconstrained by model and task [Guan et al., 2019]. Thus, we adopt information bottlenecks for attribution (IBA) to interpret transformer models at the instance level. We apply IBA to BERT [Devlin et al., 2019] across five datasets in sentiment analysis, tex-

tual entailment, and document classification. We show both qualitatively and quantitatively that the method capably captures information in the model's token-level features, as well as insight into cross-layer behavior.

Our contributions are as follows: First, we are the first to apply information bottlenecks (IB) for attribution to explain transformers. Second, we conduct quantitative analysis to investigate the quality of our method compared to other interpretability techniques. Finally, we examine the consistency of our method across layers in a case study. Across four datasets, our technique outperforms integrated gradients (IG) and local interpretable model-agnostic explanations (LIME), two widely adopted prediction attribution approaches.

3.2 Related Work

In terms of scope, interpretability methods can be categorized as model-specific or model-agnostic. Model-specific methods interpret only one family of models. For example, a prominent line of work within natural language processing (NLP) endeavors to explain BERT [Devlin et al., 2019] by probing and visualization [Clark et al., 2019; Kovaleva et al., 2019; Tenney et al., 2019a; van Aken et al., 2019]. Model-agnostic techniques aim for wide applicability across many families of parametric models. We can roughly separate model-agnostic methods into three categories: (1) gradient-based ones [Fong and Vedaldi, 2017; Li et al., 2016; Sundararajan et al., 2017]; (2) probing [Clark et al., 2019; Liu et al., 2019; Lundberg and Lee, 2017; Ribeiro et al., 2016; Tenney et al., 2019b]; (3) information-theoretical methods [Bang et al., 2019; Guan et al., 2019; Pimentel et al., 2020; Schulz et al., 2020] .

Gradient-based methods are limited to models with differentiable neural activations. They also fail to capture all the information associated with the correct prediction [Li et al., 2016]. Although probing methods provide detailed insight into specific models, they fail to capture inner mechanisms like how information flows through the network [Guan et al., 2019]. Information-theoretic methods, in contrast, provide consistent and flexible explanations, as we show in this chapter.

Guan et al. [2019] use mutual information to interpret NLP models across different tokens, layers, and neurons, but they lack a quantitative evaluation. Bang et al. [2019] also propose a

model-agnostic interpretable model using IB; however, they limit the information through the network by sampling a given number of words at the beginning, which restricts the explanation to neurons only. Our method is inspired by Schulz et al. [2020], who use IB for attribution in image classification.

3.3 Information Bottleneck for Attribution

The idea of IBA is to restrict the information flowing through the network for every single instance, such that only the most useful information is kept. Concretely, given an input $\mathbf{X} \in \mathbb{R}^N$ and output $\mathbf{Y} \in \mathbb{R}^M$, an information bottleneck is an intermediate representation $\mathbf{T}$ that maximizes the following function:

$$\mathrm{I}(\mathbf{Y}; \mathbf{T}) - \beta \cdot \mathrm{I}(\mathbf{X}; \mathbf{T}), \tag{3.1}$$

where I denotes mutual information and β controls the trade-off between reconstruction $\mathrm{I}(\mathbf{Y}; \mathbf{T})$ and information restriction $\mathrm{I}(\mathbf{X}; \mathbf{T})$. The larger the β, the narrower the bottleneck, i.e., less information is allowed to flow through the network.

We insert the IB after a given layer l in a pretrained deep neural network. In this case, $\mathbf{X} = f_l(\mathbf{H})$ represents the chosen layer's output, where $\mathbf{H}$ is the input of the layer. We restrict information flow by injecting noise into $\mathbf{X}$:

$$\mathbf{T} = \boldsymbol{\mu} \odot \mathbf{X} + (\mathbf{1} - \boldsymbol{\mu}) \odot \epsilon, \tag{3.2}$$

where $\odot$ denotes element-wise multiplication, ϵ the injected noise, $\mathbf{X}$ the latent representation of the chosen layer, $\mathbf{1}$ the all-one vector, and $\boldsymbol{\mu} \in \mathbb{R}^N$ the weight balancing signal and noise. For every dimension i, $\boldsymbol{\mu}_i \in [0, 1]$, meaning that when $\boldsymbol{\mu}_i = 1$, there is no noise injected into the original representation. To simplify the training process, we set $\boldsymbol{\mu}_i = \sigma(\boldsymbol{\alpha}_i)$, where σ is the sigmoid function and $\boldsymbol{\alpha}$ is a learnable parameter vector that we are optimizing. In the extreme case, where all the information in $\mathbf{T}$ is replaced with noise ($\mathbf{T} = \epsilon$), it's desirable to keep ϵ the same mean and variance as $\mathbf{X}$ in order to preserve the magnitude of the input to the following layer. Thus, we have $\epsilon \sim \mathcal{N}(\mu_{\mathbf{X}}, \sigma^2_{\mathbf{X}})$.

After obtaining $\mathbf{T}$, we evaluate how much information $\mathbf{T}$ still contains about $\mathbf{X}$, which is defined as their mutual information:

$$\mathrm{I}(\mathbf{X};\mathbf{T}) = \mathbb{E}_{\mathbf{X}}[D_{KL}[P(\mathbf{T}|\mathbf{X})\|P(\mathbf{T})]], \tag{3.3}$$

where D_{KL} means Kullback–Leibler (KL) divergence, $P(\mathbf{T}|\mathbf{X})$ and $P(\mathbf{T})$ represent their probability distributions. While $P(\mathbf{T}|\mathbf{X})$ can be sampled empirically, $P(\mathbf{T})$ has no analytical solution since it requires integrating over the feature map $P(\mathbf{T}) = \int P(\mathbf{T}|\mathbf{X})P(\mathbf{X})\mathrm{d}\mathbf{X}$. As is standard, we use the variational approximation $Q(\mathbf{T}) = \mathcal{N}(\mu_{\mathbf{X}}, \sigma^2_{\mathbf{X}})$ to substitute $P(\mathbf{T})$, assuming every dimension of $\mathbf{T}$ is independent and normally distributed. Even though the independence assumption does not hold in general, it only overestimates the mutual information, giving a nice upper bound of mutual information between $\mathbf{X}$ and $\mathbf{T}$:

$$\begin{aligned}
\mathrm{I}(\mathbf{X};\mathbf{T}) &= \mathbb{E}_{\mathbf{X}}[D_{KL}[P(\mathbf{T}|\mathbf{X})\|P(\mathbf{T})]] \\
&= \int_{\mathbf{X}} p(x)(\int_{\mathbf{T}} p(t|x)\log\frac{p(t|x)}{p(t)}dt)dx \\
&= \int_{\mathbf{X}}\int_{\mathbf{T}} p(x,t)\log\frac{p(t|x)}{p(t)}\frac{q(t)}{q(t)}dtdx \\
&= \int_{\mathbf{X}}\int_{\mathbf{T}} p(x,t)\log\frac{p(t|x)}{q(t)}dtdx \\
&+ \int_{\mathbf{X}}\int_{\mathbf{T}} p(x,t)\log\frac{q(t)}{p(t)}dtdx \\
&= \int_{\mathbf{X}}\int_{\mathbf{T}} p(x,t)\log\frac{p(t|x)}{q(t)}dtdx \\
&+ \int_{\mathbf{T}} p(t)(\int_{\mathbf{X}} p(x|t)dx)\log\frac{q(t)}{p(t)}dt \\
&= \mathbb{E}_{\mathbf{X}}[D_{KL}[P(\mathbf{T}|\mathbf{X})\|Q(\mathbf{T})]] \\
&- D_{KL}[Q(\mathbf{T})\|P(\mathbf{T})] \\
&\leq \mathbb{E}_{\mathbf{X}}[D_{KL}[P(\mathbf{T}|\mathbf{X})\|Q(\mathbf{T})]]
\end{aligned}$$

Since we expect $\mathrm{I}(\mathbf{X},\mathbf{T})$ to be small and mutual information to be always nonnegative, the upper bound is a desired property.

Intuitively, the purpose of maximizing $\mathrm{I}(\mathbf{Y};\mathbf{T})$ is to make accurate predictions. Therefore,

instead of directly maximizing $\mathrm{I}(\mathbf{Y};\mathbf{T})$, we minimize the loss function for the original task, e.g., the cross entropy $\mathcal{L}_{\text{CE}}$ for classification problems after inserting the information bottleneck.

Combining the above two parts, our final loss function $\mathcal{L}$ is

$$\mathcal{L} = \mathcal{L}_{\text{CE}} + \beta \cdot \mathbb{E}_{\mathbf{X}}[D_{KL}[P(\mathbf{T}|\mathbf{X})\|Q(\mathbf{T})]]. \tag{3.4}$$

Note that we negate the sign for minimization. The β hyperparameter controls the relative importance between the two loss components. After the optimization process, we obtain for every instance a compressed representation $\mathbf{T}$.

In order to get $\mathbf{T}$, we optimize the learning parameter $\boldsymbol{\alpha}$. At the beginning of the training, we start with $\mathbf{T} \approx \mathbf{X}$ to keep the information of $\mathbf{X}$ in $\mathbf{T}$ as much as possible. Thus, we initialize $\boldsymbol{\alpha}_j = 5$ for each dimension j as it results in $\boldsymbol{\mu}_j = 0.993$, which is close to 1 as desired. During optimization, we fix the training steps to 10 and repeat a sample 10 times to inject different noise, which altogether requires 100 total steps to generate an attribution map for a single instance. Another important hyperparameter is β. We empirically pick $\beta \approx 10 \times \frac{\mathcal{L}_{\text{CE}}}{\mathcal{L}_{\text{IB}}}$, consistent with what we found that works the best in the classification experiments.

We then calculate $D_{KL}[P(\mathbf{T}|\mathbf{X})\|Q(\mathbf{T})]$, indicating how much information is still kept in $\mathbf{T}$ about $\mathbf{X}$, which suggests the contribution of each token and feature. To generate the attribution map, we sum over the feature–token axis, obtaining the attribution score of each token.

Overall, we try to learn a compressed hidden representation $\mathbf{T}$ that has just enough information about the input $\mathbf{X}$ to predict the output $\mathbf{Y}$. This compression is done by adding noise, which removes the least relevant feature-level information, with $\boldsymbol{\mu}$ controlling how much to remove.

3.4 Experimental Setup

Through experimentation, we analyze IBA both quantitatively and qualitatively to understand how it interprets deep neural networks across layers.

We compare our method on BERT with two other representative model-agnostic instance-level methods—LIME [Ribeiro et al., 2016], which explores interpretable models for approximation and explanation, and integrated gradients (IG) [Sundararajan et al., 2017], a variation on

computing the gradients of the predicted output with respect to input features. For a simple baseline, we also compare with "random," whose attribution scores are assigned randomly to tokens. On each dataset, we fine-tune BERT and apply these interpretability techniques to the model. We note the test accuracy and generate an attribution score for each token. Details of all parameters are attached in Appendix A.

There is no consensus on how to evaluate interpretability methods quantitatively [Molnar, 2019]. LIME's simulated evaluation leverages the ground truth of already interpretable models like decision trees, but the ground truth is unavailable for black-box models like neural networks. Therefore, we follow Ancona et al. [2018] and Hooker et al. [2018] and carry out a *degradation test* on IMDB [Maas et al., 2011], AG News [Gulli, 2004], MNLI [Williams et al., 2018], and RTE [Wang et al., 2018a], covering sentiment analysis, natural language inference, and text classification.

The degradation test has the following steps:

1. Generate attribution scores s for each interpretability method f: $s = f(\mathcal{M}, x, y)$, where x is the test instance, y is the target label, and $\mathcal{M}$ is the model.
2. Sort tokens by their attribution score in descending order.
3. Remove top k tokens to obtain x', the degraded instance; k can be preset.
4. Test the target class probability $p(y|x')$ with the original model on the degraded instance.
5. Repeat steps 3 and 4 until all tokens are removed.

For the final visualization, we average all test instances at each degradation step to compute $\bar{p}(y|x')$. Then, we normalize the degradation test result $\bar{p}(y|x')$ to $[0, 1]$ using the normalized probability drop $\bar{d} = \frac{\bar{p}(y|x') - m}{o - m}$, where o means the original probability on the nondegraded instance, and m means the minimum of the *fully* degraded instance's probability across all interpretability models. In this way, the normalized probability drop $\bar{d}$ will be independent of the original model quality and easily comparable across models. Note that, for IBA, we perform the degradation test on the original model, not the one with the inserted bottleneck. Thus, a large β does not directly cause the probability to drop. An effective attribution map can find the most important tokens, which means $\bar{p}(y|x')$ after the degradation step will drop substantially.

	IMDB	MNLI Matched	MNLI Mismatched	AG News	RTE
Original	0.864	0.823	0.828	0.907	0.572
Random (↓)	0.011	0.106	0.106	0.008	0.012
LIME (↓)	0.038	0.244	0.260	0.033	0.014
IG (↓)	0.090	0.226	0.233	**0.036**	0.043
IBA (↓)	**0.229**	**0.374**	**0.367**	0.029	**0.059**

Table 3.1: Absolute probability drop for the target class after the top 11% most important tokens removed. The larger the score, the more effective the method. Here we use $\beta = 1e - 5$.

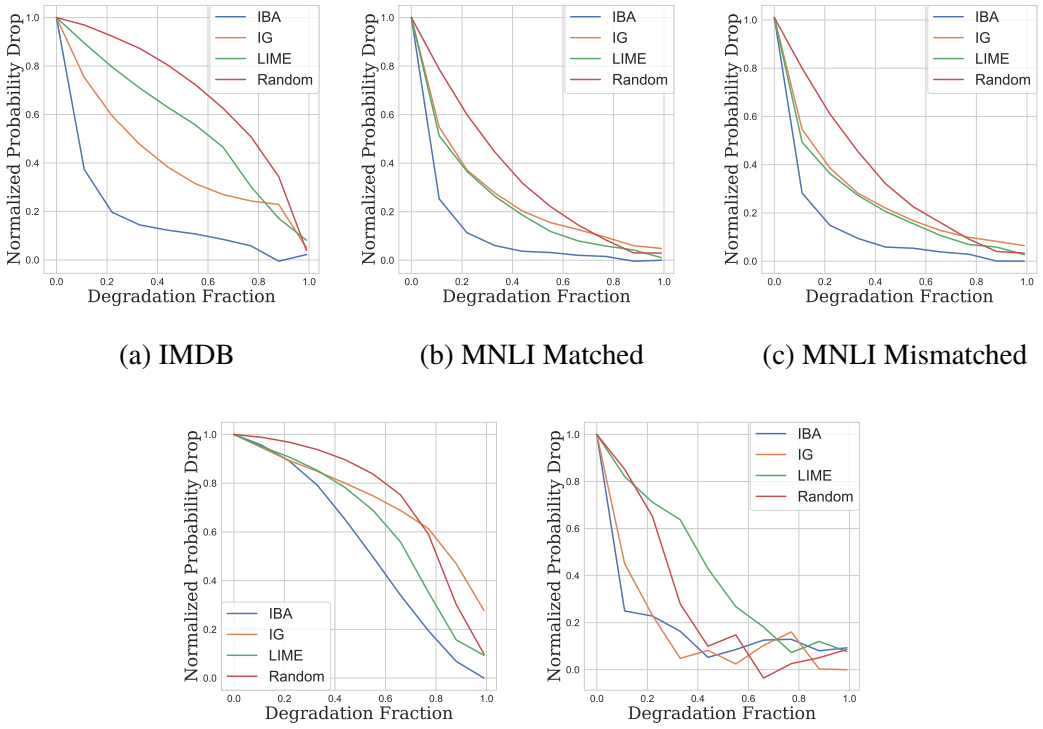
3.5 Results and Analysis

Overall, the results show that our method better identifies the most important tokens compared to other model-agnostic interpretability methods.

3.5.1 Quantitative Analysis

Table 3.1 shows the absolute probability drop $\|\bar{p}(y|x) - o\|$ with the first 11% of the important tokens removed. We further plot the normalized probability drop after each percentage of the important tokens is removed, as shown in Figure 3.1, indicating how much important information is lost for prediction: the steeper the slope, the better the ability to capture important tokens. For this experiment, we insert the information bottleneck after layer 9, and we see that removing important tokens that are identified by our method deteriorates the probability the most on IMDB and MNLI Matched/Mismatched.

Of course, choosing the right layer to insert the information bottleneck is crucial to the result. It also indicates which layer encodes the most meaningful information for prediction. To investigate differences in inserting information bottlenecks after different layers, we carry the degradation test on 1000 random test samples across layers on IMDB, as shown in Figure 3.2a—see Appendix B for all 12 layers. Insertion after layers 1, 8, and 9 generates more meaningful attribution scores. At layer 1, the tokens remain distinct (i.e., representations have not been aggregated), and it is likely that the latent representation $\mathbf{T}$ is essentially capturing per-token sentiment



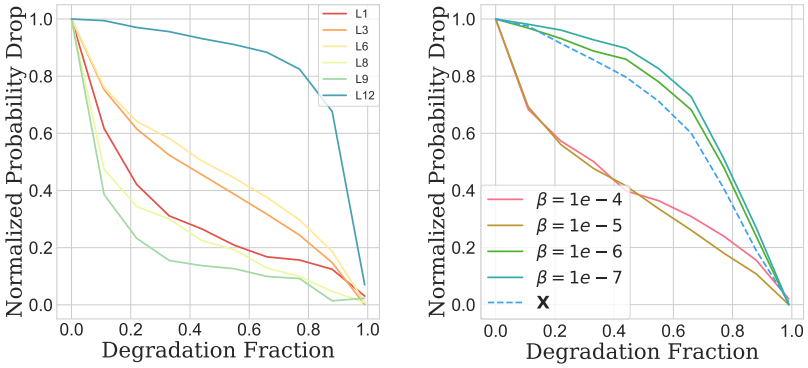
(a) IMDB (b) MNLI Matched (c) MNLI Mismatched

(d) AG News (e) RTE

Figure 3.1: Degradation test results comparing IBA, IG, LIME, and random. The steeper the slope is the better the method is.

values. The big drop of $\bar{d}$ after layers 8 and 9, on the other hand, is interesting. Recently, Xin et al. [2020] examined early exit mechanisms in BERT and found that halting inference at layers 8 or 9 produces results not much worse than full inference, which suggests that an abundance of information is encoded in those layers.

Another important parameter is β, which controls the trade-off between restricting the information flow and achieving greater accuracy. A smaller β allows more information through, and an extremely small β has the same effect of using $\mathbf{X}$ as the attribution map. As Figure 3.2b shows, when $\beta \leq 1e-6$, the degradation curve is similar to the one using $\mathbf{X}$ only.



(a) IB after different layers. (b) IB with different β.

Figure 3.2: Analysis of different layers and different β.

3.5.2 Qualitative Analysis

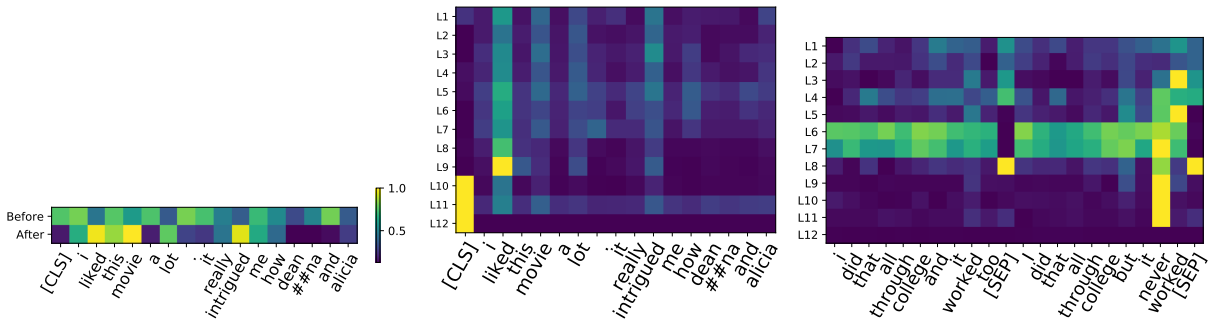
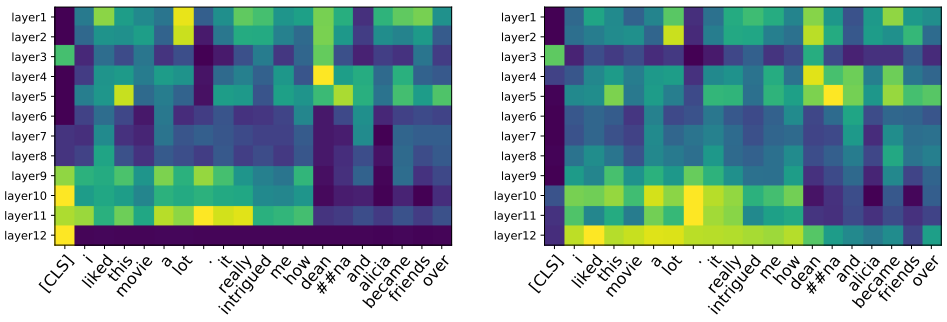


Figure 3.3: Illustrations from left to right are as follows: The before and after comparison of inserting an information bottleneck after layer 6; attribution for an IMDB example with the positive label; attribution for an MNLI example with the contradiction label.

The first plot in Figure 3.3 shows the before and after comparison of IB insertion, with positive tokens highlighted. The second and third plots visualize attribution maps for instances across layers. Consistent with our quantitative analysis in Figure 3.2a, these plots demonstrate that, for a fully fine-tuned BERT, layers 8 and 9 seem to encode the most important information for the



(a) $\beta = 10^{-3}$ (b) $\beta = 10^{-7}$

Figure 3.4: Comparison of BERT attribution maps with different values of β.

prediction. For example, in the IMDB instance, *liked* and *intrigued* have the highest attribution scores for the prediction of positive sentiment across most layers—see layer 9 in particular. In the MNLI example, *never* is mostly highlighted starting from layer 7 to predict "contradiction."

Similarly, we can tell the different effects of β on attribution maps qualitatively. Figure 3.4 shows the effects of different β on a specific example. As we can see, when β is as small as 10^{-7}, most information is allowed to flow through the network and thus most parts are highlighted. In contrast, when β is larger, the representation is more restricted.

3.6 Summary

In this chapter, we introduce an information-bottleneck-based approach to analyze attribution for transformers. Our method outperforms two widely used attribution methods across four datasets in sentiment analysis, document classification, and textual entailment. We also conduct analysis across 12 layers and have found that layer 8,9 are the most informative ones. To help us better understand the role β plays in this method, we also analyze the bottleneck representation with different β both quantitatively and qualitatively.

Chapter 4

Restrict Representations by Inserting Information Bottleneck for Attribution in Passage Reranking

In the previous chapter, we introduce the information bottleneck for attribution method and use it to interpret classification tasks. It's unclear how this method can be used to understand other NLP tasks like passage ranking in depth. In this chapter, we show how this method can be used to interpret reranking models with detailed token-wise analysis. The work is based on our paper:

- Zhiying Jiang, Raphael Tang, Ji Xin, Jimmy Lin. How Does BERT Rerank Passages? An Attribution Analysis with Information Bottlenecks. *In Proceedings of the Fourth BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP 2021*.

4.1 Overview

Pretrained language models like BERT [Devlin et al., 2019] have achieved prominent improvements in both information retrieval (IR) and natural language processing (NLP). Concurrently, researchers have raised wide awareness about the difficulty of explaining such deep learning models [Fong and Vedaldi, 2017; Guidotti et al., 2018; Robnik-Šikonja and Bohanec, 2018]. Recently,

many papers scrutinize BERT's behaviors in various tasks [Clark et al., 2019; MacAvaney et al., 2020; Qiao et al., 2019; Tenney et al., 2019a; van Aken et al., 2019]. When it comes to token-wise analysis, most of the previous works focus on intra-layer self-attention and how it relates to various linguistic characteristics. Although these analyses yield unique insights on layer-local behavior across pairs of tokens, they do not take a global perspective of how token-wise representations exactly relate to the prediction. This is crucial for answering a fundamental question in interpretability: what hidden features and tokens contribute the most to the prediction?

To faithfully compute such feature–prediction attribution maps, we leverage IBA to analyze passage reranking for pretrained transformers. We first elaborate on how we use IBA to interpret passage reranking. We then justify the choice of using IBA for attribution map generation by comparing it with two other widely adopted attribution methods. We then carry out detailed analyses of the inner mechanisms of passage reranking.

BERT reranking [Nogueira and Cho, 2019] starts a new chapter in information retrieval, as it combines the dual advantages of the speed of sparse representation (BM25) and the deep contextualization of dense representation. To be specific, given a query q, BM25 returns top-1,000 passages D. The label r is 1 if a passage $d \in D$ is relevant to q, and 0 otherwise. For BERT, the input is `[CLS] q [SEP] d [SEP]`, and the output label is r. After fine-tuning, we rerank D based on the output probabilities of relevance. This setting is different from most NLP tasks, where positive and negative labels are provided by the dataset, and only one pair of (input, probability) is required for the final output.

We use IBA to generate attribution maps for BERT-large [Devlin et al., 2019] fine-tuned on the MS MARCO dataset [Bajaj et al., 2016b]. With the attribution maps, we investigate the following questions:

Q1. What are the similarities and differences between BERT and BM25?
For the two-stage pipeline, we wonder how BERT's ranking mechanism is similar to BM25 and what it provides that BM25 doesn't. Through cross-passage examination, we find that BERT still regards lexical matching as important to some extent, similar to BM25. BERT, furthermore, manages to capture deeper-contextualized relationships between the query and the relevant passage.

Q2. How do special tokens contribute to reranking across layers?
In BERT, only the `[CLS]` token is designed to factor into prediction. Then how do those special

tokens collect information across layers to capture a contextualized relationship? We find that different from what attention analyses show, [CLS] starts to gather the evidence for prediction primarily after layer 16, especially in layer 24.

Q3. How robust is the top-ranked passage?
One of the special settings of ranking is that we do not care about the absolute score, as long as the relevant passage ranks higher than the irrelevant ones. We conduct experiments of token removal for the top-1 positive passage to test the robustness. We find that we can truncate up to 22.5% tokens on average, given reasonable attribution scores, of the top-ranked passage without affecting its order.

Q4. Does BERT have positional bias?
We then look deeper into what makes those passages rank higher. We find that BERT, after being fine-tuned on MS MARCO, prefers those passages with inverted pyramid structure—that is, passages that put important information at the start. We further confirm that it has a positional bias towards the start of the passage through various experiments.

4.2 Related Work

Generally speaking, interpretability methods are either model specific, applying to only a single architectural family, or model agnostic, covering a broad spectrum of supervised models. Since pretrained transformers represent the state of the art in NLP, for model-specific techniques we discuss those for BERT, the prototypical, most-interpreted transformer model. As this work specifically explores passage reranking, we also provide the necessary literature about recent progress.

4.2.1 BERT specific

A number of works investigate the inner mechanisms of BERT. Clark et al. [2019]; Kovaleva et al. [2019] carefully analyze BERT's attention heads, noting a positive correlation between attention heads and linguistic features, as well as special tokens.

Looking at attention, Voita et al. [2018] find that BERT captures anaphora and dependence on position and length in machine translation. Pointing out some shortfalls of these papers, Brunner et al. [2019]; Jain and Wallace [2019]; Serrano and Smith [2019] argue that attentions often do not reflect how models make predictions.

Another line of works analyzing BERT use probing classifiers to draw the connection between vector representation and specific linguistic knowledge [Hewitt and Manning, 2019; Liu et al., 2019; Tenney et al., 2019a]. Rogers et al. [2020] provide a thorough literature survey about what we already know about how BERT works and they've found different probing methods sometimes lead to contradictory interpretations. A direct remedy is to look into what BERT looks at during inference time (i.e., identify important features for prediction, also known as "attribution methods" in general). That's what our work focuses on.

4.2.2 Attribution maps

Although more commonly applied to convolutional neural networks in image classification, most attribution methods are model agnostic. They aim to assign weights to input features according to how the model makes predictions, with higher weights corresponding to greater contributions.

The most prevalent methods are gradient-based. Intuitively, gradients reflect how small changes in the input affect the final prediction to some extent. But previous work shows that raw gradients are noisy and limited to capturing only the local "importance" [Smilkov et al., 2017]. To remedy this, some of them [Smilkov et al., 2017; Sundararajan et al., 2017] incorporate global importance to mitigate this problem, while others [Binder et al., 2016; Kindermans et al., 2018; Shrikumar et al., 2017] modify or extend the back-propagation algorithms directly to emphasize positive contributions with regard to prediction. However, Sixt et al. [2020] show that most of the modified back-propagation methods fail a basic sanity check: invariance to parameter randomization and label randomization.

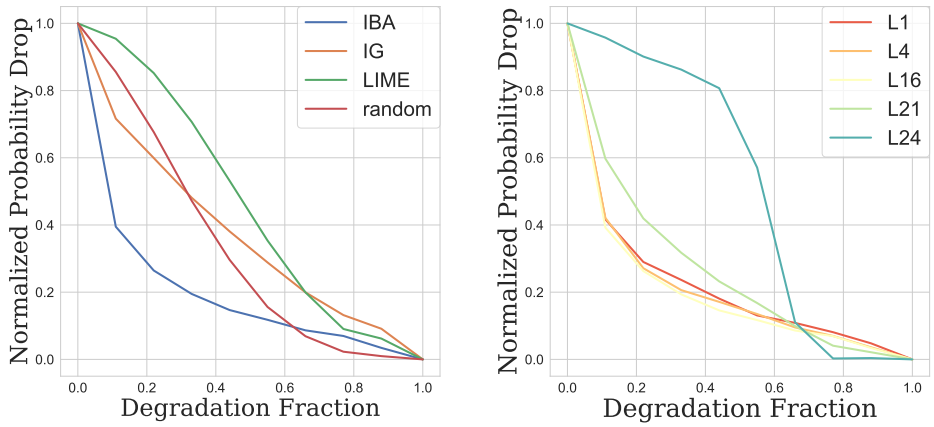
LIME [Ribeiro et al., 2016] is not limited to differentiable models. They use interpretable models like decision trees to approximate deep neural networks, and thus can theoretically interpret any classifier. However, empirically, LIME's high demand on memory may worsen its quality compared to other methods, as we will see in the later section.

Information-theoretic methods are often unconstrained by tasks and models as well, while additionally providing a unified view of how information flows across models. Guan et al. [2019] use mutual information to estimate tokens' importance across layers but don't provide a quantitative evaluation. Bang et al. [2019] also take advantage of information bottlenecks to interpret predictions, but they restrict the information by sampling tokens, which doesn't generate a complete attribution map for every token and limits the interpretation to be token-wise only. More recently, Schulz et al. [2020] propose the information bottleneck method for attribution, which empirically achieves the best result on multiple evaluation metrics in interpreting images. Jiang et al. [2020] further leverage this method in NLP and also surpass other model-agnostic methods on multiple datasets.

4.2.3 Neural IR

BERT is a game changer for information retrieval. Lin et al. [2020] even separate neural reranking techniques into "pre-BERT" and "post-BERT" eras. Nogueira and Cho [2019] start the post-BERT era by proposing a two-stage pipeline, using sparse representations like BM25 to generate candidates and then neural models like BERT to rerank them. More recent work explores merging the two-stage pipeline into an end-to-end dense retrieval, like DPR [Karpukhin et al., 2020b], which still uses BERT as the basic building block for neural information retrieval. Therefore, understanding BERT's behavior for reranking in the original setting still helps.

Toward this, a few previous works specifically analyze BERT for reranking: Qiao et al. [2019] analyze attention to see how BERT attends to stop words and regular words across layers. MacAvaney et al. [2020] have done a more thorough study of various reranking models, using carefully designed textual manipulation methods. Different from them, we use a model-agnostic method to generate a token-wise attribution map, as it provides us with the flexibility to carry out a layer-wise analysis. Besides, to the best of our knowledge, no previous work has done a cross-passage analysis to see patterns across the ranks of different passages.



(a) Comparison among attribution (b) Comparison among layers

Figure 4.1: Degradation test on MS MARCO with BERT-large.

4.3 IBA for Passage Reranking: How & Why

4.3.1 How

The procedure of using BERT to rerank passages [Nogueira and Cho, 2019] can be characterized as follows: Given query q, and a list of passages D, $d \in D$ is returned by BM25. BERT then assigns the relevance score $R(q, d)$, the logits for the probability that the passage is regarded as relevant, to each pair of q and d. $\mathcal{L}_{\text{CE}}$, in this case, is the same as the cross entropy in Nogueira and Cho [2019]. We use the BERT-large model fine-tuned on the MS MARCO dataset for experiments.

The optimization process of getting the attribution map through an information bottleneck is similar to Section 3.3.

4.3.2 Why

We first carry out experiments to justify the choice of using IBA for attribution map by showing the benchmark and examples. To compare the effectiveness of IBA with other attribution methods,

	MS MARCO
Original	0.939
Random (↓)	0.135
LIME (↓)	0.043
IG (↓)	0.265
IBA (↓)	**0.565**

Table 4.1: Probability drop after removing first 11% important tokens identified by these methods.

we carry out a degradation test. The essential idea of a degradation test is to remove the most important $k\%$ tokens, excluding special tokens, identified by different attribution methods and measure the drop of the probability with respect to the given label.

The initial value of k is 11 and we increase k until all the tokens are removed, shown as the x-axis in Figure 4.1a. y-axis means the normalized average probability drop after removing a certain percentage of tokens: $\frac{\bar{p}(y|x')-m}{o-m}$ where x' represents input with certain tokens removed, o is the original probability before tokens removal, and m is the minimum of the fully degraded instance's probability across all attribution methods. We conduct the experiment across the entire MS MARCO dev set (6980 queries).

We compare the result with two other popular model-agnostic attribution methods, LIME [Ribeiro et al., 2016] and Integrated Gradients (IG) [Sundararajan et al., 2017], each representing a different category of attribution methods: LIME uses interpretable models like decision trees and linear models to approximate the black box, while IG is a variation of using the gradient of the predicted output with respect to given input features. To provide a simple baseline, we also compare the result with "Random," where tokens are removed randomly. We expect a better attribution method will have a steeper slope, meaning removing important tokens identified by the method significantly deteriorates the performance. As shown in Figure 4.1, IBA outperforms all other three methods with a 61.3% probability drop compared with second-placed IG, which makes for a 29.0% drop. The absolute probability drop value can be seen in Table 4.1.

Table 4.2 shows a few examples with highlighted important tokens. We can see top-10 most important tokens across query and passage not only showing the token matching but also captur-

Query	Document
how much did nr ##a give to congress	m ##em ##bers of congress pay attention to these numbers , and they know that in the last election cycle the nr ##a spent $ 18 . 6 million on various campaigns , a says lee dr ##ut ##man , who has studied the role of gun money in politics for the sunlight foundation .
is mortgage and deed of trust the same document	the mortgage or deed of trust is recorded in the county land records , usually shortly after the borrow ##ers sign it . if the loan is fully paid off , the lend ##er will record a release (or satisfaction) of mortgage or a rec ##on ##vey ##ance of deed (which is used in conjunction with deeds of trust) in the county land records .
which is stronger hydro ##co ##don ##e or ox ##y ##co ##don ##e	dos ##age conversion : hydro ##co ##don ##e vs . ox ##y ##co ##don ##e . in terms of strength , 5 ##mg of ox ##y ##co ##don ##e is roughly equivalent of 7 . 5 of hydro ##co ##don ##e . that is the conversion required to bring about the same effects . hydro ##co ##don ##e would work better if you happen to be a lightweight person with a weak stomach .

Table 4.2: Top-10 most important tokens identified by IBA in three examples. '[CLS]' and '[SEP]' are ignored.

ing semantic relatedness. For example, "much" in the query of the first example is highlighted. "number", "million", and "$" signs, which are highly related to the concept of "much", are also highlighted. Similarly, in the second example, BERT identifies the core of the question – "same document". In the corresponding passage, it emphasizes "or" as well as "mortgage" before that and "trust" after that. In the third example, the query is about "stronger", which is again, captured by BERT, and related tokens like "vs" and "roughly equivalent" are highlighted.

4.4 Experimental Setup

Given the attribution maps, we are now able to study which tokens BERT looks at for reranking. To be specific, we exploit IBA to extract the top-20 most important tokens $\mathbb{M}$ for each

$(q, d), q \in Q, d \in D$, where Q and D represent the query list and the passage list. We carry out our experiment under two different settings:

1. Q consists of 105 queries from a subset of the MS MARCO passage reranking dev set, provided by Pyserini [Lin et al., 2021]. D comprises top 50 passages that BERT-large retrieves for each query. For these experiments, we fix the layer l that we insert the information bottleneck after.

2. Q consists of 1,000 randomly selected queries from the entire MS MARCO passage reranking dev set. D is composed of the human-annotated relevant passages. We then apply IBA to all 24 layers to get top-20 tokens $\mathbb{M}$ for each (q, d).

