UNIVERSITY OF COPENHAGEN

This thesis has been submitted to the PhD School of The Faculty of Science, University of Copenhagen

Efficient, Adaptable and Interpretable NLP

Nils Rethmeier

Supervised by Isabelle Augenstein

Submission date: 28th April 2023

Acknowledgements

I would like to express my heartfelt gratitude to everyone who has contributed to the successful completion of this PhD thesis, which would not have been possible without the support of so many people. First and foremost, I would like to express my deepest appreciation to my principal supervisor, Isabelle, for her unwavering support and guidance throughout this journey. Her mentorship, expertise, inspired discussions, passion for hard questions, space for growth and encouragement have been invaluable, and I am deeply grateful for the opportunity to work with and learn from her. I would also like to thank my thesis committee, Christina Lioma, Ivan Habernal and Roman Klinger.

Furthermore, I thank the European Union's Horizon 2020 research and innovation programme grant No 780495, as well as the German Federal Ministry of Education and Research under grant No 01IW17001, 01IW20010, 03COV03E, and 03WKDA1A that have funded my research.

My research would not have been possible without the support of my fellow researchers, CopeNLU and DFKI lab mates, research assistants, great infrastructure, and many reviewers. They all helped create many memorable moments, opportunities for professional and personal development. I hope to be able to continue working with many of them in the future.

Next, I would like to express my sincere gratitude to my research collaborators for their unwavering support, expertise, and enthusiasm throughout our collaborations. Their diverse perspectives and valuable input have enriched our research and challenged the state of research and its interpretation. I am grateful for the friendships that have developed from our shared passion for research and the productive environment it has provided despite the pandemic. I am honored to have had the opportunity to work with such an outstanding group of individuals.

Finally, I want to thank my wife who has been my 'motivational coach', my friends Alex, Merete, Christoph, Arne, Akiko and all my other friends and family for their continued support and for providing encouragement, beverages and moments of stillness throughout lab and industry research.

Abstract

In natural language processing (NLP), a central concern is how to develop and evaluate language model pretraining that better transfers and adapts to downstream tasks. Due to their black box character, it is hard to understand how models transfers knowledge and adapt it during pretraining and downstream application. Ultimately, the goal of language model pretraining is to develop methods that improve transfer and adaption to open-ended downstream tasks, while using training data, compute and model parameters as efficiently as possible.

This thesis presents my research for the goal of "developing efficient, adaptable, and interpretable NLP representations", which expands upon existing methodology for language model pretraining and evaluation along three dimensions. (I) Improve our understanding of adaptation at the representation level by contributing a transfer and adaptation interpretability method in two works. The first work proposes a method to quantify knowledge change during pretraining, zero-shot application and fine-tuning. A second work applies this method to in-hospital patient outcome prediction to identify knowledge redundancies, unused data sources, and quantify the impact of individual model components.

(II) Contribute best practices and new methods for contrastive learning of language models and NLP representations. A third work surveys self-supervised and supervised contrastive methods in NLP to identify important theoretical aspects like energy-based models (EBM) and properties of contrastive learning objectives to inform representation learning design in NLP. A forth work uses these insights to propose a state-of-the-art citation prediction language model that introduces an efficient contrastive citation neighborhood based pretraining method.

(III) Make self-supervised pretraining more data-efficient and supervised adaptation more label-efficient by proposing a contrastive and a non-contrastive pretraining method. The fifth work proposes a contrastive language model that unifies self-supervised pretraining and supervised fine-tuning. This enables data and compute efficient pretraining of a contrastive language model from small data to reduce costs, while markedly improving zero-shot, few-shot and long-tail performance compared to large pretrained language models. The sixth and final work proposes a retrofitting method for word-embeddings in a self-supervised manner to allow data-efficient zero-shot adaptation of representations for classification, analogy and similarity tasks without using any target data.

Resumé

I naturlig sprogbehandling (NLP) er en central bekymring, hvordan man kan udvikle og evaluere sprogmodel-forudtræning, der bedre overfører og tilpasser sig downstream-opgaver. På grund af deres black box-karakter er det svært at forstå, hvordan modeller overfører viden og tilpasser det under forudtræning og downstream-anvendelse. Målet med sprogmodel-forudtræning er at udvikle metoder, der forbedrer overførsel og tilpasning til åbne downstream-opgaver, samtidig med at man bruger træningsdata, beregninger og modelparametre så effektivt som muligt.

Denne afhandling præsenterer min forskning med det formål at udvikle effektive, tilpasningsdygtige og fortolkelige NLP-repræsentationer, der udvider eksisterende metoder til prætræning af sprogmodeller og evaluering af tre aspekter.

(I) Forbedring af vores forståelse af tilpasning på repræsentationsniveau ved at bidrage til en metode til overførsel og tilpasningstolkning i to artikler. Det første arbejde foreslår en metode til at kvantificere videnændring under forudtræning, nulskudsanvendelse og finjustering. En anden artikel anvender denne metode til at forudsige om patientresultater på hospitalet for at identificere videnredundanser, ubrugte datakilder og kvantificere virkningen af individuelle modelkomponenter.

(II) Bidrage med bedste praksis og nye metoder til kontrastlæring af sprogmodeller og NLP-repræsentationer. Det tredje arbejde gennemgår selvtilpassede og overvågede kontrastive metoder i NLP for at identificere vigtige teoretiske aspekter som energibaserede modeller (EBM) og egenskaber ved kontrastive læringsemner for at informere design af repræsentationslæring i NLP. Et fjerde arbejde anvender disse indsigter til at foreslå en state-of-the-art citation prediction sprogmodel, der introducerer en effektiv kontrastiv citation neighborhood-baseret forudtræningsmetode.

(III) Gør selv-supervised pretraining mere dataeffektiv og overvåget tilpasning mere label-effektiv ved at foreslå en kontrastiv og en ikke-kontrastiv pretraining metode. I det femte arbejde foreslås en kontrastiv sprogmodel, der forener selv-supervised pretraining og overvåget finjustering. Dette muliggør data- og computereffektiv pretraining af en kontrastiv sprogmodel fra små datamængder for at reducere omkostninger, samtidig med at det forbedrer nulskuds-, fåskuds- og langhalede ydeevne markant sammenlignet med store fortrænede sprogmodeller. Det sjette og sidste arbejde foreslår en retrofitting metode til ordbetegnelser på en selv-supervised måde for at tillade dataeffektiv nulskuds tilpasning af repræsentationer til klassificerings-, analogi- og lignende opgaver uden at bruge nogen måldata.

Note: Translation created using ChatGPT with corrections by Andreas Holm, as I do not speak Danish. Only part of thesis that used ChatGPT, i.e. **no other part of the thesis uses any language generation model or tool.**

Publications

This thesis includes the following papers as chapters, listed in the order of their appearance throughout the document:

1. Nils Rethmeier, Vageesh Kumar Saxena, Isabelle Augenstein. TX-Ray: Quantifying and Explaining Model-Knowledge Transfer in (Un-)Supervised NLP. Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence (UAI). PMLR 124:440-449, 2020. doi:10.48550/arXiv.1912.00982

2. Nils Rethmeier, Necip Oguz Serbetci, Sebastian Möller, Roland Roller. EffiCare: Better Prognostic Models via Resource-Efficient Health Embeddings. AMIA Annual Symposium Proceedings and PubMed Central. 2020, 1060–1069. PMCID: PMC8075498

3. Nils Rethmeier, Isabelle Augenstein. A Primer on Contrastive Pretraining in Language Processing: Methods, Lessons Learned & Perspectives. ACM Computing Surveys (CSUR), 2022.
doi:10.1145/3561970

4. Malte Ostendorff, Nils Rethmeier, Isabelle Augenstein, Bela Gipp, Georg Rehm, Neighborhood Contrastive Learning for Scientific Document Representations with Citation Embeddings. Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP). 2022, 11670–11688. doi:10.48550/arXiv.2202.06671

5. Nils Rethmeier Isabelle Augenstein. Long-Tail Zero and Few-Shot Learning via Contrastive Pretraining on and for Small Data. AAAI Workshop on Artificial Intelligence with Biased or Scarce Data (AIBSD) and MDPI Computer Sciences & Mathematics Forum. 2022; 3(1):10. https://doi.org/10.3390/cmsf2022003010

6. Nils Rethmeier and Barbara Plank. MoRTy: Unsupervised Learning of Task-specialized Word Embeddings by Autoencoding. In Proceedings of the 4th

Workshop on Representation Learning for NLP (RepL4NLP) at Association for Computational Linguistics. 2019, 49–54. doi:10.18653/v1/W19-4307

Contents

List of Figures

List of Tables

Part I

Executive Summary

Executive Summary 1

1.1 Introduction

Recent progresses in machine learning and natural language processing have been enabled by the pretraining and fine-tuning of computationally large language models, or language representations, on vast amounts of data (Vaswani et al., 2017b; Yogatama et al., 2019a; Brown et al., 2020). However the resulting model complexity has raised three intricately connected concerns about large pretrained language representation models.

(I) A first concern is the black box character of these models which lessens human control of their (un)desireable behavior. Both the decision process, i.e. *decision understanding* or *explainability*, and the inner workings of these models should be understood, i.e *model understanding* or *interpretability* (Gehrmann et al., 2019; Belinkov et al., 2020; Ras et al., 2022), to minimize undesired, while maximizing desired properties. More concretely, this thesis extends model understanding for *adaptation and transfer understanding* to analyze, quantify, more deeply understand and ultimately manipulate and design better transfer and adaptation mechanisms.

(II) The second concern is the research of novel language model representation learning objectives that increase the data and compute efficiency of current language models and make them better capable of few-shot learning, and learning similarity, especially to boost data and compute efficient modeling which is the third concern.

The last concern (III) directly evolves from the first two, data-efficiency (Melis et al., 2020; Merity et al., 2018) and the more commonly focused compute-efficiency and memory-efficiency (Rogers et al., 2020; Tay et al., 2022; Rae et al., 2021; Hoffmann et al., 2022; Du et al., 2022). Data-efficiency can be divided into the extensively studied supervised efficiency (Treviso et al., 2022; Beltagy et al., 2022; Ruis et al., 2022), and the rarely studied self-supervised data-efficiency (Melis et al., 2020; Radford et al., 2021). Supervised data-efficiency, or sample-efficiency, refers to the ability to predict, train and adapt from as little supervised samples (labels or annotations) as possible, i.e. zero-shot learning and few-shot learning (Brown et al., 2020; Schick and Schütze, 2021). Self-supervised data-efficiency, as a core research focus of this thesis, describes the ability of a language representation learning model (language model) to pretrain, transfer and adapt from as little (unlabeled) text data as possible using only *self-supervised learning* (Melis et al., 2020; Radford et al., 2021). Thus, this thesis is concerned with investigating the relations and synergies between (I) transfer and adaptation understanding (interpretability), (II) more efficient language (model) representation pretraining objectives, and (III) data-efficient representation transfer and adaptation.

This section introduces transfer and adaptation understanding, contrastive language model pretraining, and data-efficient pretraining while pointing to their connections. The papers described in the chapters of the thesis are cross-referenced where relevant. Section 1.2 gives a summary for each thesis publication divided into the three areas of transfer understanding, contrastive pretraining transfer, and data-efficient representation transfer. Section 1.3 provides a concise summary of the contributions by area and suggests directions for future work.

1.1.1 Decision, Model, Transfer and Adaptation Understanding

Past and recent research criticize the black box character of machine learning models (Pasquale, 2015; Zarsky, 2016; Arras et al., 2017; Samek and Müller, 2019). This resulted in the creation of visual analysis methods called explainability and interpretability methods that are used to evaluate model quality and properties in detail. In language applications, these terms are often used interchangeably by researchers, which can lead to unrealistic expectations in laypeople who expect natural language explanations and interpretations. Explainability, and the connected term *decision understanding* by Gehrmann et al. (2020), determines the relevance of specific input features, intermediate features or entire data sources (global) for individual (local) task prediction decisions (Ras et al., 2022; Rethmeier et al., 2020c). Recently, European legislation has started to require that machine learning models are "able to explain individual decisions" (Regulation, 2016). Interpretability (Molnar, 2022), and the connected term *model understanding* (Gehrmann et al., 2020), describe methods that visualize learned global features of a model (Gehrmann et al., 2020). Model understanding can be divided into the analysis of supervised and self-supervised model understanding. It can also be used to analyze the change (adaptation) or transfer (unaffectedness) of model behavior during self-supervised and supervised training stages.

Supervised model understanding in NLP helps to understand and differentiate what representations a model learns for a given set of human labeled data Gehrmann et al. (2020), while using a specific combination of pretraining objective and supervised end-task. It can be used to uncover biases or parameter redundancy (Phang et al., 2021a; Rogers et al., 2020) and is useful in improving end-task specific model design (Gehrmann et al., 2020).

Self-supervised model understanding requires scalable, exploratory representation change analysis, with manageable cognitive load to first quantify and locate useful hypotheses and entry points of analysis. This is in stark contrast to supervised model understanding, which aims to verify preconceived hypotheses (biased expectations) of what constitutes useful representations and behavior according to a very specific semantic defined by a target annotation – i.e. a probing task (Talmor et al., 2020; Elazar et al., 2021). Because of this, self-supervised model understanding is an under-researched field of study, but a central focus of this thesis. It would thus be useful to explore self-supervised representation changes using model understanding (Erhan et al., 2009a) in order to analyze which part of a representation transfers or is adapted between training stages (Ramasesh et al., 2021). This would allow a human to limit analysis overload by automatically identifying the most relevant changes in representations and model predictions (Kahng et al., 2017).

In the following subsections I will overview model understanding (interpretability) of transfer and adaptation as well as decision understanding (explainability). I will also outline where model understanding can be used to analyze and understand computational efficiency, model component efficiency, and data efficiency concerns, which is the third focus of this thesis in Section 1.1.3. Furthermore, each subsection will describe open challenges and present solution approaches developed in this thesis that address them.

1.1.1.1 Model Understanding and Interpretability

Neural models automatically build representations during training. Model interpretability (Molnar, 2022), also called model understanding (Gehrmann et al., 2020) or representation analysis (Kriegeskorte et al., 2008), is a white box approach, which aims to analyze the representations inside the model by recording and visualizing activations at each layer of a model given specific input data. In NLP, interpretability may refer to interaction and visualization of representations (Belinkov et al., 2020), which model understanding focuses on (Gehrmann et al., 2020), but it may also refer to structural analyses and behavioral studies using supervised probes (Belinkov et al., 2020). This thesis is only concerned with the former, i.e. model understanding via recording and visualizing representations and their change during training. Two prominent techniques for recording activations are activation maximization and representational similarity methods, both of which are inspired by ideas from neuroscience (Erhan et al., 2009a; Kriegeskorte et al., 2008). Activation maximization (Erhan et al., 2009a; Carter et al., 2019a; Olah et al., 2017a), visualizes the maximal activation per neuron to form a prototypical visualization of what input a neuron prefers – i.e. which inputs maximally activate a neuron. In computer vision this requires generating an input (via optimization) that maximally activates a neuron (Erhan et al., 2009a). In NLP, activation maximization is more straight forward and reduces to finding the tokens that maximally activate a neuron, because generating input embeddings of non-existing tokens may lead to arbitrary interpretations.[1] Representation (similarity) analysis in neuroscience (Kriegeskorte et al., 2008) and artificial neural network learning (Morcos et al., 2018) includes methods like Central Kernel Alignment (Kornblith et al., 2019) or predecessor methods like Singular Vector Canonical Correlation Analysis (SVCCA) (Raghu et al., 2017). In NLP and computer vision, these methods record activations for the same input data and then compare the similarity of activations between layers (Kornblith et al., 2019), or between training steps to reason about the location of task relevant representations or adaptations. In later section, I will explain how this can be used to analyze how efficiently a model uses its neurons, modules and data sources 1.1.3.

1.1.1.2 Supervised Model Understanding and Transfer Understanding

Supervised model understanding visualizes which representations are active to what extent for a specific label (prediction state) at different levels of granularity for either layers (Carter et al., 2019a; Kornblith et al., 2019; Phang et al., 2021b) or individual neurons (Dalvi et al., 2019; Rethmeier et al., 2020a; Antverg and Belinkov, 2021; Stanczak et al., 2022b). For example, when comparing training step, a typical similarity comparison of representation is between activations of a pretrained model and the same model after it has been fine-tuned in order to visualize or quantify where and how much each layer activation behavior has adapted due to supervised training (Kornblith et al., 2019; Wu et al., 2020a; Phang et al., 2021b). This can also be used to analyze whether a model uses its components and data sources efficiently and whether there are redundant representations, as will be discussed in Section 1.1.3 and Paper 3 (summary 1.2.1.2). Generally, supervised learning tasks give a clear definition of which semantic is correct, e.g. in digit or object classification tasks it is expected that specific filters form activation images of digit or object prototypes for each class at upper layers (Erhan et al., 2009a; Carter et al., 2019a). This makes it possible to understand that representations of prototypes for dog noses, ears and legs (Carter et al., 2019a) are all located in a specific region or subset of the representation space, which also applies to linguistic properties in multi-lingual models (Stanczak et al., 2022b). In NLP, supervised model understanding is typically based on evaluating a specific combination of pretrained language model and fine-tuning downstream

[1] Note that, in the case of CNNs, embedding optimization has been used to break down n-gram filter neuron information into token unigrams (Poerner et al., 2018), but this raises correctness issues.

task. Many works analyze representation at neuron or layer level (Phang et al., 2021a; Durrani et al., 2020; Stanczak et al., 2022b) for masked language models of the BERT family. Fewer works aim to understand representations in autoregressive (causal) language models. For example, Wang et al. (2022a) identify neural paths for 'indirect object identification' in GPT-2., while in Rethmeier et al. (2020a) I identify underutilized (rarely activated) neurons after fine-tuning an LSTM language model to gain performance and robustness by pruning such neurons. Existing work has used supervised model understanding to help understand algorithmic bias caused by model compression (Hooker et al., 2020b,c), or to visualize how models (mis)use their layers and individual neurons (Kahng et al., 2017; Pezzotti et al., 2018; Hoover et al., 2020) and uncover ineffective parameter use (Pezzotti et al., 2018; Michel et al., 2019; Rogers et al., 2020).

This thesis contributes to supervised model understanding with two works (Rethmeier et al., 2020a,c). First, *Paper 1* (§2) is the first work to link supervised model understanding (interpretability) to dynamic neuron pruning, which improves model robustness without the overspecialization that results from pruning actually removing the neurons. The method builds per neuron feature preferences using activation maximization to quantify and differentiate per-neuron transfer (reuse) and adaptation (change) during both pretraining and fine-tuning. The work also introduces the view of self-supervised transfer and adaptation understanding, i.e. how self-supervised learning forms representations during language model pretraining and how their behavior changes during zero-shot application. For the supervised model understanding perspective, I show how specific pretrained neurons (I) forget information, (II) reuse knowledge, (III) become adapted, and somewhat surprisingly, how supervised fine-tuning, (IV) activates previously unused neurons to store novel, end-task specific knowledge, that was not present from pretraining. This fine-grained distinction of adaptation makes it possible to distinguish prunable from critically relevant neurons, i.e.' when pruning end-task specific (IV), adapted (III), or reused (II) neurons performance on the end-task suffers, while pruning unused neurons increases generalization.

In a second work, *Paper 2* (§3), I extend the approaches developed in my previous work (Rethmeier et al., 2020a) to run efficiency analysis in medical prediction models, by uncovering unused data sources, neuron level representation redundancy and neuron level bias in medical NLP. The work introduces the idea of inverting the common approach of initial expert feature selection with large models. Instead I propose explorative model understanding that first trains with all available data and small models that make it easier to interpret, debug feature, data sources, model components usage and biases. Hospitals worldwide do not have access to large amounts of GPU computing resources. Thus, instead of training large models, I develop a novel medical sequence embedding method that uses all features and 'models away' missing data, making it possible to train a very small, highly efficient, model and thereafter use interpretability to identify which features and data sources were useful, redundant or biased. The result is a 70k parameter model compared to self-attention models with millions of parameters (Song et al., 2018b; Rethmeier et al., 2020c). Analysis of the model revealed that one data source and two neuron were redundant as well as which model components were most important for each of the four prediction tasks. Finally, I discovered potential gender and ethnic biases in a single neuron, which may indicate the need for further investigation. Interestingly, around the same time, Vig et al. (2020) manipulated neurons to influence gender bias in language models, which could be useful in assessing and removing bias in future medical prediction similar to bias considerations in recent NLP publications (Stańczak and Augenstein, 2021).

1.1.1.3 Self-supervised Model Transfer and Adaptation Understanding

In this section I describe how self-supervised model understanding (interpretability) can be used to increase the efficiency of a model via model understanding based pruning (§1.2.1.1). Self-supervised models in NLP are trained using simple objectives like token masking (Devlin et al., 2019a; Rogers et al., 2020), autoregressive token prediction (Brown et al., 2020), span (Giorgi et al., 2021a) or embedding contrast (Rethmeier and Augenstein, 2022a,c) to build language representations called language models. However, understanding the token prediction used for pretraining language models is a highly complex process that is difficult to analyze since language models often have vocabulary sizes beyond 50.000 tokens (Tay et al., 2022). Because of this, the space of useful model understanding hypotheses for self-supervised or language model pretraining is potentially infinite and testing each hypothesis one by one requires labeled data, which makes supervised model understanding very costly to scale, and thus inefficient. Another core problem is that supervised probes, or fine-tuning, introduce bias during evaluation as Talmor et al. (2020); Elazar et al. (2021) demonstrated. This is because labels require learning to change representation or only utilize a subset of the pretrained representations that are useful in predicting the probe task. Therefore, for self-supervised training it is important to switch from biased hypothesis (probing) based model understanding to explorative model understanding that can guide the evaluators attention to where model representations become most affected by representations that are biased, overspecialized, transferable, generalize and are adaptable to new domains. As a result, the utility of self-supervised model understanding should cover two important analysis aspects. First, automating guidance of evaluation makes analytical scale-up easier and thus helps generalize beyond a set of supervised probes, as Muttenthaler et al. (2020) demonstrated with self-supervised question answering. Second, it can track unforeseen effects of self-supervised training by locating and quantifying how self-supervised training builds representations, how they are transferred and how well or quickly they can be adapted by successive training. Self-supervised model understanding can thus affect future model design decisions, e.g. Muttenthaler et al. (2022) found that training objectives affect model bias more significantly than model size and data amount. Recent works on self-supervised model understanding visualized how Transformer neurons capture training information as a key-value database (Geva et al., 2021), or how different language model pretraining architectures build different representations (Wu et al., 2020a), while others study what linguistic information a pretrained language model captures at the neuron (Durrani et al., 2020; Stanczak et al., 2022a) or layer level.

However, how and when pretraining builds representations and how these can (zero-shot) transfer to new domains has not been previously studied. Prior work has also not differentiated zero-shot transfer (input domain shift) from supervised adaptation (simultaneous input and output domain shift), nor how this distinction relates to pruning. This suggests that the design of less biased, more adaptable, and more efficient models can largely benefit from developing self-supervised model understanding that makes building, transfer and adaptation of representations quantifiable and tractable.

In *Paper 1* (§2) I propose to use activation maximization to build 'feature preference distributions' per neuron to quantify neuron level changes and similarities during both self-supervised and supervised training settings. During self-supervised training, this makes it possible to explore what representations language model pretraining builds at which point of training as well as which part of the pretrained representation can be zero-shot transferred to new data. For example, the work showed that pretraining learns parts-of-speech distributions during early epochs, and allows to quantify how pretrained neuron representations are reused (transferred) or ignored during zero-shot application of the language model to new data. Such self-supervised interpretability makes it possible to precisely locate and quantify how input (text) dependent domain shift changes model behavior, as opposed to probing tasks, that analyze a mix of input (text) and output (label) domain shifts, because supervised fine-tuning biases the representations to fit an end-task (Talmor et al., 2020; Elazar et al., 2021). Thus, in this work I also studied the distinction between input domain shift (zero-shot transfer) to new domain text vs. input and output domain shift (adaptation) during supervised fine-tuning. The method revealed that zero-shot input domain shift leads to many neurons transferring representations to the new data, but also that some neurons did not transfer. During supervised fine-tuning on the other hand, much more neurons are turned down or off, while others are adapted and some new ones become preferred, i.e. encode previously unknown knowledge representations. This shows that input domain shift and label (output) domain shift have very different effects on the model.

1.1.1.4 Decision Understanding

Decision understanding (Gehrmann et al., 2020) or explainability (Ras et al., 2022) quantifies how relevant each input feature is for the prediction output (class) of a single instance. For example, in the sentence “The craft left orbit with a velocity of ...” with topic label "space" it is easy to judge that the relevant term is “orbit”, perhaps in combination with “craft left orbit” and “velocity”. Such explanations are called local or single instance explainability. Global explanations accumulate local explanations across multiple instances of an entire dataset (Radensky et al., 2022). They can be used to find the most prediction relevant and irrelevant input features across a dataset, which can be used to identify biases and debug over-reliance of a model on specific input patterns (Hoover et al., 2020). For example, in patient mortality prediction, if we find that the patient chart feature “Do not resuscitate” is strongly relevant in predicting whether a patient will die in the next 48 hours, then this relevance is technically correct, but of little predictive value (Rethmeier et al., 2020c) in the clinical process of preventive care – i.e. if it is on the patient chart any laymen knows the patient will pass away soon. Prominent techniques for explainability are gradient based methods like Integrated Gradients (Sundararajan et al., 2017b), input perturbation based methods like LIME (Ribeiro et al., 2016), or more recently, contrastive perturbation methods (Stepin et al., 2021; Atanasova et al., 2022). Such methods can be applied to analyze the relevance of input features like images or text, as well as neuron or layer relevance. As pointed out before, the distinction of explainability methods and interpretability methods depends on the interpretation given by different groups, e.g. Samek et al. (2019) use Explainable AI to include both terms, while Gehrmann et al. (2020); Belinkov et al. (2020) distinguish the two terms to focus on interpretability and model analysis. As a result, methods like global explanations can be used used for interpretability or model understanding Belinkov et al. (2020).

In *Paper 2* (§3), I apply the integrated gradients method (Sundararajan et al., 2017b) to analyze global feature reliance of a model and its reliance on local time frame features for clinical decision support. Local, per instance, explanations are used to visualize which individual patient history features are strongly (counter-)indicative of a patient going into a critical health state in the next 48h. This makes it possible for medical professionals to review a summary of important recent lab results and health markers like blood values, cardio measures, progression, medication and other relevant events in a weighted importance view, which also helps point out rare abnormalities and data issues. Global, dataset wide explanations, allow one to visualize which medical features are globally important for a specific prediction task. For example, ‘comfort-measures-only‘ is strongly indicative of patient mortality, but not informative to clinicians, while ‘diet-type=diabetic‘ is unintuitively counter-indicative on average, but likely because low sugar (diabetic) diet has vast health benefits beyond specific conditions (Petroni et al., 2020).

1.1.2 Contrastive Language Model Transfer and Adaptation

Rather than only analyzing the transfer and adaptation of language models using model understanding (interpretability), this thesis also contributes a novel contrastive language model pretraining and fine-tuning method, which delivers marked efficiency gains for pretraining transfer and fine-tuning adaptation, i.e. for aspect three (efficiency) of this thesis, detailed in §1.1.3. To do so, I use contrastive learning, which is part of metric learning (Musgrave et al., 2020a). Rather than learning to directly predict a probability for a label given an embedded input, contrastive models learn to predict a similarity metric between an embedded pair of similar (positive) or dissimilar (negative) embeddings (Jaiswal et al., 2021a). The metric is also called the compatibility (LeCun et al., 2006a) or matching score (Vinyals et al., 2016). This pair can consist of two input embeddings (Jaiswal et al., 2021a) or an input embedding and a label (output) embedding (Rethmeier and Augenstein, 2022c) – i.e. from the contrastive perspective inputs and outputs are the same. Pretraining, self-supervised contrastive learning is implemented as sampling automatically constructed pairs that combine an unaltered embedding and an augmented embedding that feed a contrastive training objective. How these augmentations are constructed has many variations in computer vision (Jaiswal et al., 2021a) and NLP (Rethmeier and Augenstein, 2022c). However, sampling semantics are crucial to make learning efficient. Many works concentrate on sampling (mining) hard negatives, which are negative pairs that are similar, but not too similar to positive pairs (Cai et al., 2020). Importantly, one can not blindly increase the number of negative samples to get better quality, because sampling has to guarantee that positive and negative pairs never become the same, as such collisions confuse the optimization process and degrade performance as Saunshi et al. (2019) show in their study. Additionally, Wang and Isola (2020) point out that it is important to sample positives such that “their embeddings preserve maximal information”, as otherwise downstream performance can be limited. Contrastive objectives are particularly useful for pretraining multi-modal representations to solve text-to-text similarity tasks (Reimers and Gurevych, 2019a; Gao et al., 2021a), text-to-image similarity and translation tasks (Radford et al., 2021; Jia et al., 2021) or text-to-audio tasks (Manco et al., 2022). In self-supervised NLP pretraining, there are a few famous examples of sampling methods for contrastive objectives. The Word2vec skipgram (Mikolov et al., 2013b) model contrastively predicts (mis-)matching context words to train token embeddings for low-resource applications. SimCSE (Gao et al., 2021a) contrasts dropout augmented sentence embeddings (positives) from embeddings of subsequent sentences (negatives) for self-supervised pretraining of Transformer representations to better perform at similarity tasks. Lastly, Sentence-BERT (Reimers and Gurevych, 2019a) and followup models use entailment as supervised pretraining of sentence similarity embeddings. Another major advantage is that contrastive learning uses embedding pairs for training, which allows for zero-shot predictions in any and across modalities by expressing a new class (similarity pattern) or task as a function of existing embeddings. In NLP this means contrastive models can run non-discrete text-to-text prediction (Rethmeier and Augenstein, 2022a), which allows generative (Wu et al., 2020b) or discriminative (Halder et al., 2020) prediction. In computer vision and multi-modal models this means predicting images from text as in CLIP (Radford et al., 2021) or images from augmented images as in SimCLR (Chen et al., 2020b).