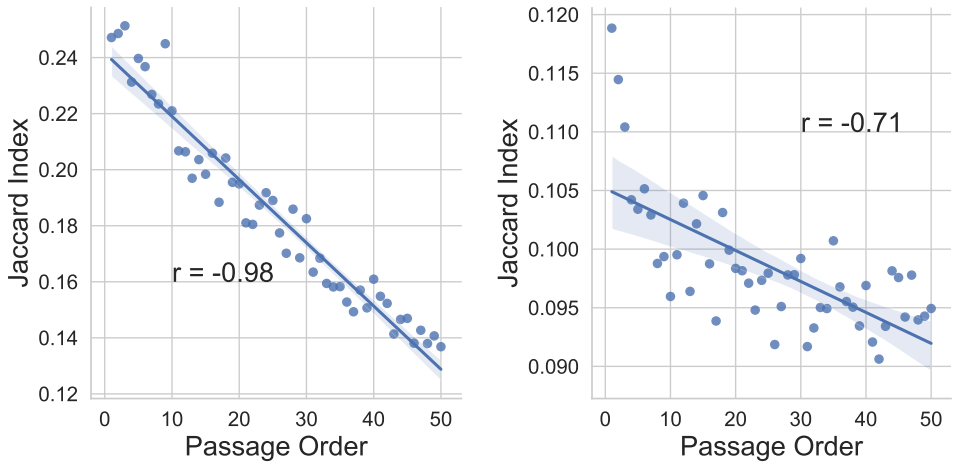
For setting 1, we perform cross-passage analysis to investigate different patterns between higher-ranked passages and lower-ranked passages. The choice of the top-50 cutoff is due to frugality: the recall@50 (0.817) is comparable to the recall@1000 (0.848), with much less computation. For setting 2, we aim at cross-layer analysis for relevant passages. Specifically, we identify if lower layers show a different focus from higher layers. This setting is similar to GLUE-like classification tasks [Wang et al., 2018a] where we want to find general patterns about BERT. The reason for using the top 20 is that, in our sampled instances, the average tokenized query length is 9.2, and we also want to see the emphasized tokens in passages.

4.5 Results and Analysis

Equipped with the attribution map and appropriate setting to conduct a detailed analysis, we are now ready to answer those questions.

4.5.1 Cross-Passage Analysis

It's well known that two-stage ranking pipelines use both exact token matching and semantic relatedness [Lin et al., 2020]. As BM25 estimates relevance purely by lexical matching, we wonder if BERT still relies on exact match and what else BERT provides.



(a) Important tokens.

(b) Whole query and passage.

Figure 4.2: Across top-50 passage analysis.

Q1. What are the similarities and differences between BERT and BM25?

To answer this question, we first study the correlation between higher ranking scores and higher lexical matching between queries and passages.

To measure the degree of lexical matching, we use the Jaccard index under experimental setting 1: $\mathbf{J} = \frac{|u_i \cap v_j|}{|u_i \cup v_j|}$, $\{u_i, v_j | u_i \in q \cap \mathbb{M}, v_j \in d \cap \mathbb{M}\}$, where $i \in [1, |q|], j \in [1, |d|]$, remember that $\mathbb{M}$ is the top-20 tokens extracted by IBA.

For each query q, we calculate the Jaccard index for every passage d_i in the top 50 passages. We then average them across all queries. We choose to insert the information bottleneck after layer 16, as it is the most informative one according to our degradation test.

As we see in Figure 4.2a, the Jaccard index decreases as the rank of the passages becomes lower. In general, the higher[1] the rank is, the more overlapped the important tokens between the query and passages are. We also calculate Spearman's correlation [Spearman, 1961] r_s to gauge the degree of monotonic association. We find that $r_s = -0.98$, indicating a strong monotonic relation between the Jaccard index and passages order. Does this correlation hold among

[1] "Higher" rank actually means lower order: passages with order 1 have a higher rank than passages with order 2, etc.

Query: What is the goal for the child with a cognitive impairment?	
BM25 Ranked 1st	BERT Reranked 1st
A cognitive impairment is a condition where your child has some problems with ability to think and learn. Children with a cognitive impairment often have trouble with such school subjects as math and reading. Cognitive impairment is a condition where your child has some problems with ability to think and learn.	Promoting optimum development. The goal for children with cognitive impairment is the promotion of optimum social, physical, cognitive, and adaptive development as individuals within a family and community. Vocational skills are only one part of the goal. The focus must also be on the family and other aspects of development.

Table 4.3: Top-1 passage by BM25 and BERT.

all tokens between the query and passage? Figure 4.2b shows the Jaccard index $\mathbf{J}' = \frac{|u'_i \cap v'_j|}{|u'_i \cup v'_j|}$, $\{u'_i, v'_j | u'_i \in q, v'_j \in d\}$ across passages. We see it shows a similar trend to $\mathbf{J}$, confirming that even if BM25 returns passages that have higher lexical matching with the query, token matching between queries and passages still plays an important role when BERT is reranking. But using all of the tokens between the query and the passage obtains a correlation coefficient of $r_s = -0.71$, which is lower than using important tokens only. We argue that it's because IBA interprets in a way that's more aligned with the specific tokens that BERT looks at when reranking.

We further investigate what BERT provides that BM25 doesn't. Specifically, we look into what d_{BERT} gets right but d_{BM25} gets wrong. We notice that BERT captures more contextualized relevance between the query and passage, while the BM25-returned answer has more "superficial" relevance - d_{BM25} seems to talk about the topic but doesn't really answer the question. The example shown in Table 4.3 demonstrates that the passage returned by BM25 seems highly related to the topic—"cognitive impairment" but instead of explaining what the goal is, it is explaining what "cognitive impairment"'s definition is. On the contrary, BERT not only returns the passage related to "cognitive impairment" but also the goal.

We find that it is hard to quantitatively measure the contextualized relevance between queries and passages by simply calculating cosine similarity ϕ between query vectors and document vectors as shown in Table 4.4. We encode q, d_{BERT}, d_{BM25} and don't find that $\phi(\eta(q), \eta(d_{\text{BERT}}))$ is

	USE	sent-bert(p)	sent-bert(n)
BM25 top-1	0.540	0.593	0.578
BERT top-1	0.483	0.731	0.563

Table 4.4: Cosine similarity between query and top-1 passage returned by different methods. "p" refers to pretrained model paraphrase-MiniLM-L6-v2", "n" refers 'bert-base-nli-mean-tokens'

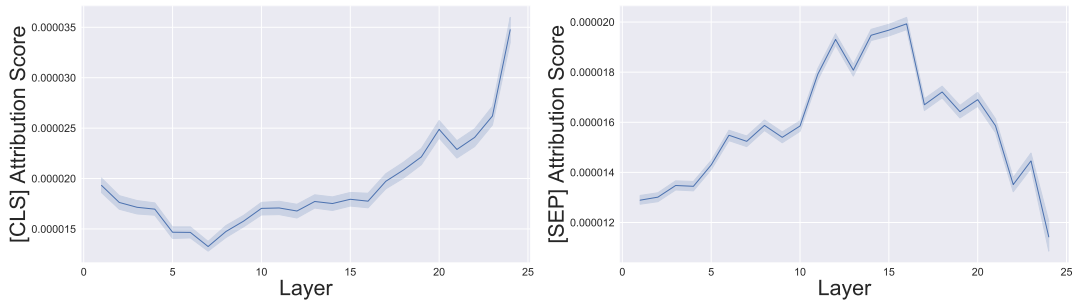
higher than $\phi(\eta(q), \eta(d_{\text{BM25}}))$ when using the Universal Sentence Encoder or Sentence-BERT [Reimers and Gurevych, 2019b], denoted as η, pretrained on an NLI dataset. However, if we use a Sentence-BERT pretrained on a paraphrase corpus (specifically the model "paraphrase-MiniLM-L6-v2") to measure semantic similarity, $\phi(\eta(q), \eta(d_{\text{BERT}}))$ is significantly higher than $\phi(\eta(q), \eta(d_{\text{BM25}}))$ as "paraphrase-MiniLM-L6-v2's" pretrained corpus includes MS MARCO triplet. Tempting as it is to conclude that BERT has indeed captured semantic similarity that BM25 hasn't, it's unfair to use a pretrained model with prior knowledge of MS MARCO to measure the semantic similarity. Therefore, we think BERT has learned a deeper relevance between the query and document, but it cannot be simply measured by vaguely defined semantic similarity.

4.5.2 Cross-Layer Analysis

Downstream tasks often rely on BERT's `[CLS]` vector at the last layer as input, and that's also true for reranking. It's intriguing to know the layer at which `[CLS]` starts to learn the relevance. Clark et al. [2019] thoroughly analyze BERT's self-attention mechanism for each layer. While they provide insight into how tokens attend to one another, the attention weights themselves often do *not* correlate with measures of feature importance [Jain and Wallace, 2019].

Q2. How do special tokens contribute to reranking across layers?

We insert an information bottleneck after each layer for 24-layer attribution maps. First, we inspect how the `[CLS]` token gets emphasized across the layers. Figure 4.3a shows the attribution score across 24 layers in experimental setting 2, with 95% confidence intervals. Note that the score is normalized between 0 to 1 for each token but it doesn't add up to 1 for each instance. We further normalize the attribution score by dividing the sum of the attribution scores at each layer to account for different layers' scales. As we can see in the plot, the attribution score for



(a) CLS attribution score across layers.

(b) SEP attribution score across layers.

Figure 4.3: Normalized attribution scores of special tokens across layers; dataset statistics under setting 2.

[CLS] across layers first decreases from layers 1–7, then goes up and fluctuates between layers 7–16, until finally increasing from layer 16 to 24. This differs from what attention analysis reveals in Kovaleva et al. [2019] and Clark et al. [2019], where they demonstrate that attention heads attend to [CLS] in earlier layers but attend to [SEP] in later layers. It's not contradictory, though, because we inspect feature importance with respect to the predicted output. Since [CLS] at the final layer is treated as a summary representation for the whole sentence to perform classification, it's intuitive that [CLS] is regarded as an important feature in the final layers.

What about the [SEP] tokens? Figure 4.3b shows the attribution score averaged between the two present [SEP] tokens—recall that BERT inserts two for every input. They become increasingly important with a certain amount of fluctuation from layer 1 to layer 16, after which [SEP]'s attribution scores drop, around the point where BERT starts to emphasize [CLS]. We combine the two [SEP] tokens because we find that both of them behave similarly, with the first [SEP] having a slightly higher attribution score. The possible explanation is that the first [SEP] is also responsible for identifying the boundary between the query and the passage, thus more important for reranking than the final [SEP]. Plots for separate [SEP]'s scores and weights can be seen in Appendix C.

Combining the above plots and the degradation tests across layers in Figure 4.1b, we conjecture that the [CLS] token initially serves as a classification prior to condition the tokens in the early layers (1–7) with [SEP] increasing participation. Then, BERT gathers more general

	# required	% required
(q, d_1)	8.9	10.39
(d_1)	18.52	22.49

Table 4.5: Truncation test on top-1 pair/passage.

Stats	Query	Passage
Average Length	9.2	74.8
Medium Length	9.0	69.0
Minimum Length	4.0	16.0
Maximum Length	44.0	214.0

Table 4.6: Stats of 1,000 relevant (q, d)

syntactic information [Hewitt and Manning, 2019], until layer 16, after which the `[CLS]` token slowly aggregates class-specific information and at layer 24 becomes the most important token for classification. Figure 4.1b (the full 24-layer degradation test is shown in Appendix D) echos the findings from previous work [Liu et al., 2019], demonstrating that the middle layers are the most informative ones for prediction. To be exact, layer 16 ($\frac{2}{3}$ of the total number of layers) is the most informative one in our experiment with BERT-large, the same fraction as what Jiang et al. [2020] find with BERT-base.

4.5.3 Truncation Test

Recent success in dense representation retrieval has made it possible to merge the two-stage process of ranking into an end-to-end single stage. However, the sacrifice of efficiency as well as the inability of doing term matching brings a flood of work about learning sparse representations [Bai et al., 2020; Formal et al., 2021] for the first-stage ranking. They learn the weight of tokens between (q, d) and tokens in the vocabulary for expansion and compression.

Although IBA doesn't serve as a weighting technique, given the attribution map we can still evaluate how dropping unimportant tokens affects the result. Empirically, we find that there is much room for tokens to be removed if the attribution map is reliable. Different from other downstream tasks, passage reranking usually involves scores for 1,000 passages to generate the final result. Instead of absolute scores for passages, we only care if relevant passages have higher scores than irrelevant passages.

Q3. How robust is the top-ranked passage?

Specifically, we want to know how many unimportant tokens we can remove before the top-1 passage falls to second place. Once again, we use the IBA-generated attribution map and then remove those tokens with the lowest attribution scores, until the ranking score for the top-1 passage drops below the second one. As in the reranking setting, the input is always a query–passage pair (q, d), and we have two experimental settings: (1) removing tokens that appear in both q and d; and (2) removing tokens that appear in only d. We include the result under both settings and report the truncated number, as well as the percentage needed in Table 4.5.

Surprisingly, even if BERT assigns an extreme score to the passage, making the score close to one another [Qiao et al., 2019], it still takes up to 22.5% tokens on average for top-1 passage to downgrade to the second place.

Obviously, removing tokens from the query quickly deteriorates the ranking score. Passages-only seems to have more redundant tokens that can be safely removed, even though sentences in the passage will become incomplete and broken after token removal. Note that this experiment removes only tokens of the top-1 passage. To further measure the trade-off between compression and quality, we do a truncation test for all passages. Specifically, given a $g\%$ of tokens kept for every single document, we measure final ranking performance—MRR score. The result is shown in Figure 4.4. From the result, we can see that truncating doc only is more robust than truncating both query and doc. Using attribution maps across different layers, on average, we have MRR $= 0.311$ with 90% tokens kept. But for the maximum, we can get MRR $= 0.392$ with 90% tokens, which is very close to the original score.

4.5.4 Positional Bias

Q4. Does BERT have positional bias?
To investigate if BERT has a positional bias, we first plot the position index of x_i where $\{x_i | x_i \in \mathbb{M} \cap d\}$. Specifically, we insert the bottleneck after every layer under experimental setting 2 — using 1000 relevant pairs of (q, d) — and then accumulate the count of each position index for all 24 layers. Statistics about randomly selected (q, d) are shown in Table 4.6. As we show in Figure 4.5a, tokens at the start of passages (e.g., position index from 0 to 20) have significantly higher occurrences than tokens appearing later.

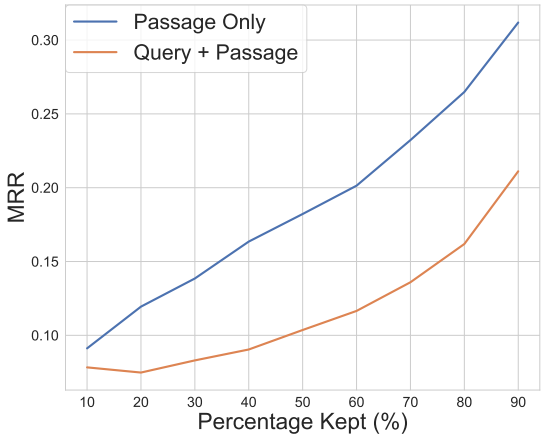
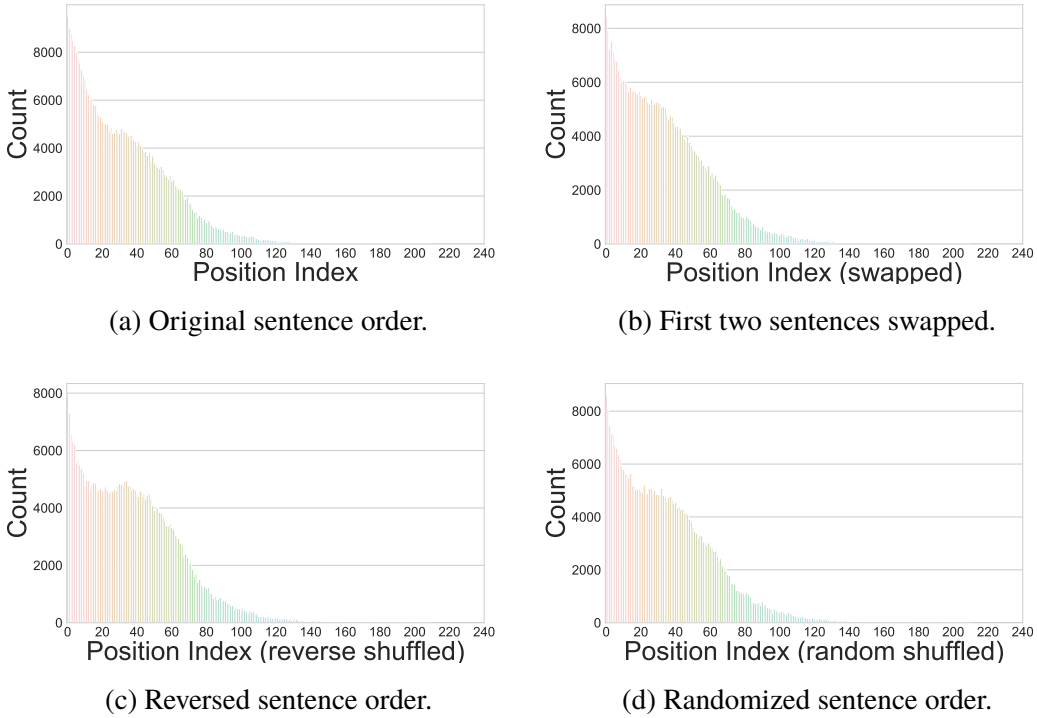


Figure 4.4: Average truncation MRR.

	Precision@1	MRR	MRR@10	Recall@3	Recall@50	Recall 1000
Original	0.276	0.411	0.403	0.427	0.817	0.848
Swapped	0.219	0.362	0.352	0.408	0.813	0.848
Randomized	0.200	0.343	0.332	0.384	0.803	0.848
Reversed	0.181	0.332	0.321	0.390	0.803	0.848

Table 4.7: Reranking metrics after changing sentence order.

We then conduct three controlled experiments: (1) swapping the first two sentences; (2) reversing the order of all sentences; and (3) randomizing the order of sentences. We maintain the order of within-sentence tokens in order to keep the discourse complete and coherent. The plots are shown in Figure 4.5. We see that, although changing the order results in more later-appearing tokens emphasized, the start of the passages still has incomparable dominance. To quantify the effect of swapping sentences, randomizing sentences, and reversing sentences, we calculate $p(\text{relevant}|\{\text{original, swap, random, reverse}\})$. We find that $p(\text{relevant}|\text{original}) = 0.939$, $p(\text{relevant}|\text{swap}) = 0.920$, $p(\text{relevant}|\text{random}) = 0.918$, $p(\text{relevant}|\text{reverse}) = 0.897$. The probability drops after every change of sentence order. The more the order changes, the more the probability drops (i.e., the negative effect is reversed order $>$ randomized $>$ swapped). Given that BERT assigns extreme reranking scores to most (q, d) pairs (e.g., scores are mostly either



(a) Original sentence order.

(b) First two sentences swapped.

(c) Reversed sentence order.

(d) Randomized sentence order.

Figure 4.5: Position index of important tokens in passages; stats of tested passages.

close to 0 or close to 1), it's unclear whether changing the sentence orders affects the final result.

Therefore, we also conduct experiments of changing the order sentence with the subset of the MS MARCO passage reranking dev set. We present these results in Table 4.7. Swapping sentences substantially deteriorates the result; randomizing and reversing the sentences further worsens the result.

The above experiments suggest that the sentence order in the passage carries high importance in reranking with BERT. Specifically, passages with the inverted pyramid structure would be preferred, as they present important information at the beginning of the passages.

We find that BERT prefers passages with important information emphasized at the beginning. But is this preference a real "bias"? Will it cause misjudgment because of emphasizing too much on the start of the passages? To answer this question, we design an experiment to see if passages with higher reranked scores (than the ground truth passages) also happen to get key

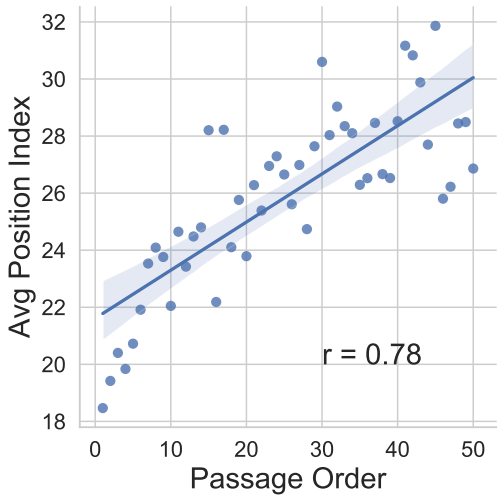


Figure 4.6: Positional index.

tokens emphasized earlier. Concretely, for those instances that have incorrectly ranked negative passages higher than the positive ones, we regard each token u_i in q as a query, and we find the position of a corresponding token v_j that appears in d where $u_i = v_j$. Then, we calculate the mean reciprocal rank for all v_j — MRR $= \frac{1}{|q \cap d|} \sum_{j=1}^{|q \cap d|} \frac{1}{\text{position}(v_j)}$. We then aggregate the MRR for all higher-ranked negative passages (HRNPs) and compare it with the MRR for the lower-ranked positive passages (LRPPs). When we aggregate by the "$\max$" function, we find that, in 86.2% of cases, HRNPs have higher MRR than LRPPs. Averaging all MRRs for HRNPs gives us 0.191, while it's 0.103 for averaging LRPPs. These numbers are 63.8%, 0.129, and 0.103, respectively if we aggregate by the arithmetic mean. We cannot say that the reason for those negative passages ranking higher is due to matched tokens appearing earlier, but we do note a correlation between HRNPs and early-appearing matched tokens.

Driven by this positional bias, we are also curious about how the positional index correlates with the passages' ranks. We compute the average positional index p_i for each document's top-20 most important tokens, and then average p_i for each query. As we show in Fig. 4.6, higher-ranked passages do have earlier tokens emphasized, meaning that passages with important tokens stressed earlier are preferred. When comparing the top-1 document returned by BERT d_{BERT} with the top-1 document returned by BM25 d_{BM25}, this preference also exists. We compute the MRR

across all tokens in the query and passages like we do in Section 4.5.4 for those passages that $d_{\text{BERT}} \neq d_{\text{BM25}}$ and d_{BERT} makes the correct prediction. We find that even BM25 is almost all about term matching, with $\mathbf{J}(q, d_{\text{BERT}}) = 0.062$, $\mathbf{J}(q, d_{\text{BM25}}) = 0.074$, considering the position, $\text{MRR}(q, d_{\text{BERT}}) = 0.127$ is still higher than $\text{MRR}(q, d_{\text{BM25}}) = 0.099$.

4.6 Summary

In this chapter, we use information bottleneck to examine BERT for reranking. We compare ranking mechanisms between BM25 and BERT, finding that BERT still values token matching, and it also learns deeper relevance between queries and passages. We further analyze special tokens across layers and demonstrate patterns that `[CLS]` aggregate evidence. We then investigate the robustness of top-ranked passages. Finally, we find that BERT fine-tuned on MS MARCO has a positional bias towards the start of the passage. In summary, attribution maps can explain models' predictions and serve well as an observation tool that helps us visualize patterns, resulting in improved hypothesis formulation and experimental design.

Chapter 5

Restrict Representations for Non-Parametric Learning by Inserting Bottleneck / Compression

In this chapter, we explore how to improve generalizability by inserting bottleneck. Most real-world problems that machine learning algorithms are expected to solve face the situation with (1) unknown data distribution; (2) little domain-specific knowledge; and (3) datasets with limited annotation. We approach these challenges by inserting bottleneck, which is indeed equivalent to doing compression as we will show in the chapter. Specifically, we propose Non-Parametric learning by Compression with Latent Variables (NPC-LV), a learning framework for any dataset with abundant unlabeled data but very few labeled ones. By only training a generative model in an unsupervised way, the framework utilizes the data distribution to build a compressor. Using a compressor-based distance metric derived from Kolmogorov complexity, together with few labeled data, NPC-LV classifies without further training. We show that NPC-LV outperforms supervised methods on all three datasets on image classification in the low data regime and even outperforms semi-supervised learning methods on CIFAR-10. We demonstrate how and when negative evidence lower bound (nELBO) can be used as an approximate compressed length for classification, which reveals the equivalence between inserting bottleneck and doing compression. By revealing the correlation between compression rate and classification accuracy, we illustrate

under NPC-LV how the improvement of generative models can enhance downstream classification accuracy. This work is presented in:

- Zhiying Jiang, Yiqin Dai, Ji Xin, Ming Li, Jimmy Lin. Few-Shot Non-Parametric Learning with Deep Latent Variable Model. *In Proceedings of the 36th Conference on Neural Information Processing Systems (NeurIPS) Spotlight, 2022*.

5.1 Overview

The progress of deep neural networks drives great success of supervised learning with huge labeled datasets [Gilmer et al., 2017; Simonyan and Zisserman, 2014; Sutskever et al., 2014]. However, large labeled datasets are luxurious in many applications and huge amounts of training parameters make the model easy to overfit and hard to generalize to other datasets. The urge to learn with a small labeled dataset prompts Few-Shot Learning (FSL). However, most few-shot classification settings require either an auxiliary "support set" [Edwards and Storkey, 2016; Snell et al., 2017a; Sung et al., 2018; Vinyals et al., 2016] that contains c classes, each with k samples (c-way k-shot); or prior knowledge about the dataset, where data augmentation can be performed within the same dataset [Hariharan and Girshick, 2017; Kwitt et al., 2016; Schwartz et al., 2018] or from other weakly-labeled/unlabeled/similar datasets [Douze et al., 2018; Gao et al., 2018; Pfister et al., 2014]. This setting is not widely applicable to every dataset in practice, as it requires either the elaborate construction of an additional "support set" or augmentation algorithms tailored to specific datasets. Pretrained models, on the other hand, do not require ad-hoc "support" and have proved to be good at few-shot learning [Brown et al., 2020] and even zero-shot learning [Puri and Catanzaro, 2019]. However, thousands of millions of training parameters make the model hard to be retrained but only fine-tuned. When the data distribution is substantially different from any datasets used in pretraining, the inductive bias from pretraining holds up fine-tuning, making the model less pliable [Yu et al., 2021].

Goals from the above learning paradigms can be summarized as to design algorithms that can be applied to any dataset and can learn with few labeled data, ideally with no training. "No Free Lunch" [Wolpert and Macready, 1997] implies that it's impossible to have an algorithm that is both "universal" and "best". But how good can a "universal" algorithm be, especially in the low

data regime, with no external data resources? Specifically, we are interested in a new setting, *Non-Supported Few-Shot Learning* (NS-FSL), defined as follows:
Given any target dataset $\mathbf{D} = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n)$ belonging to c classes. For each class, we have k labeled samples $(1 \leq k \leq 10)$. The remaining $n - ck$ unlabeled samples need to be classified into c classes without the need of support sets, any other datasets, or training parameters.

This setting is similar to the setting of semi-supervised learning but excludes labeled information in training. Ravi and Larochelle [2016] demonstrate that it's hard to optimize a neural network when labeled data is scarce. In order to make minimal assumptions about labeled data, we aim at using parameter-free methods. The goal is to grasp the data-specific distribution $p(\mathbf{x})$, with minimal premises on the conditional probability distribution $p(y|\mathbf{x})$. Deep generative models with explicit density estimation are perfect candidates for this goal. The problem then becomes: given trained generative models, how to take full advantage of the information obtained from them for classification? Using a latent representation only utilizes $p(\mathbf{z}|\mathbf{x})$, which just includes partial information. Even for those latent generative models that do not suffer from posterior collapse [Bowman et al., 2016], $p(\mathbf{z}|\mathbf{x})$'s insufficiency for classification with non-parametric methods like k-nearest-neighbor is shown in both previous works [Davidson et al., 2018; Zhao et al., 2019] and our experiments.

Inspired by previous works that use compressor-based distance metrics for non-parametric learning [Cebrián et al., 2005; Chen et al., 1999, 2004; Keogh et al., 2004b], we propose Non-Parametric learning by Compression with Latent Variables (NPC-LV), a learning framework that consists of deep-generative-model-based compressors and compressor-based distance metrics. It leverages the power of deep generative models without exploiting any label information in the probabilistic modeling procedure. With no further training, this framework can be directly used for classification. By separating probabilistic modeling from downstream tasks that require labels, we grasp the unique underlying structures of the data in every dataset and further utilize these structures in downstream tasks with no parameter tuning needed. We view this learning framework as a *baseline* in this setting, for any dataset. We argue it is "parameter-free" as there are no parameters involved in the classification stage for labeled data. Basically, it means training a generative model as is and getting a classifier for free.

Our contributions are as follows: (1) We frame existing methods into a general learning framework NPC, based on which we derive NPC-LV, a flexible learning framework with replaceable

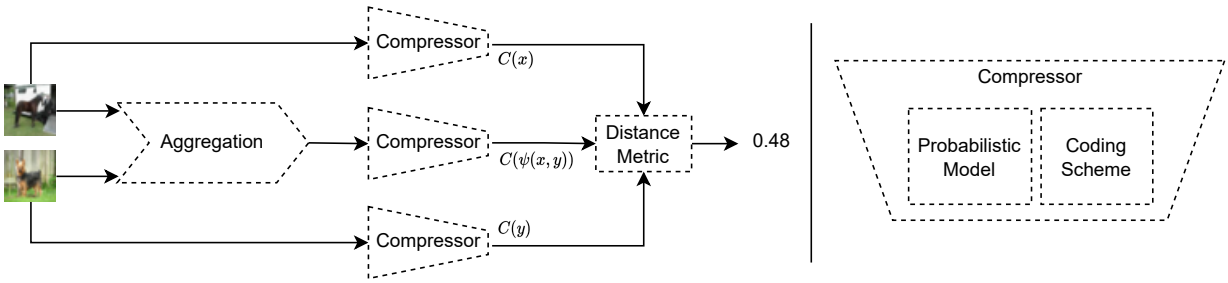


Figure 5.1: NPC framework with trainable deep probabilistic models. Replaceable modules are indicated with dashed lines.

modules. (2) We use NPC-LV as a baseline for a common learning scenario NS-FSL with neither support sets nor further training. (3) Our method outperforms supervised methods by up to 11.8% on MNIST, 18.0% on FashionMNIST, and 59% on CIFAR-10 on image classification in the low data regime. It outperforms non-parametric methods using a latent representation on all three datasets. It even outperforms semi-supervised learning methods on CIFAR-10. (4) We show how negative evidence lower bound (nELBO) can be used for classification under this framework. (5) We find the correlation between bitrate and classification accuracy. This finding suggests how improvements in the domain of deep-learning-based compressors can further boost classification accuracy under this framework.

5.2 Non-Parametric learning by Compression with Latent Variables

Non-Parametric learning by Compression (NPC) consists of three modules — a distance metric, a compressor, and an aggregation method shown in Figure 5.1. NPC-LV leverages NPC by including neural compressors based on deep generative models. We introduce the equivalence between inserting bottleneck and doing compression in Section 5.2.1; derivation of compressor-based distance metrics in Section 5.2.2; an integration of this framework with generative models in Section 5.2.3.

5.2.1 Trained Generative Models as Compressors

We approach Non-Supported Few-Shot Learning also through restricting the representation. In Chapter 3 and Chapter 4, we showed that IBA utilizes "noise" to restrict the information flow through the network. A more straightforward way to restrict the information is to use a representation with lower dimensions. This essence reflects in the auto-encoders [Rumelhart et al., 1985], where a "bottleneck" is inserted in the architecture in order to learn a compressed latent representation. Sharing similar architecture, VAE [Kingma and Welling, 2013] restricts the distribution family of the latent variable so that it is able to learn the latent representation and do generation at the same time. The power of explicit density estimation can reveal the underlying structure of the datasets without exploiting any labeled data. In other words, we grasp the data-specific distribution $p(\mathbf{x})$, with minimal premises on conditional probability distribution $p(y|\mathbf{x})$.

The relation between VAE and "bits-back" has been revealed in multiple previous works [Chen et al., 2016; Honkela and Valpola, 2004]. Townsend et al. [2019b] use latent variable models by connecting Asymmetric Numeral Systems (ANS) [Duda, 2009] to the "bits-back" argument [Frey and Hinton, 1997] (BB-ANS). In the setting of the "bits-back argument", we assume *Alice* has some extra bits of information to send to *Bob* alongside $\mathbf{x}$. It's also assumed that both *Alice* and *Bob* have access to $p(\mathbf{z})$, $p_\theta(\mathbf{x}|\mathbf{z})$ and $q_\phi(\mathbf{z}|\mathbf{x})$ where $\mathbf{z}$ is the latent variable; $p(\mathbf{z})$ is the prior distribution of $\mathbf{z}$; $p_\theta(\mathbf{x}|\mathbf{z})$ represents a generative network and $q_\phi(\mathbf{z}|\mathbf{x})$ represents an inference network. As shown in Figure 5.2, *Alice* first decodes the extra information according to $q_\phi(\mathbf{z}|\mathbf{x})$ to generate a sample $\mathbf{z}$.[1] $\mathbf{z}$ is further used to encode $\mathbf{x}$ with $p_\theta(\mathbf{x}|\mathbf{z})$ and $\mathbf{z}$ itself is encoded using $p(\mathbf{z})$. *Bob* then reverses this procedure and recovers the extra bits by encoding with $q_\phi(\mathbf{z}|\mathbf{x})$. For a single data point, the length of the final bitstream is:

$$N = n_{\text{extra}} + \log q_\phi(\mathbf{z}|\mathbf{x}) - \log p_\theta(\mathbf{x}|\mathbf{z}) - \log p(\mathbf{z}). \tag{5.1}$$

We can see the expectation of $N - n_{\text{extra}}$ is equal to the negative evidence lower bound (nELBO):

$$\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[N - n_{\text{extra}}] = -\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \log \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} = -\text{ELBO} \tag{5.2}$$

[1] Note that "encoding", "decoding" here follows data compression's convention instead of variational autoencoder's.

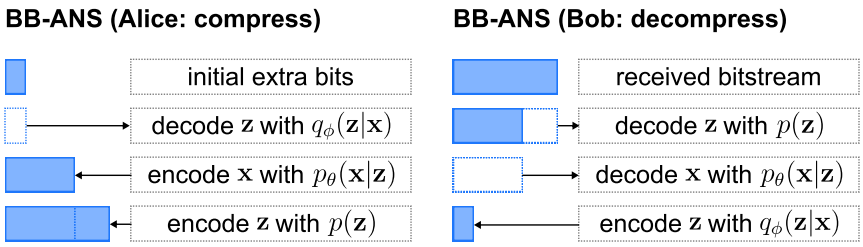


Figure 5.2: BB-ANS compress & decompress

ELBO above is derived from the "bits-back argument" in the context of compression. Now, from the perspective of latent variable models like VAE, the derivation often starts from the fact that $p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z})$ is intractable. $q_\phi(\mathbf{z}|\mathbf{x})$ is then introduced as an inference model to approximate $p(\mathbf{z}|\mathbf{x})$ in order to work around the intractability problem, which brings up the marginal log-likelihood:

$$\begin{aligned} \log p_\theta(\mathbf{x}) &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \log \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p(\mathbf{z}|\mathbf{x})}, \\ \text{ELBO} &= \log p_\theta(\mathbf{x}) - D[q_\phi(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z}|\mathbf{x})]. \end{aligned} \tag{5.3}$$

We only need to optimize the lower bound, as minimizing nELBO means maximizing $\log p_\theta(\mathbf{x})$ — the likelihood of generating real data and minimizing KL divergence between $q_\phi(\mathbf{z}|\mathbf{x})$ and $p(\mathbf{z}|\mathbf{x})$ at the same time, which is the same objective function from what we derive using "bits-back".

This equivalence demonstrates that an optimized latent variable model can be used directly for compression as, from the data compression perspective, it minimizes the code length attainable by bits-back coding using the model. With the help of ANS, we can encode symbols into bitstreams or decode bitstreams back to symbols with trained latent variable models.

Our experiments and discussion are mainly about VAE-based compressors as their architectures can be flexibly changed under the "bits-back" argument. But this doesn't limit our framework to VAE-based compressors only. We introduce some other possible generative-model-based neural compressors below.

ARM: Autoregressive models (ARMs) model $p(\mathbf{x})$ as: $p(\mathbf{x}) = p(x_0)\prod_{i=1}^{n} p(x_i|\mathbf{x}_{i-1})$. The exact likelihood estimation makes it capable of lossless compression. But instead of using ANS, which is a stack-like coding scheme, queue-like ones (e.g., Arithmetic Coding (AC) [Witten et al., 1987b]) should be used. Computational inefficiency is the main drawback of ARMs such as RNN [Rumelhart et al., 1985], but causal convolutions [Van den Oord et al., 2016a,b] can alleviate the problem.

IDF: Integer Discrete Flow (IDF) [Hoogeboom et al., 2019] can also optimize towards the exact log-likelihood. Similar to other flow-based models, it utilizes an invertible transformation f, but works on discrete distributions with additive coupling and rounding operations. For IDF, ANS can be used as the entropy coder.

5.2.2 Compressor-based Distance Metric

As we introduce in Section 2.2.2, *Normalized Information Distance* (NID) is a universal metric that can measure the similarity between two objects. Universal as NID is, the uncomputability of Kolmogorov complexity renders NID uncomputable. Cilibrasi and Vitányi [2005] propose *Normalized Compression Distance* (NCD), a quasi-universal distance metric based on real-world compressors. In this context, $K(x)$ can be viewed as the length of x after being maximally compressed. Suppose we have $C(x)$ as the length of compressed x produced by a real-world compressor, then NCD is defined as:

$$\text{NCD}(x,y) = \frac{C(xy) - \min\{C(x), C(y)\}}{\max\{C(x), C(y)\}}. \tag{5.4}$$

The better the compressor is, the closer NCD approximates NID. With a *normal* compressor (discussed in detail in Section 5.5.6), NCD has values in [0,1] and satisfies the distance metric (in)equalities up to $O(\log n/n)$ where n means the maximum binary length of a string involved [Vitányi et al., 2009]. NCD is thus computable in that it not only uses compressed length to approximate $K(x)$ but also replaces conditional Kolmogorov complexity with $C(xy)$ that only needs a simple concatenation of x, y. Li et al. [2004] simplify NCD by proposing another *compression-based dissimilarity measurement* (CDM):

$$\text{CDM}(x,y) = \frac{C(xy)}{C(x) + C(y)}. \tag{5.5}$$

Chen et al. [2004] use another variation ranging from 0 to 1:

$$\text{CLM}(x, y) = 1 - \frac{C(x) + C(y) - C(xy)}{C(xy)}. \tag{5.6}$$

NCD, CDM and CLM are different variations of Kolmogorov-based distance metrics. We empirically evaluate their performance in Section 5.5.1.