In the following section I provide an overview of basic types of contrastive learning objectives used for NLP, as well as how they can be used for supervised pretraining. The subsections also outline my contributions to data-efficient self-supervised pretraining, and how this benefits the label-efficiency and parameter-efficiency of a language model. These properties are especially desirable when pretraining models on domains where data and/or compute resources are scarce, as is often the case for medical applications and low-resource languages.

1.1.2.1 Energy-based Models and Noise Contrastive Estimation

While contrastive objectives are part of a larger family of metric objectives, Khosla et al. (2020a) point out that “recent computer vision works (Tian et al., 2019; He et al., 2020; Chen et al., 2020b) have shown that for self-supervised learning, contrastive losses are superior to other metric objectives such as the triple, max-margin or N-pair loss”. The two most popular contrastive learning objectives for self-supervised learning in computer vision and NLP are Ranking and Binary Noise Contrastive Estimation (RNCE, BNCE) (Gutmann and Hyvärinen, 2010; Mnih and Teh, 2012; Ma and Collins, 2018). In addition, contrastive learning across computer vision and

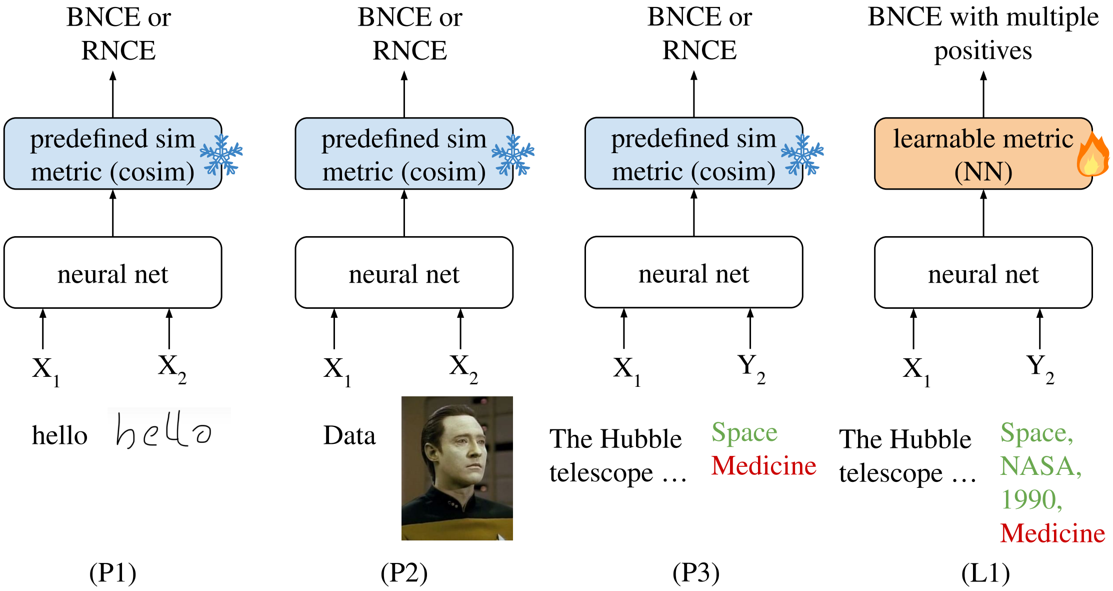


Figure 1.1: A subset of different EBM formulations.

NLP is popularly described through the lens of modern energy-based model (EBM) formulations, because EBM modeling is less restrictive than probabilistic models, and EBMs make it easy to learn relations, translations and representational fusions between modalities. Regarding being less restrictive, EBMs are “non-normalized or self-normalizing probabilistic models, that optimize an unnormalized negative log-probability called the energy function” as both LeCun et al. (2006a); Song and Kingma (2021) explain in their surveys on EBM training methods.

Regarding uni and multi-modal learning, neural EBMs and contrastive methods embed inputs X_i or labels Y_i to form embedding pairs between which to compute similarities (compatibility, or matching) using a similarity function, instead of predicting the label or score Y_i of an input X_i directly like probabilistic models. This score is then fed into a contrastive loss such as Ranking NCE or Binary NCE. Many architectures (P1-P3) (Aberdam et al., 2021; Radford et al., 2021; Gao et al., 2021a) use a predefined similarity (metric) function like cosine similarity which can only learn with exactly one positive sample and many negative samples, because only one pairing can produce maximal similarity. Figure 1.1 shows several contrastive modeling approaches expressed in the EBM notation originally introduced in LeCun and Huang (2005a); LeCun et al. (2006a). For example, EBM architecture P1 takes text X_1 and hand writing X_2 to learn handwriting recognition (Aberdam et al., 2021). Architecture (P2) on the other hand takes an image X_2 and learns compatible (matching) text X_1 as for example in CLIP, which trains to match images and their description to run zero-shot text classification on new images (Radford et al., 2021), while the prediction direction could also be reversed to build an image search. The two common losses, Ranking and Binary NCE have some connections to other, very commonly used losses. In NLP Ranking NCE is also referred to as the MultipleNegativeRanking loss (Wang et al., 2022b) while computer vision research uses the term InfoNCE (van den Oord et al., 2018; Chen et al., 2020b) to describe this objective. Ranking NCE "strongly relates to the softmax learning objective" in that it undersamples negative (non-active classes) in the normalizer (partition function) to become self-normalizing – i.e. approximately normalizing (Hjelm et al., 2019a; Rethmeier and Augenstein, 2022c). Like the softmax, it is suitable for multi-class prediction problems where classes are mutually exclusive. Binary NCE (Gutmann and Hyvärinen, 2010; Ma and Collins, 2018) originally only uses a single positive pair per sample due to the usage of cosine similarity, which only allows exactly one pairing of embeddings to match perfectly, while the normalizer (partition function) is assumed to be one (Hjelm et al., 2019a). However, this is a limitation, does not allow using Binary NCE for multi-label problems, where multiple perfect matches are required.

In Paper §6, instead of using the traditional single positive pair per sample (Gutmann and Hyvärinen, 2010; Mnih and Teh, 2012; Ma and Collins, 2018), I proposed extending Binary NCE to support using multiple positive samples per instance. Since the traditional cosine metric only allows one match, I instead used a neural network to learn a complex similarity function, see L1 in Figure 1.1, to allow multiple pairings of the same input with different label embeddings to produce multiple perfect similarities – i.e. metric multi-label (match) learning. This objective is then used for self-supervised language model pretraining by sampling in-batch words that occur (positive pairing) or are missing (negative pairing) from the current text instance (Rethmeier and Augenstein, 2022a). The result of this pretraining is an ability to zero-shot predict labels that are made up of tokens (text), which is similar to differentiable prompting, where the underlying embeddings of a label can be fine-tuned (Zhang et al., 2022). The contrastive pretraining is effective even for very small corpora and with small models, which is a property that is in line with EU regulations regarding minimizing data usage (Regulation, 2016), while also being an important criterion during application. Additionally, pretraining and subsequently fine-tuning with this loss produces markedly improved few-shot and long-tail transfer, even compared to a 10x larger Transformer, while taking 5x less compute time than fine-tuning the Transformer model. Since this approach to feeding Binary NCE allows multiple positive pairs (multi-labels), that are not mutually exclusive (no normalizer), and because the loss formulation is the same as Binary Crossentropy, except that logits are actually learned similarity scores, Binary NCE can be readily understood as and undersampled Binary Crossentropy. This relation of losses is also pointed out by (Hjelm et al., 2019a), but only for multi-class semantics rather than multi-label (multiple positives) semantics.

1.1.2.2 Supervised and Self-Supervised Contrastive Pretraining

Contrastive methods can be divided into supervised and self-supervised methods, each of which brings desirable benefits, especially regarding efficient transfer and adaptation, which is discussed in more detail in the section below 1.1.3. For example, NLP works like Gunel et al. (2021) show that supervised contrastive learning improves robustness to noise and learning from limited data. Zhang et al. (2021) add instance discrimination to learning entailment and contradiction to learn better reasoning models, while Klein and Nabi (2020) use sentence pair annotations as distantly supervised pretraining for better reasoning models. Zimmermann et al. (2021a) show that supervised contrastive learning is able to invert a data generation process efficiently and even from very little data. Self-supervised contrastive methods are effective for representation learning. Parulekar et al. (2023) show that in computer vision, self-supervised contrastive representations learned via InfoNCE (RankingNCE) preserve cluster information of image categories – e.g. the dog likeness of an image. Mikolov et al. (2013a) learned contrastive word representations, Giorgi et al. (2021a); Meng et al. (2021); Wu et al. (2020b) pretrain an already pretrained masked language model to become a contrastive language model, while Gao et al. (2021a) learn sentence representations for similarity search. However, all of these contrastive pretraining works rely on already otherwise pretrained Transformers like RoBERTa (Liu et al., 2019b).

Instead, in Paper 5 §6 I propose a contrastive language autoencoder model and train it from scratch using 'text-embedding to text/label-embedding' pretraining rather than text-to-text pretraining over discrete tokens like T5 (Raffel et al., 2020). The model is a Joint Embedding Prediction Architecture (JEPA)[2] that enables effective pretraining with very little data, but also outperforms large Transformer fine-tuning in few-shot and long-tail learning despite using a magnitude less parameters and compute compared to just fine-tuning a standard Transformer model. Because of this, the model can be considered a replacement, where the standard large pretrained language model approach is impractical or too costly.

1.1.3 Data and Compute-Efficient Transfer and Adaptation

So far I discussed how to analyze and improve transfer and adaption of language model pretraining, whereas this section ties these two aspects more closely to data-efficiency and compute efficiency concerns, to improve pretraining as a whole. Cur-

[2] A term introduced by Yann LeCun https://ai.facebook.com/blog/yann-lecun-advances-in-ai-research

rently, large pretrained models dominate NLP research. This practice is based on the experience of improved downstream performance for NLP tasks and has consolidated the assumption that pretraining with a lot of data from many domains will create representations that can be reused for any downstream task. However, failure cases of transfer such as Cleverhans effects are less widely reported (McCoy et al., 2019a). This assumption, sometimes referred to as BERTology (Rogers et al., 2020), is convenient in decreasing the effort for research and industry application progress through reuse, but can potentially be problematic for the following four reasons. First, a large pretrained NLP transformer may inherit its gender or ethnic biases (Bender et al., 2021b) to a downstream domain like medical decision support. Second, the models are large, and as a result comparatively slow on inference, which causes increased deployment time costs. For example, regulations for applications of medical NLP require that the training data is only-accessible on-site, meaning cloud compute is not an option, and that any hospital will first have to build an AI compute infrastructure, which at least in Europe is not common, because the involved security, personnel, maintenance, and running costs are a large extra cost, unlike in the IT industry (and academia), where compute is a core component of business. Third, large models cause increased development time costs, as modifications in data or models are slow to iterate because of compute requirements – regardless of domain. Four, large models increase the effort and compute of decision and model understanding techniques, especially in domains that are highly regulated and demand high model trust and decision understanding, where this additional effort can quickly increase development evaluation costs and personnel effort – each time the model needs to be updated.

From these points, it becomes evident that, large third-party pretrained models can be quite inefficient beyond their reuse for potentially improved transfer, which has lead to research into how to make them more efficient. In NLP, works on improving the training, inference and memory efficiency of Transformers aim to reduce deployment and development time of architectures used for pretrained models – Tay et al. (2022) give a good summary of recent techniques. In computer vision, contrastive self-supervised pretraining such as BYOL (Grill et al., 2020) or MOCO derivatives (He et al., 2020) have been successful, but are criticized for requiring very large amounts of compute, data and careful augmentation strategy exploration to be effective. A second core aspect of efficiency is sample-efficiency during supervised fine-tuning to adjust a pretrained model for a specific downstream task. To address this, existing research increases the few-shot capabilities of large pretrained language models using different losses (Hinton et al., 2015), continued pretraining or giving context (prompting) (Wu et al., 2022) to condition a language model into task specific text generation. Works on compute and memory efficiency limit tunable parameters to a few existing

(Ben Zaken et al., 2022a) or newly introduced ones (Houlsby et al., 2019), to pruning the trained network (Frankle and Carbin, 2019a), or quantize it (Kim et al., 2021a), i.e. reduce the compute precision of the model. Surprisingly, few to no research has studied self-supervision or pretraining data efficiency, i.e. self-supervised pretraining of a model with magnitudes less text data than current pretrained models require to be effective.

In the subsections below, I investigate the combination of small data, small model pretraining, which made it apparent that such pretraining can be very cost efficient and solve domain mismatch or compute requirement issues in situations where hardware, data, bias inheritance concerns, access and regulations prohibit using larger common setups. A welcome side-effect is that small model pretraining reduces the dependence (customer login) on other external institutions to provide pretrained models and allow low-resource domains like smaller languages or health to iterate faster in developing their own models due to the largely decreased hardware requirements. It also increases iteration speed or cost scaling efficiency in setting where data and compute are of no concern.

1.1.3.1 Data-Efficient and Parameter-Efficient Transfer and Adaptation

When seen from a transfer and adaptation perspective, it is desirable to use models that are able to train and adapt using minimal (self-)supervised data and reduce model complexity to allow easy modifications, as well as model and decision understanding, with fast iteration speeds. To reduce the amount of parameter updates, methods like Adapters (Houlsby et al., 2019) help reduce the fine-tuning parameter updates, while many efficient Transformer modification have been developed to make larger training tasks with these models more computationally efficient (Tay et al., 2022). A second aspect of efficiency is data-efficiency which can be further divided into fine-tuning data, i.e. label, efficiency and pretraining data efficiency. i.e. the ability to pretrain effectively from little data. For label-efficiency, models like GPT3 (Brown et al., 2020) or prompting methods like Schick and Schütze (2021) can produce good zero to few-shot predictions. However, the aspect of pretraining data efficiency, i.e. pretraining from little data, has rarely been studied (Melis et al., 2020). A last perspective on efficiency is model compression or simply using models with less parameters, especially while preserving robustness. For example, (Hooker et al., 2020a,c) showed that large models loose important long-tail information of rare features during model compression, which effectively disadvantages specific groups during prediction and introduces undesirable majority bias.

In Paper 2 (§3) I efficiently train a medical prediction model with a magnitude less parameters than previous LSTM and Transformer based models, while achieving significantly better performance in all prediction scores. This model is designed to be small enough to effectively be trained on a single in-hospital GPU. Additionally, since the model is small, it is computationally cheap to run models understanding and decision understanding on it, which is important because medical model have high trust requirements.

In Paper 5 (§6) I demonstrate that large Transformer models should not be assumed to be proficient at long-tail prediction. Instead of compressing large models, I pretrain a contrastive language model on a few megabytes of text data given by the task, to increase few-shot and long-tail prediction at markedly better quality than a standard pretrained transformer produces after fine-tuning. Pretraining plus fine-tuning of this contrastive model is 5x faster than fine-tuning the Transformer model, which demonstrates the computational benefit. Additionally, a contrastive Transformer variant such as TARS, that can only learn in a supervised manner (Halder et al., 2020) has a training speed of up to 2.000k samples per second on a V100 GPU, while the custom contrastive model achieves 52.000 samples per second. The contrastive model also needs no learning rate schedules, warmup, or layer normalization to pretrain stably, unlike Transformer pretraining, which is also known to be unstable during pretraining (Nguyen and Salazar, 2019) and training from scratch (Yogatama et al., 2019b), due to LayerNorm problems (Nguyen and Salazar, 2019).

In Paper 6 (§7) I propose an embedding retrofitting method which produces variations of existing word embeddings that lead to better downstream performance on 18 end-tasks. This process only used the word embedding and no target domain data which results in a zero-shot adaptation process that can be run in a few seconds on a CPU. The method can be used to produce a better word embedding for a single task or a better embedding for a set of tasks – i.e. for multi-task learning.

1.2 Scientific contributions

1.2.1 Decision, Model, Transfer and Adaptation Understanding

1.2.1.1 Paper 1: TX-Ray: Quantifying and Explaining Model-Knowledge Transfer in (Un-)Supervised NLP

State-of-the-art NLP methods for explainable artificial intelligence (XAI) focus on explaining per-sample decisions in supervised end or probing tasks – i.e. decision understanding (Arras et al., 2017; Samek et al., 2019). Interpretability, or model

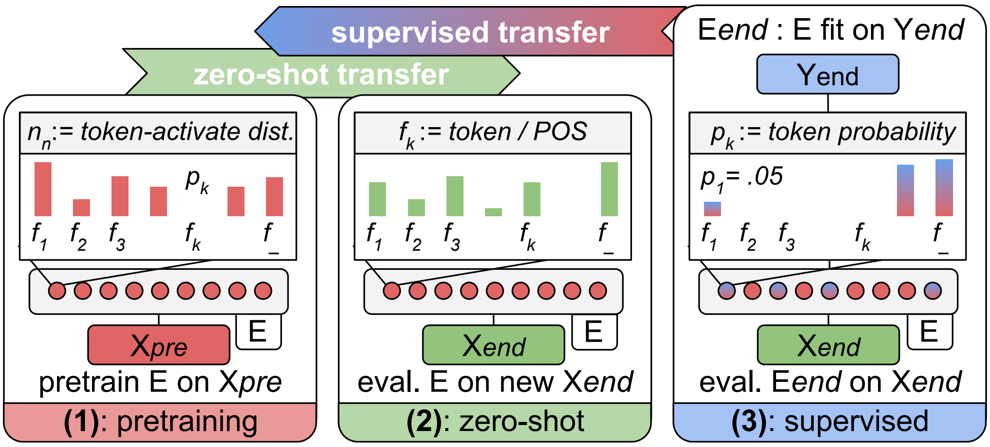


Figure 1.2: Example uses of TX-Ray: for transfer learning and model interpretability. **Left (1):** pre-train a sequence encoder E on corpus X_{pre} and collect feature preference distributions (red bars) over input features (e.g. words) f_k.[3] **Middle (2):** apply, but not re-train, the encoder E to new domain inputs X_{end} and observe the *changed neuron activation* (green). Similarities in red and green reveal zero-shot forward transfer potential or data match between X_{pre} and X_{end} according to E. **Right (3):** fine-tune encoder E on supervision labels Y_{end} to reveal 'backward' transfer of supervision knowledge into the encoder's knowledge abstractions.[4]

understanding, methods on the other hand analyze the activations of a model over a corpus of supervised probing tasks (Belinkov et al., 2020). However, this is insufficient to deeply understand and quantify how models absorb knowledge during self-supervised pretraining, zero-shot apply the learned knowledge to a new domain, or how the model knowledge transfers and adapts during supervised training. Instead, in this work, I **make model knowledge transfer and adaptation trackable and quantifiable by designing a novel method of expressing each neuron as a 'distribution of maximal input feature preference' and recording feature preference changes during pretraining, zero-shot application to a new domain, and finally during supervised fine-tuning – as seen in Figure 1.2.** Additionally, I **propose using this method for pruning that is guided by 'feature preference distribution' changes of each neuron after supervised fine-tuning.** Transfer and adaptation analysis are run in three experiments, while a fourth experiment is aimed at more deeply analyzing the knowledge sparsification induced by supervised fine-tuning via fine-grained neuron pruning and activation analysis – for full details see Paper 1 §2.

I first study model understanding during pretraining, and find that the language model learns the distribution of parts-of-speech (POS) tags during early pretraining epochs – i.e. the POS tag frequency distribution and POS tag activation strength distribution during early epochs are very similar. Furthermore, this produced two more insights. One, that **pretraining uses the majority of the neurons to learn feature preference**, i.e. 89% of model neurons build a feature preference, while only

11% do not exhibit maximal feature activation preference – i.e. 11% of neurons remain unpreferred during pretraining. Two, that during later epochs of pretraining more neurons adapt a preference, where **each neuron feature preference becomes more varied over time**, while these preferences also converge to stable distribution over later epochs in analogy to training loss convergence. This means that certain feature preferences and feature types are learned before others resulting in an implicit feature learning curriculum. In combination, this demonstrates that pretraining broadly adapts the language model neurons to build knowledge representations.

Secondly, I **study the zero-shot knowledge transfer of a pretrained model to a new text domain**, which shows that the language model still broadly activates on 99% of the same preferred neurons from pretraining. However, despite this broad reuse of neurons, the experiment also showed that the preference distributions per neuron changed significantly from the preference over the original pretraining corpus, while a subset of neurons responded similarly on both corpora. Thus, **this model understanding method allows for a more fine-grained view of model knowledge transfer by making it possible to differentiate neurons that are 'transferable but have to be adapted by way of continued training' from neurons that can 'transfer directly without the need for adaptation'**. This notion also shows that comparing data domains without the model perspective is not necessarily informative, because the transfer (overlap) between domains is determined by what the model learned from the first domain, not by what information or knowledge is actually contained in that first domain. Thus, the quality of knowledge absorption is necessarily dependent on the pretraining objective, which is a central point of motivation for the combination of model understanding and contrastive learning throughout this thesis.

Third, I study how supervised fine-tuning transfers and adapts model knowledge. I find that, as seen in Table 1.1, supervision not only sparsifies feature preferences (A), i.e. makes neurons unpreferred, but also adapts neuron preferences (B,C), and even adds preferences to the 11% of previously unused neurons (D). When pruning the removed (unpreferred) neurons (A), the model gains test set generalization, whereas when pruning adapted neurons (B,C) the model looses test set performance depending on how often the neurons are used. Specifically, rarely used preferred neurons have little impact after pruning (B), while frequently used neurons have a large impact on performance (C). Finally, when pruning neurons that were added (became preferred) due to supervision, the test set generalization of the model dropped most significantly.

Together these insights demonstrate that model understanding can not only be used to quantify model knowledge transfer and adaptation in a fine-grained manner, but also links model understanding (interpretability) to model pruning.

Which neurons prunded?	% AM of 675	F1 change %		pruning effect
		train	test	
none = baseline	100.000	0.00	0.00	–
A: 740 **avoided**	–	3.65	2.80	↓ noise, ↑ generality
B: 20 least **prefered**	0.004	-3.79	0.00	↓ over-fitting
C: 20 top **prefered**	83.120	-4.99	-1.43	↓ generalization
D: 85 **sup gained**	3.006	-3.71	-3.87	↓ sup. knowledge

Table 1.1: **Pruning avoided (removed), preferred and supervision-added neurons:** After supervised encoder fitting; (A) prunes removed (unpreferred) neurons, (B,C) prune the least and most preferred adapted neurons, and (D) prunes 85 neurons gained due to supervision. Colors represent relative score change in % from original – score drops (red,−), gains (blue).

1.2.1.2 Paper 2: EffiCare: Better Prognostic Models via Resource-Efficient Health Embeddings

In medical prognostics, existing methods adapt complex, high data-resource language models from NLP to the typically low-data resource medical data setting, which has resulted in only moderate performance improvements for clinical prediction tasks in recent works. Additionally, most hospitals do not have the compute resources to train large models and run scalable decision and model understanding on them. This is an issue, especially because trust through decision and model understanding is paramount in the clinical setting, which goes so far as regulation demanding decision understanding (Regulation, 2016). Instead, I this work, I propose combining small models that require minimal compute, while still outperforming large transformer based models. I first contribute **a novel sequence embedding method for electronic health records (EHR), which combines the language and real valued features contained in EHR data into a shared embedding format. This eliminates the traditional domain feature-engineering and makes feature selection and feature amount scaling a part of the model, rather than employing costly, manual expert domain feature selection up front.** The **proposed neural model not only very markedly outperforms the state-of-the-art results on all four patient outcome and in-hospital prediction task, but is also 17 times smaller, more data-efficient, less complex and thus more accessibly interpretable, than the previous state-of-the-art (transformer) model**. Additionally, by applying **the embedding retrofitting technique previously developed in Paper 6 §7**, Section 1.2.3.2, I **further improve the prediction performance of the model**. Running the model understanding technique developed in Paper 1, Section 1.2.1.1, **I find that there is little to no redundancy**

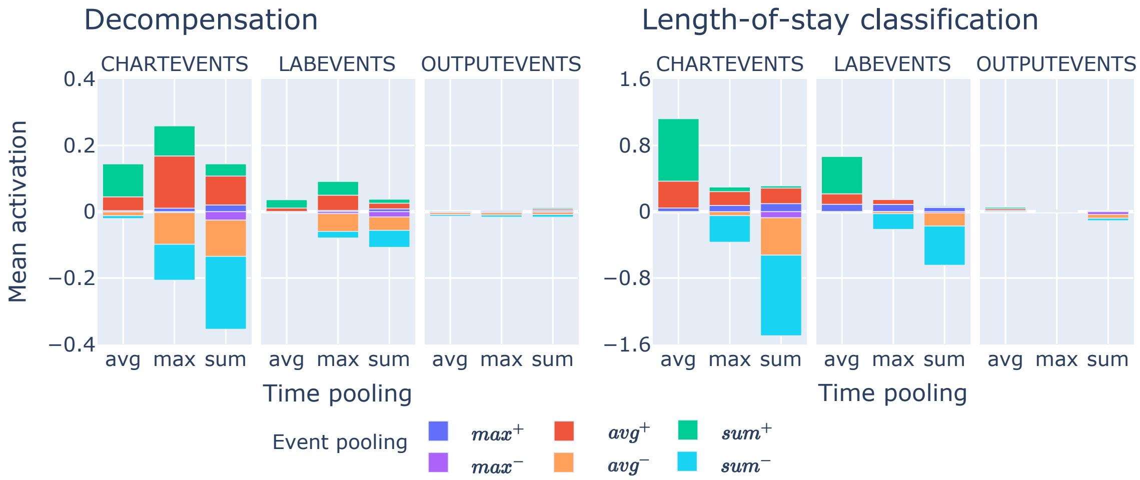


Figure 1.3: Feature type and pooling mechanism relevances for patient decompensation (organ failure) and length-of-stay classification. The x-axis shows relevances for model components such as max, average and sum pooling over time (high level patient embeddings). The color bars express relevances of pooling *over events of a single time step* Figure 3.2. Output events are not used when predicting either task.

regarding which subcategory of medical domain knowledge each neuron encodes, but also that there are neurons that capture potentially biased information such as gender and ethnicity. Furthermore, adding per-task relevance to the same technique, I determine how important each model component is per prediction task as well as how important each of the four data-sources is, which for example reveals that one data-source is not required for the tasks of length-of-stay classification, as seen in Figure 1.3. The final experiment visualizes the indicators for whether a patient dies withing the next 48h as well as which global features indicate patient mortality using the Integrated Gradients method developed by Sundararajan et al. (2017c). The global feature analysis reveals that certain features are indeed highly indicative of mortality, but that some of them are also not very informative and should thus be removed (deselected) as an input feature.

In summary, model understanding allows to develop a very data and compute efficient model design via component and data relevance analysis, as well as for post-hoc (delayed) feature selection, along with redundancy and bias checks.

1.2.2 Contrastive Language Models based Transfer and Adaptation

1.2.2.1 Paper 3: A Primer on Contrastive Pretraining in Language Processing: Methods, Lessons Learned and Perspectives

Modern natural language processing (NLP) methods employ self-supervised pretraining objectives such as masked or autoregressive language modeling to boost the performance of various downstream tasks. Contrastive pretraining on the other hand has received comparatively little attention in NLP, which is surprising since contrastive self-supervised training objectives have been the core of recent pretraining advancements in image representation learning. I thus **contributed a primer on contrastive pretraining in NLP to make it easier for NLP researchers to incorporate contrastive pretraining as well as to help them better understand, modify and become aware of important methodological considerations including the learning theoretical advantages of contrastive methods**. This is especially helpful to learn since contrastive methods are designed to improve transfer and adaptation during learning (Radford et al., 2021; Wu et al., 2020b; Chen et al., 2020b). To make it as easy as possible to learn the concepts, current application and potential future research directions of contrastive NLP methods I organize the study into four major parts. First, I introduce Noise Contrastive Estimation methods, **how they relate to commonly known Softmax and Binary Cross Entropy losses, when and how to use them effectively, which pitfalls to regard during training sample creation, as well as their connection to the more general framework of energy-based models (EBM)**. I also **introduce the notion of input-input contrastive and input-output contrastive EBMs to give special attention on how to design computationally efficient contrastive NLP methods**, since computational complexity is critical concern when pretraining using contrastive methods. The second and third part aim to provide inspiration on how to directly apply the contrastive modeling by **overviewing recent supervised contrastive pretraining methods in part two**, and **self-supervised contrastive pretraining methods in part three**. Both supervised and self-supervised contrastive pretraining methods are categorized into input-input and input-output contrastive approaches to point out their advantages, limitations and current state-of-the-art NLP applications of either. The last part focuses on open challenges and research opportunities to encourage increased research into contrastive NLP, because its properties of unified multi-modal fusion, probabilistic generalization, robustness and decades of research deep connection to self-supervised learning make contrastive methods an ideal candidate for future NLP and multi-modal research milestones in machine learning development.

Overall, the survey gives the theoretical tooling, in-practice examples and future research directions to contribute to future developments in contrastive language model pretraining as well as NLP and mutli-modal models as a whole.

1.2.2.2 Paper 4: Neighborhood Contrastive Learning for Scientific Document Representations with Citation Embeddings

Scientific document representations can use contrastive learning objectives to learn desirable similarity semantics for tasks such as document relatedness. Prior work on scientific document relatedness (Cohan et al., 2020b) has used discrete, unidirectional citation relations. However, detecting scientific document similarity despite lacking a direct citation is a central goal of document similarity tasks, which means that a hard cut-off will ignore such relations, and that a unidirectional citation, as done in (Cohan et al., 2020b) can not be sufficient for document similarity training. Instead, we use nearest neighbor sampling over citation graph embeddings that is controlled by a novel data-driven sampling margin for efficient contrastive learning of document representations, as seen in Figure 1.4.

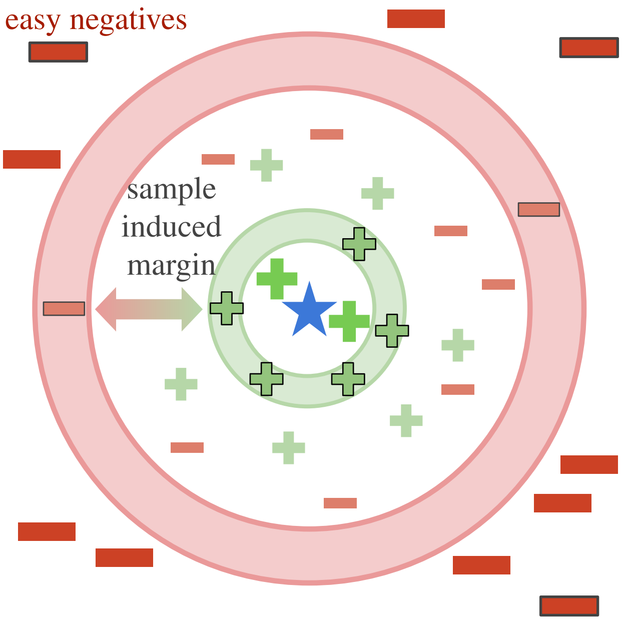


Figure 1.4: Given a query paper ★ in citation graph embedding space. A hard positive ✚ is a graph embedding that is sampled from a similar (close) context of ★, but are not so to the anchor embedding that gradients become zero. A hard (to classify) negatives ▭ (red band) is close to the positives (green band) up to a *sampling induced margin*, that can be selected as a hyperparameter. Easy negatives ▬ are very dissimilar (distant) from the query paper ★.

The sampling margin both generates informative hard-to-learn negatives and simultaneously avoids collisions with positives, as such collisions would otherwise degrade learning quality and speed. As a result, the method produces a new state-of-the-art on the SciDocs benchmark – i.e. a citation and document relatedness benchmark. Additionally, the method trains (or fine-tunes) in a sample-efficient manner with as little as 10% of the available data, while it can also be effectively combined with recent training-efficient methods like BitFit (Ben Zaken et al., 2022a). Advantageously, training a general-domain language model using the method, i.e. without using a model that was specifically pretrained on scientific text, it outperforms baselines that were pretrained on in-domain data, which nullifies the need for related domain pretraining, including data and compute costs.

1.2.3 Data-efficient Representation Transfer and Adaptation

1.2.3.1 Paper 5: Long-Tail Zero and Few-Shot Learning via Contrastive Pretraining on and for Small Data

The loss of long-tail information during model compression has been linked to algorithmic fairness considerations due to the disproportionate loss of minority (tail) information compared to majority (distribution head) information (Hooker et al., 2020a,c). Crucially, in NLP, compression is based on the assumption that large off-the-shelf language models capture long-tail information well in the first place which raises two questions. The first question is then: "how well does a standard approach of fine-tuning a pretrained Transformer language model actually capture long-tail information?". The second question then becomes whether small language models can be made to capture long-tail information, which would have implications on whether compression is a good method and necessary step for long-tail preservation. To answer these two questions, **I study the fine-tuning performance of a pretrained Transformer on a challenging long-tail, web text classification task**. Secondly, **I contribute a novel contrastive pretraining objective to pretrain a small contrastive language model from a small pretraining corpus of 60 megabytes. The method unifies self-supervised pretraining, and supervised long-tail fine-tuning, which markedly increases tail data-efficiency and tail prediction performance** – see Figure 1.5. **I additionally find that, the contrastive language model has superior zero-shot, few-shot and long-tail fine-tuning performance compared to the Transformer model.** The combined time taken for pretraining and fine-tuning of this contrastive language model is 5 times faster than the Transformer model requires for fine-tuning only, because it takes the Transformer model many epochs to capture the long-tail nature of the task. When increasing the amount of per-epoch self-supervision labels and contrastive model size to 10 million parameters, without

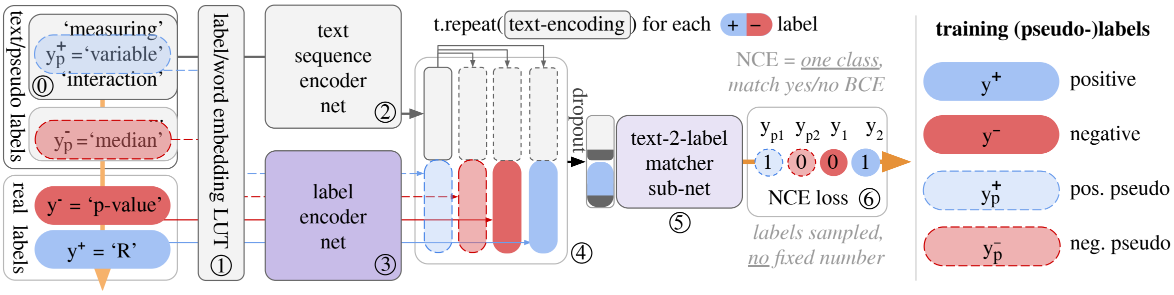


Figure 1.5: Contrastive <text, pseudo/real label> embedding pair matcher model: A word embedding layer E ① embeds text and real/pseudo labels, where labels are word IDs. CLESS embeds a text ('measuring variable interaction'), real positive (R) or negative (p-value) labels, and positive (variable) or negative (median) pseudo labels. A sequence encoder T ② embeds a single text, while a label encoder L ③ embeds c labels. Each text has multiple (pseudo) labels, so the text encoding $\mathbf{t}_i$ is repeated for, and concatenated with, each label encoding $\mathbf{l}^{\circ}_{i,l}$. The resulting batch of <text embedding, label embedding> pairs $[[\mathbf{t}_i, \mathbf{l}^{\circ}_{i,1}], \ldots, [\mathbf{t}_i, \mathbf{l}^{\circ}_{i,c}]]$ ④ are fed into a 'matcher' classifier ⑤ that is trained in ⑥ as a binary noise contrastive estimation loss L_B (Ma and Collins, 2018) over multiple label (mis-)matches $\{0, 1\}$ per text instance $\mathbf{t}_i$. Unlike older works, we add contrastive self-supervision over pseudo labels as a pretraining mechanism. Here, the word 'variable' is a positive self-supervision (pseudo) label for a text instance $\mathbf{t}_i$, while words from other in-batch texts, e.g. 'median', provide negative pseudo labels.

increasing the pretraining data size, the model markedly gains zero-shot, few-shot and long-tail retention performance that markedly surpass that of the 12 time larger Transformer language model. The result is a long-tail language model pretraining objective, that due to its data-efficiency, can effectively pretrain form small text corpora, while also observing computational and memory efficiency that are desirable in practical application.

I thus find that, model compression may not be the go-to method for obtaining good long-tail performance from compact models. More beneficially, **this also lead to the contribution of a novel contrastive language model pretraining method, that can be used in domains where pretraining and fine-tuning data or computational resources are limited**. Since, by definition, long-tail information is always limited, the method is also data-efficient regarding tail (minority) information.

1.2.3.2 Paper 6: MoRTy: Unsupervised Learning of Task-specialized Word Embeddings by Autoencoding

Publicly available pretrained word embeddings do not always work well for a specific task (Bollegala and Bao, 2018; Kiela et al., 2018a), and pretraining custom word embeddings does not always work well either, particularly in data limited setups (Dingwall and Potts, 2018). It would thus be desirable to have a word embedding learning technique that works well in low data and compute scenarios. In this work,

I contribute a simple yet effective post-processing method that retrofits word embeddings in a zero-shot fashion without target data using self-supervision to construct a set of task-specialized word embeddings. The set of specialized word embeddings can then be used as a hyperparameter during supervised evaluation to pick the most suitable word embeddings. In an experiment with 18 task ranging from classification, semantic similarity to word analogy tasks, I demonstrate that there is **always a learned retrofitting that results in better end-task performance**.

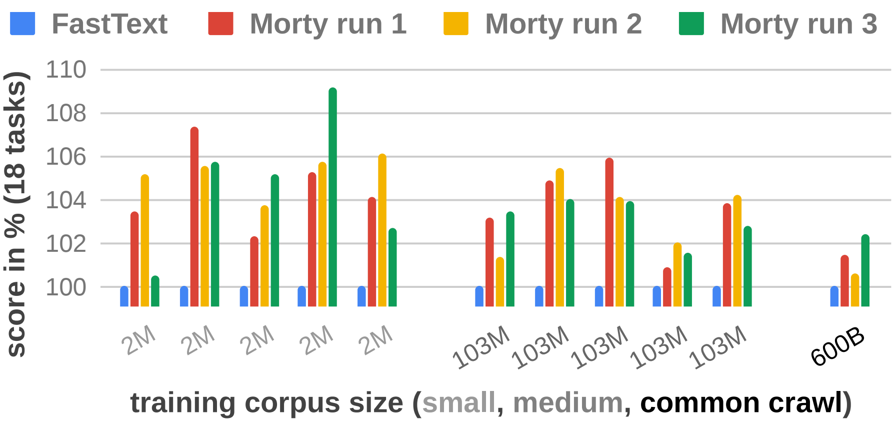


Figure 1.6: 1-epoch MoRTy (MT %) performance change over Fasttext: Blue bars show Fasttext baseline performance (100%). 3 Morty runs: trained on Fasttext for **1 epoch** (2x5 Fasttext for corpus sizes 2M and 103M and 1x for 600B). Detailed description on next page.

This post-processing technique **works especially for word embeddings pretrained from limited text data**, which is especially useful in domains where pretraining text is not abundant. In a second experiment I demonstrate that, it is also possible to produce as single retrofit embedding that is on average better over all 18 tasks, i.e. for use in a multi-task learning scenario. More importantly, I demonstrate that over three random retrofits, the resulting retrofit embedding always outperforms the original FastText word embeddings on average over 18 tasks as seen in Figure 1.6, and that this consistent improvement is especially strong for low-data embeddings.

1.3 Summary of Contributions and Future Work

The sum of publications created during this thesis advance transfer and adaptation understanding, training and efficiency of representations in NLP. While studying connections between these aspects, a particular focus is put on the relation between self-supervised and supervised training and how they interact with representation transfer and adaptation. Hence, Table 1.2 maps the contributions in understanding,

	Adaptation		Contrastive		Data-Efficient	
	Understanding		Adaptation		Adaptation	
	SL	SSL	SL	SSL	SL	SSL
Rethmeier et al. (2020a)	x	x				
Rethmeier et al. (2020c)	x	x			x	
Rethmeier and Augenstein (2022c)			x	x		
Ostendorff et al. (2022a)			x			
Rethmeier and Augenstein (2022a)			x	x	x	x
Rethmeier and Plank (2019)					x	x

Table 1.2: Summary of contributions made by each publication, organized by the three core topics: Adaptation and Transfer Understanding/ XAI, Contrastive NLP Pretraining, and Data-efficient Adaptation. Each method is further split into whether it applies to supervised learning SL, or self-supervised learning SSL.

pretraining and towards the efficiency of NLP representation transfer and adaptation to both self-supervised learning (SSL) and supervised learning (SL).

Regarding self-supervised learning aspects the proposed methods expand existing research in model understanding, transfer and efficiency gains within the framework of language model pretraining, since pretraining makes it possible to deeply study how representations evolve, transfer and adapt during training of existing and novel pretraining methods. Especially through contrastive (energy-based) pretraining methodology, the thesis contributes to the transfer, adaptation, data and compute efficiency of language models, which is generally beneficial to low resource scenarios, e.g. when data is sparse or long-tail distributed, which is sparse by definition. Additionally, compute and data-efficiency are evaluated on citation prediction, in-hospital patient outcome prediction and over various NLP tasks including classification, similarity and analogy tasks. Especially the work on patient outcome prediction verifies that NLP representation understanding and training methodology work on mixed text and scalar data, but can also be used to improve model design and trust in medical applications.

1.3.1 Decision, Model, Transfer and Adaptation Understanding

This thesis makes core contributions to model understanding of transfer, adaptation, and pruning (Paper §2), as well as to detecting model component and data source relevance (Paper §3).

Regarding model transfer understanding, Paper §2 is the first to study model understanding at all training stages from an untrained model, to a pretrained one, its zero-shot application and supervised fine-tuning. While many works have investigated pruning, this thesis is the first to link model understanding (and explainable AI) to pruning by tracking neuron feature preference changes across training stages. Another insight is that supervised adaptation of pretrained model knowledge causes representational sparsification that leads to knowledge forgetting, as the flip side of pruning, as well as to overfitting (overspecialization) of neurons caused by the supervised objective. Advantageously, this technique enables fine-grained pruning by making it possible to categorize neuron knowledge changes regarding: (I) transferred (reused without change), (II) adapted (reused with change), overspecialized (III), (IV) newly introduced and (V) actively forgetting (turned off) neurons. This makes it possible to separate end task-essential neurons and their (reused, adapted, added) knowledge (I, II, IV) from task-irrelevant (overspecialized, forgotten knowledge) neurons (III, V) to prune according to application requirements. While language model fine-tuning is often analyzed using explainability or model understanding techniques, the work in this thesis is also the first to investigate the complete pipeline of how self-supervised pretraining, zero-shot application and fine-tuning interplay at a representation level in an autoregressive (causal) language model. A direct insight of this analysis in this thesis is that a language model learns to capture basic distributional features such as parts-of-speech during early pretraining. Another profound, yet often neglected, insight is that when shifting from an original to a new domain during direct (zero-shot) transfer, it is *not only* the similarity of input feature distribution that determines transfer quality, but also the similarity of model activations (behavior) over both domain datasets. That is, transfer is maximized, if the model behaves similarly over both domain inputs and if additionally most of the two domain information (knowledge) was absorbed by the model during training on the first domain. Thus, transfer quality is largely dependent on the pretraining objective and its ability to build as much reusable representations from as little data as possible. As a result, supervised transfer can not be understood in detail without zero-shot transfer analysis, because the above argument applies even when testing on data of the same domain or dataset – i.e. this important insight was later reinforced in zero-shot probing works by Talmor et al. (2020); Elazar et al. (2021).

Additionally, in the area of patient outcome prediction (Paper §3), this thesis contributes to model component and data source relevance analysis to produce verification of model and data-efficiency as well as designing more efficient architectures. In health care models this helps to make the models small, efficient and uncover their inner workings. For example, I find that models automatically cluster medical field knowledge about nephrology or cardiology in specific neurons while minimizing redundancies between these neurons. I also find that training builds a single, potentially biased feature neuron that encodes ethnicity and gender, which may present an important factor of analysis for trustworthy in-hospital AI solutions. Specifically, compared to previous state-of-the-art Transformer based models, this design via model understanding made it possible to very significantly improve upon state-of-the-art results, while also reducing model size by an order of magnitude. This connection between model understanding and efficiency has profound benefits for in-hospital applications regarding cost, interpretability and maintenance in predictive health.

The connection between the lottery ticket hypothesis (Frankle and Carbin, 2019c) – i.e. searching for pruned networks that better generalize to a task than the larger model – and model understanding guided pruning is so far not understood. Further research into this direction could make it possible to fully control dynamic, task-wise, pruning, by finding tickets in larger networks using model understanding, and allowing the pruned neurons to be used for learning other tasks. Such task-wise pruning and transfer understanding would enable informed control of catastrophic knowledge forgetting, knowledge adaptation and its extension during exposure to new tasks and information over time. Since fine-grained neuron change understanding can distinguish task-irrelevant from task-relevant neurons it could be used to turn off forgetting by freezing neurons, or up-regulate the learning rate of specific neurons per task.

1.3.2 Contrastive Pretraining based Transfer and Adaptation

Contributions in the area of contrastive pretraining in NLP include contrastive pretraining of a language model from scratch (Paper §6), a paper §5 that contributes to contrastive transfer for scientific document and citation representation, and a primer on contrastive pretraining and applications that is aimed at helping NLP researchers better understand, use and develop contrastive NLP models (Paper §4).

Unlike contrastive pretraining in computer vision, contrastive pretraining in NLP had previously been limited to contrastive pretraining of already pretrained Transformer language models due to scaling issues with contrastive approaches and Transformer models. It was thus unclear whether language models can be pretrained contrastively from scratch to similar effectiveness as vision models. In Paper §6 this thesis contributes a novel contrastive autoencoder architecture and accompanying pretraining method that enable effective pretraining of a language model, as well as zero-shot learning by modeling pretraining and fine-tuning as a text embedding compatibility prediction task. It extends the traditional noise contrastive estimation methodology to use multiple positives, introduces a learned similarity function, scalable, compute optimized sample encoding and collision prevention for positives and negatives samples to enable data and compute efficient pretraining (see §6).

Regarding scientific document representation learning this thesis proposes a state-of-the-art knowledge fusion method for scientific document classification and citation prediction tasks – see Paper §5. It fuses knowledge of citation graph embeddings to scientific text embeddings by sampling citation embeddings per document for contrastive training. Previous work relied on discrete citation information, which relies on correct citation and does not allow citation relatedness beyond recorded citation structures – which is a core goal of document similarity. Crucially, the method introduces a data-driven sampling margin that avoids collisions between positives and hard negatives which can be tuned using only a fraction of the data while also enabling highly few-shot effective training.

Future research can apply the contrastive autoencoder pretraining method to larger pretraining data, to better understand if large language models are necessary or if data is more important. Another aspect is temporal adaptation, where first unpublished experiments indicate that a contrastive language model representation degrades significantly less over new information compared to autoregressive (causal) or masked language models. Another direction that is worth exploring is how contrastive training objective converts model representations to an energy based representations and whether these representations exhibit the same behavior during transfer and adaptation. A final area of further research is how a contrastive language model performs in neural search and recommendation tasks, especially regarding its small parameter size and compute efficient query encoding.

1.3.3 Data-efficient Representation Transfer and Adaptation

In regards to data-efficient transfer and adaptation, this thesis contributes three methods. One method to improve zero-shot (label-free) adaptation (Paper §7). A second method that enables data-efficient (self-supervised) pretraining, along with efficient zero-shot, few-shot and long-tail transfer (Paper §6). And finally a method for few-shot knowledge fusion (Paper §5).

The first method (Paper §7) retrofits word embeddings using an autoencoding approach. Existing works in this domain require adaptation (retrofitting) of word embeddings using end-task fine-tuning data, while the proposed method produces embedding variations in a fully self-supervised manner. This makes it possible to create embedding variations that lead to improved performance of a single task, where variations are treated as a hyperparameter. Additionally, the method is able to produce a single, overall better, embedding in the sense that the multi-task learning performance improves over a set of 18 tasks, which range from text classification, semantic similarity to word analogy. Finally, the method is not only extremely compute efficient, only requiring a CPU, but also generates word-embeddings that are more pretraining data efficient than standard approaches, i.e. pretrain better word embedding from small data collections.

The second work proposes a method and model for contrastive language model pretraining (Paper §6) which contributes self-supervised training that, unlike existing language models used for pretraining, can train effectively from even 10% of an already very small pretraining text data set over a much larger Transformer model. Additionally, the method allows zero-shot prediction without requiring a preceding labeled task, which also enables it to achieve markedly better few-shot, long-tail and few-shot long-tail fine-tuning performance over a standard Transformer model. This is an important property, since in the long-tail data is always limited by definition, and with larger data the complexity of the distribution's long-tail data grows. This means that, scaling up data aggravates the problem of long-tail (minority information) ignorance rather than solve it. Finally the combined pretraining and and fine-tuning time is overall five times faster than only fine-tuning a standard masked language model that has an order of magnitude more parameters.

A third work (Paper §5) contributes a state-of-art approach to knowledge distillation for scientific document representation, as well as to the few-shot learning capabilities of such models. Instead of using discrete citation graph similarity, this model computes textual similarity as continuous, data-driven citation embedding neighborhood similarity for contrastive pretraining to learn a combined representation for document relatedness. Previous methods modeled citation information unidirectionally, did not prevent positive and negative samples from colliding, and had no mechanism for hard negative sampling, which is needed to make learning efficient. To solve these issues, the proposed model introduces a data-driven sampling margin that separates for positives and hard-negatives to prevent collisions, model citation relations bidirectionally, and increase the models ability to train effectively from small data and thus adapt better to updated information. Specifically, a direct result of this controlled sampling is that the model can learn effectively from 1% to 10% of training data and works well when combined with other methods for efficient fine-tuning such as BitFit (Ben Zaken et al., 2022a), to further boost efficiency.

Future research of the mentioned contrastive pretraining methods could extend them to learning over time since the models learn better from fewer examples. Another field of improvement is to use the knowledge distillation framework to enrich language models with contextual information from other graph embeddings such as knowledge bases as retrieval augmentation and search tasks. A last future improvement could be to apply the ideas of embedding retrofitting to low resource modeling. Since the retrofitting method was already beneficially applied to the medical prediction model produced during this thesis, it is reasonable to assume that it can be applied to other low resource, low compute tasks to stabilize training and increase performance as it did in the medical domain. Together, these efficiency benefits would further decrease deployment and maintance costs of future domain applications, while still delivering better performance results, as the works produced during this thesis already demonstrated for various domains and tasks.

Part II

Explainable Representation Transfer and Adaptation

2 TX-Ray: Quantifying and Explaining Model-Knowledge Transfer in (Un-)Supervised NLP

2.1 Introduction

Continual and Transfer Learning have gained importance across fields like NLP, where the de facto standard approach is to pretrain a sequence encoder and fine-tune it to a set of supervised end-tasks (Peters et al., 2019). Analysis and understanding of transfer in NLP are currently focused on using either supervised probing tasks (Belinkov and Glass, 2019) to compare task performance metrics (Wang et al., 2019) or laborious per-instance explainability (Belinkov and Glass, 2019). *Supervised* probing annotation is costly, but not guaranteed to be reliable under domain shifts. Probing is also limited to analyzing foreseen (probed) *knowledge absorption* aspects, while unforeseen, model-knowledge properties that underlie and thus further our understanding of self-supervised pretraining remain hidden (McCoy et al., 2019b). In fact, 'decision understanding' explainability techniques, as (Gehrmann et al., 2019) term them, compute the relevance of a feature or neuron for an end-task prediction score. This makes 'decision understanding' explainability unable to answer the following research questions (**RQ1-3**) – i.e. how can we explain transfer?

(RQ1), unsupervised knowledge absorption: Can explainabilty (XAI) analyze how self-supervised models build and change knowledge abstractions during pretraining and can XAI measure knowledge changes? Do measures coincide with conventional metrics like perplexity? If and when does self-supervision learn linguistic abstractions like word function (parts-of-speech)?

(RQ2), zero-shot knowledge transfer: What knowledge subset do pretrained models apply to a new domain without re-training, e.g. in a zero-shot setting?

(RQ3), supervised/ backwards transfer: Can knowledge transfer 'backwards' from supervision labels into a pretrained model? Does XAI identify which neurons are reconfigured – i.e. become task (ir)relevant due to supervision. Can we validate XAI-based transfer measures (RQ1) empirically by pruning (ir)relevant neurons?

TX-Ray can analyze and *quantify* (self-)supervised model knowledge change: To answer RQ1-3 we propose TX-Ray. TX-Ray – i.e., Transfer eXplainability as pReference of Activations analYsis – modifies the well established activation maximization method of visualizing the preferred inputs of neurons (Erhan et al., 2009b) to suit NLP. The resulting fine-grained 'model understanding' – as (Gehrmann et al., 2019) term it – enables us to *quantify knowledge changes or transfer* during training at the level of individual neurons – without requiring or preemptively limiting analysis to probing task supervision semantics. The method is designed to explore model knowledge change at both neuron (detail) and model (overview) level to enable concise or deep explorative analysis of unforeseen knowledge transfer mechanics to help us better analyze (continual) transfer, model knowledge generalization (McCoy et al., 2019b; Frankle and Carbin, 2019c), or low-resource learning. (Adebayo et al., 2018; Sixt et al., 2019) showed that XAI methods do not guarantee faithful explanations. We thus use TX-Ray's transfer measures to guide neuron pruning and empirically verify that it can identify task (ir)relevant neurons that boost or lower test set generalization as expected. We also demonstrate that supervision not only causes catastrophic forgetting of knowledge, but also adds new knowledge into previously un-preferred (under-used) neurons in Table (2.2).

2.2 Approach

TX-Ray is inspired by the widely used activation maximization explainability method, which is based on the idea that "a pattern to which a unit is responding maximally is a good first-order abstraction of what a unit (neuron) is doing. A simple way is to find the input samples that produce the highest activation for a neuron. Unfortunately, this opens the problem of how to 'combine' these samples." (Erhan et al., 2009b). In computer vision, naively combining image maximum feature activation maps "over a corpus does not produce interpretable results" (Erhan et al., 2009b). In NLP, however, maximal activations of discrete token feature can easily be combined over many samples to form a discrete distribution of 'tokens that a neuron prefers'. These corpus-wide input feature preference distributions let us visualize how each neuron abstracts input knowledge subsets.

A major advantage of a 'feature preference' method is that it can analyze *non-supervised models over an entire corpus*, while 'prediction score relevance explainability' methods require *supervised models*, and *only explain individual instances* (Belinkov and Glass, 2019). When representing a neuron's abstracted knowledge as a feature preference distributions, we can measure knowledge change, or transfer, during learning using standard measures such as Hellinger Distance – i.e., a symmetric version of the Kullback Leibler divergence. This allows one to track changes in neuron knowledge abstractions during model pretraining, model application to new domains or due to supervised fine tuning – see experimental section. Additionally, we automatically determine neurons that change their knowledge the most over time to provide interesting starting points (see Figure 2.5, 2.7) for nuanced, per-neuron analysis (see Figure 2.6 and 2.8).

2.2.1 Neurons as Feature Preference

We thus expresses each neuron n_n as a distribution over preferred features f_k with activation probabilities p_k (Figure 3.2) that have been aggregated over an entire corpus to construct each n_n distribution as follows.

(1) Record what features neurons prefer: Given: a corpus D, text sequences $\mathbf{s_i} \in D$, input features (tokens) $f_k \in s_i$, a sequence encoder E, and hidden layer neurons $n_n \in E$, for each input token feature f_k in the corpus sequences s_i, we calculate its: encoder neuron activations $\mathbf{a} = E(f_k)$; along with $\mathbf{a}$'s maximally active neuron $\mathbf{n_{argmax}} = argmax(\mathbf{a})$ and (maximum) activation value $a_{max} = max(\mathbf{a})$; to then record a *single feature's activation* row vector $[f_k, n_{argmax}, a_{max}]$. If the encoder is part of a classifier model C, we also record the sequence's class probability $\hat{y} = C(s_i)$ and true class y as a longer vector $[f_k, n_{argmax}, a_{max}, \hat{y}, y]$. For analyses in RQ1-3, we also record part-of-speech tags (POS, see section 2.3.1) in the row vectors. This produces a matrix M of neuron feature max activations that we aggregate to express each neuron as a probability distribution over maximally activated features in Step (2).

(2) Preferred feature distribution per neuron: From rows $m_r \in M$, we generate for each neuron n_n its discrete feature activation distribution $A_{n_n} = \{(f_k, \mu(a_{max_1}, \ldots, a_{max_m})) \,|\, f_k, n_n, a_{max_j} \in m_r \wedge m_r \in M \wedge n_{argmax} = n_n\}$, where each f_k is a feature the neuron maximally activated on, and $\mu(a_{max_1}, \ldots, a_{max_m}) = \mu_{f_k}$ is the mean (maximum-)activation of that feature in n_n. We then turn each activation distribution A_{n_n} into a probability distribution P_{n_n} by calculating the sum of its feature activation means $s_{\bar{\mu}} = sum(\mu_{f_1}, \ldots, \mu_{f_l})$ and dividing each μ_{f_k} by $s_{\bar{\mu}}$ to produce the normalized distribution $P_{n_n} = \{(f_1, \mu_{f_1}/s_{\bar{\mu}}), \ldots, (f_l, \mu_{f_l}/s_{\bar{\mu}})\} = \{(f_1, p_1), \ldots, (f_l, p_l)\}\}$, where, each p_{f_k} is now the activation probability of a feature $f_k \in n_n$. Finally, for n neurons in a model, P describes their n per-neuron activation distributions $P = \{P_{n_1}, \ldots, P_{n_{n=|E|}}\}$.