5.2.3 NPC-LV

We've shown in Section 5.2.1 that we can plug in any trained latent variable model in exchange for a near-optimal compressor under the framework of BB-ANS. To show that the coding scheme is replaceable, we introduce a coding scheme variation, Bit-Swap [Kingma et al., 2019a], in the following experiments. The difference between BB-ANS and Bit-Swap is the encoding and decoding order when there is more than one latent variable. A detailed comparison is shown in Appendix F. The generative model we use for both BB-ANS and Bit-Swap is a hierarchical latent generative model (details are in Appendix E), also known as Deep Latent Gaussian Model (DLGM) [Rezende et al., 2014].

Aggregation: In addition to coding schemes and probabilistic models for compressors, aggregation methods can also be replaced. Previous works [Cilibrasi and Vitányi, 2005] assume xy in $C(xy)$ means the "concatenation" of two inputs. We expand this assumption to a more general case where "aggregation" can be another kind of aggregation function, represented as $C(\psi(x, y))$ in Figure 5.1. We justify this generalization as changing aggregations methods may make compressor-based distance metrics *not admissible* (discussed in Section 5.5.6). More sophisticated strategies of aggregation are left for future work. We also discuss other replaceable modules in detail in Section 5.5.5.

We use BB-ANS with NCD as a concrete instance to demonstrate this framework on a classification task shown in Algorithm 1. $\mathbf{D}^{U}_{\text{train}}$ and $\mathbf{D}^{L}_{\text{train}}$ mean the unlabeled and labeled training sets; predefined functions are in teal. The algorithm can be simplified into four steps: (1) Train a VAE on the unlabeled training dataset; (2) Apply ANS with discretization on the trained VAE to get a compressor; (3) Calculate the distance matrix between pairs $(\mathbf{x}_{\text{test}}, \mathbf{x}_{\text{train}})$ with the compressor and NCD; (4) Run k-Nearest-Neighbor(kNN) classifier with the distance matrix.

Algorithm 1 NPC-LV (use VAE and NCD as an example)

Input: k, $\mathbf{D}_{\text{test}}$,$\mathbf{D}_{\text{train}} = \{\mathbf{D}^U_{\text{train}}, \mathbf{D}^L_{\text{train}}\}$, $\mathbf{D}^L_{\text{train}} = \{X^L, y^L\}$
trained_vae = trainVAE($\mathbf{D}^U_{\text{train}}$)
vae_compressor = ANS(trained_vae)
for $\mathbf{x}_{\text{test}}$ in $\mathbf{D}_{\text{test}}$ **do**
  C_$\mathbf{x}_{\text{test}}$ = len(vae_compressor($\mathbf{x}_{\text{test}}$))
  distances = []
  for $\mathbf{x}_{\text{train}}$ in X^L **do**
    // Calculate NCD distance with $C(x)$, $C(y)$, and $C(\psi(x, y))$
    C_$\mathbf{x}_{\text{train}}$ = len(vae_compressor($\mathbf{x}_{\text{train}}$))
    C_agg = len(vae_compressor(aggregate($\mathbf{x}_{\text{test}}$, $\mathbf{x}_{\text{train}}$))
    NCD = (C_agg $-$ min{C_$\mathbf{x}_{\text{test}}$, C_$\mathbf{x}_{\text{train}}$}) / max{C_$\mathbf{x}_{\text{test}}$, C_$\mathbf{x}_{\text{train}}$}
    distances = push(NCD, distances)
  end for
  k_nearest_indicies = argsort(distances)[:k]
  y_{test} = majority($\{y^L_i, i \in$ k_nearest_indicies$\}$)
end for

nELBO as estimated compressed length: Specifically for VAE-based compressors, nELBO is the estimated length of the compressed bitstream as Equation (5.2) shows. Therefore, we can use it directly without actual compression. This can further simplify our method as we don't need to discretize continuous distributions or apply an entropy coder.

The reason why the underlying data distribution can help the classification is based on the *manifold assumption*, which is a common assumption in SSL [Van Engelen and Hoos, 2020]. It states that the input space consists of multiple low-dimensional manifolds, and data points lying on the same manifold have the same labels. This assumption helps alleviate the curse of dimensionality and may explain the effectiveness of using kNN with few labeled data. Due to the fact that our training process does not use labeled data, our method does not rely on other common assumptions in SSL like the *smooth assumption* and the *low-density assumption*. The fact that NPC-LV makes very few assumptions about datasets and that compressors are data-type-agnostic makes this framework extensible to other data types beyond images. For example, the

combination of an autoregressive model (e.g., character recursive neural network) and arithmetic coding [Goyal et al., 2019b] can be used in our framework for sequential data.

5.3 Related Work

5.3.1 Non-parametric learning with Information Distance

Bennett et al. [1998] propose *information distance* as a universal metric, based on which several papers [Grumbach and Tahi, 1994; Yianilos, 2002] propose more fine-grained distance metrics. Chen et al. [2004]; Cilibrasi and Vitányi [2005]; Li et al. [2004] derive more practical distance metrics based on real-world compressors. Empirical results [Cebrián et al., 2005; Chen et al., 1999, 2004; Keogh et al., 2004b] show that even without any training parameters, those compressor-based distance metrics can produce an effective distance matrix for clustering and classification on time series datasets. Cilibrasi and Vitányi [2005] further push this direction to more types of datasets, including images that are represented in "#" (black pixel) and "." (white pixel). We unify previous work in the NPC framework, expand it to real image datasets and leverage it with neural compressors.

5.3.2 Compression

Shannon [1948] establishes source coding theorems, showing that entropy rate is the limit of code rate for lossless compression. Huffman Coding [Huffman, 1952] achieves the optimal symbol code whose length is upper-bounded by $H(\mathbf{x}) + 1$ per symbol. The 1-bit overhead is due to the fact that $-\log p(x)$ is not always an integer. Stream coding techniques like AC [Witten et al., 1987b] and ANS [Duda, 2009] further optimize by representing the whole message with numeral systems. These entropy coders then can be combined with probabilistic modeling using neural networks [Goyal et al., 2019b; Mahoney, 2000; Schmidhuber and Heil, 1996] and used in our framework.

5.3.3 Semi-Supervised Learning with VAE

The evaluation paradigm in this chapter is closest to Semi-Supervised Learning (SSL). Kingma et al. [2014] design two frameworks for utilizing autoencoders in downstream classification tasks. The first (M1) is to train a tSVM [Vapnik, 1999] with latent representation output by a trained VAE. The second (M2) is to train a VAE with labels as another latent variable. M1 only requires a standard VAE but tSVM suffers from optimization difficulty [Collobert et al., 2006], making it hard to be generally applicable for VAE. More recent VAE-based methods [Joy et al., 2020; Maaløe et al., 2016] are built on M2. These methods don't train a generative model in an unsupervised way as we do.

5.3.4 Few-Shot Learning

Similar to our setting, FSL also targets the low-labeled data regime. Many previous papers [Edwards and Storkey, 2016; Snell et al., 2017a; Sung et al., 2018; Vinyals et al., 2016] on FSL are based on meta-learning, where the model is fed with an extra labeled support set, in addition to the target dataset. Another line of work [Douze et al., 2018; Gao et al., 2018; Hariharan and Girshick, 2017; Kwitt et al., 2016; Pfister et al., 2014; Schwartz et al., 2018] utilize data augmentation. Although some of them do not require extra datasets [Hariharan and Girshick, 2017; Kwitt et al., 2016; Schwartz et al., 2018], the augmentation algorithms can hardly be applied to every other dataset [Wang et al., 2020b]. Metric-based methods [Koch et al., 2015] utilize distance metrics for FSL. But instead of modeling the probability distribution of a dataset, they model the "distance" between any pair of data points with a neural network, which still requires many labeled data during "pretraining". Our work is similar to metric-based methods in that both have the essence of nearest-neighbor. The difference is that in the "pretraining" stage, our model is not trained to learn the distance but to reconstruct the image as all standard generative models do. More importantly, we use no labeled data in this stage.

5.4 Experimental Setup

We compare our method with supervised learning, semi-supervised learning, non-parametric learning, and traditional Non-Parametric learning by Compression (NPC) on MNIST, FashionMNIST, and CIFAR-10 [Krizhevsky et al., 2009; LeCun and Cortes, 2010; Xiao et al., 2017]. For each dataset, we first train a hierarchical latent generative model with *unlabeled* training sets. During the stage of calculating the distance metric using compression, we pick 1,000 samples from the test set, due to the cost of compression and pair-wise computation, together with $n = \{5, 10, 50\}$ labeled images per class from the training set. We also report the result for $n = 50$ although it is beyond our setting. We keep the selected dataset the same for every method compared. We use **bold** to highlight the cases we outperform supervised methods, use underline to highlight the case we outperform SSL and use *italic* to highlight the highest accuracy among all methods for reference. We perform compression with two coding schemes (BB-ANS and Bit-Swap) and also use nELBO for compressed length directly.

The training details for both our method and other baseline models are as follows. For both CNN and VGG, we tune hyperparameters on a validation set to obtain the optimized performance. We use VGG11 instead of VGG16 or VGG19 because VGG11 performs the best in the low data regime in our experiments. For CNN, we first normalize MNIST, FashionMNIST, and CIFAR-10. We use batch size $= 4$, epoch number $= 14$, Adadelta [Zeiler, 2012] as the optimizer with learning rate $= 1$, decaying learning rate by $\gamma = 0.7$ every step for MNIST and FashionMNIST. We use the same hyperparameters as PyTorch's official MNIST example[2] do except we increase the number of epochs to 20. For VGG11 on MNIST and FashionMNIST, we use epoch number $= 20$ when given 50 samples per class, and use epoch number $= 40$ when given less. We use Adam [Kingma and Ba, 2015] with learning rate $= 0.0001$ as the optimizer, and batch size $= 4$. For CIFAR-10, we use learning rate $= 0.00001$, epoch number $= 80$.

For MeanTeacher and VAT, we follow Zhang et al. [2021a][3] and hyperparameter settings. We use WideResNet [Zagoruyko and Komodakis, 2016] with depth $= 28$ and widen factor $= 2$ as the architecture for all three datasets and iterate 60,000 steps for MNIST and FashionMNIST, and iterate 200,000 steps for CIFAR-10. SGD with momentum is used for all three datasets, with

[2]https://github.com/pytorch/examples/tree/master/mnist
[3]https://github.com/TorchSSL/TorchSSL

learning rate $= 0.03$, momentum $= 0.9$.

For latent variable models used in this chapter, we follow the training procedure and hyper-parameters in Kingma et al. [2019a] — four 'Processing' Residual blocks at the beginning of the inference model and the end of the generative model; eight 'Ordinary' Residual blocks in total for all latent layers in both inference model and generative model. The Dropout [Srivastava et al., 2014] rate is 0.2 for MNIST and FashionMNIST, 0.3 for CIFAR-10. The learning rate for all datasets is 0.002 with the Adam optimizer. The dimension of latent variables for MNIST and FashionMNIST is $1 \times 16 \times 16$ while the dimension of latent variables for CIFAR-10 is $8 \times 16 \times 16$. During kNN, we use $k = 2$ for MNIST and FashionMNIST and $k = 3$ for CIFAR-10.

We use one NVIDIA Tesla P40 GPU for training and compression. For pairwise computation in a 50-shot setting, it takes roughly ten hours to calculate the distance matrix on MNIST and FashionMNIST; it takes about thirty hours for CIFAR10. But once the distance matrix is calculated, evaluation on 5-shot or 10-shot just takes seconds. For both CNN and VGG, it takes about half an hour to train on a 50-shot for one experiment. For VAT and MT, we need to re-train for every shot setting. For MNIST and FashionMNIST it takes about three hours to run one experiment in a single shot setting and for CIFAR10 it takes about twelve hours. The time that VAT and MT take positively relates to the number of iterations, which makes them slower than our method in the 5- and 10-shot setting but faster in the 50-shot setting.

5.5 Results and Analysis

5.5.1 Few-Shot Image Classification

Comparison with Supervised Learning: Supervised models are trained on $10n$ labeled data. In Table 5.1, when $n = 50$, CNN and VGG surpass NPC-LV. In the cases where the number of labeled data is extremely limited (e.g., $5, 10$ labeled data points per class), however, the BB-ANS variant outperforms all methods in every dataset. For MNIST, both BB-ANS and Bit-Swap produce more accurate results than supervised methods on 5-shot experiments; BB-ANS performs slightly better than the supervised methods in the 10-shot scenario. For FashionMNIST, all three variants outperform in 5-shot, 10-shot, and even 50-shot settings. For CIFAR-10, given 10 la-

Data	MNIST			FashionMNIST			CIFAR-10		
#Shot	5	10	50	5	10	50	5	10	50
Supervised Learning									
SVM	69.4±2.2	77.1±1.5	87.6±0.4	67.1±2.1	71.0±1.6	78.4±0.5	21.1±1.9	23.6±0.5	27.2±1.2
#Param	35,280			35,280			105,840		
CNN	72.4±3.5	83.7±2.6	93.2±2.8	67.4±1.9	70.6±2.5	80.5±0.7	23.4±2.9	28.3±1.9	38.7±1.9
#Param	1,199,882			1,199,882			1,626,442		
VGG	69.4±5.7	83.9±3.2	94.4±0.6	62.8±4.1	70.5±4.5	81.5±1.1	22.2±1.6	29.7±1.8	42.6±1.2
#Param	28,148,362			28,148,362			28,149,514		
Semi-Supervised Learning									
VAT	*97.0±0.3*	*97.4±0.1*	*98.4±0.1*	*74.1±0.8*	*78.4±0.3*	*87.1±0.2*	25.4±2.0	27.8±4.2	60.9±6.1
#Param	1,469,354			1,469,354			1,469,642		
MT	78.4±2.0	82.8±1.9	98.6±0.2	58.1±2.8	70.8±0.8	*87.1±0.1*	31.7±1.5	35.9±1.1	*64.3±1.6*
#Param	1,469,354			1,469,354			1,469,642		
Non-Parametric Learning									
Single	65.6±1.2	76.8±0.8	86.3±0.3	40.2±1.4	53.4±1.1	70.0±0.4	17.3±0.9	19.2±0.7	23.4±0.3
#Param	0			0			0		
Hier	73.6±3.1	82.3±2.1	90.4±1.4	69.5±3.5	72.5±1.9	78.7±1.3	22.2±1.6	24.2±4.9	26.2±2.9
#Param	0			0			0		
Non-Parametric learning by Compression with Latent Variables (NPC-LV)									
nELBO	**75.2±1.5**	81.4±1.1	91.0±1.0	**72.2±2.2**	**76.7±1.5**	**85.6±1.1**	**34.1±1.8**	**34.6±2.0**	35.6±2.5
#Param	0			0			0		
Bit-Swap	**75.7±3.6**	83.3±0.9	90.9±0.2	**73.5±3.7**	**76.0±1.4**	**82.6±1.2**	**32.2±3.5**	**32.8±1.9**	35.7±1.1
#Param	0			0			0		
BB-ANS	**77.6±0.4**	**84.6±2.1**	91.4±0.6	***74.1±3.2***	**77.2±2.2**	**83.2±0.7**	***35.3±2.9***	***36.0±1.8***	37.4±1.2
#Param	0			0			0		

Table 5.1: Test accuracy of methods with number of learnable parameters for classification. #Shot refers to the the number of training samples per class. Results report means and the 95% confidence interval over five trials. Note that "#Param" refers to parameters specifically for supervised training.

beled data points per class, NPC-LV surpasses the accuracy of CNN by 27.2% and improves the accuracy of VGG by 21.2%. This enhancement is more significant in the 5-shot setting: NPC-LV improves the accuracy of CNN by 50.9% and by 59.0% for VGG. In general, we can see as the labeled data become fewer, NPC-LV becomes more advantageous.

Comparison with Semi-Supervised Learning: The input of NS-FSL is similar to SSL in that both unlabeled data and labeled data are involved. The difference lies in the fact that (1) our training *doesn't* use any labeled data and is purely unsupervised; (2) we only need to train the model *once*, while SSL needs to retrain for different n; (3) we use fewer labeled data points, which is a more practical setting for real-world problems. We choose strong semi-supervised methods that make few assumptions about the dataset. We use consistency regularization methods instead of pseudo-labeling ones as pseudo-labeling methods often assume that the decision boundary should pass through a low-density region of the input space (e.g., Lee et al. [2013]). Specifically, we choose MeanTeacher ("MT") [Tarvainen and Valpola, 2017] and VAT [Miyato et al., 2018]. The core of both is based on the intuition that realistic perturbation of data points shouldn't affect the output. We train both models with n labeled samples per class together with an unlabeled training set. As we can see, NPC-LV achieves higher accuracy for CIFAR-10 in the low data regime, has a competitive result on FashionMNIST, and is much lower on MNIST. The strength of our method is more obvious with more complex datasets. It's a surprising result because we do not implement any data augmentation implicitly or explicitly unlike consistency regularization methods, which utilize data perturbation and can be viewed as data augmentation. It's worth noting that on all three datasets, our method using BB-ANS always outperforms ***at least one*** semi-supervised methods on the 10-shot setting, indicating that SSL methods trade "universality" for "performance" while our method is more like a baseline. Speed-wise, NPC-LV only requires training once for the generative model and can run kNN on different shots (n) with no additional cost. In contrast, SSL methods require the whole pipeline to be retrained for every n.

Comparison with Non-Parametric Learning: In this experiment, we explore the effectiveness of using latent representations directly with kNN. We train the same generative model we use in NPC-LV ("Hier"), as well as a vanilla VAE with a single latent variable ("Single"). Table 5.1 shows that the latent representation of the vanilla one is not as expressive as the hierarchical one. Although the latent representation using the hierarchical architecture performs reasonably well and surpasses supervised methods in the 5-shot setting on MNIST and FashionMNIST, it's still

	MNIST			FashionMNIST			CIFAR-10		
	NCD	CLM	CDM	NCD	CLM	CDM	NCD	CLM	CDM
gzip	86.1	85.6	85.6	81.7	82.6	82.6	31.3	30.3	30.3
bz2	86.8	86.4	86.4	81.7	79.0	79.0	28.0	27.5	27.5
lzma	87.4	88.5	88.5	80.6	82.7	82.7	31.4	30.0	30.0
WebP	86.4	87.9	87.9	69.9	67.3	67.3	33.3	34.2	34.2
PNG	86.8	89.1	89.1	74.8	76.9	76.9	32.2	28.9	28.9
BitSwap	93.2	90.9	93.2	84.3	*84.0*	*84.0*	36.9	36.9	36.9
BBANS	*93.6*	*93.4*	*93.4*	*84.5*	83.6	83.6	*40.2*	*40.8*	*40.8*

Table 5.2: Classification accuracy using different compressors and distance metrics.

significantly lower than NPC-LV in all settings. The result suggests that NPC-LV can utilize trained latent variable models more effectively than simply utilizing latent representations for classification.

Comparison with NPC: We investigate how Non-Parametric learning by Compression (NPC) with non-neural compressors performs with different distance metrics. We evaluate with NCD, CLM, and CDM as distance metrics, and gzip, bz2, lzma, WebP, and PNG as compressors, using 1,000 images from the test set and 100 samples per class from the training set. The result is shown in Table 5.2. For distance metrics, we can see CLM and CDM perform similarly well but it's not clear under what circumstances a distance metric is superior to the rest. For compressors, both Bit-Swap and BB-ANS perform much better than other compressors, indicating that generative-model-based compressors can significantly improve NPC. BB-ANS turns out to be the best compressor for classification on all three datasets.

5.5.2 nELBO as Compressed Length

As we've shown in Section 5.2.1, nELBO can be viewed as the expected length of the compressed bitstream $N - n_{\text{extra}}$. Thus, theoretically, it can be used directly to approximate compressed length. In this way, we don't need to apply ANS to VAE for the actual compression, which largely simplifies the method and boosts speed. However, as we can see in Table 5.1, using nELBO doesn't always perform better than an actual compressor like BB-ANS. This may be because nELBO in a

	5-shot	10-shot	50-shot
MNIST	4.8%	1.9%	-0.1%
FashionMNIST	22.4%	12.3%	3.9%
CIFAR-10	55.8%	38.1%	6.3%

Table 5.3: NPC-LV's excess rate compared with the average of all other methods.

well-trained model regards the aggregation of two images as out-of-distribution data points; while the discretization in the actual compressor forces close probability with a certain level of precision to be discretized into the same bin, lowering the sensitivity. Better aggregation strategies need to be designed to mitigate the gap.

5.5.3 Performance Gain and Task Difficulty

We find that our method is more advantageous on more complex datasets with a lower shot number in terms of the relative performance. Specifically, we average all methods' accuracy that NPC-LV compared across different shot settings and datasets denoted as $\bar{a}$. We then calculate the excess rate with BB-ANS variant's accuracy b by $\frac{b-\bar{a}}{\bar{a}} \times 100\%$. The result is shown in Table 5.3. The lower left part of the table represents higher task difficulty. As the shot number decreases (from right to left) and/or as the difficulty of the dataset increases (from top to bottom), our framework's performance enhancement gets higher.

5.5.4 Bitrate versus Classification Accuracy

The origin of the NPC framework comes from the intuition that the length of x after being maximally compressed by a real-world compressor is close to $K(x)$. Theoretically, the closer this length approximates the *minimum* length of the expression ($C(x) \approx K(x)$), the closer the compressor-based distance metrics are to the *normalized information distance*. We investigate, empirically, whether the bitrate actually reflects the classification accuracy. We plot bitrate versus classification accuracy for each compressor in Table 5.2 on three datasets as shown in Figure 5.3. We use the net bitrate, which is $(N - n_{\text{extra}})/d$, where N is the length of the compressed bitstream, n_{extra} is the length of the extra bits, and d is the number of pixels. As we can see, a very

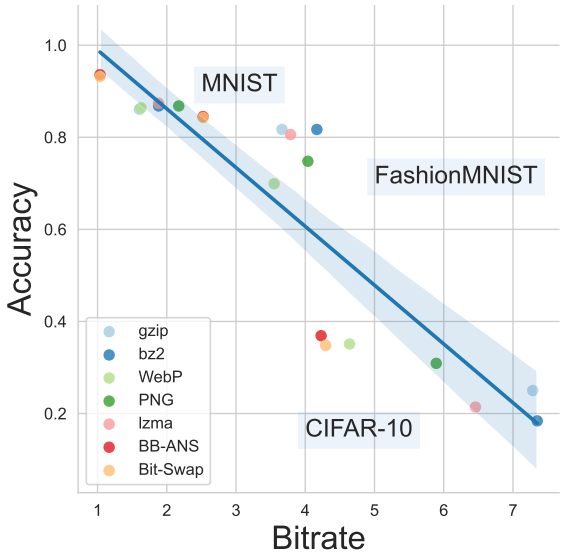


Figure 5.3: Bitrate versus Classification Accuracy

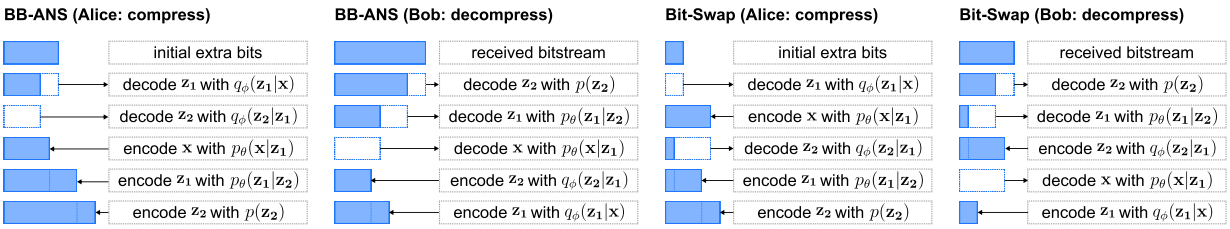


Figure 5.4: BB-ANS V.S. Bit-Swap

strong monotonic decreasing correlation between bitrate and accuracy emerges, with Spearman coefficient [Spearman, 1987] $r_s = -0.96$, meaning the lower the bitrate is, the higher the classification accuracy is. This means the correlation between bitrate and classification accuracy holds empirically regardless of datasets. It will be interesting to investigate in the future whether the correlation remains for lossy compression.

5.5.5 Ablation Study

Number of Latent Variables: The difference between how BB-ANS [Townsend et al., 2019b] and Bit-Swap [Kingma et al., 2019a] compress is shown in Figure 5.4. We can see these two compressors are only different when there is more than one latent variable, and by utilizing encoded bitstreams from previous latent variables, Bit-Swap can reduce the cost of initial bits significantly. We carry out experiments on CIFAR-10 and keep the aggregation method constant. We can see in Table 5.5, Bit-Swap performs better with more latent variables. But Bit-Swap with eight latent variables is still worse than BB-ANS with two.

Aggregation Methods: We also evaluate how different aggregation methods affect the classification accuracy, shown in the right-hand side of Table 5.5. We aggregate two images by "average", "'minimum", "maximum", "concatenation", and "greyscale+average". For "concat", the actual operation is that we compress one image after another in a way that the compressed bitstream of the first image could be used as the "extra" bits for the second image. "gs+avg" means we use the greyscale of the image and then average pixel values. Obviously, "greyscale" is not a way of aggregation, but we notate it in the table this way for simplicity and comparison, and also to stress that this operation is only used during compression. That is, the generative model is trained on the original images instead of on greyscale images. By applying simple image processing methods during compression, we find that combining aggregation with image manipulation can be effective as no compressor needs to be changed, and thus no retraining for generative models is needed. We also plot their bitrate vs. classification accuracy for various aggregation methods. Table 5.4 shows "concat" for images is an outlier - lower bitrate with low accuracy. We will show in Section 5.5.6 that "concat" disqualifies BB-ANS and Bit-Swap from being a *normal* compressor. Please note the special case of "concat" doesn't apply to other compressors like gzip which treat images as bytes in the first place.

Bit-Swap [Kingma et al., 2019a] achieves the new state of the art bitrate but why doesn't it surpass BB-ANS for classification? One of the reasons that Bit-Swap achieves a better compression rate than BB-ANS is because Bit-Swap requires fewer initial bits, details shown in Appendix F. However, initial bits can only be amortized when multiple data points need to be compressed. But in our application, at most two data points need to be compressed sequentially. This leads us to choose the net bitrate, which excludes the length of the initial bits, and makes Bit-Swap

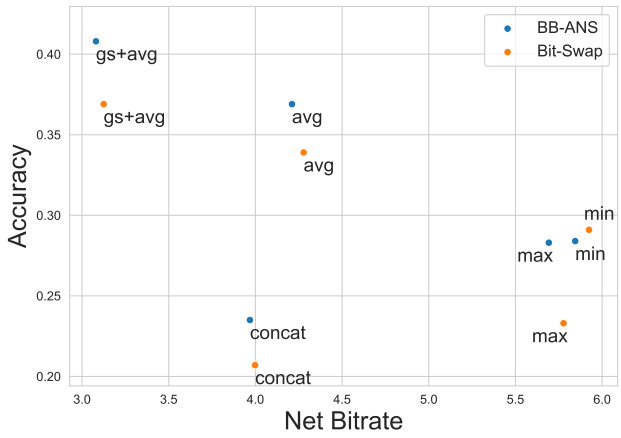


Table 5.4: Bitrate versus Accuracy on CIFAR-10 with various aggregation methods.

less advantageous. Theoretically, the net bitrate should be the same for both methods. However, empirically we can see in Table 5.4, Bit-Swap uses slightly more net bits than BB-ANS.

Although we cannot draw a definite conclusion, it appears that the more a compressor can compress, the more accurate of classification NPC can obtain with the compressor. We can see that given a compressor, there is a correlation between bitrate and accuracy even with different aggregation methods, except for aggregation methods that alter a compressor to be not *normal* (e.g., "concat').

Other Alternatives: In this work, we don't go thoroughly through all the state-of-the-art compressors, but only focus on the VAE-based lossless compressors. For this specific category, there are already numerous important factors: the choice of architectures, the choice of the number of latent variables, the choice of the hierarchy topology (e.g., asymmetrical tree structure or symmetrical one), and the choice of discretization method. Beyond this line of compressors, deep learning based compressors discussed in Section 5.2.3 can also be used under our framework.

Beyond compressors, aggregation methods also cause diverging differences in the final classification accuracy. We cover a few basic ones but there are other non-training-required aggregation methods like *linear blend operator* and even completely different aggregation strategies (e.g., using conditional VAE). On colored real-world images like CIFAR-10, image manipulation

# latent variables z	1	2	8
Bit-Swap	0.226	0.339	0.348
BB-ANS	0.226	0.369	0.356

aggregation	gs+avg	avg	min	max	concat
Bit-Swap	0.369	0.339	0.291	0.233	0.207
BB-ANS	0.408	0.369	0.284	0.283	0.235

Table 5.5: Effects of the number of latent variables and aggregation method.

can be another easy and effective way to improve accuracy. In effect, "greyscale" can be viewed as "lossy compression" and this opens up the question of how lossy compressors perform under NPC framework.

5.5.6 Discussion on Normal Compressor

Definition 2 (Normal Compressor). *A compressor is normal if it satisfies, up to an additive $O(\log n)$ term, where n means the maximal binary length of an element of Ω:*

1. *Idempotency: $C(xx) = C(x)$ and $C(\epsilon) = 0$ where ϵ is the empty string*
2. *Symmetry: $C(xy) = C(yx)$*
3. *Monotonicity: $C(xy) \geq C(x)$*
4. *Distributivity: $C(xy) + C(z) \leq C(xz) + C(yz)$*

Definition 3 (Metric). *A distance function $D : \Omega \times \Omega \to \mathbb{R}^+$ is a metric if it satisfies the following three criteria for any $x, y, z \in \Omega$, where Ω is a non-empty set, $\mathbb{R}^+$ represents the set of non-negative real number:*

1. *Identity: $D(x, y) = 0$ iff $x = y$*
2. *Symmetry: $D(x, y) = D(y, x)$*
3. *Triangle Inequality: $D(x, y) \leq D(x, z) + D(z, y)$*

Definition 4 (Admissible Distance). *A function $D : \Omega \times \Omega \to \mathbb{R}^+$ is an admissible distance if for every pair of objects $x, y \in \Omega$, the distance $D(x, y)$ is computable, symmetric, and satisfies the density condition $\sum_y 2^{-D(x,y)} \leq 1$.*

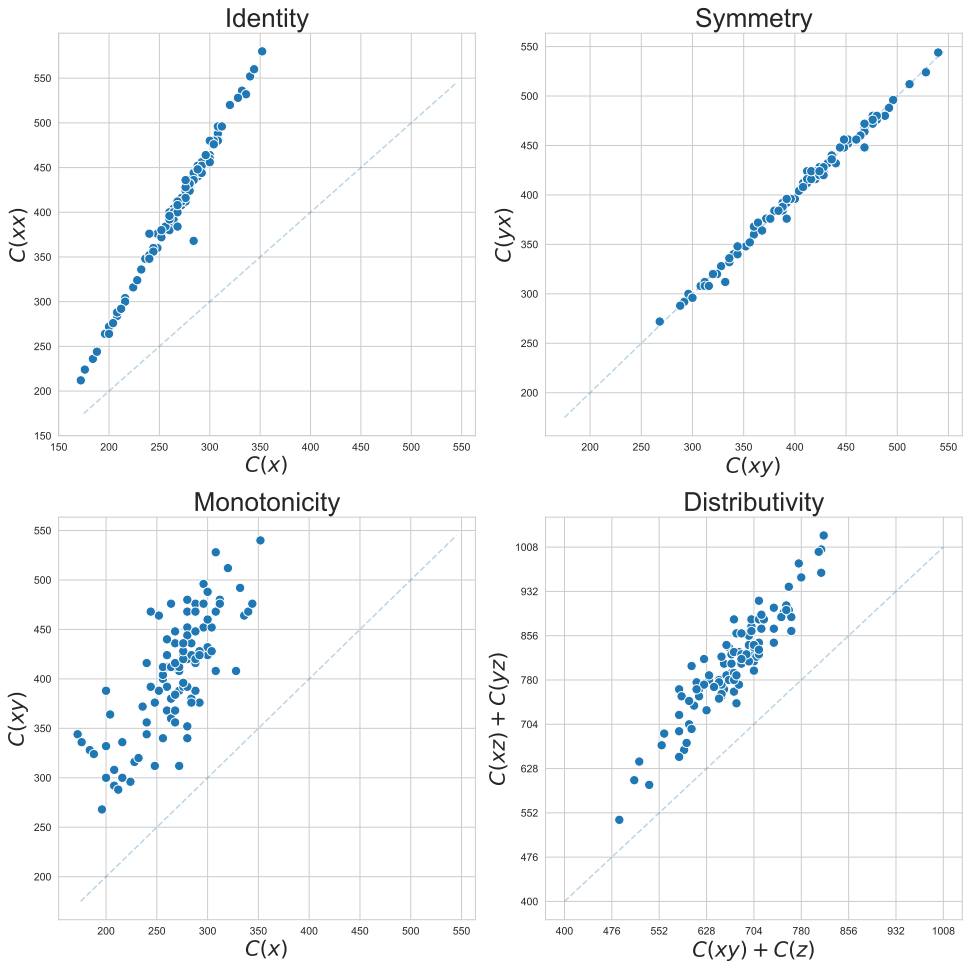


Figure 5.5: BB-ANS Normal Compressor Test for "concatenation"

Cilibrasi and Vitányi [2005] formally prove that if the compressor is *normal*, NCD is a normalized *admissible* distance satisfying the metric inequalities, which is shown in Definition 3. Cebrián et al. [2005] systematically evaluate how far real world compressors like gzip, bz2, PPMZ can satisfy the idempotency axiom. Here we empirically evaluate all 4 axioms on MNIST with the BB-ANS compressor. We randomly take 100 samples and plot $C(\cdot)$ on LHS as the x-axis, $C(\cdot)$ on RHS as the y-axis. For simplicity, we use one latent variable, under which BB-ANS equals Bit-Swap. As shown in Figure 5.5, BB-ANS satisfies monotonicity, symmetry, and distributivity.

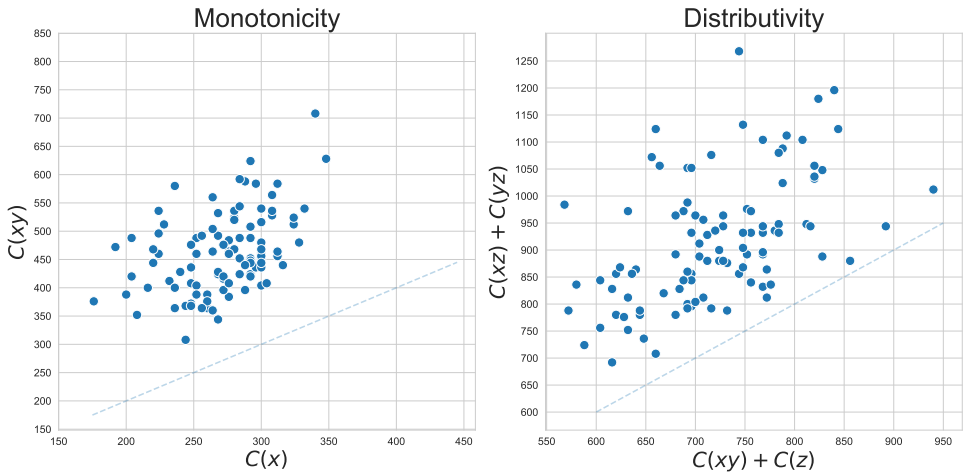


Figure 5.6: BB-ANS Normal Compressor Test for "average"

However, it fails on the identity axiom, with $C(xx) \approx 1.988C(x)$. Unlike gzip, bz2, or lzma, BB-ANS doesn't treat the concatenation of two images as a sequence of bytes but images, similar to PNG and WebP, making it hard to satisfy identity axiom. By default, the aggregation method refers to "concatenation". But simple concatenation does not perform well. Practically, BB-ANS fails the identity test when using "concatenation". Without changing the compressor, is it still possible to satisfy the above conditions so that NCD can be used as a normalized admissible distance metric?

We can change the aggregation method. We investigate whether BB-ANS with average can satisfy the above conditions. Obviously, identity and symmetry axioms can hold, as $C(\text{avg}(x, x)) = C(\frac{x+x}{2}) = C(x)$. We empirically evaluate monotonicity and distributivity, and find that they are both satisfied with "average". Figure 5.6 illustrates that $C(\text{avg}(x, y)) \geq C(x)$ and $C(\text{avg}(x, y)) + C(z) \leq C(\text{avg}(x, z)) + C(\text{avg}(y, z))$ always hold. Given the compressor is normal under the "average" function, we now prove NCD is an admissible distance metric.

Definition 5. *Let D be an admissible distance. $D^+(x)$ is defined as $D^+(x) = \max\{D(x, z) : C(z) \leq C(x)\}$, and $D^+(x, y)$ is defined as $D^+(x, y) = \max\{D^+(x), D^+(y)\}$*

Lemma 1. *If C is a normal compressor, then $E_c(x, y) + O(1)$ is an admissible distance, where $E_c(x, y) = C(xy) - \min\{C(x) - C(y)\}$ is the compression distance.*

Lemma 2. *If C is a normal compressor, then $E_c^+(x,y) = \max\{C(x), C(y)\}$*

Theorem 2. *If the compressor is normal, then the NCD is a normalized admissible distance satisfying the metric (in)equalities.*

Proof. Lemma 1 and Lemma 2 show that NCD is a normalized admissible distance. We now show how NCD satisfies the metric (in)equalities.