Features can be n-grams, and be tracked through multiple layers as in (Carter et al., 2019b). However, since in this work we focus on concisely presenting TX-Ray's transfer analysis, we only use uni-grams and a single layer.

2.2.2 Neuron Knowledge Change

We use **Hellinger distance** H (Hellinger, 1909) and neuron distribution length l to quantify differences between discrete feature preference probability distributions $p = P_{n_a}$ and $q = P_{n_b}$ of two neurons n_a and n_b as follows:

$$H(p,q) = \frac{1}{\sqrt{2}}\sqrt{\sum_{f_k=1}^{l}(\sqrt{p_{f_k}} - \sqrt{q_{f_k}})^2}; \text{knowl. change}$$

$$l(P_{n_n}) = |\{f_k \mid f_k \in P_{n_n}\}|; \text{knowledge 'diversity'}$$

Neuron length l describes the number of (unique) maximally activated features in a feature preference distribution P_{n_n}. We use Hellinger distance because it is symmetric, unlike the Kullback-Leibler divergence. Importantly, if one of the preference distributions P_{n_a} or P_{n_b} is empty, i.e. has zero features (zero length), then the resulting Hellinger distance is ill-defined. Thus, Hellinger distance allows one to easily quantify neuron feature preference shifts to measure per-neuron knowledge change during pre-training (RQ1), zero-shot transfer (RQ2), and supervised fine-tuning (RQ3).

Neuron length l on the other hand allows us to define binary states like **'un-preferred'** for empty preference distributions ($l = 0$) and non-empty ones **'preferred'** ($l > 0$). We can use the two terms to classify *three kinds of neuron preference state changes caused by different model training stages*: **'shared'**, **'avoided'**, **'gained'**. For 'shared' neurons both distributions are non-empty (preferred) – e.g. when neurons received maximum activations before and after retraining a model. 'Avoided' neurons were active 'preferred', but became less active 'un-preferred' after retraining. Finally 'gained' neurons, became more active after retraining, switching from 'un-preferred' to 'preferred' status. In RQ1-3 we will use changes in Hellinger Distance, distribution length and neuron states to identify which neurons overfit to few preferred features, which ones reuse features (transfer) and which one never specialize (unfit).

2.3 Experiments and Results

We showcase TX-Ray's usefulness for analyzing and quantifying transfer in answering the previously stated research questions. For RQ1, we pretrain an LSTM sequence encoder E[1] with 1500 hidden units on WikiText-2 similarly to (Merity et al., 2017b; Howard and Ruder, 2018), and apply (RQ2) or fine-tune it (RQ3) on IMDB (Maas

[1] Though possible, we do not pretrain Transformers, due to high computation requirements, and since LSTMs encoders perform vastly better when pretraining on small collections – compare (Wang et al., 2020b) with (Merity et al., 2017b). Instead, we focus on demonstrating TX-Ray's analytical versatility, especially for true low-resource scenarios, where large pre-training is unavailable.

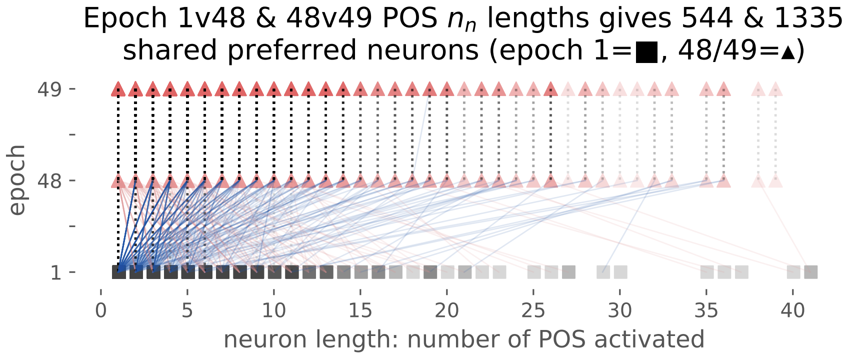


Figure 2.1: **Pretraining neuron length shifts:** where neuron length l (token variety) becomes; longer (blue /), shorter (red \), unchanged (black :) for epoch 1, 48, 49. Token variety settles (:) in later epochs.

et al., 2011), so we can analyze its zero-shot and supervised transfer properties. Each RQ's experimental setup and results are detailed below.

2.3.1 RQ1: Pretrained what Knowledge

In this experiment, we explore how pretraining builds knowledge abstractions. We first analyze neuron abstraction shift between early and later training epochs, and then verify that Hellinger distance and neuron length changes converge similar to measures like training loss.

We pretrain a single layer LSTM encoder E on paragraphs from the WikiText-2 corpus D_{wiki2} using a standard language modeling setup until loss an perplexity converge, resulting in 50 training epochs. We save model states at Epoch 1, 48 and 49 for later analysis. To produce neuron activation distributions P_{wiki_1} (gray), $P_{wiki_{48}}$ (pink) and $P_{wiki_{49}}$ (red) we feed the first 400.000 tokens of WikiText-2 into the Epoch 1, 48 and 49 model snapshots each to compare their neuron adaptation and incremental abstraction building using Hellinger distance and distribution length. Additionally, we record POS feature activation distributions using one POS tag per token, to later group tokens activations by their word function to better read, analyze and compare feature preference distributions – see Figure 2.2, 2.4, 2.6 or 2.8. POS tags are produced by the state-of-the-art Flair tagger (Akbik et al., 2019) using the Penn Treebank II[2] tag set.

We use this experiment to verify the feasibility of using a feature preference distribution approach, since comparing Epochs 1 vs. 48 should reveal *large changes* to neuron abstractions, while Epoch 48 and 49 should cause *few changes*. The resulting changes in terms of Hellinger distance, amount of 'shared' preferred neurons, and feature preference distribution lengths can be seen in Figure 2.1.

[2] https://www.clips.uantwerpen.be/pages/mbsp-tags

While the Epoch 1 vs. 48 comparison produced 544 'shared' neurons, the later 48 vs. 49 comparison shows 1335 'shared' (section 2.2.2) neurons. This means that pretraining the encoder distributes maximum input activations across increasingly many neurons. This can be seen in most neurons becoming longer (blue ■/▲ lines), and fewer neurons becoming shorter (red ▲\■ lines). As expected, for epochs 48 and 49 we see almost unchanged neuron length – seen as dotted vertical (:) lines between epochs. Additionally, in later training stages, shorter neurons are more frequent than longer ones, reflected in the opacity of dotted vertical bars decreasing with neuron length. In fact, the average length of 'shared' preferred neurons drops from 944.76 in epoch 1 to 524.55 and 519.34 in epochs 48 and 49.

Since lengths of POS class preference distributions change significantly in the early epochs, we also analyze whether the encoders activations P_{wiki_1}, $P_{wiki_{49}}$ actually learned to represent the original POS tag frequency distribution of WikiText-2. Thus, we express both corpus POS tag frequencies and encoder activation masses as proportional (relative) frequencies per token. In Figure 2.2, we see relative corpus POS tag frequencies (black), compared with encoder POS activation percentages for epoch 1 (dark grey) and 49 (red). Evidently, the encoder learns a good approximation of the original distribution (black) even after just the first epoch (dark grey), which confirms findings by (Saphra and Lopez, 2018), who showed that: "language model pretraining learns POS first", and that "during later epochs (49) the encoder POS representation changes little". Ultimately, the encoder near perfectly replicates the original POS distribution. We thus see that POS are well represented by the encoder, and that neuron adaptation and length shifts converge in later epochs in accordance with the quality of the POS match. This also tells us that TX-Ray, similar to more involved optimization-based analysis methods (Saphra and Lopez, 2018; Raghu et al., 2017), can reveal comparably deep insights into the mechanisms of unsupervised training, while being simpler and more versatile (RQ1-3).

Using Figure 2.3, a similar analysis about *neuron feature distribution changes stabilizing at later training stages* can be made using Hellinger distances. When visualizing distances, we see that they shrink as expected by 99.92% on average in later epochs and that neuron distance comparisons concentrate on medium length distributions of 10-200 features f_k each. Preference distribution changes of short, specialized, neuron seem to produce higher Hellinger distances than longer, more general neurons. Since distances over different neuron lengths are not and should not be directly compared, this visualization acts to provide an *explorable overview* of neuron distances over different preference distribution lengths, used to identify and examine interesting neurons in *detail*.

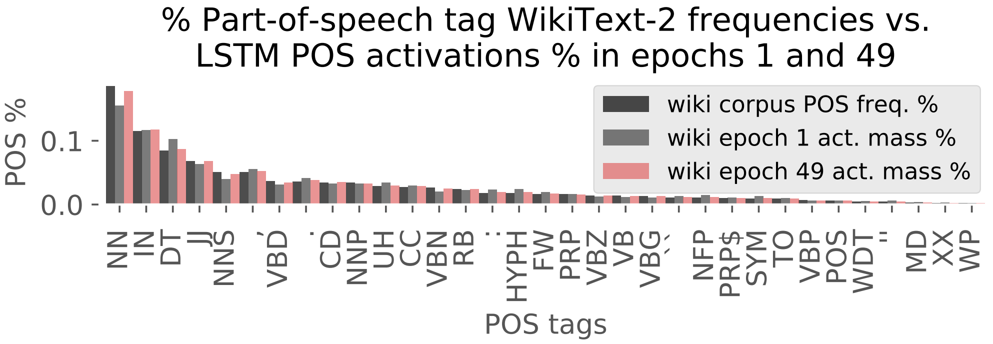


Figure 2.2: **Encoder learns POS early on:** Black: Corpus tag frequencies (y-axis) vs. encoder activations in %-per-tag. Black: corpus frequencies via FLAIR. Grey: epoch 1 encoder. Red: fully trained encoder. POS is learned early, i.e. in epoch 1, confirming findings in (Saphra and Lopez, 2018).

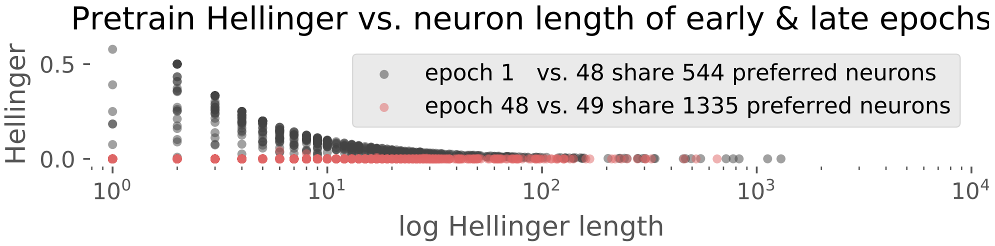


Figure 2.3: **Encoder gains knowledge preference (neurons) through pretraining:** Later epochs activate more neurons maximally (544 to 1335), while Hellinger distance (knowledge change) reduces in later epochs (48 vs. 49, red line) vs. earlier epochs (1:48, black dots).

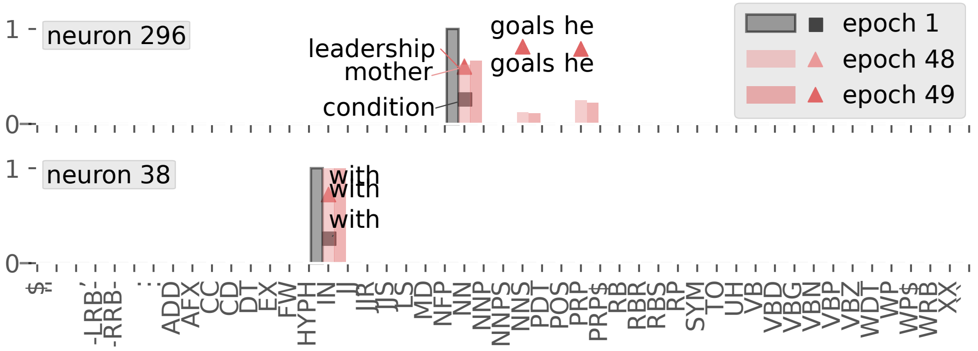


Figure 2.4: **High vs. low Hellinger H neurons:** Neuron token (■▲▲) and POS activation probabilities (bars) for epochs 1, 48, 49. Neurons with high H (296) and low H (38) between epochs 1:48.

To run such a detail analysis we pick 2 neurons from Figure 2.3 for closer inspection of their feature preference distribution changes between Epochs 1, 48 and 49. Figure 2.4 thus shows neuron **296** from the top 10 (head) most distant Epoch 1 vs. 48 neurons, and Neuron **38** from the 10 least changed ones (tail). As expected from Neuron **296**'s high Hellinger distances between Epoch 1 and 48, we see that its token and POS distribution for Epoch 1, i.e., an outlined grey bar and the word 'condition' (■), are very different from the Epoch 48 and 49 distributions (▲, ▲), which show no significant change in token and POS distribution – i.e., they look nearly the same. Equally expected from Neuron **38**'s low Hellinger distance for Epoch 1 and 48; we see that it keeps the exact same token, 'with', and POS, 'IN', across all three epochs. This demonstrates that Hellinger distance identifies neuron change, and that later epochs, as expected, lead to small neuron abstraction changes, while earlier ones, also as expected, experience larger changes.

2.3.2 RQ2: Do we Zero-Shot Transfer?

In this section, we analyze where and to what extent knowledge is zero-shot transferred when applying a pretrained encoder to text of a new domain – without re-training the encoder to fit that new data.

To do so, we apply the trained encoder E, in prediction-only mode, to both its original corpus IMDB, D_{imdb}, and to the new domain WikiText-2 corpus D_{wiki2}, to generate feature preference distributions P_{imdb} and P_{wiki2} from the encoders' hidden layer, as before. We also record activation distributions for POS, which despite the FLAIR tagger being SOTA across several datasets and tasks, had noticeably low quality on the noisy IMDB corpus. However on the WikiText-2 corpus, tagging produced comparatively sensible results. By comparing neuron token and tag activations P_{imdb} (new domain) vs. P_{wiki2} using Hellinger distances for the same neuron positions as in RQ1, we can now analyze zero-shot transfer as distribution shifts. Put differently, we estimate domain transfer between the pretrained model abstractions and text input from a new domain. High distances between the same neurons in P_{imdb} and P_{wiki2} tell us that the pretrained neuron did not abstract the new domain texts well, resulting in low transfer and poor cross-domain generalization. When comparing P_{imdb} and P_{wiki2} in terms of Hellinger distances vs. neuron lengths in Figure 2.5, we see that 1323 out of 1500 pretrained neurons (88.2%) remain 'preferred' ('shared') when applying E to the IMDB domain. A drop in the amount of 'preferred' neurons compared to the RQ1 analysis, though at 1335 to 1323 small, is expected since the pretraining corpus covers a broader set of domains.

However, to gain a *detailed* view of model abstraction behavior and zero-shot transfer, we analyze activation differences between P_{imdb} (green) and P_{wiki2} (red)

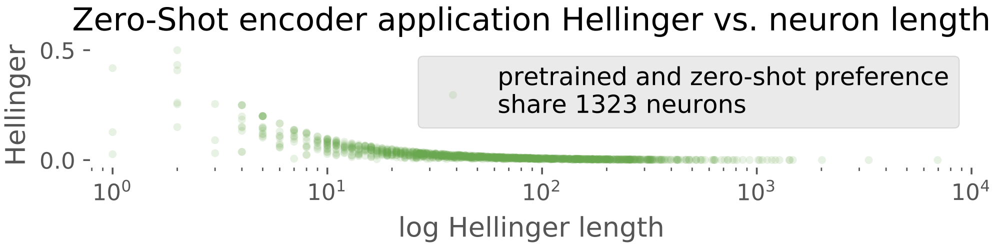


Figure 2.5: Neuron feature preference difference when applying to unseen text: Hellinger distances between neuron **1323 'shared'** preference distributions P_{wiki2} and P_{imdb} on WikiText-2 and IMDB.

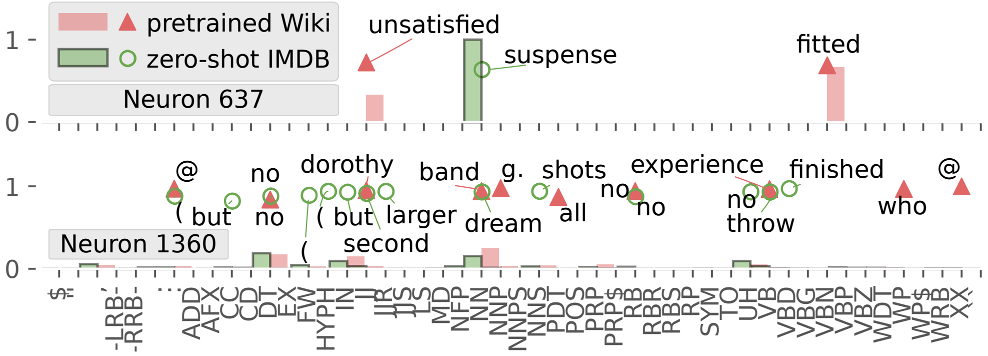


Figure 2.6: Low vs. high zero-shot transfer neurons: Neuron **637** transferred little, while the 'but-no' neuron **1360** transferred (applied) well from pretraining to the new IMDB domain.

for two specific neurons, visualizing one each from the 10 most (head) and 10 least (longtail) Hellinger-distant neurons. In Figure 2.6 (up), we see Neuron **637**, which has high Hellinger distance when comparing token feature distributions (▲, ○). As expected, the neuron's feature preference between the pretraining corpus P_{wiki2} and the new domain data P_{imdb} changes a lot. In fact, the distance in Neuron 637 is high in terms of both POS classes (word function semantics) and non-synonymous tokens – see x-axis annotated with POS tags and tokens sorted by POS class. Overall, we see very *little knowledge transfer* across data sets within Neuron 637 due to its *feature over-specialization*, which is also observable in its short distribution length l – only 2 features activate. When looking at the low Hellinger distance Neuron **1360** in Figure 2.6 (lower plot), we see that the neuron focuses on tokens such as 'no' on both datasets and 'but' on IMDB, suggesting that its pretrained sensitivity to disagreement (red), is useful when processing sentiment in the new domain dataset. Furthermore, we see that IMDB specific tokens have many strong activations for movie terms like 'dorothy' or 'shots' (green). We thus conclude that Neuron 1360 is both able to apply (zero-shot transfer) its knowledge to the new domain, as expected from the low Hellinger distance, while also being adaptive to the new domain inputs, despite not being fine-tuned to do so, which is more surprising. In summary, we find that during zero-shot application of an encoder to new domain data, the pretrained encoder exhibits broad transfer, indicated by almost equal amounts of 'shared' neurons between pretraining (1335) and application to the new domain data (1323). A supervision fit encoder however, has its knowledge reconfigured to superivsion, leading to much reduced transfer of pretrained knowledge, as we will see in RQ3.

2.3.3 RQ3: How does Supervision Back-Transfer Label Knowledge?

In this experiment, we analyze whether transfer constitutes more phenomena than just a high level observation like catastrophic *forgetting*. Here, we want to see if knowledge also transfers 'backwards' from supervised annotations to a pretrained encoder. Specifically, we analyze whether knowledge is added or discarded in two experiments. In Experiment 1, we demonstrate how TX-Ray can identify knowledge addition or loss induced by supervision at individual neuron level (section 2.3.3.1). In Experiment 2, we verify our understanding of neuron specialization and generalization by first pruning neurons that add or lose knowledge during supervision, and then measuring end-task performance changes (section 2.3.3.2). Finally, we show how neuron activity increasingly sparsifies over RQ1-3 to gain overall insights about model-neuron specialization and generalization during unsupervised and supervised transfer (section 2.3.3.3).

For this RQ, we extend the pretrained encoder E with a shallow, binary classifier[3] to classify IMDB reviews as positive or negative while *fine-tuning* E to create a domain-adapted encoder $E_{imdb-sup}$. To guarantee a controlled experiment, we freeze the embedding layer weights and do not use a language modeling objective, such that model re-fitting is exclusively based on supervised feedback – i.e., on knowledge encoded into the labels. We tune the model to produce roughly 80% F_1 on the IMDB test set, to be able to analyze the effects of *even moderate amounts of supervised fine-tuning before task (over-)fitting* occurs. To produce feature preference distributions $P_{imdb-sup}$, we feed the IMDB corpus D_{IMDB} to the newly fine-tuned encoder $E_{imdb-sup}$ – i.e. using the same IMDB text input. We also once more record POS tags for tokens. This time, since POS distributions are compared on the same corpus, their distances are more consistent than in RQ2. Analyzing Hellinger distance and neuron length change when comparing $P_{imdb-sup}$ vs. $P_{imdb-zero-shot}$ will tell us which neuron abstractions were changed the most *due to supervision* – i.e., show us 'backward knowledge transfer'. In Figure 2.7, we notice that only 675 neurons were 'shared' compared to 1323 neurons

[3] One fully connected layer with sigmoid activation that is fed by $E's$ end-of-sequence hidden state.

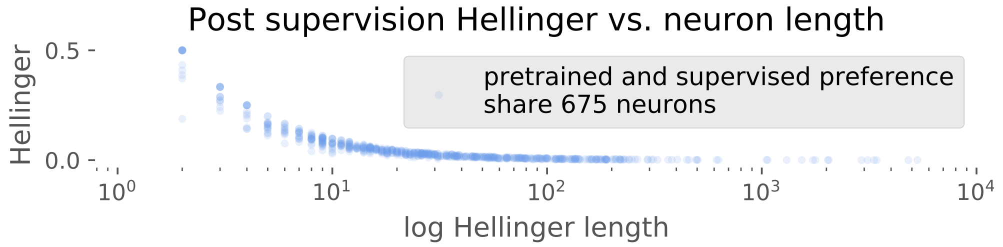


Figure 2.7: **Neuron feature preference change after supervision:** Hellinger distances of 675 'shared' neuron preferences before and after supervised encoder fine-tuning – dropped from 1323.

Table 2.1: **Preferred features of 6/ 85 noisy supervision neurons gained by supervised fine-tuning:** 3 highly active ones (top 3), 3 seldomly active ones (bottom 3).

#neuron : activation sum, features, (#features total)
200 : 1307.42 great, james, superb, famous, strange, possible, french, english, grand, final, indian, solid . . . (141)
1210 : 501.97 original, overall, good, real, some, dear, french, british, black, odd, italian, entire, many . . . (161)
125 : 299.12 more, two, best, one, few, most, three, nice, four, fellow, films, somewhat, lot, favorite, rare . . . (77)
1289 : 7.92: terrific, dull, essential, celia, unbelievable, gentle, melancholy, intended, shaggy . . . (14)
372 : 4.18: walter
688 : 0.48: archer

in the zero-shot transfer setting (Figure 2.5). In other words, *supervision re-fits the sequence-encoder to 'avoid' (unprefer) nearly half its neurons*.

2.3.3.1 Supervision adds and removes knowledge

Somewhat surprisingly, supervision not only erased neurons, but also added distributions for 85 new neurons into $P_{imdb-sup}$ that had previously empty distributions in $P_{imdb-zero-shot}$. We analyzed these neurons and found that they represent new supervision task specific feature f_k detectors. Below in Table 2.1, we show token features f_k for the top three strongest firing neurons n_n and the three least activating neurons out of the 85 – i.e. supervision-specific neurons with the highest or lowest overall activation magnitude. Note: we removed stop-words like 'the' or 'a' as well as spelling duplicates from the table's feature lists to remain brief. Features are sorted by decreasing activation mass from left to right. We see that the first three highly active neurons roughly encode movie-related locations and entities as well as sentiment terms like 'dull' or 'great', though some seem unspecialized (general), fitting many genres.

When looking at the three least activating 'supervision' neurons, we find more specialized feature lists. Some of them are short and very specialized to a specific feature – e.g. the 372 ‘walter’ neuron seems to be a ‘Breaking Bad’ review detector, while ‘archer’ (688) may detect the animated show of the same name. Somewhat surprisingly, Neuron 1289, despite only having a low activation sum, is comprised of many features that focus on sentiment like ‘terrific’ or ‘dull’, making the neuron more specialized than the top three. This suggests that ‘supervision’ neurons with low activation mass, somewhat independent of their feature variety, are more specialized than the highly active ones – which reflects in their lower ‘neuron length’, i.e. them preferring fewer features. Detailed ‘discoveries’ like supervision-gained knowledge reinforce our motivation, that an exploration-investigation approach can reveal detailed insights about a model’s inner workings if ‘drilled-down’[4] far enough, which underlines TX-Ray’s application potential.

2.3.3.2 Pruning avoided, shared and gained neurons

To understand how much the ‘avoided’, ‘shared’ and 85 neurons ‘gained’ by supervision affect predictive task performance, we run four pruning experiments (A-D) that remove neuron sets to measure the relative change from the unpruned F_1 score in $\%$ – i.e., a drop from 80 to 77 is ${}^{77-80}/_{80} = -3.75\%$. Experiment (A) cuts 740 ‘avoided’ neurons from the encoder $E_{imdb-sup}$, i.e., 740 neurons with empty feature preference distribution after supervision. Experiments B and C cut the 20 least and most active neurons from the supervision tuned encoder. To select 20 neurons each, we sort neurons by their individual activation mass, i.e. the sum of a neuron’s (max) activations, where ‘unpreferred’ neurons with an empty preference distribution have zero activity. In the last pruning experiment (D), we prune the 85 neurons that became ‘preferred’ after (due to) supervision – i.e., were ‘unpreferred’ before in $P_{imdb-zero-shot}$. Table 2.2 shows for each pruning: the relative changes in training and test set F_1 and what percentage of the encoder’s entire (max) activation mass the pruned neurons drop.

For pruning experiment **(A)**, we see that removing ‘avoided’ neurons not only does not drop performance as commonly observed when dropping irrelevant neurons (Voita et al., 2019; Sanh et al., 2020), but actually *increases both training and test set* performance by 3.65 and 2.80 $\%$ respectively, resulting in better generalization. In Experiment **(B)**, when removing seldomly activated supervision neurons, as indicated by the low activation mass percentage of 0.004%, we lose significant training performance (-3.79%), but no test set performance, telling us that those neurons were over-specialized or over-fit to the training set. It also tells us that these neurons were likely short (over-specialized), similar to those in Table 2.1 that have low activation

[4] A fundamental visualization techniques design pattern used to describe incrementally more focused analysis.

Table 2.2: **Pruning avoided, preferred and supervision-gained neurons:** After supervised encoder fitting; (A) prunes avoided (unpreferred) neurons, (B,C) prune the least and most preferred neurons, and (D) prunes 85 neurons gained by supervision – i.e., that were non-preferred in pretraining. Colors represent relative score change in % from original – score drops (red,−), gains (blue). Similar to (Frankle and Carbin, 2019c), test score increases, despite pruning $\approx 50\%$ of encoder neurons in (A).

Which neurons prunded?	**% AM of 675**	**F1 change %**		**pruning effect**
		train	**test**	
none = baseline	100.000	0.00	0.00	–
A: 740 **avoided**	–	3.65	2.80	↓ noise, ↑ generality
B: 20 least **prefered**	0.004	-3.79	0.00	↓ over-fitting
C: 20 top **prefered**	83.120	-4.99	-1.43	↓ generalization
D: 85 **sup gained**	3.006	-3.71	-3.87	↓ sup. knowledge

mass (372, 688). When we examined this intuition, we found that each of the 20 neurons has a length of exactly one – i.e. is over-specialized. When pruning the 20 most heavily used supervision neurons **(C)** with 83.12% (max) activation mass, we see the largest drop in training set performance out of all experiments (A-D). This tells us that, similar to observations in experiment (B), TX-Ray again identified neurons that strongly over-fit to the training data, while they overfit the test set to a lesser extend. Thus, Experiments (B, C) indicate that cutting supervision specific neurons after training can help preserve generalization performance, i.e., reduce generalization loss. Lastly, for **(D)**, when pruning the 85 neurons 'gained' by supervision both training and test performances drop by equal amounts. Since these 85 supervision-only neurons only became 'preferred' after supervised fine-tuning, this indicates that pretraining-exposed neurons as in (B) and (C), suffer less from overfitting on new (test set) data, even when pruned. We reason that pretraining-exposed neurons in (B) and (C) have their knowledge partially duplicated across other neurons, while the supervision-only knowledge in the 85 'gained' neurons (D) has no such backups. **(Neuron) generalization, specialization:** These observations are not only consistent with known effects of pretraining on generalization (Peters et al., 2019; Howard and Ruder, 2018), but also show that TX-Ray can identify and distinguish at *individual neuron level*, which parts of a neural network improve or preserve generalization (A, B) and which do not (C, D). Moreover, the pruning based generalization increase in experiment (A) is consistent with findings of Lottery Ticket based pruning by (Frankle and Carbin, 2019c), as well as with our notions of *neuron specialization an gener-*

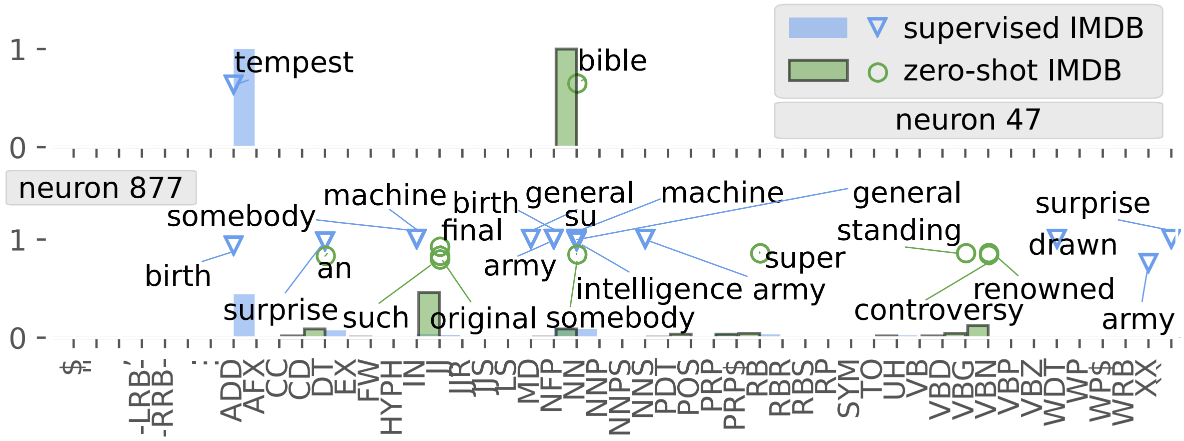


Figure 2.8: Low and transfer to supervision: Neuron **47** saw no transfer, while Neuron **877** transferred its knowledge better from before (zero-shot) to after supervised fine-tuning on IMDB.

alization used throughout TX-Ray. This demonstrates the method's effectiveness in identifying neurons that affect generalization and specialization.

To again analyze what individual neurons learned, we inspect neurons with high and low Hellinger distances between encoder activations before (green) $P_{imdb-zero-shot}$ and after supervision (blue) $P_{imdb-sup}$. In Figure 2.8, we show Neuron **47** (up), from the top 10 highest Hellinger distances. We see that the neuron 47 changed in both POS and token distributions after supervision, which suggests catastrophic forgetting, or supervised reconfiguration. For the low Hellinger distance Neuron **877** (down), we see some POS and token distribution overlap before and after supervision, and that movie review related terms (green ▽) become relevant, compared to noticeably war related tokens before supervision (green ○). This shows the neuron's semantic shift (POS, token) due to supervision – i.e., limited knowledge transfer occurred despite the low Hellinger distance. Moreover, distribution length changed for this neuron from 9 before to 15 tokens after supervision, indicating a lack of transfer. Finally, we recall that in the zero-shot case more neurons were 'shared' than after supervision, 1323 vs. 675 (Figure 2.5 vs. Figure 2.7), which should be reflected in the overall activation magnitude produced by encoder E before and after supervision.

2.3.3.3 Supervision sparsifies neuron knowledge

To investigate the distribution length shift and activation sum hypotheses formulated above, we visualize the shift of neuron length before and after supervision (Figure 2.9 and Figure 2.10), as well as the activation mass for the three research questions: (RQ1) pretraining, (RQ2) zero-shot, and (RQ3) supervision.

In Figure 2.9, we see neurons that shortened (red lines, $_\triangledown/^\circ$), or got longer (blue lines, $^\circ\backslash_\triangledown$), after supervision. Token preference distributions of neurons actually

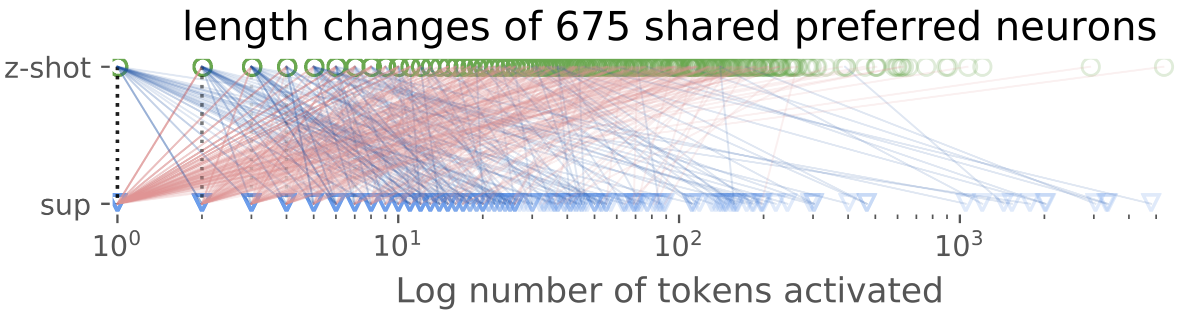


Figure 2.9: Neuron length before (○) and after **sup**ervision (▽). After supervision / neurons are shorter, \ are longer, and : are unchanged.

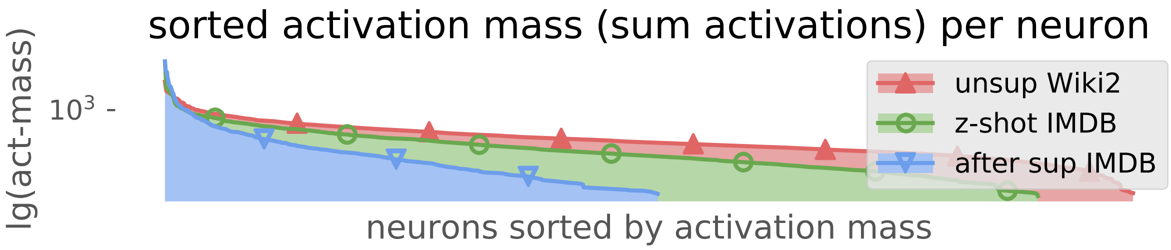


Figure 2.10: Sorted neuron activation masses, for the pretrained (large, ▲), zero-shot (middle, ⊖), and supervision tuned encoder (small, ▽). Supervision sparsifies activations – i.e. ▽ head peaks, tail shortens.

slightly lengthen by 4.62% on average over the 675 shared neurons,[5] while POS preference distributions, severely shorten at 32.83% (not shown). Similar neuron lengthening, 'feature variety increase', from supervision, was already apparent in neuron 877 (Figure 2.8), where supervision appeared to have specialized and extended a previously unspecific neuron into a movie sentiment detector[6].

In Figure 2.10, we see that the activation mass – i.e., the sum of activation values – differs across corpora and encoder activation distributions $P_{imdb-zero-shot}$, $P_{imdb-sup}$ and P_{wiki2}. A much more peaked activation mass is produced after the encoder has been fine-tuned via supervision and then again applied to IMDB (blue, ▽) compared to before supervision (green), which is a strong indicator that supervision sparsified the neuron activation and therefore the abstractions in the encoder. The activation mass of the pretrained encoder E on its pretraining corpus (WikiText-2, red ▲) is, unsurprisingly, the broadest, while it activates less strongly on the same amount of text (400k tokens) on the IMDB text (green, ⊖), due to the mismatch of domains between pretrained encoder and the new data domain – as seen in RQ2.

[5] Over the entire 1500 neurons, neuron token length shortens by 42.53% after supervision.

[6] Again, without deeper analysis, we are not claiming that this is the case, only that such points for investigation and new, interesting hypotheses can be identified via TX-Ray.

2.4 Related Work

Recent explainability methods (Gehrmann et al., 2019; Belinkov and Glass, 2019; Gilpin et al., 2018; Atanasova et al., 2020) fall into two categories: *supervised* 'model-understanding (MU)' and 'decision-understanding (DU)'. DU treats models as black boxes by visualizing how important each input is for a prediction outcome to understand model decisions. MU enables a grey-box view by visualizing internal model abstractions to understand what knowledge a model learned. Both DU and MU heavily focus on analyzing supervised models, while understanding transfer learning in self- and supervised models remain open challenges. **Supervised 'DU':** techniques explain decisions for supervised (probing) tasks to *hypothesis test* models for language properties like syntax and semantics (Conneau and Kiela, 2018; Schwarzenberg et al., 2019), or language understanding (Wang et al., 2019; Giulianelli et al., 2018). DU is limited to *supervised* analysis of *individual* samples (Gilpin et al., 2018; Arras et al., 2019). **MU:** techniques like Activation Atlas or Summit (Carter et al., 2019b; Hohman et al., 2020) explore supervised model knowledge in vision, while NLP methods like Seq2Seq-Vis (Strobelt et al., 2019) compare model behavior using many per-instance explanations. However, these methods produce a high cognitive load, showing many details, which makes it harder to understand overarching learning phenomena. **(Un-) supervised 'model and transfer understanding':** TX-Ray modifies ideas behind activation maximization (Erhan et al., 2009b; Olah et al., 2017b; Carter et al., 2019b) (see section 2.2) to enable measuring neuron knowledge change, specialization and generalization as well as to guide explorative transfer analysis by *quantifying interesting starting points*. Somewhat similarly to our setup in RQ3, (Singh et al., 2019) "calculate Helliger distances over 'neuron feature dictionaries' to measure neuron adaptation during 'supervised' task learning" in the prefrontal cortex of rats. Measuring changes in neuron feature preference distributions enables fine-grained analysis of neuron (de-)specialization and model knowledge transfer in RQ1-3. TX-Ray extends upon probing task and correlation based transfer analysis methods like (Liu et al., 2019a; Bau et al., 2019; Raghu et al., 2017), to provide more flexible, yet nuanced, (un-)supervised transfer interpretability and analysis for current and future (continual) pretraining methods (Peters et al., 2019; de Masson d'Autume et al., 2019), while also enabling *discovery of unforeseen hypotheses* to help *scale learning analysis beyond the limitations of supervised probing and approximate correlation analysis*.

2.5 Conclusion and Future Work

We presented TX-Ray, a simple, yet nuanced model knowledge explainability method for analyzing how neuron knowledge transfers between pretraining (RQ1), zero-shot knowledge application (RQ2), and supervised fine-tuning (RQ3). We showed how to extract neuron knowledge abstractions in NLP, developed extensible explainability visualizations and demonstrated how this can measure knowledge abstraction change. We find that TX-Ray enables explorative analysis of how knowledge is lost and added during supervision (RQ3), how neurons overfit or generalize (RQ1-3), and how pretraining builds knowledge abstractions (RQ1). TX-Ray is designed to reduce computational and cognitive load, but is flexible and scalable. In future, we will use TX-Ray for more advanced transfer models and metrics. The code and visualizations are available at github.com/copenlu/tx-ray.

Acknowledgements

This work was supported by the German Federal Ministry of Education and Research within the projects XAINES and DEEPLEE (01IW17001).

3 EffiCare: Better Prognostic Models via Resource-Efficient Health Embeddings

3.1 Introduction

For many machine learning tasks we can observe that more complex models with more data tend to outperform the previous state-of-art model. Particularly deep learning approaches can implicitly learn to extract and apply task relevant features(Choi et al., 2016; Kwon et al., 2019b; Lu et al., 2019; Choi et al., 2017; Ma et al., 2018; Song et al., 2018a; Ma et al., 2019b,a). However, complex models require more compute resources and large amounts of data to learn well – usually the more data the better. In clinical contexts this might be an issue as large datasets are often not available. For example, on the 2017 four-task clinical prediction benchmark by Harutyunyan(Harutyunyan et al., 2017)[1], the paradigm of applying complex models taken from other fields, but limiting training data to a small set of expert-selected features has resulted in a stagnation in performance – see Table 3.1 †. Limiting the amount of input features is based on two common assumptions. From the medical perspective, limiting features to known ones increases trust and model interpretability. From a machine learning perspective, it is common to limit the number of input variables (features) and discard rare ones, in an effort to make learning easier. A welcome side effect of such limitation is that memory, compute and manual labor requirements are minimized to meet real world time and cost limitations. However, feature limitation by expert bias prohibits the discovery of new and unforseen correlations. Moreover, a highly limited feature set combined with relatively small dataset does not provide enough input information to fully utilize deep learning methods. This problem is compounded by an increasing trend to adopt the latest large, complex models with many learnable parameters from domains like computer vision or natural language processing(Choi et al., 2016; Kwon et al., 2019b; Lu et al., 2019; Choi et al., 2017; Ma et al., 2018; Song et al., 2018a; Ma et al., 2019b,a), where data types are much more homogeneous and large scale data can be readily exploited. On the other hand, Tomašev et al.(Tomašev et al., 2019) provide a recent example of the performance benefits gained from using all instead of an expert-selected feature set. However,

[1] Originally published in 2017. Nature publication in 2019.

when using hundreds of more features with complex models, the resulting learning setup quickly becomes impractical for hospital deployment, due to the large memory, compute and complex model interpretability. However, for low-data tasks we can also use small embedding-pooling based models that were originally designed as text classifiers, where small models outperformed complex, 29-layered, convolutional networks, while using orders of magnitude less parameters and compute (Bojanowski et al., 2017).

By combining and extenting upon these insights from other fields we propose and demonstrate the following 'less-but-better' approach of using large data, with small models. We *introduce a resource-efficient feature-embedding method*, along with a *lightweight neural architecture* to process electronic health records (EHR) efficiently. This allows our approach to be able to encode and process a large variety of information performantly. We test our method on four different clinical prediction tasks found in the context of intensive care unit (ICU)(Harutyunyan et al., 2017). Using smaller models over resource-efficient feature-embedding we can handle an amount of data that would be resource-prohibitive for large models – i.e. we train over hundreds of features, while previous (complex) models used only a few hand-picked features(Ma et al., 2018; Song et al., 2018a; Ma et al., 2019b,a; Harutyunyan et al., 2017), given comparable compute setups. By combining two unsupervised pre-training methods(Bojanowski et al., 2017; Rethmeier and Plank, 2019), we simplify recent ideas of embedding patient health event histories as *time-stamped embedding sequences*(Ma et al., 2018, 2019b,a), to 'embed-away' the sparsity of health records. The proposed embedding and model approach greatly reduces the manual labor required to add data sources and test new models, which enables faster development iterations. As a result of fast development we quickly realized that less complex, pooling based models greatly outperform complex recurrent, convolutional or self-attention approaches – both in terms of task scores and resource efficiency.

Finally, we focus on model trust and transparency through 'model understanding'(Gehrmann et al., 2019), i.e. interpretability methods(Rethmeier et al., 2020a; Erhan et al., 2009b), to inspect the model, and model 'decision understanding'(Gehrmann et al., 2019), i.e. explainability(Sundararajan et al., 2017a), to identify important medical events for predictions. Combining 'model and decision understanding' allows us to verify that our models not only optimize task predictions, but also implicitly learn to identify important biomarkers for each prediction task and that they use data sources and model components efficiently, i.e. with little redundancies. Overall, we improve performance and minimize labor, time, data and hardware use to help remove these major obstacles in applying state-of-the-art prognostic models to ease the day-to-day hospital application.

3.2 Benchmark Dataset

MIMIC-III (Johnson et al., 2016), is an anonymized public database containing electronic-health-records (EHR) for over 40,000 patients from intensive care units (ICU). The data is longitudinal, heterogeneous and irregularly sampled. Furthermore, there are duplicate entries and erroneous input by the medical staff. For evaluation, we use benchmark tasks provided by Harutyunyan et al. (Harutyunyan et al., 2017). We refer to this work as *benchmark* or *Haru17/19. The benchmark includes four different tasks, namely *In-Hospital Mortality*, *Decompensation*, *Length of Stay* and *Phenotyping*. *In-Hospital Mortality* task predicts whether a patient will die during their stay at the hospital based on the first 48 hours of ICU admission. *Decompensation* task predicts at every hour whether the patient will die in the next 24 hours. *Length of Stay* is the task of predicting the remaining number of hours of a patient in ICU at each hour of stay. *Phenotyping* addresses the task of classifying the diagnosis (multilabel out of 25 acute care conditions) at the end of the patient's stay. Each task uses the data of an individual ICU stay up to the prediction time. The resulting benchmark dataset consists of 33,798 unique patients with a total of 41,902 ICU stays and over 250 million clinical events. Each sample corresponds to an individual ICU stay of a patient. Our experiments use the same cohorts, including the same training, development and test data splits. To limit information leakage, the splits are made based on individual patients, not ICU stays.

3.3 Embedding Sequential Data

Most related work in the context of prognostic deep learning models use a set of fixed features which are inserted into a sequential neural architecture(Tomašev et al., 2019; Song et al., 2018a; Harutyunyan et al., 2017; Ma et al., 2019a), expecting each particular feature at a fixed position. Opposed to that we would like to avoid a fixed set of predefined features or techniques such as imputation. In natural language processing we can technically use an unlimited amount of sequential embedding features for words to feed word co-occurence semantics into a machine learning model. Such embeddings are multi-dimensional vectors, that are pre-trained using self-supervision(Bojanowski et al., 2017; Rethmeier and Plank, 2019) such that words that share common contexts are placed close to one another in the vector space, i.e. have a smaller cosine similarity. Inspired by this idea we encode diverse set of events and their non-scalar values as vectorsas follows. To gather events, we use the same source data tables from MIMIC-III as the benchmark. The benchmark paper constructs a set of 17 hand-picked features, which they pre-process by merging duplicate events and cleaning values. We instead use all 4336 events from those

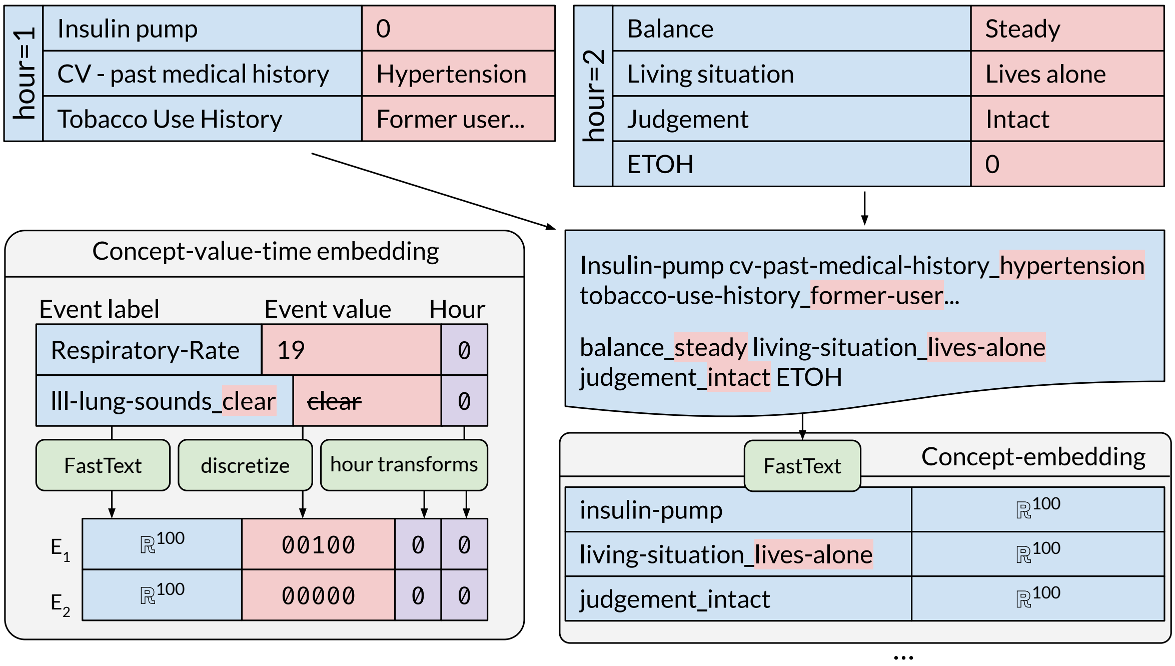


Figure 3.1: Concept-embeddings: Feature labels in an hour slot co-occur as a sequence of concepts, to train concept-embeddings via FastText – see upper and right image portion. **Concept-value-time embeddings**: Concept-embedding are concatenated with binned feature values and encoded time stamps. For example, the value 19 of label *Respiratory_Rate* is discritized as 00100, assuming that 19 is a value in the middle range of *Respiratory_Rate* based on training set values. Categorical label-values like *clear* for *lll-lung-sounds* are assigned a zero vector 00000.

tables. Namely, CHARTEVENTS containing the electronic chart that displays patient's information relevant to care, LABEVENTS containing all the laboratory test results and OUTPUTEVENTS consisting any output fluid excreted by or extracted from the patient, such as urine output or drain after surgery. In the training set, patients have 87 hour stays on average, where 92% of these hours have events in CHARTEVENTS , 15% in LABEVENTS and 12% in OUTPUTEVENTS tables. When an hour has events, it has on average 50 in CHARTEVENTS , 16 in LABEVENTS and 1 in OUTPUTEVENTS tables. The same event, e.g. heart rate measurement, can occur multiple times within an hour. Each event consists of hours elapsed since admission, a label such as Heart-rate or Heart-rhythm, and a corresponding value such as 89 or "asystole".

To embed the event label into a vector representation we use *randomly initialized embedding vectors* and pretrained *FastText embeddings*. To obtain pretrained FastText embeddings, event labels occurring together within a time step are concatenated and written into a file – see Figure 3.1. Using this sequence of feature labels any word embedding method can be applied. We use the FastText (Bojanowski et al., 2017) skip-gram method to learn those embeddings because FastText incorporates sub-word information in token embeddings. This results in embeddings, where labels that share sub-words are close-by in the embedding space, e.g. heart-rate and heart-rhythm. Learning embeddings for sub-words also helps us mitigate problems of changing the distribution of tokens by decreasing the number of categorical features by splitting each into many with concatenated values. After obtaining these embeddings, we use them as a non-trainable, or frozen, embedding look-up table in our models. That way we can feed an event sequence into this table, which turns them into an embedding sequence. Note, values in MIMIC-III are highly heterogenuous and unclean, e.g. 89, "SR (Sinus Rhythm)", "1cm", "24/24". To handle such input we first obtain unique feature labels by concatenating event names and any value that is not a scalar, e.g. judgement-intact, ventilation_rate-24/24. After obtaining the vector representation for the event label, scalar feature value and time stamp we concatenate these features into a single 'concept-value-time' embedding vector – Figure 3.1. Scalar feature values are discritized and encoded with a one-hot vector. For each individual feature we calculate uniform buckets for values using the training set. Each bucket represents a value range of that feature. In addition, two further buckets are used to deal with outliers. Categorical feature values are already included in feature label embedding and are represented as zero vectors here. Furthermore, time bucket t in hours is included both in logarithmic and exponential form: $\log(t+1)$, $\exp(t/1000)-1$. Using this calculation normalizes the hour format (which can go up to 1500 hours) in two non-linear ways(Kwon et al., 2019a). The complete pipeline is shown in Figure 3.1.

3.4 Prognostic Multitask Model

To efficiently integrate the large amount of data features occurring in the ICU we use a small model with as few parameters as empirically necessary. This enables us to train the model within a reasonable amount of time, despite using thousands of features – i.e. our best performing model trained in 40 minutes × 14 epochs on a single GPU[2]. During early experiments we also tested deep and shallow GRU-based and CNN based architectures, which were significantly slower to train but performed worse than the small time-pooling based model. Figure 3.2 provides an overview of our prognostic multitask model. The events from different sources (e.g. chart events) are embedded and collected within '1-hour-buckets' containing all embeddings occurring within a single hour t for that source. The size of each bucket is patient-dependent according to the maximum number of features occurring within an hour for that source. In case of a smaller number of features occurring within an hour, the bucket will be zero padded. Each embedding e_t from bucket at hour t is fed into a fully connected layer (FC) with ReLU activation, followed by different poolings (max, avg and normalized

[2] GeForce GTX 1080 Ti with 11GB memory, 10 Core CPU.

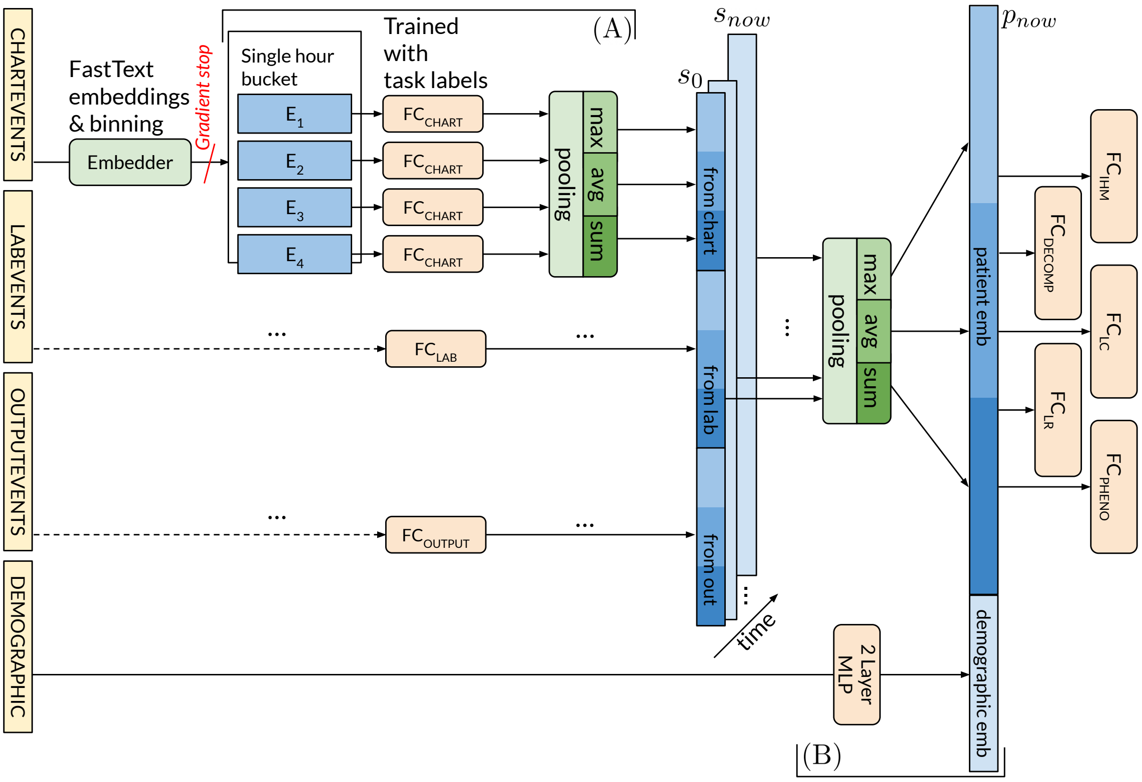


Figure 3.2: Our event-time-pooling model: Green boxes are modules without or with fixed parameters (Embedder). The left model half shows how a single time-step of event embeddings is pooled per data input source (4 yellow sources). We then pool over all time steps and data source features to discard unhelpful information.

$$\mathbf{x}_t^i = \frac{A_1^i + A_2^i + \cdots + A_t^i}{\sqrt{|A_1^i + A_2^i + \cdots + A_t^i|}} \tag{3.1}$$

sum) over the events. The output of each pooling is then concatenated to obtain an hourly patient embedding s_t, together with the pooling outputs from all other event sources (e.g. lab and output). This step (A) is applied for all hours of all events since admission (until the current prediction time s_{now}) to generate a sequence of hourly patient embeddings from s_0 to s_{now}. Finally all patient embeddings up until prediction time are fed into same poolings over time. The outputs of the poolings are then concatenated with the demographic embedding (B), resulting in a patient embedding p_{now} that represents the patient's aggregate state from admission up until hour now. This final patient embedding is then fed into the different fully connected layers, corresponding to the four different predictions (see right side of figure). Max and average poolings are commonly used in deep learning, however normalized sum pooling, see equation 3.1, was introduced by (Pham et al., 2016), to account for the accumulation of risk through time. Additionally, this sum pooling version ignores zero-padding in contrast to average pooling, which provides an additional implicit learning feature.

3.5 Configuration and Experimental Setup

For the following experiments, we trained feature representations using a FastText with a window size of 15 and default parameters with 100 dimensional embeddings. Scalar values are discritized into 10 uniform bins with two additional outlier bins for the 0.01 and 0.99 percentile values. Bins are calculated over the training set and are fixed at test time. This results in a one-hot vector of length 12 for scalar-valued features and a zero vector for any categorical feature. The fully connected layer for each table has 50 dimensions. The demographic input is sent through a two layer MLP with dimensions 40 and 20 and ReLU activations in-between. Each task is predicted by a corresponding fully connected layer using this representation.

We train with mini-batch size 16. For regularization, we apply input dropout with probabilities 0.15 for events and 0.1 for whole time-steps. The demographics network is regularized with 0.3 dropout. All tasks, but especially decompensation, have imbalanced label distributions. Thus, we use class weighting while calculating the training loss. This is equivalent to over-sampling patients that correspond to less frequent labels. For the multitask loss, we use the same weighting as in the benchmark paper(Harutyunyan et al., 2017), except for the fact that we train regression and classification for length of stay at the same time: 0.2 for in hospital mortality, 2 for decompensation, 1 for length of stay classification, 0.1 for length of stay regression and 1 for phenotyping. As optimizer we use Adam with Pytorch defaults, i.e. learning rate 0.001 and betas 0.9, 0.999. Since a few patient histories are disproportionately long, we consider only the first 30 days of each patient's stay at the hospital during training, as done by(Ma et al., 2019a,b). At test time, we predict over all time-steps. For development and model selection, we used the training and validation set defined by (Harutyunyan et al., 2017). The test set from (Harutyunyan et al., 2017) is used only to report results.

3.6 Results

See results in Table 3.1. As ROC scores overestimate performance under class imbalance(Saito and Rehmsmeier, 2015) we only discuss the results related to AUC-PR and Kappa. In case of *Phenotyping* previous works do not provide AUC-PR results. While a higher score represents an increase in performance for most evaluation measures, in case of MAD, a lower score is better.