1. For identity axiom,
$$\text{NCD}(x,x) = \frac{C(\text{avg}(x,x)) - C(x)}{C(x)} = 0. \tag{5.7}$$
2. For symmetry axiom,
$$\text{NCD}(x,y) = \frac{C(\text{avg}(x,y)) - \min\{C(x), C(y)\}}{\max\{C(x), C(y)\}} = \text{NCD}(y,x). \tag{5.8}$$
3. For triangle inequality, without loss of generality, we assume $C(x) \leq C(y) \leq C(z)$. As NCD is symmetrical, there are three triangle inequalities that can be expressed by $\text{NCD}(x,y), \text{NCD}(y,z), \text{NCD}(x,z)$. For simplicity, we prove one of them, $\text{NCD}(x,y) \leq \text{NCD}(x,z) + \text{NCD}(z,y)$ as the procedure for the other two is similar. Since BB-ANS is a *normal* compressor under "avg", we have distributivity: $C(\text{avg}(x,y))+C(z) \leq C(\text{avg}(x,z))+ C(\text{avg}(z,y))$. Subtracting $C(x)$ from both sides and rearraging results in $C(\text{avg}(x,y)) - C(x) \leq C(\text{avg}(x,z)) - C(x) + C(\text{avg}(z,y)) - C(z)$. Dividing by $C(y)$ on both sides, we have
$$\frac{C(\text{avg}(x,y)) - C(x)}{C(y)} \leq \frac{C(\text{avg}(x,z)) - C(x) + C(\text{avg}(z,y)) - C(z)}{C(y)}. \tag{5.9}$$
We know LHS≤ 1 and RHS can be ≤ 1 or > 1.
   (a) RHS≤ 1: Let $C(z) = C(y) + \Delta$; adding Δ to both the numerator and denominator of RHS increases RHS and makes it closer to 1.
$$\frac{C(\text{avg}(x,y)) - C(x)}{C(y)} \leq \frac{C(\text{avg}(x,z)) - C(x)}{C(y)+\Delta} + \frac{C(\text{avg}(z,y)) - C(z) + \Delta}{C(y)+\Delta} \tag{5.10}$$
$$= \frac{C(\text{avg}(x,z)) - C(x)}{C(z)} + \frac{C(\text{avg}(z,y)) - C(y)}{C(z)}. \tag{5.11}$$

(b) RHS$>$ 1: The procedure is similar to the case when RHS$\leq$ 1. The difference is that adding Δ to both numerator and denominator makes RHS decrease instead of increase. But RHS cannot decrease less than 1. Thus, we still have

$$\frac{C(\text{avg}(x,y)) - C(x)}{C(y)} \leq \frac{C(\text{avg}(x,z)) - C(x)}{C(z)} + \frac{C(\text{avg}(z,y)) - C(y)}{C(z)}. \quad (5.12)$$

□

5.6 Summary

In this chapter, we restrict representation by inserting bottleneck and demonstrate the equivalence between inserting bottleneck and doing compression. We propose a learning framework, Non-Parametric learning by Compression with Latent Variables (NPC-LV), to address a common learning scenario, Non-Supported Few-Shot-Learning (NS-FSL). This framework is versatile in that every module is replaceable, leading to numerous variations. We use image classification as a case study to demonstrate how to use a trained latent generative model directly for downstream classification without further training. It outperforms supervised learning and non-parametric learning on three datasets and semi-supervised learning on CIFAR-10 in the low data regime. We thus regard it as a baseline in NS-FSL. The equivalence between optimizing latent variable models and achieving the shortest code length not only shows how nELBO can be used for classification but also indicates that improvements in latent probabilistic models can benefit neural compressors. The relationship between compression rate and classification accuracy suggests that improvements in neural compressors can further benefit classification. Thus, an enhancement of any module in this chain can boost classification accuracy under this framework.

Chapter 6

Restrict Representations for Non-Parametric Learning by Compression Only

In this chapter, we illustrate how to improve the generalizability of text classification problems by doing compression only. Non-Parametric learning by Compression with Latent Variables (NPC-LV) demonstrates that adding a bottleneck in the middle can give us a neural compressor. In other words, it shows a circumstance when restricting representation in the middle results in cutting the code length at the end. That inspires us to think whether it's possible to use a shorter code length directly, which is equivalent to using a traditional compressor without latent variables. Using traditional compressors fits our Non-Parametric learning by Compression (NPC) framework.

Deep neural networks (DNNs) are often used for text classification due to their high accuracy. However, DNNs can be computationally intensive, requiring millions of parameters and large amounts of labeled data, which can make them expensive to use, optimize, and transfer to out-of-distributed (OOD) cases in practice. In this chapter, we propose a non-parametric alternative to DNNs that's easy, lightweight, and universal in text classification: a combination of a simple compressor like *gzip* with a k-nearest-neighbor classifier. Without any training parameters, our method achieves results that are competitive with non-pretrained deep learning methods on six in-distributed datasets. It even outperforms BERT on all five OOD datasets, including four low-

resource languages. Our method also excels in few-shot settings where labeled data are too scarce for DNNs to achieve satisfying accuracy. This work is presented in:

- Zhiying Jiang, Matthew Y.R. Yang, Mikhail Tsirlin, Raphael Tang, Yiqin Dai, Jimmy Lin. "Low-Resource" Text Classification: A Parameter-Free Classification Method with Compressors. *In Proceedings of The 61st Annual Meeting of the Association for Computational Linguistics (ACL) Findings, 2023.*

6.1 Overview

Text classification, as one of the most fundamental tasks in natural language processing (NLP), has improved substantially with the help of neural networks [Li et al., 2022]. However, most neural networks are data-hungry, the degree of which increases with the number of parameters. They also have many hyperparameters that must be carefully tuned for different datasets, and the preprocessing of text data (e.g., tokenization, stop word removal) needs to be tailored to the specific model and dataset. Despite their ability to capture latent correlations and recognize implicit patterns [LeCun et al., 2015], complex deep neural networks may be overkill for simple tasks such as topic classification, and lighter alternatives are usually good enough. For example, Adhikari et al. [2019b] find that a simple long short-term memory network (LSTM; Hochreiter and Schmidhuber, 1997) with appropriate regularization can achieve competitive results. Shen et al. [2018] further show that even word-embedding-based methods can achieve results comparable to convolutional neural networks (CNNs) and recurrent neural networks (RNNs).

Among all the quests for a lighter alternative to DNN methods, one stream of studies focuses on using compressors for text classification. There have been several studies in this field [Frank et al., 2000; Teahan and Harper, 2003], and most of them are based on the intuition that the minimum cross entropy between a document and a language model of a class built by a compressor indicates the class of the document. However, previous works fail to achieve competitive results with neural networks.

We propose a text classification method combining a lossless compressor, a compressor-based distance metric with a k-nearest-neighbor classifier (kNN). It utilizes compressors in capturing

regularity which then is translated into similarity measurement by a compressor-based distance metric. With the distance matrix, we use kNN to classify.

We carry out experiments on seven in-distributed datasets and five out-of-distributed ones. With a simple compressor like *gzip*, our method achieves competitive results with DNNs on six out of seven datasets and outperforms all DNNs including BERT on all OOD datasets. It also surpasses DNNs by a large margin under few-shot settings.

Our method is a simple, lightweight, and universal alternative to DNNs. It's simple because it doesn't require any preprocessing or training. It's lightweight in that it classifies without the need for parameters or GPU resources. It's universal as compressors are data-type agnostic, and non-parametric methods do not bring inductive bias by the training procedure.

Our contributions are as follows: (1) We propose a parameter-free method that achieves results comparable to non-pretrained neural network models that have millions of parameters on six out of seven in-distributed datasets; (2) We show that our method outperforms all methods including pretrained models on OOD datasets; (3) We demonstrate that our method excels in few-shot settings when labeled data are scarce.

6.2 Non-Parametric learning by Compression

Similar to NPC-LV, NPC consists of a lossless compressor, a compressor-based distance metric, and a k-Nearest-Neighbor classifier. Different from NPC-LV, NPC only requires a regular lossless compressor without any pretraining or fine-tuning. Lossless compressors aim to represent information using as few bits as possible by assigning shorter codes to symbols with higher probability. The intuition of using compressors for classification is that (1) compressors are good at capturing regularity; (2) objects from the same category share more regularity than those from different categories. The intuition can be seen clearly through the text examples below. x_1 belongs to the same category as x_2, but a different category from x_3. If we use $C(\cdot)$ to represent compressed length, we will find $C(x_1x_2) - C(x_1) < C(x_1x_3) - C(x_1)$ where $C(x_1x_2)$ means the compressed length of concatenation of x_1 and x_2. In other words, $C(x_1x_2) - C(x_1)$ can be interpreted as how many bytes we need to use to encode x_2 based on the information of x_1:

x_1 = *Japan's Seiko Epson Corp. has developed a 12-gram flying microrobot.*

x_2 = *The latest tiny flying robot has been unveiled in Japan.*

x_3 = *Michael Phelps won the gold medal in the 400 individual medley.*

As we've covered in Section 2.2.2, this intuition can be formalized as a distance metric derived from Kolmogorov complexity [Kolmogorov, 1963] and *information distance* $E(x, y)$. As the incomputable nature of Kolmogorov complexity renders E(x,y) incomputable, Li et al. [2004] propose a normalized and computable version of information distance, *Normalized Compression Distance* (NCD), utilizing compressed length $C(x)$ to approximate Kolmogorov complexity $K(x)$, same as the metric we used in our previous experiments in Chapter 5. Recall that it's defined as follows:

$$NCD(x, y) = \frac{C(xy) - \min\{C(x), C(y)\}}{\max\{C(x), C(y)\}} \tag{6.1}$$

The intuition behind using compressed length is that the length of x that has been maximally compressed by a compressor is close to $K(x)$. The higher the compression ratio, the closer $C(x)$ is to $K(x)$. As our main experiment results use *gzip* as the compressor, $C(x)$ here means the length of x after being compressed by *gzip*. $C(xy)$ is the compressed length of concatenation of x and y. With the distance matrix NCD provides, we can then use k-nearest-neighbor to classify.

Our method can be implemented with 14 lines of Python code below. The inputs are *training_set*, *test_set*, both consist of an array of *(text, label)*, and *k*:

```
import gzip
import numpy as np
for (x1, _) in test_set:
    Cx1 = len(gzip.compress(x1.encode()))
    distance_from_x1 = []
    for (x2, _) in training_set:
        Cx2 = len(gzip.compress(x2.encode())
        x1x2 = " ".join([x1, x2])
        Cx1x2 = len(gzip.compress(x1x2.encode())
        ncd = (Cx1x2 - min(Cx1,Cx2)) / max(Cx1, Cx2)
        distance_from_x1.append(ncd)
    sorted_idx = np.argsort(np.array(distance_from_x1))
    top_k_class = training_set[sorted_idx[:k], 1]
```

```
    predict_class = max(set(top_k_class), key=top_k_class.count)
```

Listing 6.1: Python Code for Text Classification with Gzip

6.3 Related Work

6.3.1 Compressor-Based Text Classification

Text classification using compressors can be divided into two main approaches: (1) Using a compressor to estimate entropy based on Shannon Information Theory; (2) Using a compressor to approximate Kolmogorov complexity and *information distance*.[1]

The first approach mainly employs a text compression technique called Prediction by Partial Matching (PPM)[2] for topic classification. This approach estimates the cross entropy between the probability distribution of a specific class c and a given document d: $H_c(d)$ [Frank et al., 2000; Teahan and Harper, 2003]. The intuition is that the lower the cross entropy is, the more likely that d belongs to c. Coutinho and Figueiredo [2015]; Kasturi and Markov [2022]; Marton et al. [2005] further improve the classification accuracy by improving representation to better cope with the compressor.

Another line of compressor-based methods [Keogh et al., 2004b; Khmelev and Teahan, 2003] takes advantage of the *information distance* [Bennett et al., 1998], a distance metric derived from Kolmogorov complexity. The intuition of *information distance* is that for two similar objects, there exists a *simple* program to convert one to another. However, most previous works focus on clustering [Vitányi et al., 2009], plagiarism detection [Chen et al., 2004] and time serires data classification [Keogh et al., 2004b]. Few [Coutinho and Figueiredo, 2015; Marton et al., 2005] explore its application to topic classification, and none applies the combination of *information distance* and k-nearest-neighbor (kNN) classifier when $k > 1$ to topic classification. Besides, to the best of our knowledge, all the previous works use relatively small datasets like 20News and

[1] This doesn't indicate these two lines of work are completely parallel. In fact, the expected value of Kolmogorov complexity equals Shannon entropy, up to a constant.

[2] PPM is a text compression scheme utilizing language modeling to estimate cross entropy.

Reuters-10. There is neither a comparison between compressor-based methods and deep learning methods nor any comprehensive study on large datasets.

6.3.2 Deep Learning for Text Classification

The deep learning methods used for text classification can be divided into two: transductive learning, represented by Graph Convolutional Networks (GCN) [Yao et al., 2019], and inductive learning, dominated by recurrent neural networks (RNN) and convolutional neural networks (CNN). We focus on inductive learning in this chapter as transductive learning assumes the test dataset is presented during the training, which is not a common scenario.

Zhang et al. [2015a] use the character-based CNN with millions of parameters for text classification. Conneau et al. [2017] extend the idea with more layers. Along the line of RNNs, Kawakami [2008] introduce a method that uses LSTMs [Hochreiter and Schmidhuber, 1997] to learn the sequential information for classification. To better capture the important information regardless of position, Wang et al. [2016] incorporate the attention mechanism into the relation classification. Yang et al. [2016] include a hierarchical structure for sentence-level attention. As the parameter number and the model complexity increase, Joulin et al. [2017] look for using a simple linear model with a hidden layer coping with n-gram features and hierarchical softmax to improve efficiency.

The landscape of classification has been further transformed by the widespread use of pretrained models like BERT [Kenton and Toutanova, 2019], with thousands of millions of parameters pretrained on a corpus containing billions of tokens. BERT achieves the state of the art on text classification [Adhikari et al., 2019a]. Built on BERT, Reimers and Gurevych [2019b] calculate semantic similarity between pairs of sentences efficiently by using a siamese network architecture and fine-tuning on multiple NLI datasets [Bowman et al., 2015; Williams et al., 2018]. We compare *gzip* with these deep learning models.

Dataset	#Training	#Test	#Classes	Avg#Words	Avg#Chars	#Vocab
AG News	120K	7.6K	4	44	236	128K
DBpedia	560K	70K	14	54	301	1M
YahooAnswers	1.4M	60K	10	107	520	1.5M
20News	11K	7.5K	20	406	1902	277K
ohsumed	3.4K	4K	23	212	1273	55K
R8	5.5K	2.2K	8	102	587	24K
R52	6.5K	2.6K	52	110	631	26K
KinyarwandaNews	17K	4.3K	14	232	1872	240K
KirundiNews	3.7K	923	14	210	1722	63K
DengueFilipino	4K	500	5	10	62.7	13K
SwahiliNews	22.2K	7.3K	6	327	2.2K	570K
SogouNews	450K	60K	5	589	2.8K	611K

Table 6.1: Details of datasets used for evaluation.

6.4 Experimental Setup

We introduce the datasets and baseline models we choose as well as the training details in this section.

6.4.1 Datasets

We choose a variety of datasets to investigate the effects of the number of training samples, the number of classes, the length of the text, and the difference in distribution on accuracy. The details of each dataset are listed in Table 6.1. Previous works on text classification have two disjoint preferences when choosing evaluation datasets: CNN and RNN-based methods favor large-scale datasets (AG News, DBpedia, YahooAnswers), whereas transductive methods like graph convolutional neural networks focus on smaller ones (20News, Ohsumed, R8, R52) [Li et al., 2022]. We include datasets on both sides in order to investigate how our method performs in both situations. Apart from dataset sizes, we also take the number of classes into account by intentionally including datasets like R52 to evaluate the performance on datasets with a large number of classes. We

also include the text length of each dataset in Table 6.1 as previous works [Marton et al., 2005] indicate that it affects the accuracy of compressor-based methods.

Generalizing to Out-Of-Distributed datasets has always been a challenge in machine learning. Even with the success of pretrained models, this problem is not alleviated. In fact, Yu et al. [2021] have shown that improved in-distributed accuracy on pretrained models may lead to poor OOD performance in image classification. In order to compare our method with pretrained models on OOD datasets, we choose five datasets that are unseen in BERT's pretrained corpus — Kinyarwanda news, Kirundi news, Filipino dengue, Swahili news, and Sogou news. Those datasets are chosen to have Latin script which means they have a very similar alphabet to English. For example, Swahili has the same vowels as English but doesn't have `q`,`x` as consonants; Sogou news is in Pinyin – a phonetic romanization of Chinese. Therefore, those datasets can be viewed as permutations of English alphabets (see Table G.1 for text examples).

6.4.2 Baselines

We compare our result with (1) neural network methods that require training and (2) non-parametric methods that use the kNN classifier directly, with or without pretraining on external data. Specifically, we choose mainstream architectures for text classification, like logistic regression, fastText [Joulin et al., 2017], RNNs with or without attention (vanilla LSTM [Hochreiter and Schmidhuber, 1997], bidirectional LSTMs [Schuster and Paliwal, 1997] with attention [Wang et al., 2016], hierarchical attention networks [Yang et al., 2016]), CNNs (character CNNs [Zhang et al., 2015a], recurrent CNNs [Lai et al., 2015], very deep CNNs [Conneau et al., 2017]) and BERT [Devlin et al., 2019]. We also include three other non-parametric methods: word2vec (W2V) [Mikolov et al., 2013], pretrained sentence BERT (SentBERT) [Reimers and Gurevych, 2019b], and the length of the instance (TextLength), all using a kNN classifier. "TextLength" is a baseline where the text length of the instance is used as the only input into a kNN classifier, whose result rules out the impact of text length in classification.

We present details of models in Table 6.2. Here we use AG News as an example to estimate the model size, as the number of parameters is affected by the number of classes and the vocabulary size. This dataset has a relatively small vocabulary size and the number of classes, making the

Model	#Param	Pretraining	Training	External Data	Pre-Process
TFIDF+LR	260K	✗	✓	✗	tok+tfidf+dict (+lower)
LSTM	5.2M	✗	✓	✗	tok+dict (+emb+lower+pad)
Bi-LSTM+Attn	8.2M	✗	✓	✗	tok+dict (+emb+lower+pad)
HAN	30M	✗	✓	✗	tok+dict (+emb+lower+pad)
charCNN	2.7M	✗	✓	✗	dict (+lower+pad)
textCNN	31M	✗	✓	✗	tok+dict (+emb+lower+pad)
RCNN	19M	✗	✓	✗	tok+dict (+emb+lower+pad)
VDCNN	14M	✗	✓	✗	dict (+lower+pad)
fastText	8.2M	✗	✓	✗	tok+dict (+lower+pad+$n$gram)
BERT-base	109M	✓	✓	✓	tok+dict+pe (+lower+pad)
W2V	0	✓	✗	✗	tok+dict (+lower)
SentBERT	0	✓	✗	✓	tok+dict (+lower)
TextLength	0	✗	✗	✗	✗
gzip (ours)	0	✗	✗	✗	✗

Table 6.2: Models with the number of parameters, the training, data augmentation, and pre-processing details.

estimated number of parameters the lower bound of the studied datasets. Some methods require pretraining either on the target dataset or on other external datasets.

We also list preprocessing required by these models in Table 6.2, including tokenization ("tok"), building vocabulary dictionaries and mapping tokens ("dict"), using pretrained word embeddings ("emb"), lowercasing the words ("lower") and padding the sequence to a certain length ("pad"). Other model-specific preprocessing includes adding an extra bag of n-grams ("$n$gram") for *fastText* and using positional embedding ("pe") for BERT. Note that for models that only require training, we do not use pretrained word embeddings as otherwise the boundary between pretraining and training will become ambiguous.

[2]We report results getting from our own implementation. We also provide previously reported results for reference in Appendix H.

Model	Pretraining	Training	AGNews	DBpedia	YahooAnswers	20News	Ohsumed	R8	R52
TFIDF+LR	✗	✓	0.898	0.982	0.715	0.827	0.549	0.949	0.874
LSTM	✗	✓	0.861	0.985	0.708	0.657	0.411	0.937	0.855
Bi-LSTM+Attn	✗	✓	0.917	0.986	0.732	0.667	0.481	0.943	0.886
HAN	✗	✓	0.896	0.986	0.745	0.646	0.462	0.960	0.914
charCNN	✗	✓	0.914	0.986	0.712	0.401	0.269	0.823	0.724
textCNN	✗	✓	0.817	0.981	0.728	0.751	0.570	0.951	0.895
RCNN	✗	✓	0.912	0.984	0.702	0.716	0.472	0.810	0.773
VDCNN	✗	✓	0.913	0.987	0.734	0.491	0.237	0.858	0.750
fastText	✗	✓	0.911	0.978	0.702	0.690	0.218	0.827	0.571
BERT	✓	✓	0.944	0.992	0.768	0.868	0.741	0.982	0.960
W2V	✓	✗	0.892	0.961	0.689	0.460	0.284	0.930	0.856
SentBERT	✓	✗	0.940	0.937	0.782	0.778	0.719	0.947	0.910
TextLength	✗	✗	0.275	0.093	0.105	0.053	0.090	0.455	0.362
gzip (ours)	✗	✗	0.937	0.970	0.638	0.685	0.521	0.954	0.896

Table 6.3: Test accuracy compared with *gzip*, red highlighting the ones outperformed by *gzip*.[3]

6.5 Results and Analysis

We conduct experiments using fourteen text classification methods on twelve datasets.

6.5.1 In-Distributed Text Classification

We train all baselines on seven datasets using their full training sets. The results are shown in Table 6.3. Our method performs particularly well on AG News, R8, and R52. On the AG News dataset, fine-tuning BERT yields the highest performance among all methods, while our method, without any pretraining, achieves competitive results, with only 0.007 points lower than BERT. On both R8 and R52, the only non-pretrained neural network that outperforms our method is HAN. For YahooAnswers, the accuracy of *gzip* is about 7% lower than the average neural methods. This may be due to the large vocabulary size of YahooAnswers, which makes it hard for the compressor to compress (detailed discussion is in Section 6.5.4).

Overall, BERT-based models are robust to the size of in-distributed datasets. Character-based models like charCNN and VDCNN perform badly when the dataset is small and the vocabulary

Dataset	average	*gzip*
AGNews	0.901	**0.937**
DBpedia	0.978	0.970
YahooAnswers	0.726	0.638
20News	0.678	**0.685**
Ohsumed	0.470	**0.521**
R8	0.914	**0.954**
R52	0.838	**0.896**

Table 6.4: Test accuracy comparison between the average of all baseline models (excluding TextLength) and *gzip*.

size is large (e.g., 20News). Word-based models are better at handling big vocabulary sizes. The result of TextLength is extremely low, indicating the compressed length used in NCD does not benefit from the length distribution of different classes.

gzip does not perform well on an extremely large dataset (e.g., YahooAnswers), but is competitive on medium and small datasets. Performance-wise, the only non-preptrained deep learning model that's competitive to *gzip* is HAN, which surpasses *gzip* on four datasets and still achieves relatively high accuracy when it's beaten by *gzip*, unlike textCNN. The difference is that *gzip* doesn't require training.

We list the average of all baseline models' test accuracy (except TextLength for its extremely low accuracy) in Table 6.4. We can see our method is either higher or close to the average on all but YahooAnswers.

6.5.2 Out-Of-Distributed Text Classification

On five OOD datasets (Kinyarwanda news, Kirundi news, Filipino dengue, Swahili news and Sogou news), we also select DNNs to make sure we cover a wide range of parameter numbers. We discard CNN-based methods due to their inferior performance when datasets are small, as shown in both Section 6.5.1 and Zhang et al. [2015a]. In addition, we also add BERT pretrained on 104 languages (mBERT). We can see in Table 6.5 that on languages that mBERT has not been

Model/Dataset	KinyarwandaNews		KirundiNews		DengueFilipino		SwahiliNews		SogouNews	
Shot#	Full	5-shot	Full	5-shot	Full	5-shot	Full	5-shot	Full	5-shot
Bi-LSTM+Attn	0.843	0.253±0.061	0.872	0.254±0.053	0.948	0.369±0.053	0.863	0.357±0.049	0.952	0.534±0.042
HAN	0.820	0.137±0.033	0.881	0.190±0.099	0.981	0.362±0.119	0.887	0.264±0.042	0.957	0.425±0.072
fastText	0.869	0.170±0.057	0.883	0.245±0.242	0.870	0.248±0.108	0.874	0.347±0.255	0.930	0.545±0.053
W2V	0.874	0.281±0.236	0.904	0.288±0.189	0.993	0.481±0.158	0.892	0.373±0.341	0.943	0.141±0.005
SentBERT	0.788	0.292±0.062	0.886	0.314±0.060	0.992	0.629±0.143	0.822	0.436±0.081	0.860	0.485±0.043
BERT	0.838	0.240±0.060	0.879	0.386±0.099	0.979	0.409±0.058	0.897	0.396±0.096	0.952	0.221±0.041
mBERT	0.835	0.229±0.066	0.874	0.324±0.071	0.983	0.465±0.048	0.906	0.558±0.169	0.953	0.282±0.060
gzip (ours)	**0.891**	**0.458±0.065**	**0.905**	**0.541±0.056**	**0.998**	**0.652±0.048**	**0.927**	**0.627±0.072**	**0.975**	**0.649±0.061**

Table 6.5: Test accuracy on out-of-distributed datasets with 95% confidence interval over five trials in five-shot setting.

pretrained on (Kinyarwanda, Kirundi, or Pinyin), it has lower accuracy than BERT. Compared with non-pretrained ones, pretrained models do not hold their advantage on low-resource languages with smaller data sizes, except for Filipino who shares a large vocabulary with English words. On large OOD datasets (i.e., SogouNews), BERT achieves competitive results with other non-pretrained neural networks.

Without any pretraining or fine-tuning, our method outperforms both BERT and mBERT on all five datasets. In fact, our experiments show that our method excels both pretrained and non-pretrained deep learning methods on OOD datasets, which back our claim that our method is universal in terms of dataset distributions. To put it simply, our method is designed to handle unseen datasets: the compressor is data-type-agnostic by nature and non-parametric methods do not introduce inductive bias during training.

6.5.3 Few-Shot Text Classification

We further compare our method with deep learning methods under the few-shot setting. We carry out experiments on AG News, DBpedia, and SogouNews across both non-pretrained deep neural networks and pretrained ones. We use n-shot labeled examples per class from the training dataset, where $n = \{5, 10, 50, 100\}$. We choose these three datasets as their scale is large enough to cover a 100-shot setting and they vary in text lengths as well as languages. We choose methods whose trainable parameters range from zero parameters like word2vec and sentence BERT to hundreds

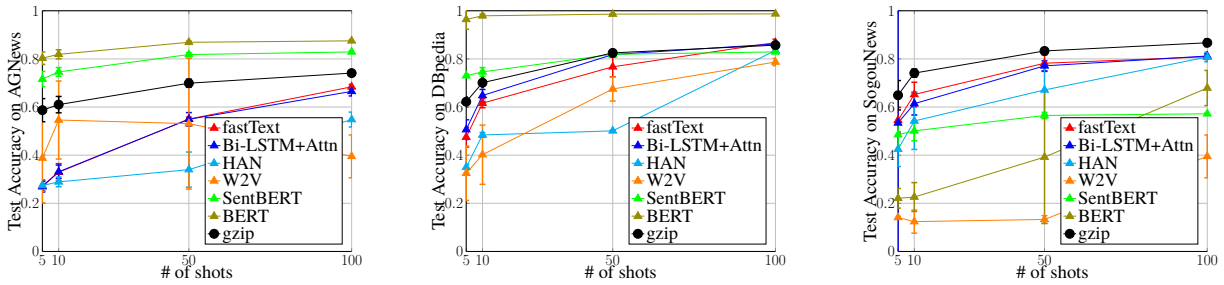


Figure 6.1: Comparison among different methods using different shots with 95% confidence interval over five trials.

of millions of parameters like BERT, covering both word-based models (HAN) and an n-gram one (fastText).

The result is plotted in Figure 6.1 (detailed numbers are shown in Table I.1). As we can see, *gzip* outperforms non-pretrained models with $5, 10, 50$ settings on all three datasets. When the number of shot is as few as $n = 5$, *gzip* outperforms non-pretrained models by large margin: the accuracy of *gzip* is 115% better than fastText on AG News 5-shot setting. In the 100-shot setting, *gzip* also outperforms non-pretrained models on AG News and SogouNews but is a little bit lower than them on DBpedia.

It's been investigated in the previous work [Nogueira et al., 2020; Zhang et al., 2021c] that pretrained models are excellent few-shot learners and this is reflected in the consistently high accuracy of BERT and SentBERT on in-distribution datasets like AG News and DBpedia under few-shot settings.[4] It's worth pointing out, though, that *gzip* outperforms SentBERT under 50 and 100-shot settings. However, as reflected in the results on SogouNews, when the dataset is distinctively different from the pretrained datasets, the inductive bias introduced from the pretraining data leads to low accuracy of BERT and SentBERT with 10, 50, and 100-shot settings, especially with the 5-shot setting. In general, when the shot number increases, the accuracy difference between *gzip* and deep learning methods becomes smaller. W2V is an exception that has a large variance in accuracy. This is due to the vectors being trained for a limited set of words, meaning that numerous tokens in the test set are out-of-vocabulary.

[4]Note that BERT reaches an almost perfect accuracy on DBpedia probably because DBpedia is extracted from Wikipedia, which BERT is pretrained on.

6.5.4 Performance Analysis

To understand the merits and shortcomings of using *gzip* for classification, we evaluate *gzip*'s performance in terms of both the absolute accuracy and the relative performance compared to the neural methods. An absolute low accuracy with a high relative performance suggests that the dataset itself is difficult, while a high accuracy with a low relative performance means the dataset is better solved by a neural network. As our method performs well on OOD datasets, we are more interested in analyzing ID cases. We carry out on seven in-distributed datasets and one out-of-distributed dataset across fourteen models to account for different ranks. We analyze both the relative performance and the absolute accuracy regarding the vocabulary size and the compression rate of both datasets (i.e., how easily a dataset can be compressed) and compressors (i.e., how well a compressor can compress).

To represent the relative performance with regard to other methods, we use the normalized rank percentage, computed as $\frac{\text{rank of gzip}}{\text{total\#methods}}$; the lower the score, the better *gzip* is. We use "bits per character"(bpc) to evaluate the compression rate. The procedure is to randomly sample a thousand instances from the training and test set respectively, calculate the compressed length, and divide by the number of characters. Sampling is to keep the size of the dataset constant.

Relative Performance Combining Table 6.1 and Table 6.3, we see that accuracy is largely unaffected by the average length of a single sample: with the Spearman coefficient $r_s = -0.220$. But the relative performance is more correlated with vocabulary size ($r_s = 0.561$) as we can see in Figure 6.2. SogouNews is an outlier in the first plot: on a fairly large vocabulary-sized dataset, *gzip* ranks first. The second plot may provide an explanation for that — the compression ratio for SogouNews is high which means even with a relatively large vocabulary size, there is also repetitive information that can be squeezed out. With $r_s = 0.785$ on the correlation between the normalized rank percentage and the compression rate, we can see when a dataset is easier to compress, our method may be a strong candidate as a classifier.

Absolute Accuracy Similarly, we evaluate the accuracy of classification with respect to the vocabulary size and we've found there is almost no monotonic relation ($r_s = 0.071$). With regard to bpc, the monotonic relation is not as strong as the one with the rank percentage ($r_s = -0.56$). Considering the effect that vocabulary size has on the relative performance, our method with *gzip* may be more susceptible to the vocabulary size than neural network methods. To distinguish be-

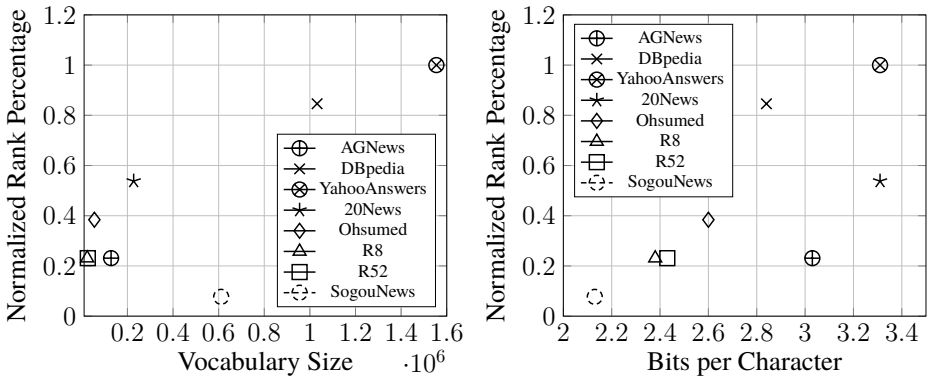


Figure 6.2: Relative performance v.s. vocabulary size and compression rate.

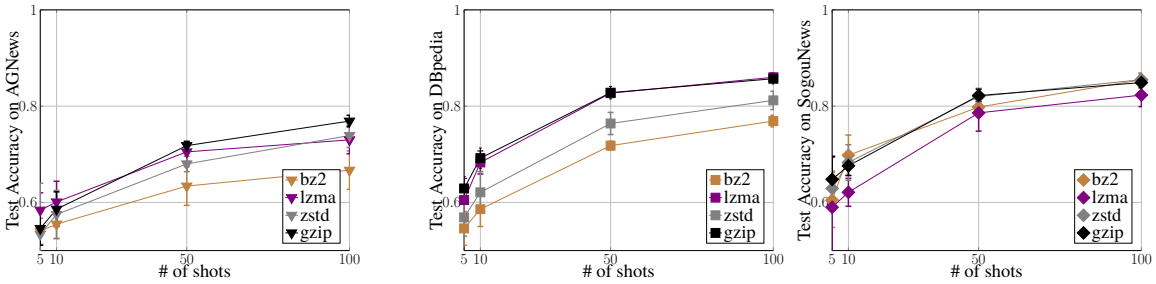


Figure 6.3: Comparison among different compressors on three datasets with 95% confidence interval over five trials.

tween a "hard" dataset and an "easy" one, we average all models' accuracies. The datasets that have the lowest accuracies are 20News and Ohsumed, which are two datasets that have the longest average length of texts.

6.5.5 Comparison among Compressors

As the compressor in our method can actually be replaced by any other compressors, we evaluate the performance of three other lossless compressors: *bz2*, *lzma*, and *zstandard*. Due to the low compression speed of *lzma*, we randomly select 1,000 test samples from the whole test set to evaluate and conduct our experiments under 5, 10, 50, and 100-shot settings. We repeat the experiments under each setting for five times to calculate the mean and the 95% confidence interval.

Each of the three compressors that we choose has different underlying algorithms from *gzip*. *bz2* uses Burrows-Wheeler algorithm [Burrows and Wheeler, 1994] to permute the order of characters in the strings to create more repeated "substrings" that can be compressed, giving it a higher compression ratio (e.g., it can achieve 2.57 bits-per-character (bpc) on AGNews while *gzip* can achieve only 3.38 bpc). *lzma* is similar to *gzip* in that they are both based on LZ77 [Ziv and Lempel, 1977], a dictionary-based compression algorithm using *(offset, length)* to represent the n-gram that has previously appeared in the search buffer.[5] *zstandard (zstd)* is a new compression algorithm that's built on LZ77, Huffman coding as well as Asymmetric Numeral Systems (ANS) [Duda, 2009]. We pick *zstd* because of its high compressing speed and a compression ratio close to *gzip*. A competitive result would suggest that *zstd* might be an alternative to *gzip* and speed up the classification.

In Figure 6.4, we plot the test accuracy and compression ratio of each compressor. Compression ratio is calculated by $\frac{\text{original size}}{\text{compressed size}}$, so the larger the compression ratio is, the more a compressor can compress.[6] Each marker type represents a dataset and markers of '+' represents the mean of each compressor's test accuracy across different shot settings.

In general, *gzip* achieves relatively high and stable accuracy across three datasets. *lzma* is competitive with *gzip* but the speed is much slower. Despite of its high compression ratio, *bz2* performs the worst across AGNews and DBpedia. Normally, a higher compression ratio of a compressor suggests that the NCD based on it approximates the information distance $E(x, y)$ better. But in *bz2*'s case, its accuracy is always lower than the regression line (Figure 6.4). We conjecture it may be because the Burrows-Wheeler algorithm used by *bz2* dismisses the information of character order by permuting characters during compression.

We investigate the correlation between accuracy and compression ratio across compressors and find that they have a moderate monotonic linear correlation as shown in Figure 6.4. As the shot number increases, the linear correlation becomes more obvious with $r_s = 0.605$ for all shot settings and Pearson correlation $r_p = 0.575, 0.638, 0.691, 0.719$ respectively on 5, 10, 50,

[5] *gzip* uses DEFLATE algorithm, which uses Huffman coding [Huffman, 1952] to further encode *(offset, length)* whereas *lzma* uses range coding to do so, resulting *lzma* has a higher compression ratio but a slower compression speed.

[6] We use compression ratio instead of bpc here as the latter one is too close to each other and cannot be differentiated from one another.