Model	Mortality		Decompensation		Length of stay		Phenotyping			
	AUC		AUC				AUC-ROC		AUC-PR	
*Single-Task (**ST**), MultiTask (**MT**)*	PR	ROC	PR	ROC	Kappa	MAD	Macro	Micro	Macro	Micro
(*) Baselines ↓										
*Haru17/19 benchmark (ST or MT)	0.533	*0.870*	0.344	0.911	*0.451*	*110.9*	0.77	0.825	n.r.	n.r.
*Ma-Care19a,b (ST)	0.532	0.870	*0.304*	*0.900*	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.
*Song18 (ST, MT)	0.519	0.863	0.327	0.908	0.429	n.r.	0.771	0.821	n.r.	n.r.
(†) Our models ↓										
† Random embeddings (MT)	0.614	0.906	0.595	0.970	0.739	58.5	0.813	0.844	0.512	0.555
† FastText embeddings (MT)	0.654	0.916	0.615	0.981	0.735	57.4	0.820	0.853	0.520	0.574
† with *Reth19 embeddings (MT)	**0.655**	0.912	**0.656**	**0.982**	0.733	57.2	0.820	**0.856**	**0.523**	**0.576**
(†) Ablations: Filtering features from our best model↓										
† FastText event count > 100 (MT)	0.654	**0.917**	0.596	0.981	0.734	**56.6**	**0.823**	0.854	0.522	0.574
† FastText only 17 *Haru17/19 (MT)	*0.506*	0.873	0.558	0.963	**0.767**	59.8	*0.769*	*0.810*	*0.421*	*0.475*
† FastText w/o demographics (MT)	0.651	0.913	0.587	0.982	0.725	56.8	0.818	0.851	0.520	0.574

Table 3.1: ***Haru17/19 Benchmark test set performances for the state-of-the-art models (*) vs. our multitask, all features model (†).** ***Haru17/19** results use only 17 expert-selected features with LSTMs(Harutyunyan et al., 2017). *Ma-Care19a,b are baseline results from (Ma et al., 2019a,b) (Dilated CNN, Transformers), while *Song18 is from (Song et al., 2018a) (Transformers). ***Reth19** are FastText embeddings retro-fit via(Rethmeier and Plank, 2019). **Best**, *worst* and n.r. (not reported) scores. (ST, MT) indicate single and multitask models.

Model	# Parameters	# Features
*Haru17/19 MC-model (MT)(Harutyunyan et al., 2017)	1,225,253	64
FastText embeddings	70,528	10461
FastText event count > 100	70,528	6165
FastText only 17 *Haru17/19	70,528	75

Table 3.2: Number of trainable parameters and features. Each categorical event-value counts a as a single feature. For comparison, the 17 features used in *Haru17/19 result in 64 individual features. *FastText only 17* counts non-preprocessed events used to construct the 64 features from *Haru17/19(Harutyunyan et al., 2017).

Sequence embeddings of patient features majorly boosts performance: First, we evaluate our embedding-based methods (†) against the previously best single and multitask models (*)(Harutyunyan et al., 2017; Ma et al., 2019a,b; Song et al., 2018a) in Table 3.1. For each baseline work(*), we list the best model results only. The *Haru17/19 benchmark has the oldest results, originally published in 2017 and then republished in a 2019 Nature issue(Harutyunyan et al., 2017). The first two works (*) only use the 17 features selected by *Haru17/19. Notably more recent complex models by Ma and Song *did not beat the less complex baselines* by *Haru17/19. We also see that multitask learning performs better. These observations suggests that we should use less complex, multitask learning models with more than 17 training features. Our approaches, that use *all 10,461 benchmark feature embeddings* with our comparatively simple multitask models(†), outperform the baselines (*) on all tasks by large margins, where some scores such as length of stay MAD and decompensation AUC-PR almost double. Furthermore, as Table 3.2 shows, even though our approach uses many more features, it has a much smaller number of trainable parameters. Thus, small neural architectures that are designed for low-resource scenarios by using fewer parameters are not only easier and faster to train, but also boost performance.

Further, we can conclude that *using all available features drastically benefits learning* and that multiple tasks may benefit each other through shared, pretrained event embeddings (see Embedder in Figure 3.2). Note that, using random embeddings does not require embedding pretraining, and therefore requires the least modeling effort. However, when we exploit embedding pre-training via FastText (Bojanowski et al., 2017) we improve on all metrics except length of stay Kappa.

Finally, we retro-fit our event embeddings using the autoencoding method and implementation[3] by *Reth19(Rethmeier and Plank, 2019). We select optimal *Reth19 embeddings using our validation set, which leads to a 4.1 point increases in the decompensation test score without using extra labor or hardware resources – i.e. they

[3] https://github.com/DFKI-NLP/MoRTy

efficiently boost performance. For completeness, we also mention three negative (dead-end) insights. We initially tested many more complex models, but since using CNN and GRU layers strongly underperformed in prediction and compute performance compared to our smaller models, we choose to not pursue complex architectures. Further, excluding the number of hours elapsed since admission in the input embeddings hurt performance moderately, while using only max pooling considerably hurt performance. Telling us that average and sum pooling are important components.

Early expert bias for feature limitation hurts performance: Next we analyze if early domain expert-based bias, like feature selection or removing rare features, impacts model performance. As done in the benchmark paper(Harutyunyan et al., 2017), we use only the 17 hand-picked features to train our model, except we do not do any feature clean-up (see *FastText only 17 *Haru17/19*). Table 3.1 shows a large drop in overall performance compared to the all-features (†) *FastText embeddings* model, except for *Length of Stay* Kappa. However, even in this low-data setting, our approach still mostly outperforms previous state-of-the-art approaches indicating that small models may be better suited to the data setup.

Additionally, when removing infrequent features from training (FastText event count > 100), we see that model performances drop for most tasks. Thus, we conclude that early domain biases like early expert feature selection and rare event filtering can greatly hurt task learning and thus test set prediction performance. Our motivation behind testing the influence of removing demographic features will be explained in the interpretability-detail section, further down.

3.7 Interpretability for 'model understanding'

Automation bias, is an *over-reliance on decision making technology due to high system complexity and low understanding*. Thus, we aim to peer into our models' black-box to increase awareness of its capabilities and its limitations by inspecting learned abstractions. We hence analyze how our best model combines and filters patient features and how important different model components are for task predicting each task. Using intepretability, we allow 'model understanding' (Gehrmann et al., 2019) at two levels – i.e. detail and overview.

First, we gain a *detail-view* of how a model represents domain knowledge from chart or lab events for all tasks by adapting the interpretability concept of "aggregating and visualizing what features the neurons in a model prefer" from (Rethmeier et al., 2020a; Erhan et al., 2009b). This concept was introduced in image recognition to visualize a model's learned abstractions like eyes or noses (Erhan et al., 2009b). The method was recently extended to visualize word category abstractions in language processing

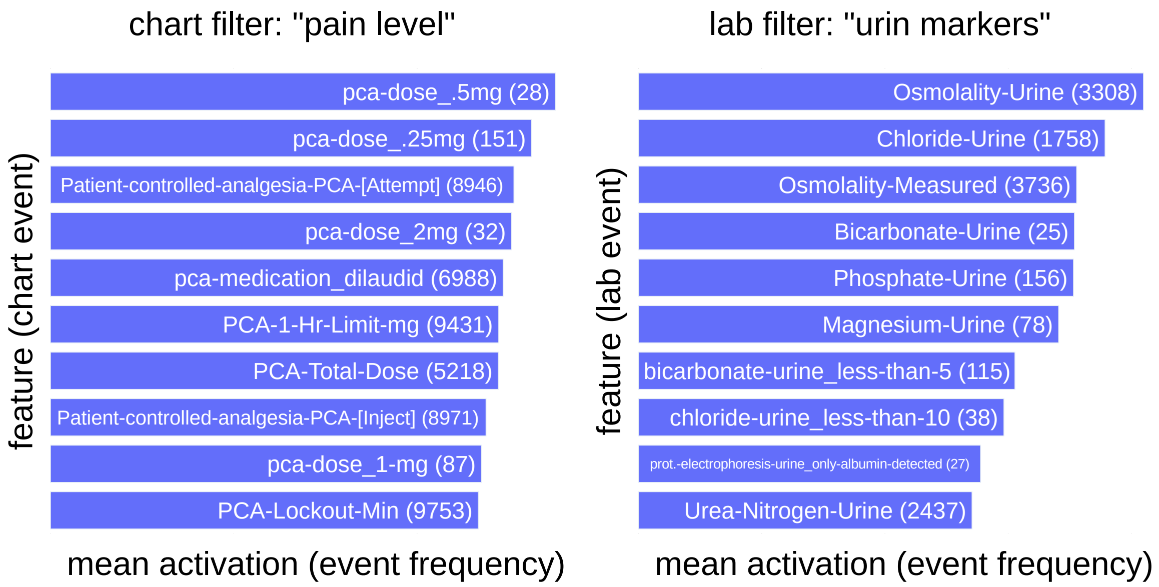


Figure 3.3: Detailed interpretability view: Top-10 "preferred features" of two patient-event level filters. Two filters H_i from FC_{CHART} and FC_{LAB} (Figure 3.2). The chart filter abstracts pain management events, the lab filter captures urine indicators. **(Rare features)** like 'Bicarbonate-Urine' matter – see **(feature count)**.

models (Rethmeier et al., 2020a), which we can use in a similar fashion to visualize which events our model components prefer during. Secondly, we aggregate activations of the learned patient representations (*patient emb* in Figure 3.2) to gain an *overview* of how important different types of either domain knowledge (data sources) or model components are for predicting each of the four tasks.

Interpretability detail-view, reveals (non-)redundancies: To visualize what low-level filters learn about individual data sources like chart or lab events, we collect the "preferred (maximally activated) features" of each neuron in FC_{CHART} and FC_{LAB} over the training set. Thus, each 'filter neuron' in FC_{CHART} or FC_{LAB} forms a distribution of 'preferred features' using the method and implementation[4] proposed by Rethmeier et al.(Rethmeier et al., 2020a), who similarly visualized knowledge abstraction during unsupervised language model training. In Figure 3.3, we see which patient features are the most active (preferred) in two neurons of our model's learned low-level feature filters. Note that, we see one chart and one lab neuron out of a total 150 filters – i.e. 50 filters each over chart, lab and body-output events.

The presented *chart data filter* shows a strong focus (in terms of activation-preference) on information regarding the need for self-controlled pain-relieve – i.e. the patient's perceived pain level. The presented *lab filter* is most active for features that involve urine lab values. Thus, these filters *implicitly learned to abstract and cluster knowledge* about pain and urine related health indicators, even though *we did not model any*

[4] https://github.com/copenlu/tx-ray

explicit feature clustering or preprocessing as done in older, manual feature-engineering approaches. When looking at the other filters, we found that many filters learned to focus on specific clinical contexts: bed-laying pressure points, cardiology-cholesterol measures, blood health, self-sufficiency, and diet.

Moreover, we found that data sources, e.g. body-output data, that have fewer features than others, have an increasing number of filters with empty feature preference distributions. This suggests that those filters may not be necessary and could be removed to shrink the model and therefore its hardware requirements. Interestingly, we also observed that non-empty filters form unique feature clusters – i.e. no duplicate clusters. This indicates that these filters learned to *avoid abstraction redundancies*, or as representation learning terminology puts it, our model learned disentangled representations[5]. Practically, efficient data representation results in a model that uses computations and memory in terms of its parameters efficiently.

Finally, we noticed that chart event filters such as those seen in Figure 3.3 already encode demographics such as sex and weight. Accordingly, after removing the demographic input data and its 2 layer MLP component, test set performances were nearly unaffected – see † *FastText w/o demographics (MT)* in Table 3.1. This leads us to conclude that, *we can use interpretabilty to identify and then remove unnecessary (redundant) model components and data sources*.

Interpretability overview, how relevant data and model components are per task? To gain an overview on how each prognostic task uses features from chart, lab and output events we collect learned patient embedding representations (see Figure 3.2) for all patient histories in the development set. Next, we multiply the representations with each of the four classifier weights. In this way we can investigate how strongly (active) each task weights specific data sources and pooling information. In Figure 3.4 we see that neither of the two tasks uses body-output feature information and that data and pooling importance vary per task. However, when we remember (see data section) that only 12% of patient-buckets have `OUTPUTEVENTS` , while 92% and 15% have `CHARTEVENTS` or `LABEVENTS` , we understand that output events still matter, and that lab events are very important[6], even though chart events dominate by raw frequency and hence weighted activation. For Decompensation sum and average pooling over a single time step (event-level) cause both strong positive and strong negative impacts on the prediction score. Also, chart events are the most important here. At event-level, average and sum pooling have strong impacts on predictions, while at the pooling-over-time-level, we see that max pooling is the most impactful.

[5] `https://deepai.org/machine-learning-glossary-and-terms/disentangled-representation-learning`

[6] Due to space limitations, we do not show frequency adjusted relevance, though this is easy in practice.

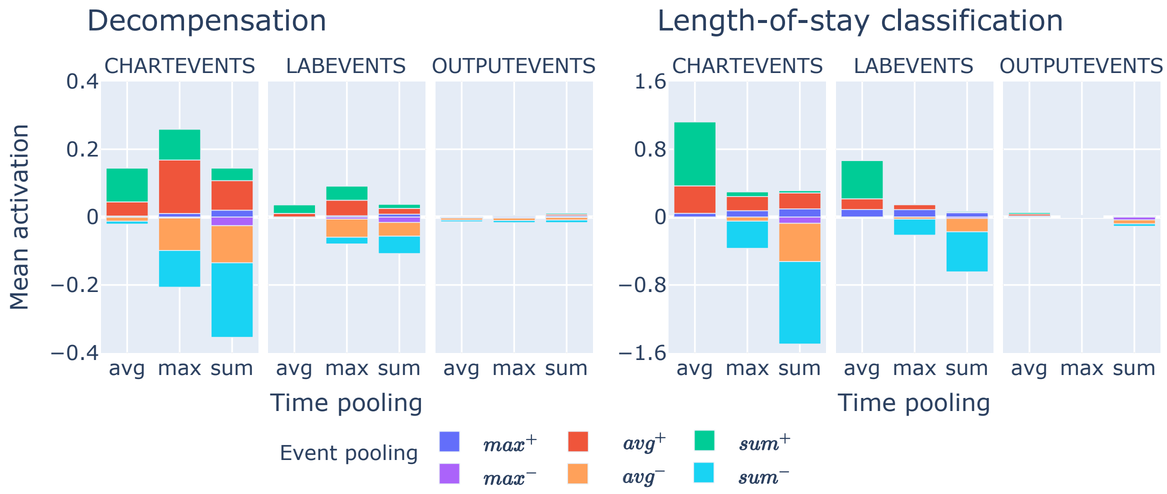


Figure 3.4: Feature type and pooling mechanism relevances for two tasks. The x-axis shows relevances for max, average and sum pooling over time (high level patient embeddings). The color bars express relevances of pooling *over events of a single time step* (low level data filters FC_{CHART}, FC_{LAB} and FC_{OUTP}) – see Figure 3.2.

We also see that average pooling over time causes only positive impacts. Thus, each task uses data sources and network components differently.

3.8 Explainability for per-patient 'decision understanding'

So far we only looked at understanding how the model abstracts data and uses its component to model the four tasks. However, to also provide 'decision understanding' (Gehrmann et al., 2019) of what features are most relevant or impactful for a specific prediction, we use the explainability method of integrated gradients (see (Gehrmann et al., 2019)). We again split our analysis into overview and detail. This allows us to see how (counter-)indicative specific event value combinations are in for predicting patient mortality overall compared to how important events are in a single patients history.

top mortality indicating features	↑ impact	frequency	top counter-indicative features	↑ impact	frequency
code-status=comfort-measures-only	0.87	161	Drain-Out-#1-Tap*	-0.89	95
heart-rhythm=asystole	0.82	194	Jackson-Pratt-#1	-0.81	4102
Gastric=Emesis	0.79	1174	Ultrafiltrate-Ultrafiltrate	-0.67	4163
Other*	0.74	119	Drain-Out-#1-Sump*	-0.63	21
code-status=cpr-not-indicate	0.68	1150	sputum-source=expectorated	-0.61	9951
code-status=comfort-measures	0.66	3044	Drain-Out-#2-Other*	-0.54	140
code-status=do-not-resuscita	0.66	50477	diet-type=diabetic	-0.54	14081

Table 3.3: **Explainability-overview for In-hospital-mortality:** Data features (events) that raise or lower correct predictions for the in-hospitality-mortality task. Left: 7 most raising features. Right: 7 most score-lowering features. * Infrequent features are important for the task.

Explainability overview of the most and least predictive features: Using the popular integrated gradients method (Sundararajan et al., 2017a), we calculate for each development set feature its impact on the prediction score of true positive in-hospitality-mortality predictions. Table 3.3 shows that features like *heart-rhythm=asystole* or *Gastric-Emesis* have a strong positive influence, whereas *Drain-Out-#1-Tap* or *Ultrafiltrate-Ultrafiltrate* have a strong negative influence on a positive prediction. "Other" describes four identically labeled hematology events from `LABEVENTS` that are related to fluids: Cerebrospinal Fluid, Joint fluid, Pleural, Ascites. As we expected, high impact features describe events that likely occur close to an in-hospital-mortality. Importantly, we see that some infrequent features have a strong positive or negative influence on the classification, meaning they should not be prematurely filtered.

Explainability detail: (counter-)predictive feature in a patient's history. Figure 3.5 shows the attributions of input events to a correct in-hospital mortality prediction of a *single patient*. The prediction is done at 48th hour as per the task definition and the input has 3438 events in total. The figure shows only the events with a high (>0.2) positive or negative impact (attribution) on the mortality prediction for a patient, where the patient eventually died at 368th hour. The events from the initial 48 hours that are most indicative for the death of the patient are Foley=3, abdominal-assessment=ascites, Urea-Nitrogen-Urine=3, patient-location=cc6b Lactate=4 and Anion-Gap=4. Most counter-indicative are features like Sputum-source=expectorated, Urea-Nitrogen=2, Lactate=2 and Foley=3. This visualization lets us analyze which events are most critical to a patients condition over time, according to the trained model.

3.9 Discussion & Conclusion

We presented a novel, resource-efficient patient event sequence embedding method and model that largely improves the state-of-the-art performance on a public benchmark for patient health prediction in intensive care units (Harutyunyan et al., 2017). Due to its resource-efficient, automated design, our method is able to learn useful features from raw and heterogeneous input without laborious feature-engineering or impractical hardware demands. The model can deal with highly sparse raw streaming inputs that have errors and missing values. As a result it improves patient health prediction over clinically used severity scores(Salluh and Soares, 2014) and modern, often much more complex neural models(Ma et al., 2019b,a). Besides identifying important features that were used by others (Harutyunyan et al., 2017; Purushotham et al., 2017), our approach uses important features that seem intuitively sound upon

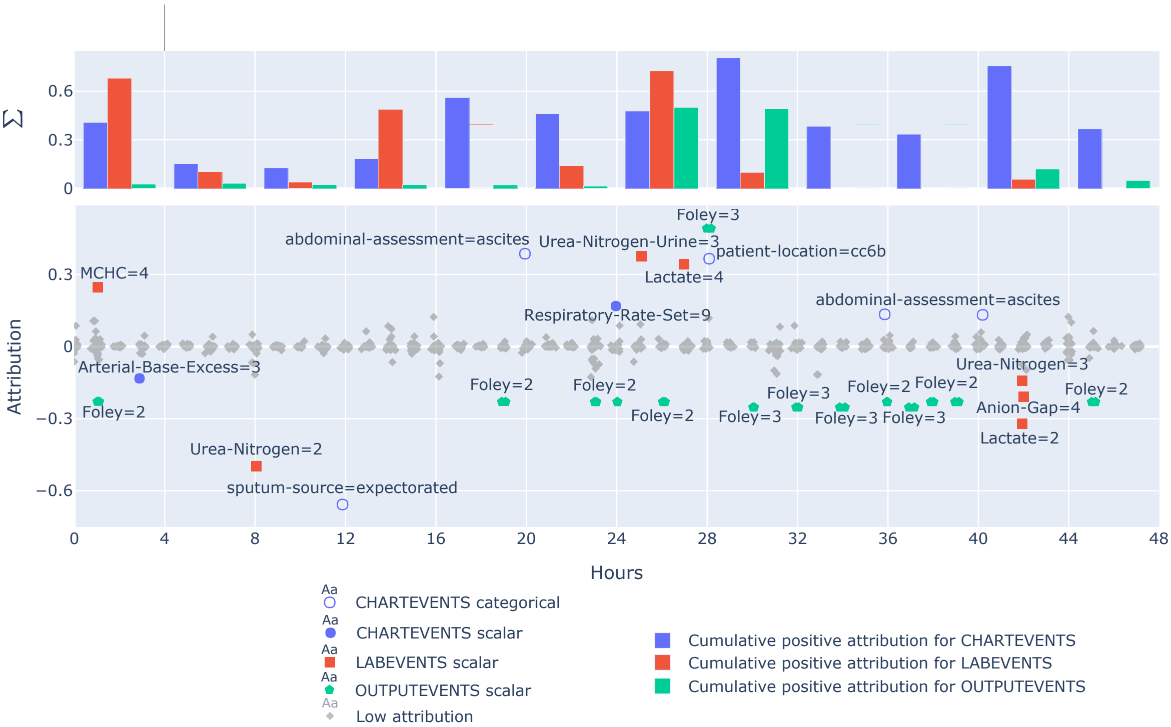


Figure 3.5: A single patients 48 hours integrated gradient explanation for a correctly predicted in-hospital mortality. Equals are categorical values, or the bin of a scalar value, where 1 is the lowest and 10 the highest bin (Outlier bins 0 and 11 are infrequent.) The bars show accumulative positive attributions for events in 4 hour blocks for different data sources. Colored events have a mortality prediction attribution (> 0.2). Negative attributions are counter-indicative.

loser inspection. Since real world application in a hospital requires minimization of human labor, time, hardware and extensibility costs, we propose multiple modeling choices to conserve those resources, while also producing substantial improvements over state-of-the-art methods.

Part III

Contrastive Language Model Adaptation

Paper 3: A Primer on Contrastive Pretraining in Language Processing: Methods, Lessons Learned & Perspectives

4

4.1 Introduction

Current downstream machine learning applications heavily rely on the effective pretraining of representation learning models. Contrastive learning is one such technique which enables pretraining of general or task-specific encoder models in a supervised or self-supervised fashion. While contrastive pretraining in computer vision has enabled the recent successes in self-supervised image representation pretraining, the benefits and best practices of contrastive pretraining in natural language processing (NLP) are less established (Jaiswal et al., 2021b). However, a first wave of works on contrastive NLP methods, seen in Figure 4.1, shows strong performance and data-efficiency benefits of (self-)supervised contrastive NLP pretraining. For example, even supervised contrastive pretraining enables zero-shot prediction of unseen text classes and improves few-shot performance (Pappas and Henderson, 2019). Moreover, task-agnostic self-supervised contrastive pretraining systems have been shown to improve overall language modeling performance (Logeswaran and Lee, 2018; Clark et al., 2020; Wu et al., 2020b; Giorgi et al., 2021a), data and label-efficiency (Radford et al., 2021; Rethmeier and Augenstein, 2020) or semantic similarity tasks (Gao et al., 2021a). Besides that, there are many task-specific uses of contrastive self-supervision, e.g. for pronoun disambiguation (Klein and Nabi, 2020), discourse representation learning (Iter et al., 2020) or text summarization (Duan et al., 2019), to name a few – see Section 4.3.

In this primer to contrastive pretraining, we therefore summarize recent supervised and self-supervised contrastive NLP pretraining methods. We then describe how they enable zero-shot learning and improve language modeling, few-shot learning, pretraining data-efficiency or rare event prediction. We cover basic concepts and crucial design lessons of contrastive NLP, while detailing the resulting benefits such

	supervised	self-supervised
end-task agnostic		COCO-LM: Meng, 2021 CLEAR: Wu, 2021 Electric: Clark, 2020 CLESS: Rethmeier, 2020 CoDA: Qu, 2020 MixText: Chen, 2020 DeCLUTR: Giorgi, 2020 OLFMLM: Aroca, 2020 CERT: Fang, 2020 CPC: Oord, 2018 QT: Logeswaran, 2018 Word2vec: Mikolov, 2013 SimCSE: Gao, 2021
end-task specific	SimCSE:	Gao, 2021
	👁 CLIP:	Radford, 2020
	CLESS:	Rethmeier, 2020
	CSS:	Klein, 2020
	👁 TCN:	Jian, 2019
	UST: Uehara, 2020	CONPONO: Iter, 2020
	GILE: Pappas, 2019	👁 ALIGN: Jia, 2019
		BIT: Duan, 2019

Figure 4.1: **Types of contrastive pretraining:** and works that fall within these categories. 👁 marks text-image contrastive works.

as zero-shot prediction and efficient training. Then, we structure existing research as supervised or self-supervised contrastive pretraining and explain connections to Energy-Based models (EBMs), since many works refer to EBMs. Finally, we point out open challenges and outline future and underrepresented research directions in contrastive NLP pretraining.

4.2 Contrastive Learning and its Benefits

At their core, contrastive methods learn to distinguish between pairs of similar or dissimilar data points. A pair of similar data points is called a positive sample, which in self-supervised contrastive learning is generated by augmenting an original data point. For example, SimCSE by (Gao et al., 2021a) applies two dropout masks to an input sentence to create two slightly different sentence embeddings that are then used as a pair of positive (matching) sentence embeddings for self-supervised pretraining. Negative samples are pairs where the two data points are of different data instances, e.g. in SimCSE the authors simply use the embeddings of other sentences in a training batch as negatives. Contrastive objectives have been demonstrated to have certain desirable properties over other common losses. (Graf et al., 2021a) have shown that a contrastive loss is more resistant to label noise than the commonly used softmax objective. Additionally, Zimmermann et al. (2021a) demonstrate that contrastive self-supervision effectively “inverts a data generating process”. This results in very data-efficient pretraining as both they and Rethmeier and Augenstein (2020) demonstrate. Other potential benefits include modelling redundancy reduction (Zbontar et al., 2021) and disentangling representations (Ren et al., 2021). Below, in Section 4.2.1, we overview popular contrastive losses for NLP and summarize how to avoid pitfalls. Then we overview connections to other machine learning fields and specifically outline Energy-Based models in Section 4.2.2, since they are used in much of the cited research. Finally we organize methods into input-input contrastive and the NLP specific input-output contrastive methods to highlight their respective benefits.

4.2.1 Noise Contrastive Estimation (NCE)

Noise contrastive estimation (NCE) is the objective used by most contrastive learning approaches within NLP. Thus, we briefly outline its main variants, Binary and Ranking NCE, and the core ideas behind them, while pointing to Ma and Collins (2018)[1] for detailed, yet readily understandable explanations of the two main NCE variants. Both variants can intuitively be understood as classification with undersampling of negative (non-active) classes. Either method is used to predict a similarity score between two text embeddings, where the prediction score 1 means similar, and a score of 0 or -1 means dissimilar, i.e. a direct analog to the standard (self-)supervised classification objectives. Training is done with positive and negative samples, where x_i is an original text input embedding, while a_i^- is an augmented text embedding that is dissimilar to x_i, and a_i^+ is a text embedding that is considered similar to x_i. A positive sample is then describes as a pair of similar text embeddings $<x_i, a_i^+>$ that is annotated, manually or automatically, with a similarity (class) of 1 via an indicator variable. A negative sample is a pair $<x_i, a_i^->$ of dissimilar text embeddings that is annotated with a similarity of 0, or -1, depending of the similarity function (e.g. cosine) to indicated dissimilarity or a mismatch. Either method uses one positive sample $<x_i, a_i^+>$, and sub-samples K negative samples $<x_i, a_i^->$ for contrast. Below we describe both variants, and will point out an easy way to remember both variants at the end.

Binary NCE: The first variant expresses NCE as a binary objective (loss) in the form of maximum log likelihood, where only K negatives are considered.

$$L_B(\theta,\gamma) = log\,\sigma(s(x_i, a_{i,0}^+;\theta),\gamma) + \sum_{k=1}^{K} log(1 - \sigma(s(x_i, a_{i,k}^-;\theta),\gamma) \tag{4.1}$$

[1] https://vimeo.com/306156327 talk by Ma and Collins (2018).

Here, $s(x_i, a_{i,\circ}; \theta)$ is a similarity or scoring function that measures the compatibility between a single text input x_i and a contrast sample $a_{i,\circ}$. This sample is another input text or an output label (text) to model NLP tasks as 'text-to-text' prediction similar to language models. The similarity or scoring function is typically a cosine similarity, a dot product, or a small neural network that computes a similarity or matching score between a pair of text embeddings (Pappas and Henderson, 2019; Rethmeier and Augenstein, 2020). The $\sigma(z, \gamma)$ is a scaling function, which for use in Equation 4.1, is typically the sigmoid $\sigma(z) = \exp(z - \gamma)/(1 + \exp(z - \gamma))$ with a hyperparameter $\gamma \geq 0$ (temperature), that is tuned or omitted depending on how negative samples a_i^- are attained (Ma and Collins, 2018).

Ranking NCE, InfoNCE, NT-Xent: learns to rank a single positive pair $(x_i, a_{i,0}^+)$ above K negative pairs $(x_i, a_{i,k}^-)$:

$$L_R(\theta) = log \frac{e^{\bar{s}(x_i, a_{i,0}^+; \theta)}}{e^{\bar{s}(x_i, a_{i,0}^+; \theta)} + \sum_{k=1}^{K} e^{\bar{s}(x_i, a_{i,k}^-; \theta)}} \tag{4.2}$$

In Jaiswal et al. (2021c), section 5, it can be seen that the Ranking NCE objective has the same form as the InfoNCE objective in CPC (van den Oord et al., 2018) or the NT-Xent objective in SimCLR (Chen et al., 2020b), except that the SimCLR version uses a similarity scaling factor (temperature) τ. The names InfoNCE and NT-Xent are commonly used in computer vision, while all names are used in NLP. Interestingly, van den Oord et al. (2018) also proved that "minimizing this loss maximizes a lower bound on the mutual information" of a positive sample (between a pair) $<x_i, a_{i,0}^+>$ – i.e. in their notation between $x_{t+k}, c_t := <a_{i,0}^+, x_i>$. This means that the Ranking NCE can also be understood as approximate mutual information maximization objective.

Additionally, as discussed in Ma and Collins (2018) section 3.2, some older works define a modified similarity (scoring) function $\bar{s}(x_i, a_{i,\circ}) = s(x_i, a_{i,\circ}) - log\, p_{\mathcal{N}}(a_{i,\circ})$ to subtract the probability of the current sample $a_{i,\circ}$ under a chosen noise distribution $p_{\mathcal{N}}(a_{i,\circ})$. For example, Mikolov et al. (2013c) set $p_{\mathcal{N}}(a_{i,\circ})$ as corpus word-unigram probabilities p_{word} (Mikolov et al., 2013c) to make the learning of word embeddings more robust. Works like Deng et al. (2020) set the noise distribution to the probability p_{LM} of a sequence under a language model LM, to learn contrastive sequence prediction. As Ma and Collins (2018) state, when desirable, adding this noise term can make the Binary NCE objective self-normalizing allowing it to converge faster towards the MLE solution. Ranking NCE is already self-normalizing, but Ma and Collins (2018) showed that adding the noise term can still improve RankingNCE results. While some older works like (Mnih and Teh, 2012) set the parameters of the noise distribution term to zero, for computational reasons, recent models do not use a noise distribution term (Wu et al., 2020b; Rethmeier and Augenstein, 2020).

Generalization to an Arbitrary Number of Positives: As Khosla et al. (2020b) discuss, original contrastive losses use only one positive sample per text instance (see e.g. Mikolov et al. (2013c); Logeswaran and Lee (2018)), while recent methods mine multiple positives per sample (Qu et al., 2021; Rethmeier and Augenstein, 2020). This means that the positive term in Equation 4.1 extends to become a sum over P positive samples.

$$\sum_{p=1}^{P} log\, \sigma(s(x_i, a_{i,p}^{+}; \theta, \gamma)) \tag{4.3}$$

A way to easily remember Binary and Ranking NCE: Binary NCE can be remembered as an undersampled version of Binary Cross Entropy over similarity scores. When used with multiple positives, but *without* a noise distribution term for self-normalization, it can learn multi-label problems, where an instance can have multiple active labels (classes) that are independent of each other. When using a single positive class, Binary NCE learns a contrastive version of multi-class classification, especially when adding the optional noise distribution term for self-normalization, which makes it act even closer to an undersampled softmax objective.

Ranking NCE can be remembered as an undersampled softmax over similarity scores, as it uses an undersampled normalization, which suits multi-class learning, where classes are mutually exclusive and normalization can be thought of as 'inducing a ranking and mutual exclusivity' between classes. While this objective if often appropriate when learning representations, other ranking losses can be used to induce ranking oriented semantics, e.g. SPECTER and SciNCL (Cohan et al., 2020b; Ostendorff et al., 2022b) use the triplet (ranking) loss for contrastive pretraining of citation representations. However, for practical applications the impact of ranking loss variant may be minor, as pointed out by Musgrave et al. (2020a), who give a concise overview of relevant losses and a critical analysis of realistic benefits and drawbacks.

Lessons on Effective Negative and Positive Sampling. A key component (and pitfall) of effective contrastive learning is how positive and negative samples are generated. Saunshi et al. (2019) prove and empirically validate that "sampling more negatives improves performance, *but only if they do not collide with positive samples*", which otherwise deteriorates performance. Instead, in section 6.3. of their work they propose to "sample negatives from blocks of similar data points", i.e. from similar contexts such as the same paragraph or sentence. Instances of such contextual contrast sampling can be found in (Saunshi et al., 2019; Rethmeier and Augenstein, 2020;

Iter et al., 2020). For example, Rethmeier and Augenstein (2020) sample words from a current text instance to construct positives for self-supervised pretraining of a contrastive text autoencoder model. Recent works use multiple positive samples to boost supervised contrast (Khosla et al., 2020b). Additionally, during self-supervision, multiple positives should be sampled from similar contexts (Wang and Isola, 2020) or "diversely from *common and rare* positives when pretraining long-tail recognition language models." (Rethmeier and Augenstein, 2020).

4.2.2 Contrastive Learning in Machine Learning

Contrastive learning methods are related to other machine learning concepts, all of which describe the same underlying intuitions of how to learn representations in either a supervised or self-supervised fashion. All these methods are related in that they are learning from the similarity (contrastive, metric learning), shared information (mutual information), or compatibility (Energy-Based Models) between views or augmentations of the same input or output data.

Mutual information: For one, as explained in Section 4.2.1, InfoNCE (or Ranking NCE) has been shown to maximize the lower bound of mutual information (MI) between similar augmented and non-augmented inputs (van den Oord et al., 2018; Hjelm et al., 2019b), while works like (Boudiaf et al., 2020; Kong et al., 2020) show this MI perspective between inputs and labels or inputs and input sub-sequences. Importantly Tschannen et al. (2020) demonstrate that maximizing mutual information with contrastive losses can deteriorate end-task performance, and does not necessarily lead to learning useful representations. They also state that the “mutual information gets biased by the end-task objective, such that end task performance is maximized”. In contrast, when thinking of the recent successes with contrastive pretraining in computer vision (Chen et al., 2020b,c), it becomes apparent that the bias introduced via the sampling design, i.e. the input augmentations used to produce positive and negative samples, is the tool for introducing task biases that lead to learning relevant representations. For this reason, starting from Section 4.3, we will discuss contrastive works in the context of NLP subfields to provide pointers to what kinds of sampling and augmentations induce desirable biases that help a model learn NLP task-relevant representations.

Metric learning: Because contrastive methods learn classification over similarity scores between instances, it is a part of the more general field of metric learning. Metric learning uses losses like triplet loss, focal loss, Neighborhood Component Analysis, and many others to learn (dis-)similarities between inputs (Musgrave et al., 2020a). Interestingly, Musgrave et al. (2020a) find that even basic contrastive losses perform surprisingly well when fairly compared to advanced metric losses, while

Zimmermann et al. (2021a) point out that contrastive objectives are “theoretically more deeply understood than most metric losses”.

Energy-Based Models: Many recent works describe contrastive learning as Energy-Based Models (EBMs). LeCun and Huang (2005a); LeCun et al. (2006b) define an Energy Base Model $E(W, X, Y)$ as one that “instead of trying to classify inputs X to labels Y, we would like to predict if a certain pair of $<x, y>$ fit together or not under the model parameters W – i.e. find whether a y is compatible with x according to W. Especially LeCun and Huang (2005a) describes how an EBM E(X,Y,W) can be expressed in probabilistic model notation $P(Y|X, W)$ or as a non-normalized model, as we have already seen in Section 4.2.1 with Ranking NCE and Binary NCE. The modernized graphical and mathematical notion, as used in the excellent EBM lecture by LeCun and Canziani[2], still heavily relates to the ones used in (LeCun and Huang, 2005a; LeCun et al., 2006b).

Therefore, we overview the two most NLP-relevant EBM formulations and reuse not only their mathematical notation but also adopt the graphical notation from LeCun et al. (2006b) for figures. In keeping with this notion, we categorize methods as either input-input (x_i, x_j) or input-output (x_i, y_j), which also allows us to better discuss their respective benefits. Contrastive computer vision methods learn from input-input (image-image) pairs (x_i, x_j) (Jaiswal et al., 2021b; Chen et al., 2020c). As a result, using 10 samples incurs 10 times the computational load, which spawned efforts to reduce this load by reusing computation (He et al., 2020). In NLP, some recent methods reduce this burden by instead using input-output (text, label) pairs (x_i, y_c), where the labels are produced by a separate, very lightweight, encoder – see Section 4.2.2. Here x_i, x_j are input text embeddings, while y_c are embeddings of “a short text span that describes a real or self-supervision label”, i.e. an extreme summarization. This works as follows.

Input-Output Contrastive EBM: The binary NCE variant from Equation 4.1 is a special case of a “Contrastive Free Energy” loss as described in Fig. 6b of LeCun et al. (2006b), while Fig. 2 and Sec. 3.3 of LeCun and Huang (2005b) describe it as the negative log-likelihood loss with negative undersampling. LeCun et al. (2006b) devise an input-output EBM E variation that learns the compatibility between input-output pairs (x_i, y_c) with $x_i \in X$ and $y_c \in Y$

$$E(X, Y) \text{ or } E(W, X, Y) \tag{4.4}$$

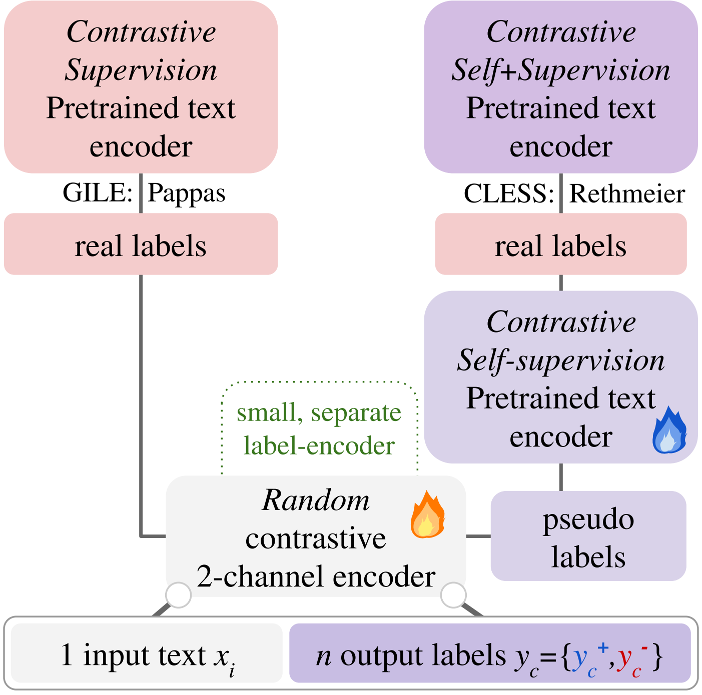
[2] `https://atcold.github.io/pytorch-Deep-Learning/en/week07/07-1/` – EBM definition by Yann LeCun and Alfredo Canziani.

Here, W (θ in Equation 4.1) are model parameters that encode inputs X and labels Y, while these X and Y are views of either the same data point (positives), or different data points (negatives). The energy function E measures the compatibility between views (X, Y), where $E(\circ){=}0$ indicates optimal compatibility – e.g. $E(X{=}Tiger, Y{=}felidae){=}0$ means X and Y match. Note that in the probabilistic framework $P(Y{=}felidae|X{=}Tiger, W){=}1$. Figure 4.2 shows two recent works (Pappas and Henderson, 2019; Rethmeier and Augenstein, 2020) that use an input-output contrastive learning approach. These methods encode an input text x_i using a text-encoder T and a label description text y_c using a very small, computationally cheap, label-encoder L. The input text and label text encoding is then concatenated into a single text input-output encoding pair $(T(x_i), L(y_c))$, which feeds a classifier that trains a binary NCE objective L_B, as in Equation 4.1. The left method in Figure 4.2 by Pappas and Henderson (2019) uses *supervised text-to-label pair pretraining* to allow zero-shot prediction of unseen test time classes. The right-hand side method (Rethmeier and Augenstein, 2020) instead samples input words $x_i \in X$ to use them as 'pseudo label' encodings $y_c'{=}L(x_i)$ for *contrastive self-supervised pretraining*. Once this method pretrains on sampled input words (pseudo labels), the prediction head is directly reused during supervision via textual labels. This enables zero-shot prediction, *without using supervision labels*, by unifying supervision and self-supervision as a single task of learning to contrast (mis)matching (real) label encodings $L(y_c)$ or pseudo label encodings $y_c'{=}L(x_i)$.

Input-Input Contrastive EBM: Expressing input-input contrastive learning as an EBM is straight forward (LeCun et al., 2006b). Input-input methods in Figure 4.3 contrast input texts X from augmented input texts X' rather than from labels Y as in Figure 4.2. For example, Clark et al. (2020) replace a subset of input text words $x_{i,w}$ with other words $x_{i,w'}$ sampled from the vocabulary for self-supervised contrastive pretraining. The original text x_i is augmented into a text a_i to provide a positive sample augment a_i^+ or a negative sample augment a_i^-. Self-supervised pretraining then contrasts pairs (x_i, a_i) of original texts against augmented ones via the binary NCE as in Equation 4.1. As a direct analog to the EBM in Equation 4.4, this can be written as

$$E(X, X') \text{ or } E(W, X, X') \tag{4.5}$$

Methods on the left in Figure 4.3 re-pretrain an already pretrained language model such as BERT, using a contrastive objective. Methods on the right implement what amounts to contrastive (self-supervised) language model pretraining. Though the



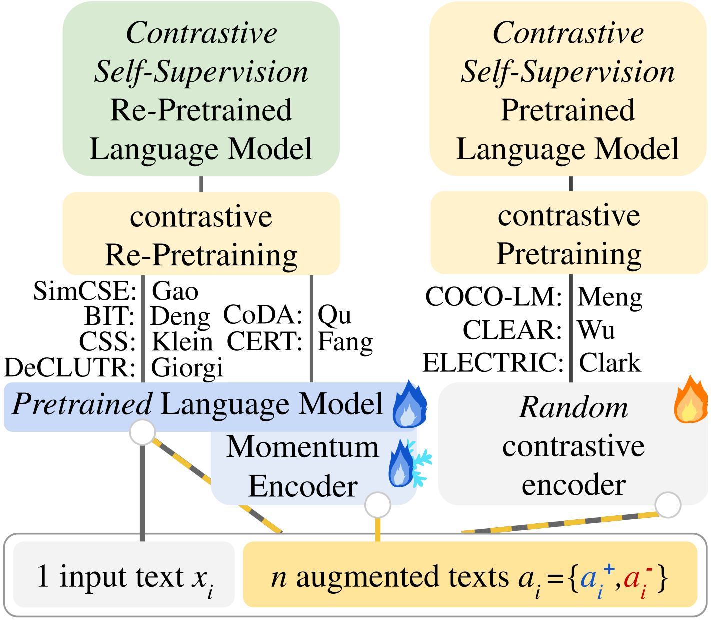
Contrastive Supervision Pretrained text encoder
GILE: Pappas
real labels
Contrastive Self+Supervision Pretrained text encoder
CLESS: Rethmeier
real labels
Contrastive Self-supervision Pretrained text encoder
small, separate label-encoder
Random contrastive 2-channel encoder
pseudo labels
1 input text x_i
n output labels $y_c=\{y_c^+, y_c^-\}$

Figure 4.2: **Contrastive input-output** (X, Y) **pretraining:** Texts and labels are encoded independently via a medium sized text encoder and a very small label-encoder. This encodes 1 text for n labels with minimal computation to enable large-scale K negative sampling.

difference between input-input and input-output contrast seems semantical at first, each approach implies specific practical benefits and drawbacks as follows.

Furthermore, input-input contrastive methods (EBM) such as the ones listed in Figure 4.3 recently explored improving large-scale NLP pretraining. Re-pretraining methods like Fang et al. (2020a); Deng et al. (2020); Giorgi et al. (2021a); Qu et al. (2021); Gao et al. (2021a) (see Figure 4.3 left tower) apply 'a second stage of contrastive pretraining' to an otherwise pretrained Transformer model, to save computation by contrastively training with less input augmentations. This can be seen as either making use of (an advantage) or requiring (a limitation) otherwise pretrained models. Other input-input contrastive methods such as Meng et al. (2021); Wu et al. (2020b); Clark et al. (2020) *do not rely* on otherwise pretrained encoders – see Figure 4.3 right tower. Additionally, works such as Radford et al. (2021) provide evidence that the same efficiency benefit of contrastive learners apply to large-scale models across modalities. For instance, Radford et al. (2021) replace a Transformer by a CNN to speed up self-supervised zero-shot prediction learning by a factor of 3, and add text contrastive pretraining to speed up learning by another factor of 4.

The Benefits and Weaknesses of Input-Output or Input-Input Contrastive Methods. *Input-Output contrastive methods* in Figure 4.2 are capable of zero-shot prediction



Contrastive Self-Supervision Re-Pretrained Language Model
Contrastive Self-Supervision Pretrained Language Model
contrastive Re-Pretraining
contrastive Pretraining
SimCSE: Gao
BIT: Deng
CSS: Klein
DeCLUTR: Giorgi
CoDA: Qu
CERT: Fang
COCO-LM: Meng
CLEAR: Wu
ELECTRIC: Clark
Pretrained Language Model
Momentum Encoder
Random contrastive encoder
1 input text x_i
n augmented texts $a_i = \{a_i^+, a_i^-\}$

Figure 4.3: **Contrastive input-input** (X, X') **Pretraining:** Input-input methods contrast an original text with augmented positive a_i^+ and negative a_i^- texts $a_i \in X'$, which requires more computation than input-output methods.

since they learn a pretraining NCE classifier, which can be reused or re-tuned to suit any downstream task labels, without having to initialize a new classifier per task – i.e. multi-task learning becomes single-task learning. As a contrastive analog to text-to-text Transformers like T5 (Raffel et al., 2020), they unify self and supervised prediction with zero-shot transfer as a 'text-to-text embedding similarity prediction' objective, whereas most, but not all, input-input methods, still have to initialize a new classifier for supervised downstream tasks. Input-output contrastive methods encode labels by using a small, compute efficient label-encoder, while encoding the more compute intensive text input X encoding only once. Input-input methods on the other hand construct 'augmented views' X' by running a large (Transformer) text encoder over an augmented version X' of the original input X. This multiplies their training time by the number $K+P$ of negative and positive samples and presents the most important challenge to the more wide-spread adaptation of contrastive pretraining. For this reason, input-input research often argues that fewer negative samples should be used. Instead, input-output contrastive self-supervision (Rethmeier and Augenstein, 2020) and contrastive supervision (Pappas and Henderson, 2019; Jiang et al., 2019a; Hardalov et al., 2021) enable very data-efficient pretraining and improved zero to few-shot, as well as long-tail learning. They can be pretrained on very small text collections with commodity hardware, which is very important for many applications in industry, medicine, and places where large amounts of GPU compute are less easy to attain. Thus, combining *highly expressive input-input with compute efficient input-output methods* provides a logical progression for future research.

Both input-input and input-output methods are well suited for self-supervised pretraining of language representations. These contrastively pretrained representations can subsequently be used for transfer learning of later supervised end-tasks during fine-tuning just as masked or autoregressive language model pretraining improves transfer. To make it easy to understand which contrastive methods have been explored within the different subfields of NLP we overview popular self-supervised and supervised contrastive NLP methods Section 4.3 and later point to open questions and opportunities Section 4.4.

4.3 Self- or Supervised Contrastive Pretraining

The goal of contrastive pretraining is to initialize model weights for efficient zero-shot transfer or fine-tuning to downstream tasks. Pretraining is either supervised or self-supervised. Supervised contrastive pretraining methods use corpora of hand-annotated data such as paraphrased parallel sentences, textual labels or text summarizations to define text data augmentations for contrastive pretraining. Self-supervised contrastive methods aim to scale pretraining by contrasting automatically augmented input texts X' or textual output pseudo-labels $Y'{\sim}P(X)$ – see Section 4.2.2 for input-input vs. input-output contrastive methods. Both self-supervised and supervised contrastive methods are used to train language models from scratch, or can 're-pretrain' or fine-tune a model that was already pretrained using another pretraining method, e.g. a masked language model such as RoBERTa (Liu et al., 2019b). Below, we structure self- and supervised contrastive pretraining by technique and application.

4.3.1 Self-supervised Contrastive Pretraining

Input-input Contrastive Text Representation Pretraining via Automated Text Augmentation. Figure 4.3 compares methods that use input-input contrastive (EBM) learning as overviewed in Section 4.2.2. Qu et al. (2021) use combinations of recently proposed text data augmentations like "cutoff, back translation, adversarial augmentation and mixup". They find that mixing augmentations is most useful when the augmentations provide sufficiently different views of the data. Further, since constructing text augmentations which do not alter the meaning (semantics) of a sentence is very difficult, they introduce two losses to ensure both sufficient difference and semantic consistency of sentence augmentations. They define a consistency loss to guarantee that augmentations lead to similar predictions y_c and a contrastive loss that makes augmented text representations a_i similar to the original text x_i. To ensure that a sufficiently large amount of negative text augmentations are sampled, they use an augmentation-embedding memory bank with a momentum encoder. Fang et al. (2020a) only use back-translation, Wu et al. (2020b); Meng et al. (2021) investigate sentence augmentation methods, Giorgi et al. (2021a) contrast text spans, Clark et al. (2020); Meng et al. (2021) replace input words by re-sampling a language model, (Gao et al., 2021a) sample positive text embeddings via dropout, and Simoulin and Crabbé (2021) investigate contrastive sentence structure pretraining. Finally, Meng et al. (2021) also contrasts cropped sentences after augmentation via word re-sampling.

Contrasting Next or Surrounding Sentence (or Word) Prediction (NSP, SSP). Sentence prediction is a popular input-input contrastive method as in Section 4.2.2. Next sentence prediction, NSP, and surrounding sentence prediction, SSP, take inspiration from the skip-gram model (Mikolov et al., 2013c), where surrounding and non-surrounding words are contrastively predicted given a central word to learn word embeddings using an NCE Section 4.2.1 variant (Mikolov et al., 2013c). Methods mostly differ in how they sample positive and negative sentences, where negative sampling strategies such as undersampling frequent words, in Mikolov et al. (2013a), are crucial. Logeswaran and Lee (2018) propose contrastive NSP, to predict the next sentence as a positive sample against n random negative sample sentences. Instead of generating the next sentence, they learn to discriminate which sentence encoding follows a given sentence. This allows them to train a better text encoder model with less computation but sacrifices the ability to generate text. Liu et al. (2019b) investigate variations of the contrastive NSP objective used in the BERT model. The method contrasts a consecutive sentence as a positive text sample against multiple non-consecutive sentences from other documents as negative text samples. They find that sampling negatives from the same document during self-supervised BERT pretraining is critical to downstream performance, but that removing the original BERT NSP task improves downstream performance. Iter et al. (2020) find that predicting surrounding sentences in a k-sized window around a given central anchor sentence “improves discourse performance of language models”. They sample surrounding sentences: (a) randomly from the corpus to construct easy negatives, and (b) from the same paragraph, but outside the context window as hard (to contrast) negative samples. Contextual negative sampling is theoretically and empirically proven by Saunshi et al. (2019), who demonstrate that: “increased negative sampling only helps if negatives are taken from the original texts’ context or block of information”, i.e. the same document, paragraph, or sentence. Aroca-Ouellette and Rudzicz (2020) study how to combine different variants of the NSP pretraining tasks with non-contrastive, auxiliary self-supervision signals, while Simoulin and Crabbé (2021) explore contrastive sentence structure learning.

Input-output Contrastive Text Representation Pretraining. Rethmeier and Augenstein (2020) use output label embeddings as an alternative view Y (labels) of text input embeddings X for contrastive learning of (dis)-similar text-label embedding pairs (X, Y) via binary NCE from Section 4.2.1. Using a separate label and text encoder enables the model to efficiently compute many negative label samples, while encoding the text X only once, unlike input-input view methods in Figure 4.3. They pretrain with random input words as pseudo-labels for self-supervised pretraining on a very small corpus, which despite the limited pretraining data enables unsupervised zero-shot prediction, largely improved few-shot and markedly better rare concept (long-tail) learning.

Contrastive Distillation. Sun et al. (2020) propose CoDIR, a contrastive language model distillation method to pretrain a smaller student model from an already pretrained larger teacher such as a Masked Transformer Language Model. Compressing a pretrained language model is challenging because nuances such as interactions between the original layer representation are easily lost – without noticing. For distillation, they extract layer representations from both the large teacher and the small student network over the same or two different input texts, to create a student and teacher view of said texts. Using the constrastive InfoNCE loss (van den Oord et al., 2018), they then learn to make the student representation similar to teacher representations for the same input texts, and dissimilar if they receive different texts. The score or similarity function in InfoNCE is measured as the cosine distance between mean pooled student and teacher Transformer layer representations. For negative sampling in pretraining, they use text inputs from the same topic, e.g. a Wikipedia article, to mine hard negative samples – i.e. they sample views from similar contexts as recommended for contrastive methods in (Saunshi et al., 2019).

Text Generation as a Discriminative EBM. Deng et al. (2020) combine an auto-regressive language model, with a contrastive text continuation EBM model for improved text generation. During pretraining, they learn to contrast real data text continuations and language model generated text continuations via conditional NCE from Section 4.2.1. For generation, they sample the top-k text completions from the auto-regressive language model and then score the best continuation via the trained EBM, to markedly improve model perplexity. However, the current approach is computationally expensive.

Cross-modal Contrastive Representation Pretraining. Representations for zero-shot image classification can be pretrained using image caption text for contrastive self-supervised pretraining. Jia et al. (2021) automatically mine a large amount of noisy text captions for images in ALIGN, to then noise-filter and use them to construct matching and mismatching pairs of image and augmented text captions for contrastive training. Radford et al. (2021) use the same idea in CLIP, but pretrain on a large collection of well annotated image caption datasets. Both methods allow for zero-shot image classification and image-to-text or text-to-image generation and are inherently zero-shot capable. Radford et al. (2021) also run a zero-shot learning efficiency analysis for CLIP and find two things. First, they find that using a data-efficient CNN text encoder increases zero-shot image prediction convergence 3-fold compared to a Transformer text encoder, which they state to be computationally prohibitive. Second, they find that adding contrastive self-supervised text pretraining increases zero-shot image classification performance 4-fold. Thus, CLIP (Radford et al., 2021) shows that contrastive self-supervised CNN text encoder pretraining can substantially outperform current Transformer pretraining methods, while ALIGN (Jia et al., 2021) also automates the image and caption data collection process to increase data scalability.

Vision-language grounding Shi et al. (2021) show that contrastive RoI-Feature prediction pretraining can increase performance on vision language grounding tasks compared to self-supervised masked token prediction or image-sentence matching predictions, especially when there is a gap between the pretraining domain and end-task domain. Akula et al. (2020) use a crowdsourced dataset to create a harder vision-language grounding task for robustness tests against common models and find that a contrastive method they design does increase performance, but is outperformed by a multi-task learning approach. Since contrastive losses have been shown to not overfit random labels (robustness) by Graf et al. (2021a), it is not clear whether there may be a label noise problem due to the crowdsourced nature of the data.

4.3.2 Supervised Contrastive Pretraining

Input-output Contrastive Supervised Text Representation Pretraining. Seen in Figure 4.2, Pappas and Henderson (2019) train a two-input-lane Siamese CNN network, which encodes text as the input view x_i in one lane, and labels via a label encoder in a second data view y_c, to learn to contrast pairs of (x_i, y_x) as similar (1) or not (0). Rather than encoding labels as multi-hot vectors such as $[0, 1, 0, 0, 1]$, they express each label by a textual description of said label. These textual label descriptions can then be encoded by a label encoder subnetwork, which in the simplest case constructs a label embedding by averaging over the word embeddings of the words that describe a label. However, this requires manually describing each label. Using embeddings of supervised labels, they pretrain a contrastive text classification network on known positive and negative labels, and later apply the pretrained network to unseen classes for zero-shot prediction. Their method thus provides supervised, but zero-shot capable pretraining. While Rethmeier and Augenstein (2020) also support supervised contrastive input-output pretraining, they automate label descriptions construction, and conjecture that in real-world scenarios, most labels, e.g. the word 'elephant', are already part of the input vocabulary and can thus be pretrained as word embeddings via methods such as Word2Vec (Mikolov et al., 2013a). They also note that: "once input words are labels, one can sample input words as pseudo label embeddings for contrastive self-supervised pretraining", as described in section Section 4.3.1. Either method is contrastively pretrained via binary NCE as described in Section 4.2.1. Furthermore, both methods markedly boost few-shot learning and enable zero-shot predictions, while Rethmeier and Augenstein (2020) enables unsupervised zero-shot learning via self-supervised contrastive pretraining. The added contrastive self-supervision further boosts few-shot and long-tailed learning performance, while also increasing convergence speed over supervised-only contrastive learning in Pappas and Henderson (2019).

Contrastive Commonsense Pretraining. (Klein and Nabi, 2020) use contrastive self-supervised pretraining to refine a pretrained BERT language model to drastically increase performance on pronoun disambiguation and the Winograd Schema Commonsense Reasoning task. Their method contrasts over candidate trigger words that affect which word a pronoun refers to. They first mine trigger word candidates from text differences in paraphrased sentences and then maximize the contrastive margin between candidate pair likelihoods. While general pretraining provides little pronoun disambiguation learning signal, their method demonstrates the design of task-specific contrastive learning to produce strong performance increases in *un- and supervised commonsense reasoning*.