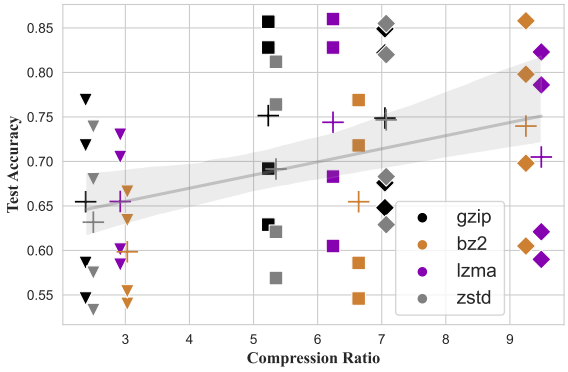


Figure 6.4: Compression ratio V.S. Test Accuracy across different compressors on three datasets under different shot settings

and 100-shot settings across four compressors. We have also found that for a single compressor, the easier a dataset can be compressed, the higher the accuracy *gzip* can achieve (details are in Section 6.5.4). Combining our findings, we can see that a compressor performs best when it has a high compression ratio on datasets that are highly compressible unless crucial information is disregarded by its compression algorithm.

6.5.6 Comparison with Other Compressor-based Methods

A majority of previous compressor-based text classification is built on estimating cross entropy between the probability distribution built on class c and the document d: $H_c(d)$, as we mention in Section 6.3.1. Summarized in Russell [2010], the procedure of using compressor to estimate $H_c(d)$ is: (1) for each class c, concatenate all samples d_c in the training set belonging to c; (2) compress d_c as one long document to get the compressed length $C(d_c)$; (3) concatenate the given test sample d_u with d_c and compress to get $C(d_c d_u)$; (4) the predicted class is $\arg\min_c C(d_c d_u) - C(d_c)$. The distance metric used by previous work [Marton et al., 2005; Russell, 2010] is mainly $C(d_c d_u) - C(d_c)$. Although using this distance metric is faster than pair-wise distance matrix computation on small datasets, it has several drawbacks: (1) Most compressors have a limited "size", for *gzip* it's the sliding window size that can be used to search back of the repeated string while for *lzma* it's the dictionary size it can keep a record of. This means that even if there are a

Method	AGNews	SogouNews	DBpedia	YahooAnswers
gzip(ce)	0.739±0.046	0.741±0.076	**0.880±0.010**	**0.408±0.012**
gzip(kNN)	**0.752±0.041**	**0.862±0.033**	0.852±0.008	0.352±0.014

Table 6.6: Comparison with other compressor-based methods under the 100-shot setting.

large number of training samples, the compressor cannot take full advantage of those samples; (2) When d_c is large, compressing $d_c d_u$ can be really slow, which cannot be solved by parallelization. These two main drawbacks stop this method from being applied to a really large dataset. Thus, we limit the size of the dataset to 1,000 randomly picked test samples and 100-shot from each class in the training set to compare our method with this method.

In Table 6.6, "*gzip* (ce)" means using the cross entropy $C(d_c d_u) - C(d_c)$ while "*gzip* (kNN)" refers to our method. We carry out each experiment five times and calculate the mean and 95% confidence interval. Our method outperforms the cross-entropy method on AGNews and SogouNews. The reason for the large accuracy gap between the two methods on SogouNews is probably because each instance in SogouNews is very long, and the size of each sample can be 11.2K, which, when concatenated, makes d_c larger than 1,000K under 100-shot setting, while *gzip* typically has 32K window size only. When the search space is tremendously smaller than the size of d_c, the compressor fails to take advantage of all the information from the training set, which renders the compression ineffective. The cross-entropy method does perform very well on YahooAnswers. This might be because on a divergent dataset like YahooAnswers, which is created by numerous online users, concatenating all the samples in a class allows the cross-entropy method to take full advantage of all the information from a single class.

We also test the performance of the compressor-based cross-entropy method on *full* AGNews dataset, as it is a relatively smaller one with a shorter single instance. The accuracy is 0.745, not much higher than the 100-shot setting, which further confirms that using $C(d_c d_u) - C(d_c)$ as a distance metric cannot take full advantage of the large datasets. In general, the result suggests that the compressor-based cross-entropy method is not as advantageous as ours on large datasets while cross-entropy method is faster. To mitigate the computational efficiency problem of NCD, Raff and Nicholas [2017] propose an alternative distance metric that is the order of magnitude faster than NCD.

6.6 Summary

In this chapter, we restrict representation by doing compression directly. Specifically, we use *gzip* with a compressor-based distance metric to do text classification. Our method achieves an accuracy comparable to non-pretrained neural network classifiers on in-distributed datasets and outperforms both pretrained and non-pretrained models on out-of-distributed datasets. We even achieve the state-of-the-art performance on four low-resource language classification datasets. We also find that our method has greater advantages under few-shot settings, which echoes our mission of Non-Supported Few-Shot Learning (NS-FSL).

Chapter 7

Restrict Representations by Compression without Actual Compression under Weak Supervision

In Chapter 5, we show that nELBO can be used directly as an estimated compressed length for downstream classification. Chapter 6 further shows the potential of using compressors for NLP tasks. The idea of using estimated compressed for NLP tasks instead of doing actual compression is intriguing as it not only indicates a new way to utilize the probability distribution captured in pretrained models but also releases the potential of using it without overheads about the discretization or coding schemes. In this chapter, we introduce a way to utilize pretrained language models with the idea of doing compression but without the actual process of compression on passage ranking tasks.

To be specific, this work proposes a novel adaptation of a pretrained sequence-to-sequence model to the task of passage ranking. By taking advantage of the underlying probability of "target" words, we utilize the probabilistic distribution captured by pretrained generative models. Our approach is fundamentally different from a commonly-adopted discriminative formulation of ranking, which relies on encoder-only pretrained transformer architectures such as BERT. On the popular MS MARCO passage ranking task, experimental results show that our approach is at least on par with previous classification-based models and can surpass them with larger gen-

erative models. On the test collection from the TREC 2004 Robust Track, we demonstrate a zero-shot transfer-based approach that outperforms previous state-of-the-art models requiring in-dataset cross-validation. Furthermore, we find that our approach significantly outperforms an encoder-only model in a low-data regime.

This work is presented in:

- Rodrigo Nogueira*, Zhiying Jiang*, Ronak Pradeep, Jimmy Lin. Document Ranking with a Pretrained Sequence-to-Sequence Model. *In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP) Findings, 2020.*

7.1 Overview

Given a query q, the objective of passage ranking is to return a ranked list of passages from a large corpus that maximizes certain types of ranking metrics such as average precision or nDCG. The task of ranking is often converted into a classification problem. The formulation is straightforward: rank based on the probability that a candidate passage belongs to the desired class.

Pretrained language models, exemplified by BERT [Devlin et al., 2019], have proven highly effective in a variety of classification and sequence labeling tasks in NLP. Nogueira and Cho [2019] were the first to demonstrate its effectiveness in ranking tasks. Since it is impractical to apply inference to *every* passage in a corpus with respect to a query, these techniques are typically applied to *rerank* a list of candidates. In a typical end-to-end system, these candidates are from the results of a keyword search based on a "classic" IR scoring function such as BM25 [Robertson et al., 1995]. This gives rise to the standard multi-stage pipeline architecture of keyword retrieval followed by reranking using one or more machine learning models [Asadi and Lin, 2013; Nogueira et al., 2019a].

In this work, we adapt a pretrained generative model, specifically sequence-to-sequence model (in our case, T5 [Raffel et al., 2020b]), to the task of passage reranking. To our knowledge, this is a novel use of this class of models that has not been previously described in the literature. In a data-rich regime, with lots of training examples, our method can outperform a pure classification-based encoder-only approach. More importantly, the generative model appears to be far more

data-efficient: in a data-poor regime, our approach significantly outperforms BERT with limited training examples.

Echoing to the effectiveness shown in Chapter 5, the result in this work also demonstrates that the rich latent knowledge has been captured in probabilistic modeling during pretraining and how can it be utilized for downstream tasks efficiently with small amounts of labeled data. By "connecting" fine-tuned latent representations of relevance to related output "target words", we can exploit the model's latent knowledge (e.g., of semantics, linguistic relations, etc.) that has been honed through pretraining. We describe probing experiments that attempt to verify our intuitions by deliberately altering the target words to capture different aspects of "semantic relatedness".

7.2 Passage Reranking with Generative Models

7.2.1 Method Overview

Our reranking method is based on T5 [Raffel et al., 2020b], which is a sequence-to-sequence model that uses a similar masked language modeling objective as BERT to pretrain its encoder–decoder architecture. In this model, all target tasks are cast as sequence-to-sequence tasks. For our task, the input sequence is:

$$\text{Query: } \texttt{q} \text{ Document: } \texttt{d} \text{ Relevant: } , \tag{7.1}$$

where `q` and `d` are the query and passage texts respectively. The model is fine-tuned to produce the words "true" or "false" depending on whether the passage is relevant or not to the query. That is, "true" and "false" are the "target words" (i.e., ground truth predictions in the sequence-to-sequence transformation).

At inference time, to compute probabilities for each query–passage pair (in a reranking setting), we apply a softmax only on the logits of the "true" and "false" tokens. Hence, we rerank the passages according to the probabilities assigned to the "true" token.

Note that T5 tokenizes sequences using the SentencePiece model [Kudo and Richardson, 2018], which might split a word into subwords. We choose target words ("true" and "false")

that are represented as single tokens; thus each class is represented by a single logit. In the case where target words are split in multiple subwords, we would need a method to aggregate their logits into a single score; it is best to avoid this complexity in the design of the reranking setup.

7.2.2 Reformulate through Information Distance

Recall that in Chapter 6, we introduce the intuition of *information distance* between x_1 and x_2 represents how many bits of information we need for x_2 given x_1. In the passage reranking setting, the intuition corresponds to that a less relevant passage needs more bits compared to a relevant passage, given the same query:

$$C(\mathtt{qd}_r) - C(\mathtt{q}) < C(\mathtt{qd}_n) - C(\mathtt{q}), \tag{7.2}$$

where $\mathtt{d}_r$ means a relevant passage and $\mathtt{d}_n$ refers to a non-relevant passage.

Typically, if we want to calculate the compressed length of $\mathtt{qd}$ of autoregressive model, we can apply entropy coding. As briefly introduced in Section 5.2.1, autoregressive models are generative models with explicit density estimation, $p(\mathbf{x}) = p(x_0)\prod_{i=1}^{n} p(x_i|\mathbf{x}_{i-1})$, making it capable of lossless compression. Applying a queue-like coding scheme (e.g., Arithmetic Coding [Witten et al., 1987b]) we have a neural compression algorithm. In practice, we can set a sliding window size, and use the probability distribution of the next predicted token as input to the coding scheme like Goyal et al. [2019b] do.

But the fact that we only need the compressed length makes actual compression unnecessary. In fact, we can use $-\log p(\mathbf{x})$ to represent the estimated compressed length. Furthermore, to avoid unfair comparison between passages with different lengths, we only use the probability distribution of the last token for the estimated code length. In our case, it's the target token "true". As $C(\mathtt{q})$ is the same for every passage given one query, we can sort $C(\mathtt{qd})$ directly to generate the ranking list of passages.

7.3 Related Work

The advent of deep learning has transformed the information retrieval community. Prior to deep learning, researchers and practitioners mostly adopt the paradigm known as "learning to rank", which is heavily driven by manual feature engineering [Li, 2011; Liu, 2009]. For example, commercial web search engines are known to incorporate thousands of features (or more) in their models. The introduction of continuous vector space representations coupled with neural models was exciting as it provides a potential path away from the need for handcrafted features. Well-known early neural ranking models include DRMM [Guo et al., 2016], DUET [Mitra et al., 2017], KNRM [Xiong et al., 2017], and Co-PACRR [Hui et al., 2018]; the literature is too vast for an exhaustive review here, and thus we refer readers to past overviews [Mitra and Craswell, 2019; Onal et al., 2018]. Interestingly, however, a meta-analysis by Yang et al. [2019] finds that without sufficient training data, these neural models still perform worse than well-tuned bag-of-words query expansion baselines.

However, in recent years, we have witnessed a dramatic shift to ranking models based on BERT [Dai and Callan, 2019; Li et al., 2020; Yilmaz et al., 2019], starting with Nogueira and Cho [2019]. Our work belongs to this large family of models based on transformers, although our exploration of a sequence-to-sequence ranking formulation based on encoder–decoder architectures sets us apart from previous classification-based formulations using encoder-only architectures.

7.4 Experimental Setup

7.4.1 Datasets

Datasets used in our experiments are listed below:

MS MARCO passage [Bajaj et al., 2016a] is a dataset with about 1M natural language queries and 8.8M passages obtained from Bing search engine results. The training set contains approximately 530K (query, relevant passage) pairs, with on average one relevant passage per unique query; non-relevant passages are also provided as part of the training set. The development and test sets

contain approximately 6,900 queries each, but relevance labels are only publicly available for the development set.

Robust04 [Voorhees, 2004] is the test collection from the TREC 2004 Robust Track. It comprises 249 topics, with relevance judgments on a collection of aroudn 528K passages (TREC Disks 4 and 5).

Core17 [Allan et al., 2017] is the test collection from the TREC 2017 Common Core Track, with relevance judgments for 50 topics on about 1.86M articles from the New York Times Annotated Corpus.

Core18 [Allan et al., 2018] is the test collection from the TREC 2018 Common Core Track, with relevance judgments for 50 topics on ∼600K articles from the TREC Washington Post Corpus.

For Robust04, Core17, and Core18, we use the topic "titles" (i.e., short keyword phrases) as queries to our bag-of-words retrieval methods and the topic "descriptions" (i.e., sentence-length statements) as input to our sequence-to-sequence models. These topic descriptions are more similar to MS MARCO's natural language questions, and others have found that using well-formed questions improves the effectiveness of pretrained reranking models [Dai and Callan, 2019].

It's worth reemphasizing that our models are *not* trained on Robust04, Core17, or Core18 data. We use their queries and relevance judgments only as held-out test sets; thus, for those collections, our evaluation adopts a zero-shot transfer setting.

7.4.2 Training and Inference

We follow Raffel et al. [2020b] and use the cross-entropy to fine-tune T5 models with various sizes (base, large, and 3B) using a constant learning rate of 10^{-3} for 100K iterations (approx. ten epochs) with class-balanced batches of size 128. To simplify our training procedure (and related hyperparameters) as well as to eliminate the need for convergence checks, we simply train for a fixed number of iterations, selected based on the computational demands of our largest model and the (self-allotted) time for running experiments. We report results using the model state at the final checkpoint. This procedure is consistent with the advice of Kaplan et al. [2020] and recommendations by Dodge et al. [2019], since we quantify effectiveness for a particular

computational budget. We use a maximum of 512 input tokens and two output tokens (one for the target token and another for the end-of-sequence token). In the MS MARCO passage dataset, none of the inputs exceed this length limitation. Training T5 base, large, and 3B take approximately 12, 48, and 160 hours overall, respectively, on a single Google TPU v3-8.

For inference, we adopt greedy decoding. Since we only use the logits of the first decoding step, beam search and top-k random sampling [Fan et al., 2018] would give the same results.

Because Robust04, Core17, and Core18 contain full-length documents, during inference it is not possible to directly feed the *entire* text at once to our model due to length restrictions. To address this issue, we first segment each document into passages by applying a sliding window of 10 sentences with a stride of 5. We then obtain a relevance probability for each passage by classifying it independently. We select the highest probability among these passages as the relevance probability of the document; that is, we do not use the original (BM25) retrieval scores.[1] This procedure is the same as the MaxP technique of Dai and Callan [2019] although our definition of passages differs.

7.4.3 Baselines

We compare against the following baselines:

BM25: For a baseline bag-of-words retrieval method, we use the BM25 implementation in the Anserini open-source IR toolkit [Yang et al., 2017],[2] which is based on Lucene. We adopt all the default settings. At inference time, we retrieve the top 1000 passages per query.

BM25+RM3: To examine the effects of query expansion, we apply the BM25+RM3 model as described in Yang et al. [2019], where it is shown to be a competitive baseline for pre-BERT neural ranking models. We use the implementation in Anserini, with all default settings.

BM25+BERT-large: We additionally compare our method against the BERT-large from Nogueira et al. [2019a], which is a two-stage pipeline with bag-of-words retrieval (BM25) followed by a

[1] We also examined the alternative of interpolating model scores with retrieval scores, but this did not improve effectiveness and additionally introduces an extra parameter to tune.

[2] http://anserini.io/

BERT reranker. Architecturally, it is the same as our method, the only difference being BERT vs. T5 as the reranking model. Nogueira et al. [2019a] can be characterized as the baseline of the best methods from the official MS MARCO passage leaderboard; all higher-ranked submissions can be described as improvements upon this basic approach, and thus it represents a competitive comparison point. Note that we do not apply reranking on top of BM25+RM3 because RM3 is known to reduce effectiveness when evaluated using these relevance judgments [Nogueira et al., 2019b].

Our T5 rerankers are applied directly to the output of BM25 (and BM25+RM3) from Anserini (1000 hits), thus providing a contrastive setup that isolates the impact of our method.

7.5 Results and Analysis

7.5.1 Ranking Results on Full Datasets

Main results on the MS MARCO passage retrieval task are shown in Table 7.1, comparing BERT-large [Nogueira et al., 2019a] to T5 models of different sizes. MRR@10 is the official metric for the task. Based on the Student's paired t-test, the effectiveness of T5-3B (bolded) on the development set is significantly better ($p < 0.01$) than BERT-large. Effectiveness increasing with larger models is an expected trend [Brown et al., 2020], and with T5-11B we might obtain an even higher MRR@10; unfortunately, we are not able to run these experiments due to their high computational costs.

Results on Robust04, Core17, and Core18 are shown in Table 7.2, where we apply our T5 reranker on top of retrieval results from BM25 and BM25+RM3 (see Section 7.4.2). The T5-3B results in bold are significantly better ($p < 0.05$) than BERT-large, T5-base, and the corresponding baseline (BM25 or BM25+RM3), based on the Student's paired t-test with Bonferroni corrections. We compare our model with Birch [Yilmaz et al., 2019], BERT–MaxP [Dai and Callan, 2019], and PARADE [Li et al., 2020], which are BERT-based models that represent the state of the art. BERT–MaxP and PARADE results are from fine-tuning on the MS MARCO data and then fine-tuning again on Robust04 (via cross-validation).[3] Birch uses Robust04, Core17, and

[3]MaxP numbers are from the reimplementation by Li et al. [2020], which are higher than the original paper due

		MS MARCO Passage	
	# Params	Dev	Test
BM25	-	0.184	0.186
+ BERT-large	340 M	0.372	0.365
+ T5-base	220 M	0.381	-
+ T5-large	770 M	0.393	-
+ T5-3B	3 B	**0.398**	0.388

Table 7.1: MRR@10 figures on the MS MARCO passage, with BERT-large figures from Nogueira et al. [2019a]. Model sizes are also shown.

Core18 for tuning weighting parameters. In contrast, we apply inference directly using our model trained on the MS MARCO passage data; Robust04, Core17, and Core18 relevance judgments are only used as a test set, which makes our results zero-shot. To our knowledge, our T5-3B model produces the highest known scores reported on these test collections.

Note that results from our T5 models have lower proportions of judged passages in the top-20 (Jdg@20) than BM25 and BM25+RM3. In other words, our models are retrieving passages that have never been evaluated, for which we have no relevance labels. Since standard evaluation tools such as `trec_eval` treat "unknown" as non-relevant, the results for our models represent a lower bound on true effectiveness. This finding confirms recent observations that test collections built before the advent of BERT-based rerankers place transformer-based models at a disadvantage [Yilmaz et al., 2020].

As we expect, effectiveness increases with larger models, but in all cases T5 improves over both a bag-of-words as well as a query expansion baseline. Note that the latter is considered to be a strong baseline, even for pre-BERT neural ranking models [Yang et al., 2019]. In many cases, we notice that the effectiveness improvement of T5-large over T5-base is small; we investigate this curious finding further in Section 7.5.3.

to additional fine-tuning on MS MARCO.

	Robust04				Core17				Core18			
Model	AP	P@20	nDCG@20	Jdg@20	AP	P@20	nDCG@20	Jdg@20	AP	P@20	nDCG@20	Jdg@20
Birch	0.3697	0.4669	0.5325	-	0.3323	0.6200	0.5092	-	0.3522	0.4920	0.4953	-
BERT–MaxP	-	-	0.5453	-	-	-	-	-	-	-	-	-
PARADE	-	-	0.5713	-	-	-	-	-	-	-	-	-
BM25	0.2531	0.3631	0.4240	0.9770	0.2087	0.4620	0.3877	0.9550	0.2495	0.4000	0.4100	0.9620
+ T5-base	0.3279	0.4464	0.5298	0.9158	0.2758	0.6070	0.5180	0.8840	0.3125	0.4700	0.4741	0.8020
+ T5-large	0.3288	0.4448	0.5345	0.8906	0.2799	0.6210	0.5356	0.9090	0.3330	0.5070	0.5057	0.8200
+ T5-3B	**0.3876**	**0.5165**	**0.6091**	0.9632	**0.3193**	**0.6530**	**0.5629**	0.9260	**0.3749**	**0.5410**	**0.5493**	0.8600
BM25 + RM3	0.2903	0.3821	0.4407	0.9764	0.2823	0.5500	0.4467	0.9620	0.3135	0.4700	0.4604	0.9390
+ T5-base	0.3340	0.4440	0.5532	0.9058	0.3067	0.6010	0.5203	0.8840	0.3364	0.4620	0.4698	0.7990
+ T5-large	0.3382	0.4446	0.5287	0.8840	0.3109	0.6070	0.5299	0.8880	0.3557	0.4900	0.5007	0.8070
+ T5-3B	**0.4062**	**0.5223**	**0.6122**	0.9588	**0.3564**	**0.6470**	**0.5612**	0.9100	**0.3998**	**0.5330**	**0.5492**	0.8540

Table 7.2: Results on Robust04, Core17, and Core18. The T5 models are trained only on MS MARCO passage data and thus represent zero-shot transfer. Jdg@20 is the percentage of top-20 retrieved passages that were judged.

7.5.2 Effect of Model Size and Training Data

Results from the MS MARCO passage ranking task in Table 7.1 represent a direct comparison between BERT and T5 as the retrieval pipeline is otherwise the same. For Robust04, Core17, and Core18, we adopt a different architecture than PARADE, BERT–MaxP, and Birch, but effectiveness clearly improves as the size of the T5 model increases, as shown in Table 7.2. While T5 achieves better results, it is possible that the improvements come from simply having a bigger model, as opposed to any intrinsic advantages over an encoder-only architecture. Since we do not have pretrained T5 and BERT models of comparable sizes, it is difficult to conduct a fair empirical comparison. However, we do note from Table 7.1 that T5-base outperforms the larger BERT-large model.

Another important dimension of size is the amount of training data available, as it is often expensive to annotate high-quality data for information retrieval. In Figure 7.1, we report the results of experiments fine-tuning BERT-base and T5-base with 1K, 2.5K, and 10K positive instances (and an equal number of negative instances) sampled from the full MS MARCO passage dataset. We select these two "base" models due to their more modest computational demands for fine-tuning. We train them using a batch size of 32 for three epochs. For BERT, we use a learning

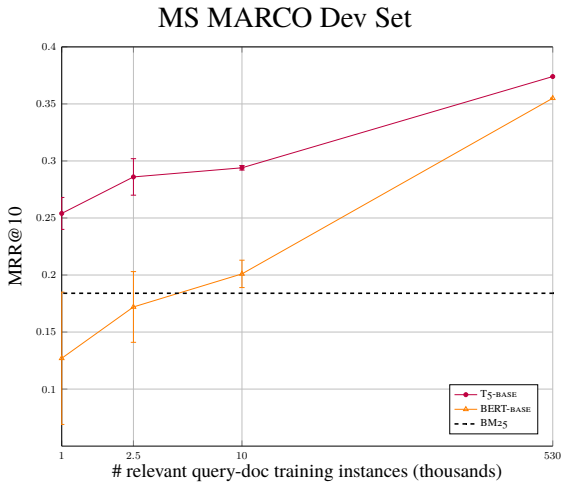


Figure 7.1: Comparisons between T5-base and BERT-base trained with different numbers of training instances (note the log scale in the x-axis). Results report means and 95% confidence intervals over five trials.

rate 10^{-6} and no warm-up step. For T5, we use a learning rate of 10^{-3}. Note that these differences in experimental methodology render the results not directly comparable to those in Table 7.1. For all conditions (2K, 5K, and 20K samples in total), we repeat the experiment five times, drawing different samples each time; the 95% confidence intervals are shown in Figure 7.1. We run the setting with 530K training instances only once due to its high computational cost.

As we expect, effectiveness improves as we fine-tune both models with more data. Interestingly, in a data-poor setting with only a modest amount of training data, T5 can learn far more effectively than BERT. We see clearly that with the same amount of *limited* training data (10K positive instances is only about 2% of the entire dataset), T5 is significantly more effective than BM25. In fact, with only 1K positive and 1K negative training instances, BERT performs worse than the BM25 baseline (i.e., worse than just exact term matching), while T5 is 7 points better than the BM25 baseline. With 10K training instances, BERT is able to modestly improve upon BM25, but remains nine points behind T5 fine-tuned on the same amount of data. It is also intriguing that T5 is able to achieve roughly 10 points above the BM25 baseline, which accounts for nearly 60% of its total gain, with only 2% of the training data.

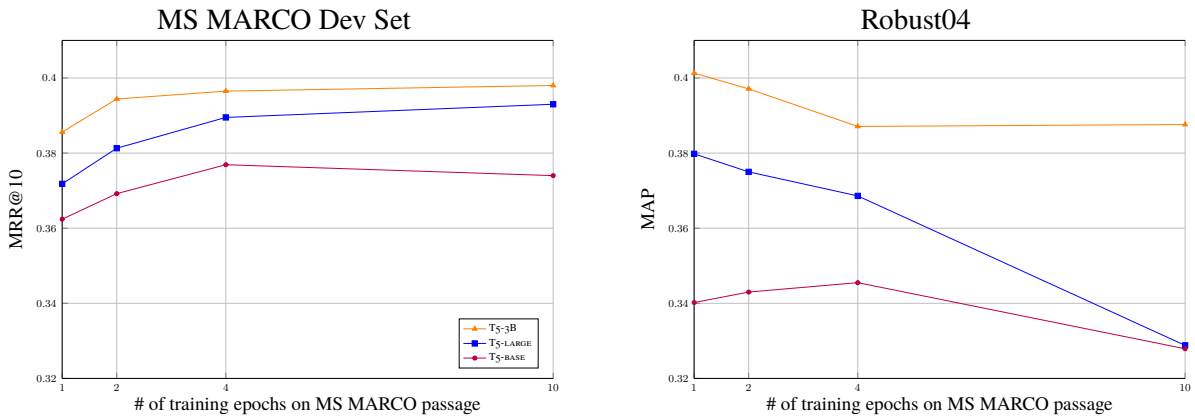


Figure 7.2: **(a)** MRR@10 vs. number of training epochs on MS MARCO. **(b)** MAP on Robust04 vs. number of training epochs on MS MARCO.

7.5.3 Effect of Checkpoint Selection

The application of our T5 approach to Robust04, Core17, and Core18 is zero shot since the model is never exposed to labeled training data from those collections.[4] We apply the fine-tuning procedure described in Section 7.4.2 and directly evaluate on those test collections. Results in Table 7.2, however, revealed an oddity: the effectiveness of T5-large is not substantially better than T5-base, contrary to our expectations. Further investigation reveals this to be an issue of "how much to fine-tune".

In Figure 7.2(a), we show MRR@10 vs. number of training epochs on MS MARCO, and in Figure 7.2(b), a similar graph for MAP on Robust04 (reranking BM25 results). On MS MARCO, effectiveness increases overall as we fine-tune the model for more epochs, with the exception of T5-base, which exhibits signs of over-training. These findings are expected. On Robust04, however, exhibits signs of over-training for all model sizes. It makes sense that fine-tuning more and more on a specific dataset would reduce the model's ability to generalize to other domains. This observation also suggests that we can obtain even better results than those in Table 7.2 if we apply our model on an earlier checkpoint.

[4]It is possible, however, that during pretraining the model was exposed to passages from the target corpus.

Proper checkpoint selection, however, requires in-domain validation data, which no longer qualifies as zero shot. We emphasize that this diagnostic experiment was conducted *after* obtaining the zero-shot results reported in Table 7.2 and thus does not invalidate our zero-shot claims. We are unsure if our observations are merely idiosyncrasies of passages ranking, or a more general problem with transfer learning using transformers. Nevertheless, this is an issue deserving further exploration.

7.5.4 Effect of Logit Normalization

As we mentioned in Section 7.2.2, calculating the probability of the target token can be viewed as a way of estimating compressed length. But there are still different ways of choosing the vocabulary — that is, whether we normalize through softmax over all the possible tokens or over "true" and "false" only. Both of them can be viewed from *information distance* perspective, as Section 7.2.2 shows, just with different vocabulary. Another possible alternative is to use the logit of the target token "true" directly. But this alternative doesn't have interpretable meaning through the *information distance* lens. Here, we investigate the effectiveness of these alternative normalization techniques.

In Table 7.3, we show T5-base results on the development set of the MS MARCO passage dataset. In the first row, we simply use the logit of the "true" token as the score of the passage. This method performs poorly, with an MRR@10 close to zero. Normalizing with a softmax over either all logits (row 2) or only the "true" and "false" logits (row 3) yields similarly high MRR@10 figures. These results demonstrate that the logits of a particular token (in this case, the "true" token) are not comparable across different examples, but they become comparable once normalized appropriately. The method in row 3 is the default method throughout the chapter because it gives slightly better results.

7.5.5 Target Token Probing Experiments

The experimental results above immediately raise two questions:

Logit Normalization Technique	MRR@10
(1) None ("true" logit only)	0.026
(2) Softmax on all logits	0.379
(3) Softmax on "true"/"false" logits only	0.381

Table 7.3: T5-base results on the development set of the MS MARCO passage dataset comparing different logit normalization techniques.

1. Why is our approach more data-efficient than BERT? That is, why does T5 significantly outperform BERT when fine-tuned with far fewer training examples?

2. How is our approach fundamentally different from classification with an encoder-only model, given that the softmax in our case reduces the model to a binary classifier?

We believe these two issues are closely related. Let's consider the second question first: At a high level, both neural models are learning latent representations important to the task at hand (in this case, relevance classification), starting from a pretrained model, and then mapping these latent representations into task-specific decisions. Thus, end-to-end effectiveness depends on a combination of the knowledge imparted via pretraining (already present at the start) and the knowledge gained via fine-tuning on task-specific data. In the classification-based approach using BERT proposed by Nogueira and Cho [2019], the model relies on a single fully-connected layer to map the latent representation (i.e., the `[CLS]` token) into this binary decision. While the approach can exploit pretrained knowledge when fine-tuning the latent representations, the final mapping (i.e., the fully-connected layer) needs to be learned from scratch (since it is randomly initialized).[5]

In contrast, T5 can exploit both pretrained knowledge and knowledge gleaned from fine-tuning in learning task-specific latent representations as well as the mapping to relevance decisions; specifically, we note that T5 is pretrained with tasks whose outputs are "true" and "false". Unlike the fully-connected layer in the encoder-only approach, T5 can exploit the part of the network used for generating output tokens. Embedded in that neural machinery is latent knowledge about

[5]While other models such as PARADE [Li et al., 2020] layer additional neural components on top of BERT, our basic argument still holds since these additional parts of the model are also randomly initialized.

	Target Token		Training Size (query-relevant doc pairs)		
Type	Relevant	Non-Relevant	1K	10K	530K (all)
Baseline	true	false	0.254 ±0.014	0.294 ±0.002	0.374
Alternate	yes	no	0.218 ±0.040	0.301 ±0.004	0.378
Reverse	false	true	0.243 ±0.025	0.282 ±0.006	0.374
Antonyms	hot	cold	0.240 ±0.021	0.246 ±0.005	0.375
Related Words	apple	orange	0.206 ±0.026	0.260 ±0.003	0.376
Unrelated Words	hot	orange	0.194 ±0.018	0.242 ±0.005	0.377
Subwords	_ab	_de	0.179 ±0.014	0.228 ±0.005	0.377

Table 7.4: Results with T5-base on the development set of the MS MARCO passage dataset comparing different target token manipulations.

semantics, linguistic relations, and lexical features that are necessary to generate fluent text. In other words, T5 has access to an additional source of knowledge that BERT does not.

This explanation, we believe, also answers the first question. With plenty of training data, BERT has no trouble learning the final fully-connected layer (mapping latent representations to decisions), even from scratch (i.e., random initialization). However, faced with few training examples, BERT still must learn the classification layer, but without any benefit from pretraining—and the experiments in Figure 7.1 show that it is unable to do so effectively. In contrast, in a low-data setting, T5 can "fall back" on pretrained neural machinery for generating fluent textual output. In other words, the pretraining objective in T5 seems to transfer well to *generating* relevance labels, which resonates with the main findings in Chapter 5.

To turn our intuition into testable hypotheses, we can vary the target tokens used as the prediction targets and manipulate their "linguistic relatedness"—to deliberately "disrupt" linguistic knowledge that may be captured in the model. As Puri and Catanzaro [2019] show, the choice of target tokens impacts effectiveness. Recall that in our baseline, "true" indicates a relevant passage and "false", a non-relevant passage. We investigate the following contrastive variants:

- **"Alternate".** Instead of "true" and "false", we use "yes" and "no", respectively. Here we are probing with an equally intuitive formulation of the targets, except that these words have not been used in pretraining, and thus the model is less likely to have strong prior associations.

- **"Reverse".** We swap the target tokens; that is, "false" indicates a relevant passage and "true", a non-relevant passage. If the model is indeed exploiting latent knowledge about linguistic relations, then forcing the model to make opposite associations on the same polarity scale should lower effectiveness with respect to the baseline.
- **"Antonyms".** We map a relevant passage to "hot" and a non-relevant passage to "cold". This preserves the use of adjectives at opposite ends of a polarity scale, but a scale that is completely unrelated to relevance. If the model is exploiting latent knowledge, we would expect effectiveness to be lower than the baseline.
- **"Related Words".** We map a relevant passage to "apple" and a non-relevant passage to a related word "orange". These words are semantically related, but do not present a polarity contrast as before. We would expect effectiveness to be lower than the baseline.
- **"Unrelated Words".** We map a relevant passage to "hot" and a non-relevant passage to a completely unrelated word "orange". Thus, we force the model to build an arbitrary semantic mapping. We would expect effectiveness to be lower than the baseline and also lower than using related words.
- **"Subwords".** We map a relevant passage to the subword "_ab" and a non-relevant passage to the subword "_de". Note that we carefully select single tokens after tokenization by SentencePiece. Here, we remove all "semantics" from the input–output mapping and thus expect effectiveness to be lower than the above conditions.

Using these target token configurations, we conduct experiments on T5-base with either 1K (or 10K) positive and 1K (or 10K) negative instances sampled from the full MS MARCO passage dataset, same as in Section 7.5.2. Once again, for each of the conditions, we repeat the experiment five times, drawing different samples every time. For reference, we also fine-tune with all available data. Note that the effectiveness of T5-base is different from the values in Table 7.1 because we use slightly different (more computationally-efficient) hyperparameters: here, we train for 40K steps using a batch of size 256. Experimental results are shown in Table 7.4, with means and 95% confidence intervals.

When fine-tuning with all available data, the choice of target tokens has negligible impact on effectiveness. These small differences can be explained by the stochastic nature of the training

process. This does appear consistent with our hypothesis that with sufficient training data, T5 is able to learn arbitrary mappings between passage relevance and target tokens.

In the data-poor setting, the results are also consistent with our hypotheses. With minimal amounts of training data (the 1K condition), the confidence intervals from different samples mostly overlap (with the exception of subwords), so we do not have the benefit of greater certainty that comes with statistical significance. In the 10K condition, our target token manipulations all significantly reduce effectiveness, except for the "Alternate" condition, which performs slightly better than the baseline condition. This seems somewhat idiosyncratic, but we suspect that the prevalence of the target tokens in the data used for pretraining might have an impact: yes/no appear more often in the pretraining corpus than true/false. Overall, it is clear that the semantics of the target tokens, even small differences, can affect the overall effectiveness of the model. The "Unrelated Words" and "Subwords" conditions are clearly less effective. Finally, we note that the 95% confidence intervals are smaller under the 10K condition, which illustrates the greater instability in effectiveness when training on smaller datasets (which is expected).

These results support our hypothesis that T5 is exploiting latent knowledge to aid in predicting relevance. As the strongest piece of evidence, in the 1K condition, "Subwords" performs worse than the BM25 baseline; i.e., it exhibits difficulty achieving any predictive power at all. There are at least two potential factors at play: we are removing semantic associations, as the subwords are token fragments, and furthermore, we are forcing the model to produce tokens in an order (and context) that it has not encountered during pretraining. We are unable to tease apart these effects currently, but either explanation is consistent with our intuitions. For all other target token manipulations, we are at least able to beat the BM25 baseline under the 1K condition.

Finally, our experiments are inconclusive regarding the importance of having a polarity scale in the low-data setting. Quite clearly, reversing "true" and "false" has a noticeable impact (especially in the 10K condition), but T5 is more effective learning targets that are semantically related but do not present a polarity contrast ("apple" and "orange") than targets that encode an unrelated polarity contrast ("hot" and "cold"). Due to computational limitations (primarily from the number of trials necessary to obtain confidence intervals), we experiment with only one target token pair for each category; additional trials with different targets will be required to draw firmer conclusions.

7.6 Summary

In this chapter, we restrict representation by doing compression but without the process of actual compression. We take the advantage of pretrained language model's exceptional ability in capturing probabilistic distribution by introducing a novel generation-based approach to passage reranking. Our models outperform a classification-based encoder-only approach, especially in the low-data regime with limited training data. We attempt to explain these observations in terms of hypotheses about the knowledge that a model gains from pretraining vs. fine-tuning on task-specific data. These hypotheses are operationalized into target token probing experiments, where we demonstrate that the model appears to exploit knowledge from its ability to generate fluent natural language text. Overall, leveraging the perspective of compression helps us exploit the pretrained generative models effectively especially in a low-data regime.

Chapter 8

Restrict Representations by Compression without Actual Compression under No Supervision

In Chapter 7, we show how pretrained language models can be utilized in a generative way for NLP tasks. However, the requirement of fine-tuning set two major limitations for this method: (1) high demand for computational resources required; (2) a fair amount of labeled data needed. The heavy reliance on a single token's log probability in Chapter 7 creates rigid format requirements that need to be fine-tuned on about one thousand labeled data in order to unleash the power of pretrained language models. In this chapter, we introduce how to utilize large language models (LLMs) in a non-parametric way under one-shot and zero-shot settings, echoing with the possibility of using log-likelihood as estimated compressed length for autoregressive models proposed in Chapter 5.

To be specific, we present a novel approach that utilizes the Generative Pretrained Transformer (GPT) to approximate Kolmogorov complexity, to estimate the optimal *information distance* for few-shot learning. We first propose using GPT as a prior for lossless text compression, achieving a noteworthy compression ratio. Experiment with LLAMA2-7B backbone achieves a compression ratio of 15.5 on enwik9. We justify the pretraining objective of GPT models by demonstrating its equivalence to the compression length, and, consequently, its ability to approximate the *information distance* for texts. Leveraging the approximated *information distance*, our method allows

for the direct application of pretrained GPT models in quantitative text similarity measurements. Experiment results show that our method in most cases achieves superior performance compared to fine-tuning and prompt baselines on challenging NLP tasks, including semantic similarity, zero and one-shot text classification, and zero-shot passage reranking with *no* parameters.

This work is in submission:

- Cynthia Huang*, Yuqing Xie*, Zhiying Jiang*, Jimmy Lin, Ming Li. Approximating Human-Like Few-shot Learning with GPT-based Compression. *In Submission, 2023.*

8.1 Overview

Large labeled datasets are often scarce in the real world where annotation is expensive and time consuming. This has prompted the development of few-shot learning, where the model is learned using only a few annotated samples [Finn et al., 2017]. One resort to the few-shot scenario is to utilize the pretrained models like Generative Pretrained Transformers (GPTs) [Brown et al., 2020; OpenAI, 2023; Radford et al., 2019] with in-context learning [Brown et al., 2020; Zhao et al., 2021], fine-tuning [Liu et al., 2022] or the combination [Ben-David et al., 2022]. However, in-context learning requires heavy engineering to achieve a high accuracy [Liu et al., 2023], and its ability to generalize to different tasks is constrained by the input size and the need for precise formatting. On the other hand, fine-tuning also has its limitations, the most notable of which is its inability to generalize to out-of-distribution datasets when labeled samples are extremely scarce [Nogueira et al., 2020; Yu et al., 2021].

Extending the work in Chapter 5, Chapter 6, we use information distance to help us utilize the distribution captured in LLMs and exploit it using as few labeled data as possible without parameters. Recall that *information distance*, consisting of Kolmogorov complexity [Li and Vitányi, 2008], is both data-type-agnostic and distribution-agnostic. Moreover, its capability to be used without any parameter tuning allows this metric to be generalized across different domains. Building on this *information distance*, we propose a novel method that leverages the knowledge that GPT models learned during pretraining to tasks traditionally challenging for prompt engineering or fine-tuning, including semantic similarity, zero-shot passage reranking and zero-shot text

classification. The premise of our method is that pretrained language models can optimize the compression ratio and thus better approximate Kolmogorov complexity. Specifically, achieving a higher compression ratio during lossless compression requires a richer prior capable of more accurate next-token predictions based on preceding content. For example, Huffman Coding employs shorter bit arrays for higher-frequency elements to yield a shorter compressed sequence. This can be further enhanced by adopting bi-gram or tri-gram Huffman Coding. Large language models can then refine the next token distribution accuracy, thereby optimizing the compression ratio.

Despite the potential of large language model-based compressors, their direct application to downstream tasks is nearly infeasible due to speed constraints. In addition to the inference speed required by the language model itself, the overhead of the compressor is even more substantial. Fortunately, the *information distance* only requires the compressed length instead of the actual compression of the text sequence. We demonstrate an equivalence of the compression length under arithmetic coding to the negative log probability for the text tokens under GPT. This easy-to-compute compression length enables us to efficiently approximate the *information distance* without the overheads of the actual compression. By approximating *normalized information distances* [Cilibrasi and Vitányi, 2003; Li et al., 2001b, 2004] using GPT-based compression, we significantly enhance GPT's ability to quantitatively capture text similarities, which forms the foundation for its application in downstream NLP tasks.

Our contributions are as follows: (1) We propose a novel way that utilizes pretrained GPT models to approximate *information distance* to solve various downstream NLP tasks, without fine-tuning or prompt engineering. (2) By connecting arithmetic coding's compression length to the cumulative negative log probabilities in GPT models, we efficiently compute and approximate the *information distance* derived from Kolmogorov complexity. (3) We validate the effectiveness of our method through experiments in semantic textual similarity, text classification and re-ranking under zero-shot and one-shot settings, exhibiting notable improvement over fine-tuning and prompt baselines. (4) We also demonstrate that our lossless text compression method GPT-AC achieves SOTA compression ratio with Llama2-7B backbone, highlighting the potential of pre-trained large language models as powerful priors in compression.

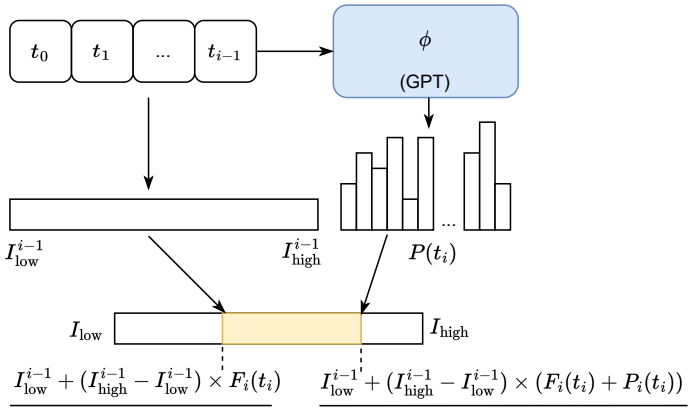


Figure 8.1: Illustration of GPT-based Arithmetic Encoding

8.2 Non-Parametric Learning by Compression with GPT

In Section 6.2 and Section 5.2, we introduce the framework of NPC that uses a compressor to approximate *information distance*. We use the same basic component here: a compressor and a compressor-based distance metric. The major difference is that we use a GPT-based compressor.

8.2.1 GPT-based Compression

In this section, we first introduce GPT-based Arithmetic Coding (GPT-AC) where GPT is integrated into adaptive arithmetic coding, and then demonstrate how to use the compressed length directly without applying a coding scheme.

GPT as the Entropy Model Consider a text $T = (t_1, t_2, ..., t_n)$, which is composed of a sequence of tokens. Let ϕ represent a GPT model, where $\phi(t_{1:(i-1)}) = P_i(t_i|t_1, t_2, ..., t_{i-1})$ models the probability distribution P_i of the next token t_i. The function $\phi(T)$ outputs all next-token probability distributions $(P_2, \cdots, P_{n+1})$. To derive the distribution for P_1, an EOS (End Of Sentence) token is added at the start of the text as t_0. For each token t_i, the associated P_i serves as the probability distribution of the entropy model for both encoding and decoding in the compressor.

GPT-based Arithmetic Encoding In the encoding phase, under the schema of adaptive arith-

metic coding, we start with an initial interval $I^0 = [0, 1)$. For each token t_i in the sequence, we calculate the cumulative distribution function $F_i(t_i)$ and $P_i(t_i)$ based on $\phi(t_{1:(i-1)})$. Then, the interval $I = [I_{\text{low}}, I_{\text{high}})$ is updated according to the range assigned to t_i:

$$\begin{aligned} I^i_{\text{low}} &= I^{i-1}_{\text{low}} + (I^{i-1}_{\text{high}} - I^{i-1}_{\text{low}}) \times F_i(t_i), \\ I^i_{\text{high}} &= I^{i-1}_{\text{low}} + (I^{i-1}_{\text{high}} - I^{i-1}_{\text{low}}) \times (F_i(t_i) + P_i(t_i)). \end{aligned} \tag{8.1}$$

After updating I for each token in the sequence, we can pick any number, say x, within the final interval to represent the entire text sequence.

GPT-based Arithmetic Decoding When decoding the encoded message x, the token t_1 can be identified by finding the range $[F_1(t_1), F_1(t_1) + P_1(t_1)]$ that includes x. The value of x is then updated by normalizing it within the range of t_1, using the formula: $x \leftarrow \frac{x - F_1(t_1)}{P_1(t_1)}$. With this updated x and the next-token probability distribution $\phi(t_2)$, we can decode the next token t_2. This process is repeated until an EOS token is encountered, indicating the end of the text. The text can be losslessly decoded using x and ϕ alone.

Negative Log Probability as Compression Length During the arithmetic encoding, the length of the interval I^i equals to $I^{i-1} * P_i(t_i)$. From an initial interval of length 1, the entire message's encoding results in a final interval with a length of $\prod_{i=1}^{n} P_i(t_i)$. The number of bits required to represent this final interval, and thus the message T, is $\sum_{i=1}^{n} -\log_2 P_i(t_i)$. This reveals a method to approximate the compression length directly without exactly performing the compression. With the triangular forward attention masking utilized by GPT, we can pass the full tokenized text sequence to the model and obtain probability distributions for all tokens.

GPT Pretraining Optimizes for Compression Length The optimization target during pretraining for auto-regressive models such as GPT is defined as:

$$L(T|p_{model}) = -\log p_{model}(T) = -\log p_{model}(t_1, t_2, ..., t_n) = \sum_{i=1}^{n} -\log_2 P_i(t_i).$$

For entropy coding, $H(T) \triangleq \mathbb{E}(-\log p_{data}(T))$, defining the optimal code length. While p_{data} is often unknown, we thus use the observation $p_{\bar{data}}$ to approximate p_{data}:

$$H(T) = \mathbb{E}_{p_{data}}[-\log p_{model}(T)] \simeq \mathbb{E}_{p_{\bar{data}}}[-\log p_{model}(T)]$$

According to The Shannon–Fano–Elias coding scheme [Cover, 1999], we can construct a prefix-free code of length $-\log p_{\text{model}}(t_1, t_2, ..., t_n) + 2$ bits. Consequently, the pretraining phase of GPT models is essentially an exercise in learning a compressor that optimizes the coding length.

The Rank Coding Variant In the method outlined above, we primarily employ arithmetic coding for text compression. An alternative variant of lossless coding uses rank instead of probability [Herrera and Luo, 2021]. After the GPT model predicts the distribution of the next token, we can rank all tokens according to their probability. The target token is assigned the corresponding rank index, with tokens of higher probability having a smaller rank index. In this variant, we approximate the compression length as $\sum_{i=1}^{n} \log_2(rank_i)$ where $rank_i$ denotes the rank for token i. We call this variant "Log-Rank".

8.2.2 Information Distance Approximation and Variation

Having computed the compression length using the GPT-AC method, we can now utilize it to approximate the *information distance*. Let $x = \{x_1, \cdots, x_n\}$ and $y = \{y_1, \cdots, y_m\}$ denote two tokenized text sequences, where each x_i or y_i represents a token in the sequence. We approximate $K(x)$ using the compression length $C(x) = \sum_{i=1}^{n} -\log_2 P_i(x_i)$ where P_i represents the probability distribution for x_i based on the GPT model prediction.

In both Section 5.2 and Section 6.2, we approximate $K(xy)$ instead of $K(x|y)$ as concatenation provides more flexibility. But autoregressive models make conditioning on input feasible. We can approximate $K(x|y)$ as follows: let $P'_i = \phi(y, x_{1:i-1})$ denotes the probability distribution for token x_i outputed by the GPT model, given $y = (y_1, \cdots, y_m)$ and previous tokens in x, $K(x|y)$ can be estimated as $C(x|y) = \sum_{i=1}^{n} -\log_2 P'_i(x_i)$. A similar approach can be used to estimate $K(y)$ and $K(y|x)$. We denote all compressed-based approximations in $C(\cdot)$.

Similar to previous sections, we use Normalized Information Distance (NID). To make the notation align with other variations, we use $\mathcal{M}_{max}$ to represent it:

$$\mathcal{M}_{max}(x, y) = \text{NID} = \frac{\max\{K(x|y), K(y|x)\}}{\max\{K(x), K(y)\}} \tag{8.2}$$

To tackle challenges such as partial matching [1] , Bennett et al. [1998] proposed the following

[1] Partial matching means situations where only portions of two objects match each other.

variants of the universal distances for broader application scenarios:

$$\mathcal{M}_{min}(x,y) = \frac{\min\{K(x|y), K(y|x)\}}{\min\{K(x), K(y)\}}. \tag{8.3}$$

Keogh et al. [2004a] proposed the Compression-Based Dissimilarity Measure (CDM) for data mining applications, demonstrating its effectiveness in practice. We rescale it to fit within the range $[0, 1)$:

$$\mathcal{M}_{mean}(x,y) = \frac{C(x|y) + C(y|x)}{C(x) + C(y)} = 2 * CDM - 1, CDM = \frac{C(xy)}{C(x) + C(y)} \tag{8.4}$$

8.3 Related Works

8.3.1 Few-shot Learning

Prior to the emergence of large pretrained models, the majority of previous works on few-shot learning can be divided into two streams: meta/transfer-learning based methods [Edwards and Storkey, 2016; Snell et al., 2017b; Sung et al., 2018; Vinyals et al., 2016] and data augmentation based methods [Douze et al., 2018; Gao et al., 2018; Kwitt et al., 2016; Pfister et al., 2014]. However, the former relies on constraining the hypothesis space by prior knowledge from other tasks or support datasets while the later depends on the prior knowledge of the datasets, often accompanied with the smoothness assumption [Van Engelen and Hoos, 2020] (i.e., closeness of data points in the input space shares the same label). Pretrained models, on the other hand, have incorporated prior knowledge during the pretraining stage and are proved to be excellent few-shot learners [Brown et al., 2020]. However, pretrained models suffer from (1) high computational cost and (2) unsatisfactory performance in out-of-distributed datasets [Yu et al., 2021]. The problem of computational cost is especially prominent for large language models like GPT-3 where it is infeasible to fine-tune locally. We utilize pretrained language model for one-shot and zero-shot classification tasks with no fine-tuning required.

8.3.2 Kolmogorov Complexity and Compression Distance

Information distance was first proposed by Bennett et al. [1998] as a universal metric. Due to the problem of incomputability, Chen et al. [2004]; Cilibrasi and Vitányi [2003]; Li et al. [2004] have derived computable versions of *information distances* for tasks like clustering and plagiarism detection, shedding light on the possibility of using real-world compressors to approximate Kolmogorov complexity. Recently, Jiang et al. [2022a] propose non-parametric learning by compression with latent variables (NPC-LV) where neural networks are incorporated into *information distance*. They demonstrate that trained generative models like variational autoencoders can be used directly with zero parameters for downstream few-shot image classification. However, it remains open in how to incorporate pretrained language models into this framework, which we aim to address in this chapter.

8.3.3 Neural Compression

Our GPT-based compressor belongs to the neural compression where neural networks are used for data compression. Shannon's source coding theorem [Shannon, 1948] establishes the limit of the lossless compression on random variables with probability distribution. With the near optimal coding schemes, the bottleneck is the entropy model. Fortunately, deep generative models with explicit density estimation serve as the entropy model that can learn adaptively. Townsend et al. [2019a] propose Bits-Back with Asymmetric Numeral Systems (BB-ANS), a lossless compression algorithm based on VAE. Bit-Swap [Kingma et al., 2019b] further improves BB-ANS by incorporating multiple latent variables and hierarchical networks. In addition to autoencoders, Flow [Rezende and Mohamed, 2015; Wang et al., 2022]-based lossless compression [Hoogeboom et al., 2019] outperform Bit-Swap and achieve the state of the art compression ratio. The development of deep neural networks also benefit lossless text compression. Goyal et al. [2019b] use recurrent neural networks [Schuster and Paliwal, 1997] combining with arithmetic coding [Witten et al., 1987a] to achieve higher compression ratio than *gzip*. Recent advancements, such as the fast transformer-based general-purpose lossless compressor TRACE [Mao et al., 2022], have demonstrated promising results in enhancing compression performance with transformer architecture.

8.3.4 Pretrained Models

Pretraining has been adopted in numerous deep learning models with the rise of transformer [Vaswani et al., 2017] due to its ability of learning task-agnostic representation. In NLP, encoder-only transformer like BERT [Devlin et al., 2019] has achieved the state of the art performance on GLUE benchmark [Wang et al., 2018a]including tasks like natural language inference and sentiment analysis with only MLP and fine-tuning. Decoder-only transformer like GPT [Brown et al., 2020; Radford et al., 2019] can treat downstream discriminative tasks in a generative way. However, previous works on few-shot learning using language models are either prompt-based [Li and Liang, 2021; Perez et al., 2021; Petroni et al., 2019] or fine-tuning-based [Nogueira et al., 2020; Yamada et al., 2020; Zhao et al., 2021] while we propose a new way to leverage pretrained language models for few-shot learning without fine-tuning or additional parameters.

8.4 Experimental Setup

Our experimental evaluation consists of four key components: lossless text compression and three downstream tasks, namely semantic textual similarity, text classification, and passage reranking. For downstream applications, we mainly conduct experiments with the GPT-2 small (124M)[2] and BERT-base-uncased(110M) models from HuggingFace Hub[3].

We now introduce how the distances discussed in Section 8.2.2 can be applied to various NLP tasks. To calculate the text similarity score between two texts x, y, we first compute their individual compression lengths $C(x), C(y)$. We also calculate the joint and conditional compression lengths $C(xy), C(x|y), C(y|x)$. Subsequently, we use these values to compute the distance metrics defined in Section 8.2.2 as $\mathcal{M}$. We then apply these distance measures to specific tasks.

- For semantic textual similarity, we treat the two sentences as x and y, and use the metric $\mathcal{M}$ to predict similarity.

[2] https://huggingface.co/gpt2 We opted for this version mainly due to the lack of complete access to the token probability distribution in the APIs of newer versions.

[3] https://huggingface.co/gpt2, https://huggingface.co/bert-base-uncased

Model →	GPT-AC (Ours)				*gzip*	7z	DZIP	TRACE
Dataset ↓	Llama2-7B	L-744M	M-355M	S-124M				
Enwik9	**15.56**	8.05	7.71	6.53	3.09	4.35	4.47	5.29
BookCorpus	**10.55**	8.34	7.89	7.22	2.77	3.80	3.95	4.58

Table 8.1: Compression Ratio by Compression Method. Note that the compression ratio equals to *Original text length* / *Compressed text length*. Under GPT-AC, we also list the model's size (L,M,S) and its number of parameters.

- For zero-shot text classification, we treat the label descriptions as x and the multiple-choice options as y. For one-shot text classification, we consider the training sample as x and the test sample as y. We classify the text sample by comparing $\mathcal{M}$ for different classes.
- For passage reranking, we treat the documents as x and the query as y, we perform ranking according to $\mathcal{M}$.

8.5 Results and Analysis

8.5.1 Lossless Text Compression

In the Lossless Text Compression task, we assess our method on the Enwik9 [Singh et al., 2020] and BookCorpus [Zhu et al., 2015] datasets. GPT-AC[4] is benchmarked against both traditional methods, such as *gzip* [Deutsch, 1996], and contemporary neural network-based techniques like DZIP [Goyal et al., 2019a] and TRACE [Mao et al., 2022].

As shown in Table 8.1, GPT-AC significantly outperforms both conventional compression methods like *gzip* and 7z, in terms of the compression ratio. Even with the GPT-2 small model, GPT-AC achieves a more than 2-fold enhancement in compression ratio compared to the widely-used GZIP, on both the Enwik9 and Book datasets. On Enwik9, GPT-AC with Llama2-7B records

[4]In actual implementation, GPT-AC processes every chunk of 2500 characters independently. Though this may slightly compromise the compression ratio, it enables parallel computing.

Dataset	# Test	GPT-AC (Ours)	*gzip*	GPT-emb	BERT-emb
STS-12	3,108	40.2	**50.4**	5.4	30.9
STS-13	1,500	**66.0**	48.4	14.6	59.9
STS-14	3,750	**55.3**	43.3	10.9	47.7
STS-15	3,000	**70.3**	59.1	9.6	60.3
STS-16	1,186	**69.5**	59.4	26.8	60.3
STS-b	1,379	**55.0**	50.7	12.4	47.3

Table 8.2: Semantic Textual Similarity Performance. Spearman Rank Correlation ρ between the distance metrics and given labels for the STS datasets. $\rho * 100$ is reported.

a compression ratio of 15.56, a 67% enhancement over the previous state of the art of 9.33, based on NNCP [Bellard, 2021] [5]. As the language model increases, the compression ratio consistently improves, suggesting that larger and better-trained language models will further amplify these results.

8.5.2 Semantic Textual Similarity

For Text Semantic Modeling, we test the models on the Semantic Textual Similarity benchmark (STS-b) [Cer et al., 2017]. The datasets consist of sentence pairs with labels from 0 to 5 indicating semantic relatedness. We compare GPT-AC against GPT-2 [Radford et al., 2019], taking the last token embedding vector, and BERT [Devlin et al., 2019], taking the averaged token embedding vector[6]. We then calculate the cosine similarity between these vectors to serve as the distance measure. For *gzip*, we follow Section 6.2 and use the normalized compression distance as the metric.

As shown in Table 8.2, our method substantially outperforms the cosine similarity distance metrics derived from GPT-2 embeddings and also shows moderate enhancement over those utilizing BERT embeddings. These results demonstrate the effectiveness of the approximated *infor-*

[5] http://mattmahoney.net/dc/text.html

[6] These embeddings have proven to be effective in previous studies [Sheng et al., 2022].

Dataset↓	# C	Model→ Domain↓	GPT-AC (Ours)	*gzip*	GPT-prompt	GPT	BERT	(SBERT)
Zero-shot Multiple Choice								
PIQA	2	Reasoning	**61.5**	53.4	50.5	49.2	50.1	(56.5)
CaseHOLD	5	Legal	**58.3**	52.4	20.3	19.9	35.0	(50.6)
One-shot								
AGNews	4	News	**47.8**±3.3	30.2±3.0	47.2±2.9	37.7±7.2	45.5±3.1	(45.8±10.2)
Medical	5	Bio-Med	**27.9**±3.2	25.6±2.8	22.1±1.2	23.7±3.5	23.8±4.8	(39.7±9.1)
SST5	5	Sentiment	26.8±3.1	21.2±2.7	**29.8**±1.6	22.7±2.8	21.1±3.3	(26.2±2.3)
Banking77	77	Finance	**34.0**±1.3	20.3±1.5	-	21.7±1.7	24.5±3.9	(53.1±1.9)

Table 8.3: Text Classification Accuracy (100%). We report the averaged accuracy across 5 runs with different random seeds, together for the standard deviations. This does not apply to zero-shot experiments because the models do not contain randomness.

mation distance in capturing semantic similarities.

8.5.3 Text Classification

For Text Classification, we evaluate the models on PIQA (Physical Interaction: Question Answering [Bisk et al., 2020]) and CaseHOLD (Case Holdings On Legal Decisions [Zheng et al., 2021]) for zero-shot classification, and SST-5 (sentiment analysis) [Socher et al., 2013], Medical abstracts [Schopf et al., 2023], AG-News (news headlines) [Zhang et al., 2015b], and Banking77 (banking and financial) [Casanueva et al., 2020] for one-shot classification. We compare our method with two main approaches: 1) fine-tuning GPT-2 or BERT with a classification layer, 2) in-context learning with GPT-2, denoted as GPT-prompt (detailed settings are in Appendix K), and 3) cosine similarity using Sentence-BERT[7] embeddings as a metric for classification.

As depicted in Table 8.3, in the zero-shot multiple-choice classification context, the *information distance* approximated by GPT-AC delivers superior results compared to cosine similarity distance metrics based on the embeddings from GPT-2, BERT, and even SBERT. Note that

[7] all-MiniLM-L12-v2

SBERT, which is fine-tuned on 1 billion high-quality labeled sentence pairs, does not fall under our setting of NS-FSL; it is included to provide a point of reference.

In one-shot text classification, our method surpasses both fine-tuned GPT and BERT on all datasets. Additionally, our method also outperforms the GPT-prompt version in all datasets except SST-5. Given that SST-5 is a widely used classification benchmark, we hypothesize that the superior performance of the prompt approach could be due to data leakage during GPT pre-training. Moreover, we did not apply the GPT-prompt method to the banking77 dataset because accommodating one-shot samples of 77 classes [Casanueva et al., 2020] within the GPT-2 prompt proves challenging, and adjusting the prompt can be complex. This issue represents a significant hurdle when applying GPT-2 with in-context learning.

8.5.4 Passage Reranking

For Passage Reranking, we evaluate the models on different domain-specific zero-shot text retrieval datasets, including Trec-Covid [Voorhees et al., 2021], Trec-News [Soboroff et al., 2019], SciFact [Wadden et al., 2020], BioASQ [Tsatsaronis et al., 2015], FiQA-2018 [Maia et al., 2018], and ArguAna [Wachsmuth et al., 2018]. Given a query, we first retrieve the top relevant document with BM25 [Robertson et al., 1995] with Elastic Search API[8]. We then rerank the documents with the models. We compare our system with the original BM25 ranking, and the Dense Passage Retrieval (DPR) [Karpukhin et al., 2020a] model, a BERT-based model already fine-tuned on the MS MARCO [Campos et al., 2016] for ranking, and a text *gzip* compressor.

As shown in Table 8.4, our proposed method outperforms the BERT-based DPR model across all settings. Despite the DPR model being fine-tuned on the massive labeled MS MARCO dataset, our performance remains superior. We also benchmark our model against the *gzip* compression method. The improvements observed indicate that GPT-AC can provide significant semantic information, leading to improved ranking results. Notably, BM25 is a strong baseline in that the domain-specific texts can contain many out-of-distribution terms, potentially hampering the performance of the language model. Despite this, our method demonstrates comparable performance across majority of the datasets and surpasses BM25 in certain domains, particularly in Bio-Medical and Finance, which requires much domain-specific understanding.

[8] https://github.com/elastic/elasticsearch

Dataset ↓	# Test	Model → Domain	GPT-AC(Ours)	BM25	*gzip*	(DPR)
TREC-COVID	50	COVID	**0.694**	0.656	0.447	(0.332)
TREC-NEWS	57	News	0.225	**0.398**	0.142	(0.161)
SciFact	300	Scientific	0.648	**0.665**	0.053	(0.318)
BioASQ	500	Bio-Med	**0.517**	0.465	0.157	(0.127)
FiQA-2018	648	Finance	**0.239**	0.236	0.032	(0.112)
ArguAna	1406	Argument	**0.327**	0.315	0.073	(0.175)

Table 8.4: Text Re-Ranking Effectiveness (NDCG@10). Retrieving top 100 relevant passages using BM25 and re-ranking using GPT-AC. We present the best result among different metric combinations.

8.5.5 Comparison of Distance Metrics

In Table 8.5, we compare the results of all distance metrics with two different coding variants. As stated in Section 8.2.2, we always take the longer sequence results with $\mathcal{M}_{max}$, the shorter sequence results with $\mathcal{M}_{min}$, and the average over both of them with $\mathcal{M}_{mean}$. We aim to avoid bias towards any single sequence when both texts contain similar or equally important information, suggesting that $\mathcal{M}_{mean}$ is the most suitable. This intuition is justified by the results for STS and one-shot classification. However, the approach differs for re-ranking. Typically, queries contain fewer than 20 tokens, while documents often contain hundreds of tokens. We thus want to focus the measurement on the query part, without being disturbed by the extra or non-relevant information in the document. Therefore, $\mathcal{M}_{min}$ is the preferred choice. ArguAna, however, is a unique case as both the query and the document contain about 250 tokens with similar content, making $\mathcal{M}_{mean}$ more suitable. For zero-shot classification, despite the varying content and text length between the two sequences, both pieces of information remain important, making $\mathcal{M}_{mean}$ the optimal choice. Finally, we observe $\mathcal{M}_{mean}$ works better with Log-Rank, while $\mathcal{M}_{min}$ works better with Log-Prob. However, no specific trend is evident for a broader conclusion.

			Log-Prob			Log-Rank		
Dataset	$len(x)$	$len(y)$	$\mathcal{M}_{max}$	$\mathcal{M}_{min}$	$\mathcal{M}_{mean}$	$\mathcal{M}_{max}$	$\mathcal{M}_{min}$	$\mathcal{M}_{mean}$
Semantic Textual Similarity (Spearman Rank Correlation $\rho * 100$)								
sts-b	13	13	44.7	47.6	48.2	50.4	54.2	**55.0**
Zero-shot Classification (Accuracy %)								
PIQA	24	10	60.0	56.4	**62.0**	59.5	55.0	61.5
CaseHold	29	219	56.0	27.2	57.9	55.2	27.6	**58.3**
One-shot Classification (Accuracy %)								
AGNews	53	53	41.3	41.1	**47.8**	40.5	39.9	43.3
Medical	296	268	25.4	27.0	**27.9**	25.5	26.3	27.7
SST5	23	24	24.8	26.2	26.7	24.3	26.0	**26.8**
Banking77	15	14	30.6	25.2	33.9	29.8	24.7	**34.0**
Re-ranking (NDCG@10)								
TREC-COVID	303	17	0.459	0.655	0.467	0.473	**0.694**	0.489
TREC-NEWS	808	16	0.173	0.161	0.167	0.184	0.161	**0.186**
BioASQ	304	14	0.306	**0.517**	0.364	0.267	0.507	0.330
FiQA-2018	247	15	0.128	**0.239**	0.154	0.118	0.222	0.143
ArguAna	276	247	0.277	0.257	0.301	0.282	0.262	**0.307**
SciFact	345	21	0.389	**0.648**	0.519	0.351	0.635	0.478

Table 8.5: Distance metric Analysis. We also list the token length of both x and y

8.5.6 Information Distance and Classification Accuracy

In Figure 8.2, we aim to illustrate the performance variance when the test cases have different distance scoring. For each test case, we compute the prediction distance ratio $R_{pred}(x) = \frac{\mathcal{M}(x,D_{c^*})}{\frac{1}{|C|}\sum_c \mathcal{M}(x,D_c)}$. Here, D_c represents the one-shot example in class c, C embodies all the classes, and c^* stands for the class predicted under metric $\mathcal{M}$. A smaller R_{pred} suggests that the predicted class is more distant from the average distance. We then categorize all the test samples according to their R_{pred} value, with each group containing 10% of the data. In Figure 8.2, the x-axis represents the average R_{pred} within each group, and the y-axis represents the group accuracy. The plot

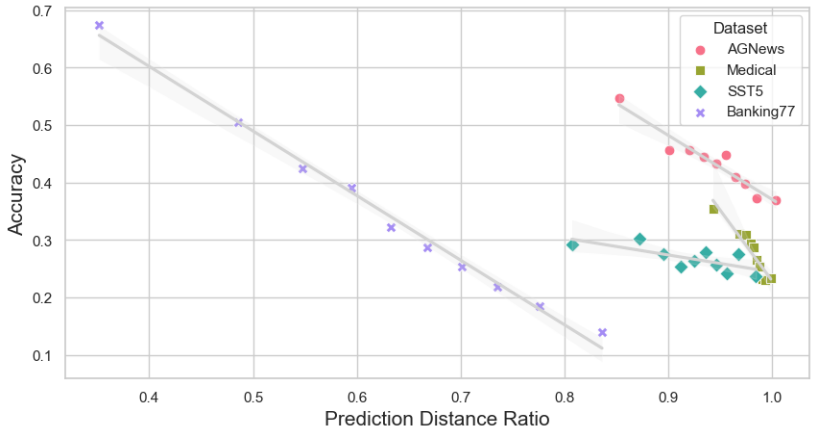


Figure 8.2: Relation between Prediction Distance Ratio and One-shot Classification Accuracy. Experiment result under $\mathcal{M}_{mean}$ with Log-Prob.

indicates that the further the predicted class deviates from the average, the better performance of our method.

8.6 Summary

In this chapter, we introduced GPT-based Compression, a novel approach that employs GPT models to estimate the optimal *information distance* for few-shot learning, under the NS-FSL setting like in Chapter 5. We demonstrated how the pretraining objectives of GPT models align with the minimization of compression length and proposed an efficient method for approximating this length using negative log probability. By applying this compression technique, we are able to approximate the *information distance* and demonstrate its efficacy in various downstream NLP tasks, including semantic similarities, zero-shot and one-shot text classification, and zero-shot passage reranking with *no* parameters.

Chapter 9

Conclusion and Future Work

9.1 Conclusion

Deep neural networks have achieved remarkable performance in various fields. The pace of progress is even faster with the help of pre-trained models. Powerful as pre-trained models are, they face two major challenges: interpretability and generalizability. We approach both two problems by restricting representation.

We propose two major directions to restrict representation — restricting the representation in the middle (i.e., inserting a bottleneck) and restricting the representation at the end (i.e., doing compression). We show the effectiveness of restricting representation for both interpretability and generalizability. Additionally, we also show the connection between inserting bottleneck and doing compression. Specifically, our main contribution can be summarized as follows:

- We implement an attribution method with information bottleneck to analyze the contribution of features to the prediction that is model-agnostic (i.e., regardless of whether the models are pretrained or not), excels in quantitative evaluation and is interpretable in qualitative analysis.
- With an attribution method that can faithfully generate attribution maps, we are able to carry out deep analysis of tasks like passage re-ranking using BERT. Through the cross-layer and

cross-passage analysis, we are able to find: (1) BERT still cares about token-wise matching but is better at capturing contextual information; (2) BERT fine-tuned on MSMARCO has a positional bias towards the start of the passage; (3) Top-ranked passages are robust to token-removal.

- Cross-layer analysis through multiple tasks on multiple-sized BERT shows that $\frac{2}{3}$ of all the layers — layer 8 for `bert-base` and layer 16 for `bert-large` are the most informative layer.

- We unify previous works on compression-based methods into a framework Non-Parametric learning by Compression (NPC), based on which we develop a new framework Non-Parametric learning by Compression with Latent Variables (NPC-LV) that are flexible with multiple replaceable modules. This framework can utilize a trained generative model with no parameters for classification with few-labeled data efficiently — it outperforms VGG [Simonyan and Zisserman, 2014] by up to 59.0% on CIFAR-10 in the few-shot setting.

- We demonstrate how negative lower bound (nELBO) can be used directly for classification as it can be used to estimate the compressed length.

- We reveal the correlation between compression rate and classification accuracy, which suggests that the improvement of generative models may further improve the classification.

- We showcase the effectiveness of NPC on text classification tasks — with absolute *zero* training, *no* pre-processing, and a simple compressor like *gzip*, we are able to achieve the text classification result competitive to non-pretrained deep neural networks on six in-distributed (English) datasets and outperform both BERT and mBERT on all five Out-Of-Distributed (OOD) datasets in both full-dataset and few-shot scenario.

- We achieve the state-of-the-art result on text classification for four low-resource languages (`Kinyarwanda`, `Kirundi`, `Dengue`, `Swahili`), outperforming KinyaBERT [Nzeyimana and Rubungo, 2022], a pretrained model specialized in the language, in the few-shot scenario.

- We demonstrate the generative-model-based passage reranking method can be viewed from the NPC perspective and it outperforms encoder-based methods significantly with weak supervision.

- We extended the method of NPC to GPT models that perform well with almost no supervision. We show that this method can be used for various NLP tasks like semantic similarity, zero-shot text ranking and classification, and one-shot classification without any parameters. We've found that the compression-based methods leverage the prior knowledge better than in-context learning and fine-tuning under extremely limited or no labeled data.

9.2 Future Work

To further investigate interpretability and generalizability with restricted representation, here are a few future directions worth investigating.

9.2.1 Control Restrictiveness by Restricting Latent Representation

The superficial relationship between information bottleneck in Section 2.1.1 and variational autoencoder in Section 5.2.1 is that they are different ways of restricting information. However, information bottleneck and negative evidence lower bound (nELBO), the optimization function that we derive from utilizing variational autoencoder, share a deeper connection through an information-theoretic perspective.

In an unsupervised scenario, the information bottleneck is just the mutual information between the input X and the latent representation Z:

$$\mathrm{I}(X;Z) = \mathbb{E}_x[D_{KL}[p(z|x)\|p(z)])] = \mathbb{E}_z[D_{KL}[p(x|z)\|p(x)]]. \tag{9.1}$$

As marginal distribution $p(z)$ is intractable, we use $q_\phi(z)$ to approximate. Thus, we have:

$$\mathrm{I}(X;Z) = \mathbb{E}_x[D_{KL}[p(z|x)\|q_\phi(z)]] - D_{KL}[q_\phi(z)\|p(z)] \leq \mathbb{E}_x[D_{KL}[p(z|x)\|q_\phi(z)]] = R, \tag{9.2}$$

where R means *rate*, measuring the relative KL divergence between encoding distribution and marginal approximation. Note that this upper bound assumes $p(z|x)$ is known.
Writing $\mathrm{I}(X;Z)$ from another direction, we use $p_\theta(x|z)$ to approximate $p(x|z)$:

$$\mathrm{I}(X;Z) = \mathbb{E}_z[D_{KL}[p_\theta(x|z)\|p(x)]] + \mathbb{E}_z[D_{KL}[p_\theta(x|z)\|p(x|z)]], \tag{9.3}$$

assuming we have inference network like the setting in Section 5.2.3, we have:

$$\mathrm{I}(X;Z) \geq \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] + H(X) = H(X) - D, \tag{9.4}$$

where D means *distortion*, measuring reconstruction error, and H(X) is the *data entropy*. Therefore, we have:

$$H(X) - D \leq \mathrm{I}(X;Z) \leq R. \tag{9.5}$$

As H(X) is the data entropy, which is out of control, our optimization focuses on D and R. In order to find the optimal rate and distortion, we can perform Legendre transformation by minimizing:

$$\min_{q_\phi(z|x), q_\phi(z), p_\theta(x|z))} D + \beta R. \tag{9.6}$$

If we set $\beta = 1$, Equation (9.6) is the optimization function of VAE — nELBO. Note that in nELBO, $p(z|x) \to q_\phi(z|x)$ in Equation (9.2) as in VAE, inference network is present and upper bound assumes $p(z|x)$ is known.

In a more general case, Equation (9.6) is the $\beta-$VAE objective [Higgins et al., 2016]. When $\beta \ll 1$, the behavior of $\beta-$VAE is close to an autoencoder, and when $\beta \gg 1$, D is high and the optimization emphasizes on autodecoding. β controls the trade-off between autoencoding and autodecoding [Alemi et al., 2018]. In other words, the larger the β is, the more independent of latent representation is. From this perspective, nELBO is just a special case of a more general objective that controls the relative importance of latent representation.

In Section 5.2.3, we've shown using a VAE-based neural compressor for few-shot text classification. However, we don't know the role that latent representation plays in the effectiveness of this framework. Thus, in the next step, we want to reveal the relationship between latent representation and classification by tuning β in $\beta-$VAE.

9.2.2 Control Restrictiveness by Using Lossy Compressor

In Section 9.2.1, we discuss a possible extension of using R' to further restrict $R(h(x))$ so that $L(R'(R(h(x)))) \leq L(R(h(x)))$. In this section, we discuss another extension of further restricting $R(f(x))$ by using lossy compressors.

As we have discussed in Jiang et al. [2022b], simple image manipulation like greyscale can improve the classification accuracy, and greyscale can be viewed as a lossy compressor. Inspired by that, we are wondering how actual lossy compressors work under NPC-LV. The possible selection of lossy compressors include Ballé et al. [2020]; Yang et al. [2020a,b].

References

Ashutosh Adhikari, Achyudh Ram, Raphael Tang, and Jimmy Lin. Docbert: Bert for document classification. *arXiv preprint arXiv:1904.08398*, 2019a.

Ashutosh Adhikari, Achyudh Ram, Raphael Tang, and Jimmy Lin. Rethinking complex neural network architectures for document classification. In *Proceedings of the 2019 Conference of NAACL-HLT, Volume 1 (Long and Short Papers)*, pages 4046–4051, 2019b.

Alexander Alemi, Ben Poole, Ian Fischer, Joshua Dillon, Rif A Saurous, and Kevin Murphy. Fixing a broken elbo. In *International Conference on Machine Learning*, pages 159–168. PMLR, 2018.

James Allan, Donna Harman, Evangelos Kanoulas, Dan Li, Christophe Van Gysel, and Ellen Voorhees. TREC 2017 common core track overview. In *Proceedings of the Twenty-Sixth Text REtrieval Conference (TREC 2017)*, 2017.

James Allan, Donna Harman, Evangelos Kanoulas, and Ellen Voorhees. TREC 2018 common core track overview. In *Proceedings of the Twenty-Seventh Text REtrieval Conference (TREC 2018)*, 2018.

David Alvarez-Melis and Tommi S Jaakkola. On the robustness of interpretability methods. *arXiv preprint arXiv:1806.08049*, 2018.

Marco Ancona, Enea Ceolini, Cengiz Öztireli, and Markus Gross. Towards better understanding of gradient-based attribution methods for deep neural networks. In *International Conference on Learning Representations*, 2018.

Nima Asadi and Jimmy Lin. Effectiveness/efficiency tradeoffs for candidate generation in multi-stage retrieval architectures. In *Proceedings of the 36th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2013)*, pages 997–1000, 2013.

Yang Bai, Xiaoguang Li, Gang Wang, Chaoliang Zhang, Lifeng Shang, Jun Xu, Zhaowei Wang, Fangshan Wang, and Qun Liu. SparTerm: Learning term-based sparse representation for fast text retrieval. *arXiv preprint arXiv:2010.00768*, 2020.

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. MS MARCO: A human generated MAchine Reading COmprehension dataset. *arXiv:1611.09268*, 2016a.

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, et al. MS MARCO: A human generated machine reading comprehension dataset. *arXiv:1611.09268*, 2016b.

Johannes Ballé, Philip A Chou, David Minnen, Saurabh Singh, Nick Johnston, Eirikur Agustsson, Sung Jin Hwang, and George Toderici. Nonlinear transform coding. *IEEE Journal of Selected Topics in Signal Processing*, 15(2):339–353, 2020.

Seojin Bang, Pengtao Xie, Heewook Lee, Wei Wu, and Eric Xing. Explaining a black-box using deep variational information bottleneck approach. *arXiv:1902.06918*, 2019.

Fabrice Bellard. Lossless data compression with transformer, 2021.

Eyal Ben-David, Nadav Oved, and Roi Reichart. Pada: Example-based prompt learning for on-the-fly adaptation to unseen domains. *Transactions of the Association for Computational Linguistics*, 10:414–433, 2022.

Charles H Bennett, Péter Gács, Ming Li, Paul MB Vitányi, and Wojciech H Zurek. Information distance. *IEEE Transactions on information theory*, 44(4):1407–1423, 1998.

David Berthelot, Nicholas Carlini, Ian Goodfellow, Avital Oliver, Nicolas Papernot, and Colin Raffel. Mixmatch: A holistic approach to semi-supervised learning.

Alexander Binder, Sebastian Bach, Gregoire Montavon, Klaus-Robert Müller, and Wojciech Samek. Layer-wise relevance propagation for deep neural network architectures. In *Information science and applications (ICISA) 2016*, pages 913–922. Springer, 2016.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020.

Samuel Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, 2015.

Samuel Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 10–21, 2016.

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

Gino Brunner, Yang Liu, Damian Pascual, Oliver Richter, Massimiliano Ciaramita, and Roger Wattenhofer. On identifiability in transformers. In *International Conference on Learning Representations*, 2019.

Michael Burrows and David Wheeler. A block-sorting lossless data compression algorithm. In *Digital SRC Research Report*. Citeseer, 1994.

Daniel Fernando Campos, Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, Li Deng, and Bhaskar Mitra. Ms marco: A human generated machine reading comprehension dataset. *ArXiv*, abs/1611.09268, 2016.

Iñigo Casanueva, Tadas Temcinas, Daniela Gerz, Matthew Henderson, and Ivan Vulic. Efficient intent detection with dual sentence encoders. In *Proceedings of the 2nd Workshop on NLP for ConvAI - ACL 2020*, mar 2020. URL https://arxiv.org/abs/2003.04807. Data available at https://github.com/PolyAI-LDN/task-specific-datasets.

Manuel Cebrián, Manuel Alfonseca, and Alfonso Ortega. Common pitfalls using the normalized compression distance: What to watch out for in a compressor. *Communications in Information & Systems*, 5(4):367–384, 2005.

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada, August 2017. Association for Computational Linguistics. doi: 10.18653/v1/S17-2001. URL https://aclanthology.org/S17-2001.

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

Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey E Hinton. Big self-supervised models are strong semi-supervised learners. *Advances in Neural Information Processing Systems*, 33:22243–22255, 2020b.

Xi Chen, Diederik P Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. Variational lossy autoencoder. 2016.

Xin Chen, Sam Kwong, and Ming Li. A compression algorithm for dna sequences and its applications in genome comparison. *Genome informatics*, 10:51–61, 1999.

Xin Chen, Brent Francia, Ming Li, Brian Mckinnon, and Amit Seker. Shared information and program plagiarism detection. *IEEE Transactions on Information Theory*, 50(7):1545–1551, 2004.

Rudi Cilibrasi and Paul M. B. Vitányi. Clustering by compression. *CoRR*, cs.CV/0312044, 2003. URL http://arxiv.org/abs/cs/0312044.

Rudi Cilibrasi and Paul MB Vitányi. Clustering by compression. *IEEE Transactions on Information theory*, 51(4):1523–1545, 2005.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. What does BERT look at? An analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, 2019.

Ronan Collobert, Fabian Sinz, Jason Weston, Léon Bottou, and Thorsten Joachims. Large scale transductive svms. *Journal of Machine Learning Research*, 7(8), 2006.

Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. Very deep convolutional networks for text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1107–1116, 2017.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

David Pereira Coutinho and Mario AT Figueiredo. Text classification using compression-based dissimilarity measures. *International Journal of Pattern Recognition and Artificial Intelligence*, 29(05):1553004, 2015.

Thomas M Cover. *Elements of information theory*. John Wiley & Sons, 1999.

Zhuyun Dai and Jamie Callan. Deeper text understanding for IR with contextual neural language modeling. In *Proceedings of the 42nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2019)*, pages 985–988, Paris, France, 2019.

Tim R Davidson, Luca Falorsi, Nicola De Cao, Thomas Kipf, and Jakub M Tomczak. Hyperspherical variational auto-encoders. In *34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018*, pages 856–865. Association For Uncertainty in Artificial Intelligence (AUAI), 2018.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

L. Peter Deutsch. GZIP file format specification version 4.3. RFC 1952, May 1996. URL https://www.rfc-editor.org/info/rfc1952.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, 2019.

Jesse Dodge, Suchin Gururangan, Dallas Card, Roy Schwartz, and Noah A. Smith. Show your work: Improved reporting of experimental results. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2185–2194, 2019.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. 2020.

Matthijs Douze, Arthur Szlam, Bharath Hariharan, and Hervé Jégou. Low-shot learning with large-scale diffusion. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3349–3358, 2018.

Jarek Duda. Asymmetric numeral systems. *arXiv preprint arXiv:0902.0271*, 2009.

Jarek Duda, Khalid Tahboub, Neeraj J Gadgil, and Edward J Delp. The use of asymmetric numeral systems as an accurate replacement for huffman coding. In *2015 Picture Coding Symposium (PCS)*, pages 65–69. IEEE, 2015.

Harrison Edwards and Amos Storkey. Towards a neural statistician. *International Conference on Learning Representations*, 2016.

R. Fagin and L. Stockmeyer. Relaxing the triangle inequality in pattern matching. pages 219–231, 1998.

Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical neural story generation. *arXiv:1805.04833*, 2018.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

Ruth C. Fong and Andrea Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. In *Proceedings of the IEEE International Conference on Computer Vision*, 2017.

Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. SPLADE: Sparse lexical and expansion model for first stage ranking. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2288–2292, 2021.

Eibe Frank, Chang Chui, and Ian H Witten. Text categorization using compression models. 2000.

Brendan J Frey and Geoffrey E Hinton. Free energy coding. In *Proceedings of Data Compression Conference-DCC'96*, pages 73–81. IEEE, 1996.

Brendan J. Frey and Geoffrey E. Hinton. Efficient stochastic source coding and an application to a bayesian network source model. *The Computer Journal*, 40(2_and_3):157–165, 1997.

Jerome H Friedman. On bias, variance, 0/1—loss, and the curse-of-dimensionality. *Data mining and knowledge discovery*, 1(1):55–77, 1997.

Hang Gao, Zheng Shou, Alireza Zareian, Hanwang Zhang, and Shih-Fu Chang. Low-shot learning via covariance-preserving adversarial augmentation networks. 2018.

Fabian Giesen. Interleaved entropy coders. *arXiv preprint arXiv:1402.3392*, 2014.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf.

Mohit Goyal, Kedar Tatwawadi, Shubham Chandak, and Idoia Ochoa. Dzip: improved general-purpose lossless compression based on novel neural network modeling. *CoRR*, abs/1911.03572, 2019a. URL http://arxiv.org/abs/1911.03572.

Mohit Goyal, Kedar Tatwawadi, Shubham Chandak, and Idoia Ochoa. Deepzip: Lossless data compression using recurrent neural networks. In *2019 Data Compression Conference (DCC)*, pages 575–575. IEEE, 2019b.

Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

Karol Gregor, Ivo Danihelka, Andriy Mnih, Charles Blundell, and Daan Wierstra. Deep autoregressive networks. In *International Conference on Machine Learning*, pages 1242–1250. PMLR, 2014.

Stéphane Grumbach and Fariza Tahi. A new challenge for compression algorithms: genetic sequences. *Information Processing & Management*, 30(6):875–886, 1994.

Chaoyu Guan, Xiting Wang, Quanshi Zhang, Runjin Chen, Di He, and Xing Xie. Towards a deep and unified understanding of deep neural models in NLP. In *International Conference on Machine Learning*, 2019.

Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A survey of methods for explaining black box models. *ACM computing surveys (CSUR)*, 2018.

Antonio Gulli. *AGNews*. 2004. http://groups.di.unipi.it/~gulli/AG_corpus_of_news_articles.html.

Jiafeng Guo, Yixing Fan, Qingyao Ai, and W. Bruce Croft. A deep relevance matching model for ad-hoc retrieval. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 55–64, 2016.

Mark H Hansen and Bin Yu. Model selection and the principle of minimum description length. *Journal of the American Statistical Association*, 96(454):746–774, 2001.

Bharath Hariharan and Ross Girshick. Low-shot visual recognition by shrinking and hallucinating features. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3018–3027, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Michael Herrera and Kasey Luo. Lossless neural text compression, 2021. URL https://web.stanford.edu/class/cs224n/reports/custom_116635402.pdf.

John Hewitt and Christopher D. Manning. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2019.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. 2016.

Geoffrey E Hinton and Drew Van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the sixth annual conference on Computational learning theory*, pages 5–13, 1993.

Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Antti Honkela and Harri Valpola. Variational learning and bits-back coding: an information-theoretic view to bayesian learning. *IEEE transactions on Neural Networks*, 15(4):800–810, 2004.

Emiel Hoogeboom, Jorn Peters, Rianne Van Den Berg, and Max Welling. Integer discrete flows and lossless compression. *Advances in Neural Information Processing Systems*, 32, 2019.

Sara Hooker, Dumitru Erhan, Pieter-Jan Kindermans, and Been Kim. Evaluating feature importance estimates. *arXiv:1806.10758*, 2018.

Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417, 1933.

David A Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952.

Kai Hui, Andrew Yates, Klaus Berberich, and Gerard de Melo. Co-PACRR: A context-aware neural IR model for ad-hoc retrieval. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining (WSDM 2018)*, pages 279–287, 2018.

Sarthak Jain and Byron C. Wallace. Attention is not explanation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3543–3556, 2019.

Zhiying Jiang, Raphael Tang, Ji Xin, and Jimmy Lin. Inserting information bottleneck for attribution in transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 3850–3857, 2020.

Zhiying Jiang, Yiqin Dai, Ji Xin, Ming Li, and Jimmy Lin. Few-shot non-parametric learning with deep latent variable model. In *Advances in Neural Information Processing Systems*, 2022a. URL https://openreview.net/forum?id=24fiAU_9vT.

Zhiying Jiang, Yiqin Dai, Ji Xin, Ming Li, and Jimmy Lin. Few-shot non-parametric learning with deep latent variable model. *Advances in Neural Information Processing systems*, 2022b.

Zhiying Jiang, Rui Wang, Dongbo Bu, and Ming Li. A theory of human-like few-shot learning, 01 2023.

Peter D Johnson Jr, Greg A Harris, and DC Hankerson. *Introduction to information theory and data compression*. CRC press, 2003.

Armand Joulin, Edouard Grave, and Piotr Bojanowski Tomas Mikolov. Bag of tricks for efficient text classification. *EACL 2017*, page 427, 2017.

Tom Joy, Sebastian M Schmon, Philip HS Torr, N Siddharth, and Tom Rainforth. Rethinking semi-supervised learning in vaes. *CoRR*, 2020.

Tom Joy, Sebastian Schmon, Philip Torr, N Siddharth, and Tom Rainforth. Capturing label characteristics in vaes. In *International Conference on Learning Representations*, 2021.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv:2001.08361*, 2020.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online, November 2020a. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.550. URL https://aclanthology.org/2020.emnlp-main.550.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, 2020b.

Alexandros Kastanos and Tyler Martin. Graph convolutional network for swahili news classification. *arXiv preprint arXiv:2103.09325*, 2021.

Nitya Kasturi and Igor L Markov. Text ranking and classification using data compression. In *I (Still) Can't Believe It's Not Better! Workshop at NeurIPS 2021*, pages 48–53. PMLR, 2022.

Kazuya Kawakami. Supervised sequence labelling with recurrent neural networks. *Ph. D. thesis*, 2008.

Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186, 2019.

Eamonn Keogh, Stefano Lonardi, and Chotirat Ann Ratanamahatana. Towards parameter-free data mining. In ., KDD '04, page 206–215, New York, NY, USA, 2004a. Association for Computing Machinery. ISBN 1581138881. doi: 10.1145/1014052.1014077. URL https://doi.org/10.1145/1014052.1014077.

Eamonn Keogh, Stefano Lonardi, and Chotirat Ann Ratanamahatana. Towards parameter-free data mining. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 206–215, 2004b.

Dmitry V Khmelev and William J Teahan. A repetition based measure for verification of text collections and for text categorization. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 104–110, 2003.

Pieter-Jan Kindermans, Kristof T. Schütt, Maximilian Alber, Klaus-Robert Müller, Dumitru Erhan, Been Kim, and Sven Dähne. Learning how to explain neural networks: PatternNet and PatternAttribution. In *ICLR (Poster)*, 2018.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR (Poster)*, 2015.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. 2013.

Diederik P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing systems*, pages 3581–3589, 2014.

Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. *Advances in Neural Information Processing systems*, 29:4743–4751, 2016.

Friso Kingma, Pieter Abbeel, and Jonathan Ho. Bit-swap: Recursive bits-back coding for lossless compression with hierarchical latent variables. In *International Conference on Machine Learning*, pages 3408–3417. PMLR, 2019a.

Friso H. Kingma, P. Abbeel, and Jonathan Ho. Bit-swap: Recursive bits-back coding for lossless compression with hierarchical latent variables. In *International Conference on Machine Learning*, 2019b.

Jack Klys, Jake Snell, and Richard Zemel. Learning latent subspaces in variational autoencoders. *Advances in Neural Information Processing systems*, 2018.

Gregory Koch, Richard Zemel, Ruslan Salakhutdinov, et al. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2. Lille, 2015.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International Conference on Machine Learning*, 2017.

A.N. Kolmogorov. Three approaches to the quantitative definition of information. *Problems of Information Transmission*, 1(1):1–7, 1965.

A.N. Kolmogorov. On tables of random numbers. *Theoretical Computer Science*, 207(2):387–395, 1998. ISSN 0304-3975. doi: https://doi.org/10.1016/S0304-3975(98)00075-9. URL https://www.sciencedirect.com/science/article/pii/S0304397598000759.

Andrei N Kolmogorov. On tables of random numbers. *Sankhyā: The Indian Journal of Statistics, Series A*, pages 369–376, 1963.

Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. Revealing the dark secrets of BERT. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019.

Deutsches Krebsforschungzentrum. The transformation distance: A dissimilarity measure based on movements of segments.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25, 2012.

Taku Kudo and John Richardson. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, 2018.

Roland Kwitt, Sebastian Hegenbart, and Marc Niethammer. One-shot learning of scene locations via feature trajectory transfer. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, pages 78–86, 2016.

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

Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning. 2016.

Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010. URL http://yann.lecun.com/exdb/mnist/.

Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

Dong-Hyun Lee et al. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, volume 3, page 896, 2013.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.703. URL https://aclanthology.org/2020.acl-main.703.

Canjia Li, Andrew Yates, Sean MacAvaney, Ben He, and Yingfei Sun. PARADE: Passage representation aggregation for document reranking. *arXiv:2008.09093*, 2020.

Hang Li. *Learning to Rank for Information Retrieval and Natural Language Processing*. Morgan & Claypool Publishers, 2011.

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. Visualizing and understanding neural models in NLP. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2016.

Ming Li. Information distance and its applications. pages 1–9, 2006.

Ming Li and Paul M. B. Vitányi. *An Introduction to Kolmogorov Complexity and its Applications*. Springer, 2008.

Ming Li, Jonathan H Badger, Xin Chen, Sam Kwong, Paul Kearney, and Haoyong Zhang. An information-based sequence distance and its application to whole mitochondrial genome phylogeny. *Bioinformatics*, 17(2):149–154, 2001a.

Ming Li, Jonathan H. Badger, Xin Chen, Sam Kwong, Paul Kearney, and Haoyong Zhang. An information-based sequence distance and its application to whole mitochondrial genome phylogeny . *Bioinformatics*, 17(2):149–154, 02 2001b. ISSN 1367-4803. doi: 10.1093/bioinformatics/17.2.149. URL https://doi.org/10.1093/bioinformatics/17.2.149.

Ming Li, Xin Chen, Xin Li, Bin Ma, and Paul MB Vitányi. The similarity metric. *IEEE transactions on Information Theory*, 50(12):3250–3264, 2004.

Ming Li, Paul Vitányi, et al. *An introduction to Kolmogorov complexity and its applications*, volume 3. Springer, 2008.

Qian Li, Hao Peng, Jianxin Li, Congying Xia, Renyu Yang, Lichao Sun, Philip S Yu, and Lifang He. A survey on text classification: From traditional to deep learning. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 13(2):1–41, 2022.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. 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)*, pages 4582–4597, 2021.

Kaiqu Liang, Cem Anil, Yuhuai Wu, and Roger Grosse. Out-of-distribution generalization with deep equilibrium models. 2021.

Jimmy Lin, Rodrigo Nogueira, and Andrew Yates. Pretrained transformers for text ranking: BERT and beyond. *arXiv:2010.06467*, 2020.

Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. Pyserini: An easy-to-use Python toolkit to support replicable ir research with sparse and dense representations. *arXiv:2102.10073*, 2021.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin A Raffel. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *Advances in Neural Information Processing Systems*, 35:1950–1965, 2022.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E Peters, and Noah A. Smith. Linguistic knowledge and transferability of contextual representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2019.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35, 2023.

Tie-Yan Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, 2009.

Xien Liu, Song Wang, Xiao Zhang, Xinxin You, Ji Wu, and Dejing Dou. Label-guided learning for text classification. *arXiv preprint arXiv:2002.10772*, 2020.

Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing systems*, 2017.

Calvin Luo. Understanding diffusion models: A unified perspective. *arXiv preprint arXiv:2208.11970*, 2022.

Yi Ma, Doris Tsao, and Heung-Yeung Shum. On the principles of parsimony and self-consistency for the emergence of intelligence. *Frontiers of Information Technology & Electronic Engineering*, pages 1–26, 2022.

Lars Maaløe, Casper Kaae Sønderby, Søren Kaae Sønderby, and Ole Winther. Auxiliary deep generative models. In *International conference on machine learning*, pages 1445–1453. PMLR, 2016.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, June 2011.

Sean MacAvaney, Sergey Feldman, Nazli Goharian, Doug Downey, and Arman Cohan. ABNIRML: Analyzing the behavior of neural IR models. *arXiv preprint arXiv:2011.00696*, 2020.

David JC MacKay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

Matt Mahoney. Large text compression benchmark, 2006. URL http://mattmahoney.net/dc/text.html.

Matthew V Mahoney. Fast text compression with neural networks. In *FLAIRS conference*, pages 230–234, 2000.

Macedo Maia, Siegfried Handschuh, André Freitas, Brian Davis, Ross McDermott, Manel Zarrouk, and Alexandra Balahur. Www'18 open challenge: Financial opinion mining and question answering. In *Companion Proceedings of the The Web Conference 2018*, WWW '18, page 1941–1942, Republic and Canton of Geneva, CHE, 2018. International World Wide Web Conferences Steering Committee. ISBN 9781450356404. doi: 10.1145/3184558.3192301. URL https://doi.org/10.1145/3184558.3192301.

Yu Mao, Yufei Cui, Tei-Wei Kuo, and Chun Jason Xue. Trace: A fast transformer-based general-purpose lossless compressor. In *Proceedings of the ACM Web Conference 2022*, WWW '22, page 1829–1838, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450390965. doi: 10.1145/3485447.3511987. URL https://doi.org/10.1145/3485447.3511987.

Gary Marcus. Deep learning: A critical appraisal. *arXiv preprint arXiv:1801.00631*, 2018.

Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. 1993.

Yuval Marton, Ning Wu, and Lisa Hellerstein. On compression-based text classification. In *European Conference on Information Retrieval*, pages 300–314. Springer, 2005.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

Rishabh Misra. News category dataset, 06 2018.

Rishabh Misra and Jigyasa Grover. *Sculpting Data for ML: The first act of Machine Learning*. Jigyasa Grover and Rishabh Misra, 01 2021. ISBN 978-0-578-83125-1.

Bhaskar Mitra and Nick Craswell. An introduction to neural information retrieval. *Foundations and Trends in Information Retrieval*, 13(1):1–126, 2019.

Bhaskar Mitra, Fernando Diaz, and Nick Craswell. Learning to match using local and distributed representations of text for web search. In *Proceedings of the 26th International Conference on World Wide Web (WWW 2017)*, pages 1291–1299, 2017.

Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1979–1993, 2018.

Christoph Molnar. *Interpretable Machine Learning*. 2019. https://christophm.github.io/interpretable-ml-book/.

Rubungo Andre Niyongabo, Qu Hong, Julia Kreutzer, and Li Huang. Kinnews and kirnews: Benchmarking cross-lingual text classification for kinyarwanda and kirundi. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5507–5521, 2020.

Rodrigo Nogueira and Kyunghyun Cho. Passage re-ranking with BERT. *arXiv preprint arXiv:1901.04085*, 2019.

Rodrigo Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. Multi-stage document ranking with BERT. *arXiv:1910.14424*, 2019a.

Rodrigo Nogueira, Wei Yang, Jimmy Lin, and Kyunghyun Cho. Document expansion by query prediction. *arXiv:1904.08375*, 2019b.

Rodrigo Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. Document ranking with a pretrained sequence-to-sequence model. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 708–718, 2020.

Antoine Nzeyimana and Andre Niyongabo Rubungo. Kinyabert: a morphology-aware kinyarwanda language model. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5347–5363, 2022.

Kezban Dilek Onal, Ye Zhang, Ismail Sengor Altingovde, Md Mustafizur Rahman, Pinar Karagoz, Alex Braylan, Brandon Dang, Heng-Lu Chang, Henna Kim, Quinten McNamara, Aaron Angert, Edward Banner, Vivek Khetan, Tyler McDonnell, An Thanh Nguyen, Dan Xu, Byron C. Wallace, Maarten de Rijke, and Matthew Lease. Neural information retrieval: At the end of the early years. *Information Retrieval*, 21(2–3):111–182, 2018.

OpenAI. Gpt-4 technical report, 2023.

Ethan Perez, Douwe Kiela, and Kyunghyun Cho. True few-shot learning with language models. *Advances in Neural Information Processing systems*, 34:11054–11070, 2021.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, 2019.

Tomas Pfister, James Charles, and Andrew Zisserman. Domain-adaptive discriminative one-shot learning of gestures. In *European Conference on Computer Vision*, pages 814–829. Springer, 2014.

Tiago Pimentel, Josef Valvoda, Rowan Hall Maudslay, Ran Zmigrod, Adina Williams, and Ryan Cotterell. Information-theoretic probing for linguistic structure. *arXiv:2004.03061*, 2020.

Raul Puri and Bryan Catanzaro. Zero-shot text classification with generative language models. *arXiv preprint arXiv:1912.10165*, 2019.

Yifan Qiao, Chenyan Xiong, Zhenghao Liu, and Zhiyuan Liu. Understanding the behaviors of BERT in ranking. *arXiv:1904.07531*, 2019.

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

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

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI*, 2019.

Edward Raff and Charles Nicholas. An alternative to ncd for large sequences, lempel-ziv jaccard distance. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1007–1015, 2017.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(1), jan 2020a. ISSN 1532-4435.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020b.

Md Atiqur Rahman and Mohamed Hamada. Lossless text compression using gpt-2 language model and huffman coding. In *SHS Web of Conferences*, volume 102, page 04013. EDP Sciences, 2021.

Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. *International Conference on Learning Representations*, 2016.

Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*

(EMNLP-IJCNLP), pages 3982–3992, Hong Kong, China, November 2019a. Association for Computational Linguistics. doi: 10.18653/v1/D19-1410. URL https://aclanthology.org/D19-1410.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, 2019b.

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pages 1530–1538. PMLR, 2015.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*, pages 1278–1286. PMLR, 2014.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

Jorma Rissanen. Modeling by shortest data description. *Automatica*, 14(5):465–471, 1978.

Jorma Rissanen. *Stochastic complexity in statistical inquiry*, volume 15. World scientific, 1989.

Stephen Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, and M. Gatford. Okapi at trec-3. In *Overview of the Third Text REtrieval Conference (TREC-3)*, pages 109–126. Gaithersburg, MD: NIST, January 1995. URL https://www.microsoft.com/en-us/research/publication/okapi-at-trec-3/.

Stephen E Robertson and K Sparck Jones. Relevance weighting of search terms. *Journal of the American Society for Information science*, 27(3):129–146, 1976.

Andrew Robinson. Did einstein really say that? *Nature*, 557(7703):30–31, 2018.

Marko Robnik-Šikonja and Marko Bohanec. Perturbation-based explanations of prediction models. In *Human and machine learning*. Springer, 2018.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. A primer in BERTology: What we know about how BERT works. *Transactions of the Association for Computational Linguistics*, 8: 842–866, 2020.

Michal Rolinek, Dominik Zietlow, and Georg Martius. Variational autoencoders pursue pca directions (by accident). In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12406–12415, 2019.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

Stuart J Russell. *Artificial intelligence a modern approach*. Pearson Education, Inc., 2010.

Tim Salimans and Durk P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *Advances in Neural Information Processing systems*, 29:901–909, 2016.

Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. 2016.

Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. One-shot learning with memory-augmented neural networks. *arXiv preprint arXiv:1605.06065*, 2016.

Jürgen Schmidhuber. *Neural Predictors for Detecting and Removing Redundant Information*, pages 1152–1167. Springer Netherlands, Dordrecht, 2000. ISBN 978-94-010-0870-9. doi: 10.1007/978-94-010-0870-9_73. URL https://doi.org/10.1007/978-94-010-0870-9_73.

Jürgen Schmidhuber and Stefan Heil. Sequential neural text compression. *IEEE Transactions on Neural Networks*, 7(1):142–146, 1996.

Tim Schopf, Daniel Braun, and Florian Matthes. Evaluating unsupervised text classification: Zero-shot and similarity-based approaches. In *2022 6th International Conference on Natural Language Processing and Information Retrieval (NLPIR)*, NLPIR 2022, New York, NY, USA, 2023. Association for Computing Machinery.

Karl Schulz, Leon Sixt, Federico Tombari, and Tim Landgraf. Restricting the flow: Information bottlenecks for attribution. In *International Conference on Learning Representations*, 2020.

Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681, 1997.

Eli Schwartz, Leonid Karlinsky, Joseph Shtok, Sivan Harary, Mattias Marder, Abhishek Kumar, Rogério Schmidt Feris, Raja Giryes, and Alexander M Bronstein. Delta-encoder: an effective sample synthesis method for few-shot object recognition. In *NeurIPS*, 2018.

David Sculley and Carla E Brodley. Compression and machine learning: A new perspective on feature space vectors. In *Data Compression Conference (DCC'06)*, pages 332–341. IEEE, 2006.

Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, 2017.

Sofia Serrano and Noah A. Smith. Is attention interpretable? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2931–2951, 2019.

Claude Elwood Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

Dinghan Shen, Guoyin Wang, Wenlin Wang, Martin Renqiang Min, Qinliang Su, Yizhe Zhang, Chunyuan Li, Ricardo Henao, and Lawrence Carin. Baseline needs more love: On simple word-embedding-based models and associated pooling mechanisms. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 440–450, 2018.

Tianye Sheng, Lisong Wang, Zongfeng He, Mingjie Sun, and Guohua Jiang. An unsupervised sentence embedding method by maximizing the mutual information of augmented text representations. In *Artificial Neural Networks and Machine Learning – ICANN 2022: 31st International Conference on Artificial Neural Networks, Bristol, UK, September 6–9, 2022,*

Proceedings, Part II, page 174–185, Berlin, Heidelberg, 2022. Springer-Verlag. ISBN 978-3-031-15930-5. doi: 10.1007/978-3-031-15931-2_15. URL https://doi.org/10.1007/978-3-031-15931-2_15.

Eyal Shnarch, Ariel Gera, Alon Halfon, Lena Dankin, Leshem Choshen, Ranit Aharonov, and Noam Slonim. Cluster & tune: Boost cold start performance in text classification. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7639–7653, 2022.

Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *International Conference on Machine Learning*, pages 3145–3153. PMLR, 2017.

Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*, 2017.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *International Conference on Learning Representations*, 2014.

Harshdeep Singh, Robert West, and Giovanni Colavizza. Wikipedia citations: A comprehensive data set of citations with identifiers extracted from english wikipedia. *Quantitative Science Studies*, pages 1–19, 2020.

Leon Sixt, Maximilian Granz, and Tim Landgraf. When explanations lie: Why many modified BP attributions fail. In *International Conference on Machine Learning*, pages 9046–9057. PMLR, 2020.

Noam Slonim and Naftali Tishby. Agglomerative information bottleneck. *Advances in Neural Information Processing systems*, 12, 1999.

Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. SmoothGrad: removing noise by adding noise. *arXiv:1706.03825*, 2017.

Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. *Advances in Neural Information Processing Systems*, 30:4077–4087, 2017a.

Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017b.

Ian Soboroff, Shudong Huang, and Donna Harman. Trec 2019 news track overview. In *TREC*, 2019. URL https://trec.nist.gov/pubs/trec28/papers/OVERVIEW.N.pdf.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA, October 2013. Association for Computational Linguistics. URL https://aclanthology.org/D13-1170.

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. *Advances in Neural Information Processing systems*, 28: 3483–3491, 2015.

Kihyuk Sohn, David Berthelot, Nicholas Carlini, Zizhao Zhang, Han Zhang, Colin A Raffel, Ekin Dogus Cubuk, Alexey Kurakin, and Chun-Liang Li. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *Advances in Neural Information Processing Systems*, 33, 2020.

Charles Spearman. The proof and measurement of association between two things. 1961.

Charles Spearman. The proof and measurement of association between two things. *The American Journal of Psychology*, 100(3/4):441–471, 1987.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 23(5):828–841, 2019.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International Conference on Machine Learning*, 2017.

Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1199–1208, 2018.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 3104–3112, 2014.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Raphael Tang, Akshat Pandey, Zhiying Jiang, Gefei Yang, Karun Kumar, Jimmy Lin, and Ferhan Ture. What the daam: Interpreting stable diffusion using cross attention. *arXiv preprint arXiv:2210.04885*, 2022.

Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. *Advances in Neural Information Processing Systems*, 2017.

William J Teahan and David J Harper. Using compression-based language models for text categorization. In *Language modeling for information retrieval*, pages 141–165. Springer, 2003.

Andreia Teixeira, Armando Matos, André Souto, and Luís Antunes. Entropy measures vs. kolmogorov complexity. *Entropy*, 13(3):595–611, 2011. ISSN 1099-4300. doi: 10.3390/e13030595. URL https://www.mdpi.com/1099-4300/13/3/595.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019a.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R. Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel Bowman, Dipanjan Das, and Ellie Pavlick. What do you learn from context? Probing for sentence structure in contextualized word representations. In *International Conference on Learning Representations*, 2019b.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. URL https://openreview.net/forum?id=wCu6T5xFjeJ.

MTCAJ Thomas and A Thomas Joy. *Elements of information theory*. Wiley-Interscience, 2006.

Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. In *2015 ieee information theory workshop (itw)*, pages 1–5. IEEE, 2015.

Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000.

James Townsend, Thomas Bird, and David Barber. Practical lossless compression with latent variables using bits back coding. In *International Conference on Learning Representations*, 2019a. URL https://openreview.net/forum?id=ryE98iR5tm.

James Townsend, Thomas Bird, and David Barber. Practical lossless compression with latent variables using bits back coding. In *International Conference on Learning Representations*, 2019b.

George Tsatsaronis, Georgios Balikas, Prodromos Malakasiotis, Ioannis Partalas, Matthias Zschunke, Michael R Alvers, Dirk Weissenborn, Anastasia Krithara, Sergios Petridis, Dimitris Polychronopoulos, et al. An overview of the bioasq large-scale biomedical semantic indexing and question answering competition. *BMC bioinformatics*, 16(1):138, 2015.

Betty van Aken, Benjamin Winter, Alexander Löser, and Felix A. Gers. How does BERT answer questions? A layer-wise analysis of transformer representations. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019.

Aäron Van den Oord, Nal Kalchbrenner, Lasse Espeholt, Koray Kavukcuoglu, Oriol Vinyals, and Alex Graves. Conditional image generation with pixelcnn decoders. In *Advances in Neural Information Processing systems*, 2016a.

Aaron Van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *International Conference on Machine Learning*, pages 1747–1756. PMLR, 2016b.

Jesper E Van Engelen and Holger H Hoos. A survey on semi-supervised learning. *Machine learning*, 109(2):373–440, 2020.

Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 1999.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing systems*, 30, 2017.

R.C. Veltkamp. Shape matching: Similarity measures and algorithms. pages 188–197, 2001.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. *Advances in Neural Information Processing systems*, 29:3630–3638, 2016.

Paul M. B. Vitányi. Conditional kolmogorov complexity and universal probability. *CoRR*, abs/1206.0983, 2012. URL http://arxiv.org/abs/1206.0983.

Paul MB Vitányi, Frank J Balbach, Rudi L Cilibrasi, and Ming Li. Normalized information distance. In *Information theory and statistical learning*, pages 45–82. Springer, 2009.

Elena Voita, Pavel Serdyukov, Rico Sennrich, and Ivan Titov. Context-aware neural machine translation learns anaphora resolution. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1264–1274, 2018.

Ellen Voorhees, Tasmeer Alam, Steven Bedrick, Dina Demner-Fushman, William R. Hersh, Kyle Lo, Kirk Roberts, Ian Soboroff, and Lucy Lu Wang. Trec-covid: Constructing a pandemic information retrieval test collection. *SIGIR Forum*, 54(1), February 2021. ISSN 0163-5840. doi: 10.1145/3451964.3451965. URL https://doi.org/10.1145/3451964.3451965.

Ellen M. Voorhees. Overview of the TREC 2004 Robust Track. In *Proceedings of the Thirteenth Text REtrieval Conference (TREC 2004)*, pages 52–69, 2004.

Henning Wachsmuth, Shahbaz Syed, and Benno Stein. Retrieval of the best counterargument without prior topic knowledge. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 241–251. Association for Computational Linguistics, 2018. URL http://aclweb.org/anthology/P18-1023.

David Wadden, Shanchuan Lin, Kyle Lo, Lucy Lu Wang, Madeleine van Zuylen, Arman Cohan, and Hannaneh Hajishirzi. Fact or fiction: Verifying scientific claims. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7534–7550, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.609. URL https://www.aclweb.org/anthology/2020.emnlp-main.609.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, 2018a.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*, 2018b.

Siyu Wang, Jianfei Chen, Chongxuan Li, Jun Zhu, and Bo Zhang. Fast lossless neural compression with integer-only discrete flows. In *International Conference on Machine Learning*, pages 22562–22575. PMLR, 2022.

Yaqing Wang, Quanming Yao, James T. Kwok, and Lionel M. Ni. Generalizing from a few examples: A survey on few-shot learning. *ACM Comput. Surv.*, 53(3), jun 2020a. ISSN 0360-0300. doi: 10.1145/3386252. URL https://doi.org/10.1145/3386252.

Yaqing Wang, Quanming Yao, James T Kwok, and Lionel M Ni. Generalizing from a few examples: A survey on few-shot learning. *ACM computing surveys (csur)*, 53(3):1–34, 2020b.

Yequan Wang, Minlie Huang, Xiaoyan Zhu, and Li Zhao. Attention-based lstm for aspect-level sentiment classification. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 606–615, 2016.

Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2018.

Ian H. Witten, Radford M. Neal, and John G. Cleary. Arithmetic coding for data compression. *Commun. ACM*, 30(6):520–540, jun 1987a. ISSN 0001-0782. doi: 10.1145/214762.214771. URL https://doi.org/10.1145/214762.214771.

Ian H Witten, Radford M Neal, and John G Cleary. Arithmetic coding for data compression. *Communications of the ACM*, 30(6):520–540, 1987b.

David H Wolpert and William G Macready. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82, 1997.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *International conference on machine learning*, pages 478–487. PMLR, 2016.

Qizhe Xie, Zihang Dai, Eduard Hovy, Thang Luong, and Quoc Le. Unsupervised data augmentation for consistency training. *Advances in Neural Information Processing Systems*, 33, 2020a.

Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V Le. Self-training with noisy student improves imagenet classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10687–10698, 2020b.

Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. DeeBERT: Dynamic early exiting for accelerating BERT inference. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, July 2020.

Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. End-to-end neural ad-hoc ranking with kernel pooling. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2017)*, pages 55–64, 2017.

Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. Luke: Deep contextualized entity representations with entity-aware self-attention. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6442–6454, 2020.

Peilin Yang, Hui Fang, and Jimmy Lin. Anserini: Enabling the use of Lucene for information retrieval research. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1253–1256, 2017.

Wei Yang, Kuang Lu, Peilin Yang, and Jimmy Lin. Critically examining the "neural hype" weak baselines and the additivity of effectiveness gains from neural ranking models. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1129–1132, 2019.

Xiangli Yang, Zixing Song, Irwin King, and Zenglin Xu. A survey on deep semi-supervised learning. *arXiv preprint arXiv:2103.00550*, 2021.

Yibo Yang, Robert Bamler, and Stephan Mandt. Improving inference for neural image compression. *Advances in Neural Information Processing Systems*, 33:573–584, 2020a.

Yibo Yang, Robert Bamler, and Stephan Mandt. Variational bayesian quantization. In *International Conference on Machine Learning*, pages 10670–10680. PMLR, 2020b.

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

Liang Yao, Chengsheng Mao, and Yuan Luo. Graph convolutional networks for text classification. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 7370–7377, 2019.

Peter N Yianilos. Normalized forms for two common metrics. *NEC Res. Inst., Rep*, pages 91–082, 2002.

Emine Yilmaz, Nick Craswell, Bhaskar Mitra, and Daniel Campos. On the reliability of test collections for evaluating systems of different types. In *Proceedings of the 43rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2020)*, pages 2101–2104, 2020.

Zeynep Akkalyoncu Yilmaz, Wei Yang, Haotian Zhang, and Jimmy Lin. Cross-domain modeling of sentence-level evidence for document retrieval. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3481–3487, 2019.

Yaodong Yu, Heinrich Jiang, Dara Bahri, Hossein Mobahi, Seungyeon Kim, Ankit Singh Rawat, Andreas Veit, and Yi Ma. An empirical study of pre-trained vision models on out-of-distribution generalization. In *NeurIPS 2021 Workshop on Distribution Shifts: Connecting Methods and Applications*, 2021.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

Bowen Zhang, Yidong Wang, Wenxin Hou, Hao Wu, Jindong Wang, Manabu Okumura, and Takahiro Shinozaki. Flexmatch: Boosting semi-supervised learning with curriculum pseudo labeling. *Advances in Neural Information Processing Systems*, 34, 2021a.

C Zhang, S Bengio, M Hardt, B Recht, and O Vinyals. Understanding deep learning requires rethinking generalization. *ICLR*, 2017.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3): 107–115, 2021b.

Haode Zhang, Yuwei Zhang, Li-Ming Zhan, Jiaxin Chen, Guangyuan Shi, Xiao-Ming Wu, and Albert YS Lam. Effectiveness of pre-training for few-shot intent classification. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 1114–1120, 2021c.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. *Advances in Neural Information Processing systems*, 28, 2015a.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *NIPS*, 2015b.

Shengjia Zhao, Jiaming Song, and Stefano Ermon. Infovae: Balancing learning and inference in variational autoencoders. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, pages 5885–5892, 2019.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. Calibrate before use: Improving few-shot performance of language models. In *International Conference on Machine Learning*, pages 12697–12706. PMLR, 2021.

Lucia Zheng, Neel Guha, Brandon R. Anderson, Peter Henderson, and Daniel E. Ho. When does pretraining help?: assessing self-supervised learning for law and the casehold dataset of 53,000+ legal holdings. *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Law*, 2021.

Yukun Zhu, Ryan Kiros, Richard S. Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 19–27, 2015.

Jacob Ziv and Abraham Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on information theory*, 23(3):337–343, 1977.

A Zvonkin and L Levin. The complexity of finite objects and the development of the concepts of information and randomness by means of the theory of algorithms. *Russian Mathematical Surveys*, 25:83, 10 2007. doi: 10.1070/RM1970v025n06ABEH001269.

APPENDICES

Appendix A

Parameters and Datasets for IBA

Dataset	Number of Dev/Test
IMDB	25000
MNLI Matched	9815
MNLI Mismatched	9832
AG News	7600
RTE	277

Table A.1: Dataset Details.

To keep as much information as possible at the beginning, $\boldsymbol{\mu}_i$ should be set close to 1,$\forall i$, in which case $\mathbf{T} \approx \mathbf{X}$. So we initialize with $\alpha_i = 5, \forall i$ and therefore $\boldsymbol{\mu}_i \approx 0.993$. In order to stabilize the result, the input of the bottleneck ($\mathbf{X}$) is duplicated 10 times with different noise added. We set the learning rate to 1 and the number of training steps to 10. We use empirical estimation for $\beta \approx 10 \times \frac{\mathcal{L}_{CE}}{\mathcal{L}_{IB}}$. For IMDB, MNLI Matched/Mismatched, and AGNews, we insert the IB after layer 9 and β is set to 10^{-5}. For RTE, we insert the IB after layer 10 and β is set to 10^{-4}.

We carry out experiments on NVIDIA RTX 2080 Ti GPUs with 11GB VRAM running PyTorch 1.4.0 and CUDA 10.0. A full technical description of our computing environment is released alongside our codebase. For LIME, we set N, the number of permuted samples drawn

from the original dataset, to 100 as this reaches the limitation of GPU memory. Similarly, the number of steps of integrated gradients is set to 10 because it is more memory intensive. The average time of running 25000 instances on the described GPU is about 10 hours for IBA, 13 hours for LIME, and 2 hours for IG.

We use the test sets when the label is provided and use the dev sets otherwise. See Table A.1 for details. Note that "IMDB" refers to the sentiment analysis dataset provided by Maas et al. [2011]. "MNLI Matched" means that the training set and the test set have the same set of genres while "MNLI Mismatched" means that genres that appear in the test set don't appear in the training set. Detailed information of the MNLI dataset can be found in Williams et al. [2018].

Appendix B

Degradation Test across All Layers

Figure B.1 shows the complete version of the degradation test across all 12 layers. In general, the earlier we insert the bottleneck, the larger the probability drop is, except for layers 8 and 9, which are the only two layers with steeper slopes than layer 1.

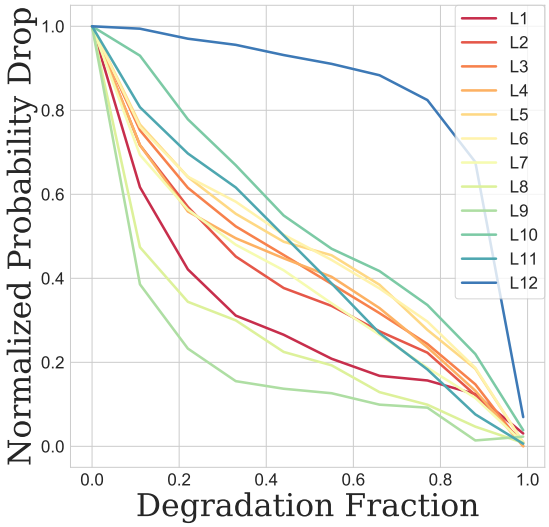
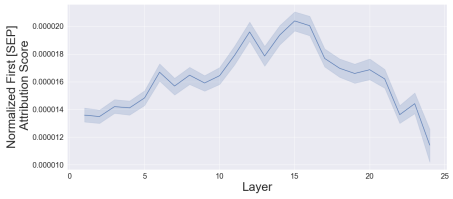


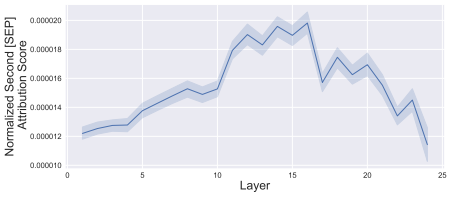
Figure B.1: Degradation test results across all layers.

Appendix C

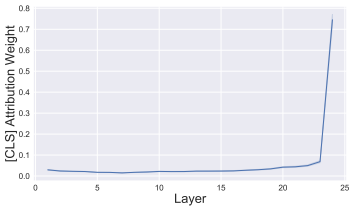
Detailed [SEP] Attribution Score across Layers



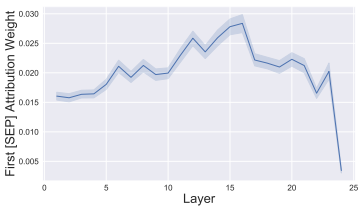
(a) Firt [SEP] Normalized Score



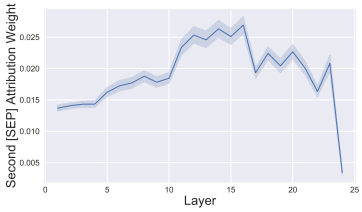
(b) Second [SEP] Normalized Score



(c) CLS attribution weight.



(d) Firt [SEP] Weight



(e) Second [SEP] Weight

Figure C.1: Special tokens attribution scores and weights.

Figure C.1 contains plots showing [CLS] weight and two [SEP] scores as well as weights across layers. Figure C.1c shows how important [CLS] is compared with other tokens across

layers—that is, we divide the attribution score by the sum of all of the tokens' attribution scores. It's even more clear that the [CLS] token aggregates all tokens' information in the final layer and becomes the most important token for prediction. The first [SEP] has slightly higher weight than the second one. It's probably because the first [SEP] indicates the boundary between query and document, which is an important information to learn for reranking. But in general they show similar patterns.

Appendix D

All 24-Layer Degradation Test

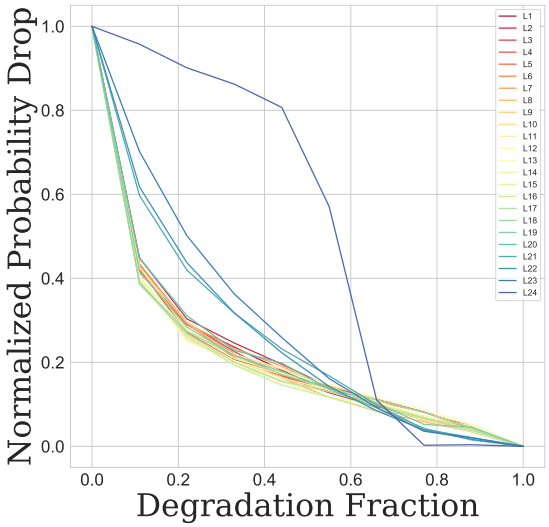


Figure D.1: 24-layer degradation test result

Figure D.1 shows the degradation test result for all 24 layers. As we can see, middle layers show the steepest slope at first, indicating they are the most capable ones of capturing important tokens. The reason why layer 24 gets a slow probability drop is because special tokens like `[CLS]`

and [SEP] are not removed in degradation test while [CLS] is regarded as the most important token in layer 24.

Appendix E

Hierarchical Latent Variable Models

The hierarchical autoencoder in the paper uses deep latent Gaussian models (DLGM) [Rezende et al., 2014] following the sampling process based on Markov chains, whose marginal distributions are:

$$\begin{aligned} p_\theta(\mathbf{x}) &= \int p_\theta(\mathbf{x}|\mathbf{z}_1)p_\theta(\mathbf{z}_1)d\mathbf{z}_1, \\ p_\theta(\mathbf{z}_1) &= \int p_\theta(\mathbf{z}_1|\mathbf{z}_2)p_\theta(\mathbf{z}_2)d\mathbf{z}_2, \\ &\dots \\ p_\theta(\mathbf{z}_{L-1}) &= \int p_\theta(\mathbf{z}_{L-1}|\mathbf{z}_L)p_\theta(\mathbf{z}_L)d\mathbf{z}_L. \end{aligned} \tag{E.1}$$

Combining the above equations, the marginal distribution of $\mathbf{x}$ is:

$$p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}|\mathbf{z}_1)p_\theta(\mathbf{z}_1|\mathbf{z}_2)...p_\theta(\mathbf{z}_{L-1})|\mathbf{z}_L)p_\theta(\mathbf{z}_L)d\mathbf{z}_{1:L}. \tag{E.2}$$

Accordingly, inference models $q_\phi(\mathbf{z}_{i+1}|\mathbf{z}_i)$ need to be defined for every latent layer. ELBO that includes multiple latent variables then becomes:

$$\mathbb{E}_{q_\phi(\cdot|\mathbf{x})}[\log p_\theta(\mathbf{x}, \mathbf{z}_{1:L}) - \log q_\phi(\mathbf{z}_{1:L}|\mathbf{x})]. \tag{E.3}$$

In this paper, we use Logistic distribution ($\mu = 0, \sigma = 1$) as the prior $p(\mathbf{z}_L)$, and use conditional Logistic distribution for both inference models $q_\phi(\mathbf{z}_{i+1}|\mathbf{z}_i), q_\phi(\mathbf{z}_1|\mathbf{x})$ and generative models

$p_\theta(\mathbf{z}_i|\mathbf{z}_{i+1})$. These distributions are modeled by neural networks, which is stacked by Residual blocks [He et al., 2016] as hidden layers. More architecture details can be referred to Kingma et al. [2019a], where they also discusses other possible topologies regarding to hierarchical latent variable models.

Appendix F

Initial Bits of BB-ANS and Bit-Swap

The main difference between BB-ANS and Bit-Swap is that BB-ANS requires the sender *Alice* to decode $\mathbf{z}_{i+1}$ with $q_\phi(\mathbf{z}_{i+1}|\mathbf{z}_i)$ for i from 1 to $L-1$ first, and then encode $\mathbf{z}_i$ with $p_\theta(\mathbf{z}_i|\mathbf{z}_{i+1})$ for i from 1 to $L-1$. While Bit-Swap interleaves this encoding and decoding procedure and applies it recursively for latent variables, as illustrated in Figure 5.4. The advantage of Bit-Swap's procedure is that, after decoding $\mathbf{z}_1$, the bits encoded from x can be used in decoding $\mathbf{z}_2$; then bits encoded from $\mathbf{z}_1$ can be used for decoding $\mathbf{z}_3$. As a result, the initial bits required for Bit-Swap is much less than BB-ANS. Concretely, for BB-ANS, the minimum initial bits required:

$$-\log q_\phi(\mathbf{z}_1|\mathbf{x}) - \sum_{i=1}^{L-1} \log q_\phi(\mathbf{z}_{i+1}|\mathbf{z}_i). \tag{F.1}$$

For Bit-Swap, the minimum initial bits required:

$$\begin{aligned} &-\max\ (0, \log q_\phi(\mathbf{z}_1|\mathbf{x})) + \sum_{i=1}^{L-1} \max\ \left(0, \log \frac{p_\theta(\mathbf{z}_{i-1}|\mathbf{z}_i)}{q_\phi(\mathbf{z}_{i+1}|\mathbf{z}_i)}\right) \\ &\leq -\log q_\phi(\mathbf{z}_1|\mathbf{x}) - \sum_{i=1}^{L-1} \log q_\phi(\mathbf{z}_{i+1}|\mathbf{z}_i). \end{aligned} \tag{F.2}$$

The initial bits Bit-Swap required is less than BB-ANS, making Bit-Swap reach the optimal compression rate.

Appendix G

Text Examples for Text Classification

Dataset	Sample Text
AGNews	“Wall St. Bears Claw Back Into the Black (Reuters) Reuters - Short-sellers, Wall Street’s dwindling band of ultra-cynics, are seeing green again.”
DBpedia	“European Association for the Study of the Liver”, “The European Association for the Study of the Liver (EASL) is a European professional association for liver disease.”
YahooAnswers	“Is a transponder required to fly in class C airspace?”,“I’ve heard that it may not be for some aircraft. What are the rules?”,“the answer is that you must have a transponder in order to fly in a class C airspace.”

20News	“Subject: WHAT car is this!? Nntp-Posting-Host: rac3.wam.umd.edu Organization: University of Maryland, College Park Lines: 15 I was wondering if anyone out there could enlighten me on this car I saw the other day. It was a 2-door sports car, looked to be from the late 60s/ early 70s. It was called a Bricklin. The doors were really small. In addition, the front bumper was separate from the rest of the body. This is all I know. If anyone can tellme a model name, engine specs, years of production, where this car is made, history, or whatever info you have on this funky looking car, please e-mail. Thanks,- IL —- brought to you by your neighborhood Lerxst —-”
Ohsumed	“Protection against allergen-induced asthma by salmeterol.The effects of the long-acting beta 2-agonist salmeterol on early and late phase airways events provoked by inhaled allergen were assessed in a group of atopic asthmatic patients.In a placebo-controlled study, salmeterol 50 micrograms inhaled before allergen challenge ablated both the early and late phase of allergen-induced bronchoconstriction over a 34 h time period.Salmeterol also completely inhibited the allergen-induced rise in non-specific bronchial responsiveness over the same time period.These effects were shown to be unrelated to prolonged bronchodilatation or functional antagonism.These data suggest novel actions for topically active long-acting beta 2-agonists in asthma that extend beyond their protective action on airways smooth muscle.”
R8	“champion products ch approves stock split champion products inc said its board of directors approved a two for one stock split of its common shares for shareholders of record as of april the company also said its board voted to recommend to shareholders at the annual meeting april an increase in the authorized capital stock from five mln to mln shares reuter ”

R52	“january housing sales drop realty group says sales of previously owned homes dropped pct in january to a seasonally adjusted annual rate of mln units the national association of realtors nar said but the december rate of mln units had been the highest since the record mln unit sales rate set in november the group said the drop in january is not surprising considering that a significant portion of december s near record pace was made up of sellers seeking to get favorable capital gains treatment under the old tax laws said the nar s john tuccillo reuter”
KinNews	“mutzig beer fest itegerejwe n’abantu benshi kigali mutzig beer fest thedition izabera juru parki rebero hateganyijwe imodoka zizajya zifata abantu buri minota zibakura sonatubei remera stade kumarembo areba miginai remera mugiporoso hamwe mumujyi rond point nini kigali iki gitaramo kizaba cyatumiwemo abahanzi batandukanye harimo kizigenza mugihugu cy’u burundi uzwi izina kidum benshi bakaba bamuziho gucuranga neza live music iki gitaramo kikazatangira isaha saa kumi n’ebyiri z’umugoroba taliki kugeza saa munani mugitondo taliki kwinjira bizasaba amafaranga y’u rwanda kubafite mutzig golden card aha niho tike zigurirwa nakumat la gallette simba super market flurep”

KirNews	“sentare yiyungurizo ntahangwa yagumije munyororo abamenyeshamakuru bane abo bamenyeshamakuru bakaba bakorera ikinyamakuru iwacu bakaba batawe mvuto kwezi kw’icumi umwaka bakaba bagiye ntara bubanza kurondera amakuru yavuga hari abagwanya leta binjiye gihugu abajejwe umutekano baciye babafata bagishika komine bukinanyana ahavugwa bagwanyi bakaba baciye bashikirizwa sentare nkuru bubanza umushikirizamanza akaba yaciye abagiriza icaha co kwifatanya n’abagwanyi gutera igihugu icaha cahavuye gihindurwa citwa icaha co gushaka guhungabanya umutekano w’igihugu iyo sentare yaciye ibacira imyaka ibiri nusu n’amande y’amafaranga umuriyoni umwe umwe icabafashe cane n’ubutumwe bwafatanwe umwe muribo buvuga ’bagiye i bubanza gufasha abagwanyi” ababuranira bakaba barageragejе kwerekana kwabo bamenyeshamakuru ataco bapfana n’abagwanyi ikinyamakuru iwacu kikaba carunguruje sentare yiyungurizo ntahangwa ariko sentare yafashe ingingo kubagumiza mumunyororo ikinyamakuru iwacu kikavuga kigiye kwitura sentare ntahinyuzwa”
Filipino	“Kung hindi lang absent yung ibang pipirma sa thesis namen edi sana tapos na hardbound”

SwahiliNews	“TIMU ya taifa ya Tanzania, Serengeti Boys jana ilijiweka katika nafasi fi nyu katika mashindano ya Mataifa ya Afrika kwa wachezaji wenye umri chini ya miaka 17 baada ya kuchapwa mabao 3-0 na Uganda kwenye Uwanja wa Taifa, Dar es Salaam.Uganda waliandika bao lao la kwanza katika dakika ya 15 lililofungwa na Kawooya Andrew akiunganisha wavuni krosi ya Najibu Viga huku lile la pili likifungwa na Asaba Ivan katika dakika ya 27 Najib Yiga.Serengeti Boys iliendelea kulala, Yiga aliifungia Uganda bao la tatu na la ushindi na kuifanya Serengeti kushika mkia katika Kundi A na kuacha simanzi kwa wapenzi wa soka nchini. Serengeti Boys inasubiri mchezo wa mwisho dhidi ya Senegal huku Nigeria ikisonga mbele baada ya kushinda mchezo wake wa awali kwenye uwanja huo na kufikisha pointi sita baada ya kushinda ule wa ufunguzi dhidi ya Tanzania.”
SogouNews	“2008 di4 qi1 jie4 qi1ng da3o guo2 ji4 che1 zha3n me3i nv3 mo2 te4 ”,“2008di4 qi1 jie4 qi1ng da3o guo2 ji4 che1 zha3n yu2 15 ri4 za4i qi1ng da3o guo2 ji4 hui4 zha3n zho1ng xi1n she4ng da4 ka1i mu4 . be3n ci4 che1 zha3n jia1ng chi2 xu4 da4o be3n yue4 19 ri4 . ji1n nia2n qi1ng da3o guo2 ji4 che1 zha3n shi4 li4 nia2n da3o che2ng che1 zha3n gui1 mo2 zui4 da4 di2 yi1 ci4 , shi3 yo4ng lia3o qi1ng da3o guo2 ji4 hui4 zha3n zho1ng xi1n di2 qua2n bu4 shi4 ne4i wa4i zha3n gua3n . yi3 xia4 we2i xia4n cha3ng mo2 te4 tu2 pia4n .”

Table G.1: Sample text for each dataset.

Appendix H

Other Reported Results on Text Classification Baselines

In Table 6.3 and Table 6.5, we report the result from our hyper-parameter setting and implementation. However, we find that we couldn't replicate previously reported results in some cases — we get higher or lower results than previously reported ones, which may be due to different experiment setting (e.g., they may use pre-trained word embeddings while we don't) or different hyper-parameter settings. Thus, we provide results reported by some previous paper for reference in Table H.1, Table H.2 and Table H.3. Note that SogouNews is listed in the first table as it has abundant resources and is commonly used as benchmark for DNNs that excel at large datasets. As the studies carried out in low-resource languages and few-shot learning scenarios are insufficient, in Table H.2 and in Table H.3, we also report the result of variants of our models like BiGRU using Kinyarwanda embeddings (Kin. W2V) and BERT$_{MORPHO}$ incorporating morphology and pre-trained on Kinyarwanda corpus (Kin. Corpus) in addition to models we use in the paper. We don't find any result reported for DengueFilipino as previous works' evaluation uses multi-label metrics.

Paper	Model	Emb	AGNews	DBpedia	YahooAnswers	20News	Ohsumed	R8	R52	SogouNews
Zhang et al. [2015a]	LSTM	✓	0.860	0.985	0.708	-	-	-	-	0.951
	charCNN	✗	0.914	0.985	0.680	-	-	-	-	0.956
Yang et al. [2016]	HAN	✓	-	-	0.758	-	-	-	-	-
Joulin et al. [2017]	charCNN	✗	0.872	0.983	0.712	-	-	-	-	0.951
	VDCNN	✗	0.913	0.987	0.734	-	-	-	-	0.968
	fastText	✗	0.915	0.981	0.720	-	-	-	-	0.939
Conneau et al. [2017]	VDCNN	✗	0.908	0.986	0.724	-	-	-	-	0.962
Yao et al. [2019]	LSTM	✗	-	-	-	0.657	0.411	0.937	0.855	-
	fastText	✓	-	-	-	0.797	0.557	0.947	0.909	-
Liu et al. [2020]	fastText	✓	0.925	0.986	0.723	0.114	0.146	0.860	0.716	-
	BiLSTM	✓	-	-	-	0.732	0.493	0.963	0.905	-
	BERT	✗	-	-	-	0.679	0.512	0.960	0.897	-

Table H.1: Results reported in previous works on datasets with abundant resources with embedding (Emb) information.

Paper	Model	Emb	PT	KinyarwandaNews	KirundiNews	SwahiliNews	DengueFilipino
Niyongabo et al. [2020]	charCNN	✗	✗	0.717	0.692	-	-
	BiGRU	✓(Kin. W2V)	✗	0.887	0.859	-	-
	CNN	✓(Kin. W2V)	✗	0.875	0.857	-	-
Kastanos and Martin [2021]	fastText	✗	✗	-	-	0.675	-
Nzeyimana and Rubungo [2022]	BERT_{BPE}	✗	✓(Kin. Corpus)	0.883	-	-	-
	BERT_{MORPHO}	✗	✓(Kin. Corpus)	0.869	-	-	-
	KinyaBERT	✗	✓(Kin. Corpus)	0.880	-	-	-

Table H.2: Results reported in previous works on low resource languages with embedding (Emb) and pre-training (PT) information.

Paper	Model	AGNews	DBpedia
Shnarch et al. [2022]	BERT	0.619	0.312
	$\text{BERT}_{\text{IT:CLUSTER}}$	0.807	0.670

Table H.3: Results reported in previous works on 64-sample learning, corresponding to 14-shot for AGNews and ≈5-shot for DBpedia.

Appendix I

Numerical Results of Few-Shot Learning

The exact numerical values of accuracy shown in Figure 6.1 is listed in three tables below.

Dataset	AGNews			
#Shot	5	10	50	100
fastText	0.273±0.021	0.329±0.036	0.550±0.008	0.684±0.010
Bi-LSTM+Attn	0.269±0.022	0.331±0.028	0.549±0.028	0.665±0.019
HAN	0.274±0.024	0.289±0.020	0.340±0.073	0.548±0.031
W2V	0.388±0.186	0.546±0.162	0.531±0.272	0.395±0.089
BERT	0.803±0.026	0.819±0.019	0.869±0.005	0.875±0.005
SentBERT	0.716±0.032	0.746±0.018	0.818±0.008	0.829±0.004
gzip (ours)	0.587±0.048	0.610±0.034	0.699±0.017	0.741±0.007

Dataset	DBpedia			
#Shot	5	10	50	100
fastText	0.475±0.041	0.616±0.019	0.767±0.041	0.868±0.014
Bi-LSTM+Attn	0.506±0.041	0.648±0.025	0.818±0.008	0.862±0.005
HAN	0.350±0.012	0.484±0.010	0.501±0.003	0.835±0.005
W2V	0.325±0.113	0.402±0.123	0.675±0.05	0.787±0.015
BERT	0.964±0.041	0.979±0.007	0.986±0.002	0.987±0.001
SentBERT	0.730±0.008	0.746±0.018	0.819±0.008	0.829±0.004
gzip (ours)	0.622±0.022	0.701±0.021	0.825±0.003	0.857±0.004

Dataset	SogouNews			
#Shot	5	10	50	100
fastText	0.545±0.053	0.652±0.051	0.782±0.034	0.809±0.012
Bi-LSTM+Attn	0.534±0.042	0.614±0.047	0.771±0.021	0.812±0.008
HAN	0.425±0.072	0.542±0.118	0.671±0.102	0.808±0.020
W2V	0.141±0.005	0.124±0.048	0.133±0.016	0.395±0.089
BERT	0.221±0.041	0.226±0.060	0.392±0.276	0.679±0.073
SentBERT	0.485±0.043	0.501±0.041	0.565±0.013	0.572±0.003
gzip (ours)	0.649±0.061	0.741±0.017	0.833±0.007	0.867±0.016

Table I.1: Few-Shot result on AG News, DBpedia, and SogouNews.

Appendix J

Choice of k and Tie-Breaking Strategies

For the results in the paper, we use $k = 2$ and "maximum" as the tie-breaking strategy. This does mean the reported accuracy is the upperbound of the method. We thus carry out the experiments using various k and various tie-breaking strategies. The "minimum" and "maximum" serves as the lowerbound and the upperbound while "random" is to pick either label when tied and the reported "random" result is based on *one* run so it may create fluctuation, shown in Table J.1. For comparison, we also add results of other two non-parametric learning methods — W2V and SentBERT under the same setting. The result is shown in Table J.2. For "random" case, choosing an odd number for k performs better than choosing an even number no matter what non-parametric methods we use.

Dataset	Tie-Breaking	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$	$k=9$	$k=10$
Kinyarwanda	min	**0.835**	0.752	0.822	0.806	0.812	0.806	0.814	0.812	0.812	0.810
	random	**0.835**	0.822	0.834	0.833	0.831	0.829	0.830	0.826	0.828	0.823
	max	0.835	**0.891**	0.868	0.867	0.853	0.855	0.845	0.847	0.842	0.840
Kirundi	min	**0.858**	0.722	0.775	0.732	0.774	0.746	0.779	0.731	0.751	0.758
	random	**0.858**	0.803	0.788	0.790	0.800	0.789	0.792	0.789	0.781	0.784
	max	0.858	**0.906**	0.827	0.828	0.824	0.848	0.800	0.835	0.831	0.800
Swahili	min	0.850	0.759	0.873	0.854	0.876	0.865	**0.883**	0.873	0.883	0.877
	random	0.850	0.839	0.880	0.883	0.885	0.883	**0.889**	0.888	0.889	0.888
	max	0.850	**0.927**	0.895	0.912	0.896	0.905	0.897	0.903	0.896	0.900
Filipino	min	0.864	0.746	0.867	0.855	0.873	0.893	0.887	0.890	**0.899**	0.893
	random	0.864	0.850	0.893	0.887	0.910	0.910	**0.913**	0.910	0.913	0.902
	max	0.864	**0.945**	0.922	0.916	0.928	0.934	0.928	0.928	0.922	0.928
Sogou	min	**0.957**	0.930	0.956	0.951	0.956	0.953	0.956	0.954	0.955	0.955
	random	0.957	0.951	**0.959**	0.959	0.959	0.959	0.959	0.958	0.959	0.958
	max	0.957	**0.975**	0.967	0.968	0.964	0.966	0.963	0.964	0.962	0.962

Table J.1: Accuracy using *gzip*($k$nn) on full OOD datasets with various tie-breaking strategies and various k.

Dataset	Non-Parametric Methods	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$	$k=9$	$k=10$
Kinyarwanda	W2V	0.812	0.806	0.814	0.814	0.816	0.823	**0.825**	0.825	0.823	0.825
	SentBERT	**0.729**	0.703	0.712	0.704	0.707	0.702	0.700	0.698	0.692	0.688
Kirundi	W2V	**0.868**	0.783	0.723	0.727	0.737	0.763	0.766	0.750	0.743	0.742
	SentBERT	**0.869**	0.742	0.646	0.645	0.674	0.664	0.657	0.660	0.644	0.641
Swahili	W2V	0.811	0.804	0.839	0.839	0.847	0.847	0.848	0.849	0.849	**0.850**
	SentBERT	0.712	0.696	0.732	0.738	0.745	0.747	**0.751**	0.751	0.750	0.750
Filipino	W2V	0.500	0.506	0.474	0.526	0.561	0.517	0.581	0.584	0.592	**0.601**
	SentBERT	0.913	0.884	0.902	0.893	**0.916**	0.916	0.905	0.899	0.910	0.905

Table J.2: Accuracy using other non-parametric methods (x+$k$nn) on full OOD datasets with *random* strategy and various k.

Appendix K

GPT In-Context Learning

CaseHold Prompt	Citing: {citing}. Holding 0: {holding_0}. Holding 1: {holding_1}. Holding 2: {holding_2}. Holding 3: {holding_3}. Holding 4: {holding_4}. Which holding is correct (0, 1, 2, 3, or 4)? Answer:
PIQA Prompt	Goal: {goal}. Option 0: {option_0}. Option 1: {option_1}. Which option is correct (0 or 1)? Answer:

Table K.1: Zero-Shot Prompts

AG News Prompt	Please classify text input into one of the following categories: World, Sports, Business, and Science/Technology. Here are some examples: Input: {Example 1 Text}, Label: {Example 1 Label} Input: {Example 2 Text}, Label: {Example 2 Label} Input: {Testing Text}, Label:
Label Token Mapping	'World': 10603, 'Sports': 18153, 'Business': 24749, 'Science/Technology': 26959
Medical Prompt	Please classify text input into one of the following categories: World, Sports, Business, and Science/Technology. Here are some examples: Input: {Example 1 Text}, Label: {Example 1 Label} Input: {Example 2 Text}, Label: {Example 2 Label} Input: {Testing Text}, Label:
Label Token Mapping	'Neoplasms': 10603, 'Digestive system diseases': 18153, 'Nervous system diseases': 24749, 'Cardiovascular diseases': 26959, 'Pathological conditions': 0
SST5 Prompt	Please classify text input into one of the following categories: World, Sports, Business, and Science/Technology. Here are some examples: Input: {Example 1 Text}, Label: {Example 1 Label} Input: {Example 2 Text}, Label: {Example 2 Label} Input: {Testing Text}, Label:
Label Token Mapping	'very negative': 10603, 'negative': 18153, 'neutral': 24749, 'positive': 26959, 'very positive': 0

Table K.2: One-Shot Prompt

Appendix L

List of Evaluation Metrics

- Normalized probability drop for degradation normalizes the degradation test result $\bar{p}(y|x')$ to $[0, 1]$ by:

$$\bar{d} = \frac{\bar{p}(y|x') - m}{o - m}, \tag{L.1}$$

  where o means the original probability on the nondegraded instance and m means the minimum of the *fully* degraded instance's probability across all interpretability models.

- Mean Reciprocal Rank (MRR) measures the reciprocal of the rank of the first relevant item in the prediction list:

$$\text{MRR} = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{\text{rank}_i}, \tag{L.2}$$

  where rank_i is the rank position of the first relevant passage returned for i-th query.

- Normalized Discounted Cumulative Gain (nDCG) takes into account the relevance of the

predictions and their positions:

$$\begin{aligned} \text{nDCG}_k &= \frac{\text{DCG}_k}{\text{IDCG}_k} \\ \text{DCG}_k &= \sum_{i=1}^{k} \frac{\text{rel}_i}{\log_2(i+1)} \\ \text{IDCG}_k &= \sum_{i=1}^{|\text{REL}_k|} \frac{\text{rel}_i}{\log_2(i+1)}, \end{aligned} \tag{L.3}$$

where REL_k is the list of relevant passages ordered by their relevance score in the corpus up to position k, so IDCG is the ideal discounted cumulative gain.