Contrastive Text Summarization. Duan et al. (2019) use a Transformer attention mechanism during abstractive sentence summarization learning to optimize two contrasting loss objectives. One loss maximizes the contributions of tokens with the most attention when predicting the summarized sentence. The other loss is connected to a second decoder head, which learns to minimize the contribution of the attention to other, non-summarization relevant, tokens. This method can perhaps best be understood as contrastive, layer attention noise reduction. The main draw back of this method is the current dual network head prediction, which introduces a larger complexity compared to other contrastive methods.

Cross and Multi-modal Supervised Contrastive Text Pretraining for Representation Learning. Recent work from computer vision and time series prediction train with contrastive supervised losses to enable zero-shot learning or improve data-to-text generation. Jiang et al. (2019a) fuse image and text description information into the same representation space for generalized zero-shot learning – i.e. where at test time some classes are unseen, zero-shot, while other classes were seen during training. To do so, they first pretrain a supervised text-image encoder network to contrast $(image, text, label)$ triplets of human annotated image classes. At test time, this contrastive network decides which text description best matches a given image. This works for seen and unseen classes, because classes are represented as text descriptions. Radford et al. (2021); Li et al. (2021) perform pretraining on manually annotated textual image descriptions to enable better generalization to unseen image classes. Uehara et al. (2020) turn stock price value time series into textual stock change descriptions where the contrastive objectives markedly increase the fluency and non-repetitiveness of generated texts, especially when trained with little data.

Datasets Construction for Contrastive pretraining. Raganato et al. (2019) automatically create a corpus of contrastive sentences for word sense disambiguation in machine translation by first identifying sense ambiguous source sentence words, and then creating replacement word candidates to mine sentences for contrastive evaluation.

4.4 Challenges and Future Opportunities:

Here we outline current challenges and promising next steps towards improving contrastive NLP, as well as how its learning properties may help better handle issues like data efficiency or algorithmic bias.

Data Efficiency, Fairness and Small-scale Pretraining. Zimmermann et al. (2021a) proved that contrastive methods effectively recover data properties even from very limited data, which explains their few-shot label efficiency in both supervised contrastive fine-tuning (Gunel et al., 2021) and contrastive re-pretraining of pretrained language model (Fang et al., 2020a; Iter et al., 2020; Su et al., 2021). Additionally, contrastive language models like Clark et al. (2020); Wu et al. (2020b); Rethmeier and Augenstein (2020); Meng et al. (2021) do not require other pretrained models, while largely improving pretraining data efficiency (zero-shot learning) or label efficiency (few-shot learning). For example, Rethmeier and Augenstein (2020) propose a small contrastive language model to markedly improve *long-tail learning* over large pretrained language models. Contrastive modeling thus provides a promising direction to reducing algorithmic fairness issues that have been linked to a loss of minority (tail) information by Hooker et al. (2020a). These aspects indicate that contrastive self-supervised models require far less pretraining data than other objectives, which opens their applications to data sparse domains, languages, productivity gains, and scalable or budget friendly language model pretraining.

Better Negative and Positive Generation: Current methods require sampling many negative instances for contrastive learning to work well. Sampling hard (Cai et al., 2020), or context relevant negatives (Saunshi et al., 2019) are known to boost sample efficiency during contrastive self-supervised learning, while sampling diverse negatives (Mikolov et al., 2013c; Musgrave et al., 2020a; Rethmeier and Augenstein, 2020) have been demonstrated to improve generalization in open class set (or open tasks) applications such as pretraining. Sampling multiple positives for supervised contrast is common in multi-label metric learning and therefore somewhat studied. However, explicit experiments on the benefits of generating multiple positive samples for *contrastive self-supervision* are poorly understood, especially in NLP. To date, Wang and Isola (2020) showed that positive self-supervision samples should be sampled close to one another (in computer vision), while sampling more contextual positives has been linked to largely improved language model pretraining sample efficiency gains in Rethmeier and Augenstein (2020). Works like BYOL (Grill et al., 2020) or Barlow Twins (Zbontar et al., 2021) do not require negative sampling. Their momentum contrast or redundancy reduction based learning, may be adapted for contrastive language modeling to overcome current compute challenges of input-input contrastive NLP.

Self-supervised text augmentation research in NLP (Section 4.3.1) is gaining momentum and Qu et al. (2021); Chen et al. (2020a) and many others analyze using mixes of recent text data augmentations. However, these input-input contrastive methods often use computationally expensive or non-robust mechanisms like: back translation, initializing a new prediction head per downstream task, or rely on already otherwise pretrained models like RoBERTa. Fortunately, more scalable and robust input augmentations have already been proposed by Wu et al. (2020b); Iter et al. (2020), which is a promising step to cost effective future extensions.

Data Limited NLP Sub-fields: (Chi et al., 2021) use contrastive pretraining to reduce data limitations in multi-lingual models, while Jiang et al. (2020b) use adversarial sample generation to make contrastive pretraining more sample efficient and robust. The contrastive word sense disambiguation (WSD) dataset construction method by Raganato et al. (2019) is potentially adaptable to automatically mine inputs for contrastive pronoun learning in Klein and Nabi (2020). Such automation would help to scale contrastive common sense learning.

Underresearched NLP applications: An underresearched direction for contrastive NLP are data-to-text tasks that turn non-text inputs into a textual description. Uehara et al. (2020), for instance, contrastively learn to generate stock change text descriptions from stock price time series using limited data, while works such as Radford et al. (2021); Jia et al. (2021) show that contrastive text supervision and self-supervision can multiply the zero-shot learning efficiency in cross-modal representation learning. Deng et al. (2020) improve text generation with contrastive importance resampling of language model generated text continuations, while Duan et al. (2019) propose contrastive abstractive sentence summarization, which using Momentum Contrast can potentially improve. Sun et al. (2020) compress a large language model. Future work could adapt their method to fuse multiple language models or mutually transfer knowledge between models. Jiang et al. (2021) present long-tail preserving vision model compression by contrasting easily pruned (forgotten) model information with large model information. Together with Wu et al. (2020b); Rethmeier and Augenstein (2020) this may be used to learn and compress large long-tail capable language models that retain more tail class information to reduce compression-induced minority fairness losses as first identified by Hooker et al. (2020a).

XAI and active learning: Works like Luss et al. (2021) and Ross et al. (2021) generate contrastive (counterfactual) explanations in vision or NLP, because humans give contrastive explanations. This could be used to ease the creation of semantically sensible input augmentations, which are used as negative or positive samples for contrastive learning. This would result in an optimization loop between explanation and contrast pair generation that amounts to energy minimization towards an equilibrium. Additionally, human annotations could be incorporated for data-efficient, explanation guided human-in-the-loop learning.

4.5 Conclusion

This primer on contrastive pretraining, surveys contrastive learning concepts and their relations to other sub-fields like EBMs to ease advanced reading into the connected literature. It highlights recent methodological and theoretical insights that are important to designing effective contrastive learners for NLP. Finally, the primer structures contrastive pretraining as self- vs. supervised learning summarises challenges, and provides pointers to future research directions.

5 Paper 4: Neighborhood Contrastive Learning for Scientific Document Representations with Citation Embeddings

5.1 Introduction

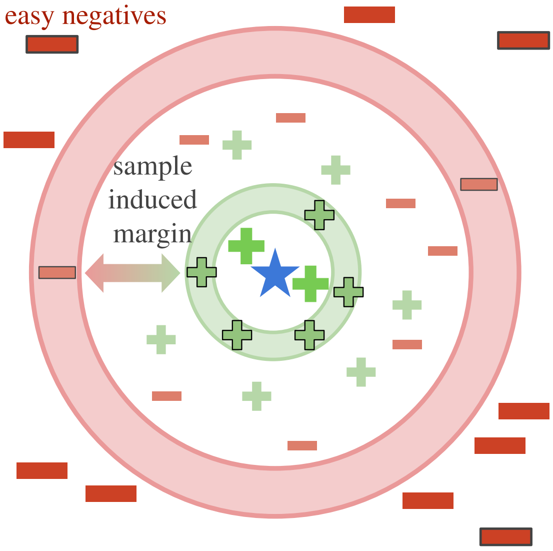


Figure 5.1: Starting from a query paper ★ in a citation graph embedding space. Hard positives ✚ are citation graph embeddings that are sampled from a similar (close) context of ★, but are not so close that their gradients collapse easily. Hard (to classify) negatives ▭ (red band) are close to positives (green band) up to a *sampling induced margin*. Easy negatives ▬ are very dissimilar (distant) from the query paper ★.

Large pretrained language models (LLMs) achieve state-of-the-art results through fine-tuning on many NLP tasks (Rogers et al., 2020). However, the sentence or document embeddings derived from LLMs are of lesser quality compared to simple baselines like GloVe (Reimers and Gurevych, 2019b), as their embedding space suffers from being anisotropic, i.e. poorly defined in some areas (Li et al., 2020).

One approach that has recently gained attention is the combination of LLMs with contrastive fine-tuning to improve the semantic textual similarity between document representations (Wu et al., 2020c; Gao et al., 2021b). These contrastive methods learn to distinguish between pairs of similar and dissimilar texts (positive and negative samples). As recent works show (Tian et al., 2020b; Rethmeier and Augenstein, 2022d,b; Shorten et al., 2021), the selection of these positive and negative samples is crucial for efficient contrastive learning.

This paper focusses on learning scientific document representations (SDRs). The core distinguishing feature of this domain is the presence of citation information that complement the textual information. The current state-of-the-art SPECTER by Cohan et al. (2020a) uses citation information to generate positive and negative samples for contrastive fine-tuning of a SciBERT language model (Beltagy et al., 2019). SPECTER relies on 'citations by the query paper' as a discrete signal for similarity, i.e., positive samples are cited by the query while negative ones are not cited.

However, SPECTER's use of citations has its pitfalls. Considering only one citation direction may cause positive and negative samples to collide since a paper pair could be treated as a positive and negative instance simultaneously. Also, relying on a single citation as a discrete similarity signal is subject to noise, e.g., citations may reflect politeness and policy rather than semantic similarity (Pasternack, 1969) or related papers lack a direct citation (Gipp and Beel, 2009). This discrete cut-off to similarity is counter-intuitive to (continuous) similarity-based learning.

Instead, the generation of non-colliding contrastive samples should be based on a continuous similarity function that allows us to find semantically similar papers, even without direct citations. With SciNCL, we address these issues by generating contrastive samples based on citation embeddings. The citation embeddings, which incorporate the full citation graph, provide a continuous, undirected, and less noisy similarity signal that allows the generations of arbitrary difficult-to-learn positive and negative samples.

Contributions:

- We propose neighborhood contrastive learning for scientific document representations with citation graph embeddings (SciNCL) based on contrastive learning theory insights.
- We sample positive (similar) and negative (dissimilar) papers from the k nearest neighbors in the citation graph embedding space, such that positives and negatives do not collide but are also hard to learn.

- We compare against the state-of-the-art approach SPECTER (Cohan et al., 2020a) and other strong methods on the SciDocs benchmark and find that SciNCL outperforms SPECTER on average and on 9 of 12 metrics.
- Finally, we demonstrate that with SciNCL, using only 1% of the triplets for training, starting with a general-domain language model, or training only the bias terms of the model is sufficient to outperform the baselines.
- Our code and models are publicly available.[1]

5.2 Related Work

Contrastive Learning pulls representations of similar data points (positives) closer together, while representations of dissimilar documents (negatives) are pushed apart. A common contrastive objective is the triplet loss (Schroff et al., 2015) that Cohan et al. (2020a) used for scientific document representation learning, as we describe below. However, as Musgrave et al. (2020b) and Rethmeier and Augenstein (2022d) point out, contrastive objectives work best when specific requirements are respected. **(Req. 1)** Views of the same data should introduce new information, i.e. the mutual information between views should be minimized (Tian et al., 2020b). We use citation graph embeddings to generate contrast label information that supplements text-based similarity. **(Req. 2)** For training time and sample efficiency, negative samples should be hard to classify, but should also not collide with positives (Saunshi et al., 2019). **(Req. 3)** Recent works like Musgrave et al. (2020b) and Khosla et al. (2020c) use multiple positives. However, positives need to be consistently close to each other (Wang and Isola, 2020), since positives and negatives may otherwise collide, e.g., Cohan et al. (2020a) consider only 'citations by the query' as similarity signal and not 'citations to the query'. Such unidirectional similarity does not guarantee that a negative paper (not cited by the query) may cite the query paper and thus could cause collisions, the more we sample (Appendix 5.10.6.10). Our method treats both citing and being cited as positives (Req. 2), while it also generates hard negatives and hard positives (Req. 2+3). Hard negatives are close to but do not overlap positives (red band in Figure 5.1). Hard positives are close, but not trivially close to the query document (green band in Figure 5.1). The sample induced margin (space between red and green band in Figure 5.1) ensures that contrast samples do not collide.

Triplet Mining remains a challenge in NLP due to the discrete nature of language which makes data augmentation less trivial as compared to computer vision (Gao et al., 2021b). Examples for augmentation strategies are translation, word deletion, or word reordering (Fang et al., 2020b; Wu et al., 2020c). Positives and negatives can

[1] https://github.com/malteos/scincl

be sampled based on the sentence position within a document (Giorgi et al., 2021b). Gao et al. (2021b) utilize supervised entailment datasets for the triplet generation. Language- and text-independent approaches are also applied. Kim et al. (2021b) use intermediate BERT hidden state for positive sampling and Wu et al. (2021) add noise to representations to obtain negative samples. Xiong et al. (2020) present an approach similar to SciNCL where they sample hard negatives from the k nearest neighbors in the embedding space derived from the previous model checkpoint. While Xiong et al. rely only on textual data, SciNCL integrates also citation information which are especially valuable in the scientific context as Cohan et al. (2020a) have shown.

Scientific Document Representations based on Transformers (Vaswani et al., 2017c) and pretrained on domain-specific text dominate today's scientific document processing. There are SciBERT (Beltagy et al., 2019), BioBERT (Lee et al., 2019) and SciGPT2 (Luu et al., 2021), to name a few. Recent works modify these domain LLMs to support cite-worthiness detection (Wright and Augenstein, 2021), document similarity (Ostendorff et al., 2020) or fact checking (Wadden et al., 2020).

Aside from text, citations are a valuable signal for the similarity of research papers. Paper (node) representations can be learned using the citation graph (Wu et al., 2019; Perozzi et al., 2014; Grover and Leskovec, 2016). Especially for recommendations of papers or citations, hybrid combinations of text and citation features are often employed (Han et al., 2018; Jeong et al., 2020; Brochier et al., 2019; Yang et al., 2015; Holm et al., 2022).

Closest to SciNCL are Citeomatic (Bhagavatula et al., 2018) and SPECTER (Cohan et al., 2020a). While Citeomatic relies on bag-of-words for its textual features, SPECTER is based on SciBERT. Both leverage citations to learn a triplet-based document embedding model, whereby positive samples are papers cited in the query. Easy negatives are random papers not cited by the query. Hard negatives are citations of citations – papers referenced in positive citations of the query, but are not cited directly by it. Citeomatic also uses a second type of hard negatives, which are the nearest neighbors of a query that are not cited by it.

Unlike our approach, Citeomatic does not use the neighborhood of citation embeddings, but instead relies on the actual document embeddings from the previous epoch. Despite being related to SciNCL, the sampling approaches employed in Citeomatic and SPECTER do not account for the pitfalls of using discrete citations as signal for paper similarity. Our work addresses this issue.

Cross-Modal Transfer. SciNCL transfers knowledge across modalities, i.e., from citations into a language model. According to Cohan et al. (2020a), SciNCL can be considered as a "*citation-informed Transformer*". This cross-modal transfer learning is applied for various modalities (see Kaur et al. (2021) for an overview): text-to-image

(Socher et al., 2013), RGB-to-depth image (Tian et al., 2020a), or graph-to-image (Wang et al., 2018). While the aforementioned methods incorporate cross-modal knowledge through joint loss functions or latent representations, SciNCL transfers knowledge through the contrastive sample selection, which we found superior to the direct transfer approach Appendix 5.10.6.9.

5.3 Methodology

Our goal is to learn citation-informed representations for scientific documents. To do so we sample three document representation vectors and learn their similarity. For a given query paper vector $\mathbf{d}^Q$, we sample a positive (similar) paper vector $\mathbf{d}^+$ and a negative (dissimilar) paper vector $\mathbf{d}^-$. This produces a 'query, positive, negative' triplet $(\mathbf{d}^Q, \mathbf{d}^+, \mathbf{d}^-)$ – represented by (★,✚,▬) in Figure 5.1. To learn paper similarity, we need to define three components: (Section 5.3.1) how to calculate document vectors $\mathbf{d}$ for the loss over triplets $\mathcal{L}$; (Section 5.3.2) how citations provide similarity between papers; and (Section 5.3.3) how negative and positive papers $(\mathbf{d}^-, \mathbf{d}^+)$ are sampled as (dis-)similar documents from the neighborhood of a query paper $\mathbf{d}^Q$.

5.3.1 Contrastive Learning Objective

Given the textual content of a document d (paper), the goal is to derive a dense vector representation $\mathbf{d}$ that best encodes the document information and can be used in downstream tasks. A Transformer language model f (SciBERT; Beltagy et al. (2019)) encodes documents d into vector representations $f(d) = \mathbf{d}$. The input to the language model is the title and abstract separated by the `[SEP]` token.[2] The final layer hidden state of the `[CLS]` token is then used as a document representation $f(d) = \mathbf{d}$.

Training with a masked language modeling objectives alone has been shown to produce sub-optimal document representations (Li et al., 2020; Gao et al., 2021b). Thus, similar to the SDR state-of-the-art method SPECTER (Cohan et al., 2020a), we continue training the SciBERT model (Beltagy et al., 2019) using a self-supervised triplet margin loss (Schroff et al., 2015):

$$\mathcal{L} = \max\left\{\|\mathbf{d}^Q - \mathbf{d}^+\|_2 - \|\mathbf{d}^Q - \mathbf{d}^-\|_2 + \xi, 0\right\}$$

Here, ξ is a slack term ($\xi = 1$ as in SPECTER) and $\|\Delta\mathbf{d}\|_2$ is the L^2 norm, used as a distance function. However, the SPECTER sampling method has significant drawbacks. We will describe these issues and our contrastive learning theory guided improvements in detail below in Section 5.3.2.

[2] Cohan et al. (2019) evaluated other inputs (venue or author) but found the title and abstract to perform best.

5.3.2 Citation Neighborhood Sampling

Compared to the textual content of a paper, citations provide an outside view on a paper and its relation to the scientific literature (Elkiss et al., 2008), which is why citations are traditionally used as a similarity measure in library science (Kessler, 1963; Small, 1973). However, using citations as a discrete similarity signal, as done in Cohan et al. (2020a), has its pitfalls. Their method defines papers cited by the query as positives, while paper citing the query could be treated as negatives. This means that *positive and negative learning information collides* between citation directions, which Saunshi et al. (2019) have shown to deteriorate performance. Furthermore, a cited paper can have a low similarity with the citing paper given the many motivations a citation can have (Teufel et al., 2006). Likewise, a similar paper might not be cited.

To overcome these limitations, we learn citation embeddings first and then use the citation neighborhood around a given query paper d^Q to construct similar (positive) and dissimilar (negative) samples for contrast by using the k nearest neighbors. This builds on the intuition that nodes connected by edges should be close to each other in the embedding space (Perozzi et al., 2014). Using citation embeddings allows us to: (1) sample paper similarity on a continuous scale, which makes it possible to: (2) define hard to learn positives, as well as (3) hard or easy to learn negatives. Points (2-3) are important in making contrastive learning efficient as will describe below in Section 5.3.3.

5.3.3 Positives and Negatives Sampling

Positive samples: d^+ should be semantically similar to the query paper d^Q, i.e. sampled close to the query embedding $\mathbf{d}^Q$. Additionally, as Wang and Isola (2020) find, positives should be sampled from comparable locations (distances from the query) in embedding space and be dissimilar enough from the query embedding, to avoid gradient collapse (zero gradients). Therefore, we sample c^+ positive (similar) papers from a close neighborhood around query embedding $\mathbf{d}^Q$ $(k^+ - c^+, k^+]$, i.e. the green band in Figure 5.1. When sampling with kNN search, we use a small k^+ to find positives and later analyze the impact of k^+ in Figure 5.2.

Negative samples: can be divided into easy ▬ and hard ▭ negative samples (light and dark red in Figure 5.1). Sampling more hard negatives is known to improve contrastive learning (Bucher et al., 2016; Wu et al., 2017). However, we make sure to sample hard negatives (red band in Figure 5.1) such that they are close to potential positives but do not collide with positives (green band), by using a tunable 'sampling induced margin'. We do so, since Saunshi et al. (2019) showed that sampling a larger number of hard negatives only improves performance *if the negatives do not collide* *with positive samples*, since collisions make the learning signal noisy. That is, in the margin between hard negatives and positives we expect positives and negatives to collide, thus we avoid sampling from this region. To generate a diverse self-supervised citation similarity signal for contrastive SDR learning, we also sample easy negatives that are farther from the query than hard negatives. For negatives, the k^- should be large when sampling via KNN to ensure samples are dissimilar from the query paper.

5.3.4 Sampling Strategies

As described in Figure 5.3.2 and Figure 5.3.3, our approach improves upon the method by Cohan et al. (2020a). Therefore, we reuse their sampling parameters (5 triplets per query paper) and then further optimize our methods' hyperparameters. For example, to train the triplet loss, we generate the same amount of $(\mathbf{d}^Q, \mathbf{d}^+, \mathbf{d}^-)$ triplets per query paper as SPECTER (Cohan et al., 2020a). To be precise, this means we generate c^+=5 positives (as explained in 5.3.3). We also generate 5 negatives, three easy negatives c^-_{easy}=3 and two hard negatives c^-_{hard}=2, as described in Section 5.3.3.

Below, we describe three strategies (I-III) for sampling triplets. These either sample neighboring papers from citation embeddings (I), by random sampling (II), or using both strategies (III). For each strategy, let c' be the number of samples for either positives c^+, easy negatives c^-_{easy}, or hard negatives c^-_{hard}.

Citation Graph Embeddings: We train a graph embedding model f_c on citations extracted from the Semantic Scholar Open Research Corpus (S2ORC; Lo et al., 2020) to get citation embeddings C. We utilize PyTorch BigGraph (Lerer et al., 2019), which allows for training on large graphs with modest hardware requirements. The resulting graph embeddings perform well using the default training settings from Lerer et al. (2019), but given more computational resources, careful tuning may produce even better-performing embeddings. Nonetheless, we conducted a narrow parameter search based on link prediction – see Appendix 5.10.4.

(I) K-nearest neighbors (KNN): Assuming a given citation embedding model f_c and a search index (e.g., FAISS, Section5.4.3), we run $KNN(f_c(d^Q), C)$ and take c' samples from a range of the $(k - c', k]$ nearest neighbors around the query paper d^Q with its neighbors N=$\{n_1, n_2, n_3, \ldots\}$, whereby neighbor n_i is the i-th nearest neighbor in the citation embedding space. For instance, for c'=3 and k=10 the corresponding samples would be the three neighbors descending from the tenth neighbor: n_8, n_9, and n_{10}. To reduce computing effort, we sample the neighbors N only once via $[0; \max(k^+, k^-_{\text{hard}})]$, and then generate triplets by range-selection in N; i.e. positives $= (k^+ - c^+; k^+]$, and hard negatives $= (k^-_{\text{hard}} - c^-_{\text{hard}}; k^-_{\text{hard}}]$.

(II) Random sampling: Sample any c' papers without replacement from the corpus.

(III) Filtered random: Like (II) but excluding the papers that are retrieved by kNN, i.e., all neighbors within the largest k are excluded. This is analog to SPECTER's approach of selecting random candidates that are not cited by the query.

The kNN sampling introduces the hyperparameter k that allows for the *controlled sampling of positives or negatives* with different difficulty (from easy to hard depending on k). Specifically, in Figure 5.1 the hyperparameter k defines the tunable *sample induced margin* between positives and negatives, as well as the width and position of the positive sample band (green) and negative sample band (red) around the query sample. Besides the strategies above, we experiment with similarity threshold, k-means clustering and sorted random sampling, neither of which performs well (Appendix 5.10.6).

5.4 Experiments

In the following, we introduce our experiments including the data sets and implementation details.

5.4.1 Evaluation Dataset

We evaluate on the SciDocs benchmark (Cohan et al., 2020a). A key difference to other benchmarks is that embeddings are the input to the individual tasks without explicit fine-tuning. The SciDocs benchmark consists of the following four tasks:

Document classification (CLS) with Medical Subject Headings (MeSH) (Lipscomb, 2000) and Microsoft Academic Graph labels (MAG) (Sinha et al., 2015). **Co-views and co-reads** (USR) prediction based on the L2 distance between embeddings. **Direct and co-citation** (CITE) prediction based on the L2 distance between the embeddings. **Recommendations** (REC) generation based on embeddings and paper metadata.

5.4.2 Training Datasets

The experiments mainly compare SciNCL against SPECTER on the SciDocs benchmark. However, we found 40.5% of SciDocs's papers leaking into SPECTER's training data (the leakage affects only the unsupervised paper data but not the gold labels – see Appendix 5.10.2. To be transparent about this leakage, we train SciNCL on two datasets:

SPECTER replication (w/ leakage): We replicate SPECTER's training data including its leakage. Unfortunately, SPECTER provides neither citation data nor a mapping to S2ORC, which our citation embeddings are based on. We successfully map 96.2% of

SPECTER's query papers and 83.3% of the corpus from which positives and negatives are sampled to S2ORC. To account for the missing papers, we randomly sample papers from S2ORC (without the SciDocs papers) such that the absolute number of papers is identical with SPECTER.

S2ORC subset (w/o leakage): We select a random subset from S2ORC that does not contain any of the mapped SciDocs papers. This avoids SPECTER's leakage, but also makes the scores reported in Cohan et al. (2020a) less comparable. We successfully map 98.6% of the SciDocs papers to S2ORC. Thus, only the remaining 1.4% of the SciDocs papers could leak into this training set.

The details of the dataset creation is described in Appendix 5.10.1 and 5.10.3. Both training sets yield 684K triplets (same count as SPECTER). Also, the ratio of training triplets per query remains the same (Section 5.3.4). Our citation embedding model is trained on the S2ORC citation graph. In *w/ leakage*, we include all SPECTER papers even if they are part of SciDocs, the remaining SciDocs papers are excluded (52.5 nodes and 463M edges). In *w/o leakage*, all mapped SciDocs papers are excluded (52.4M nodes and 447M edges) such that we avoid leakage also for the citation embedding model.

5.4.3 Model Training and Implementation

We replicate the training setup from SPECTER as closely as possible. We implement SciNCL using Huggingface Transformers (Wolf et al., 2020), initialize the model with SciBERT's weights (Beltagy et al., 2019), and train via the triplet loss (Equation 5.3.1). The optimizer is Adam with weight decay (Kingma and Ba, 2015; Loshchilov and Hutter, 2019) and learning rate $\lambda{=}2^{-5}$. To explore the effect of computing efficient fine-tuning we also train a BitFit model (Ben Zaken et al., 2022b) with $\lambda{=}1^{-4}$ (Section 5.7.2). We train SciNCL on two NVIDIA GeForce RTX 6000 (24G) for 2 epochs (approx. 24 hours of training time) with batch size 8 and gradient accumulation for an effective batch size of 32 (same as SPECTER). The graph embedding training is performed on an Intel Xeon Gold 6230 CPU with 60 cores and takes approx. 6 hours. The kNN strategy is implemented with FAISS (Johnson et al., 2021) using a flat index (exhaustive search) and takes less than 30min for indexing and retrieval of the triplets.

5.4.4 Baseline Methods

Task →	Classification		User activity prediction				Citation prediction				Recomm.		Avg.
Subtask →	MAG	MeSH	Co-View		Co-Read		Cite		Co-Cite				
Model ↓ / Metric →	F1	F1	MAP	nDCG	MAP	nDCG	MAP	nDCG	MAP	nDCG	nDCG	P@1	
Oracle SciDocs †	*87.1*	*94.8*	*87.2*	*93.5*	*88.7*	*94.6*	*92.3*	*96.8*	*91.4*	*96.4*	*53.8*	*19.4*	*83.0*
USE (2018)	80.0	83.9	77.2	88.1	76.5	88.1	76.6	89.0	78.3	89.8	53.7	19.6	75.1
Citeomatic* (2018)	67.1	75.7	81.1	90.2	80.5	90.2	86.3	94.1	84.4	92.8	52.5	17.3	76.0
SGC* (2019)	76.8	82.7	77.2	88.0	75.7	87.5	91.6	96.2	84.1	92.5	52.7	18.2	76.9
BERT (2019b)	79.9	74.3	59.9	78.3	57.1	76.4	54.3	75.1	57.9	77.3	52.1	18.1	63.4
SciBERT* (2019)	79.7	80.7	50.7	73.1	47.7	71.1	48.3	71.7	49.7	72.6	52.1	17.9	59.6
BioBERT (2019)	77.2	73.0	53.3	74.0	50.6	72.2	45.5	69.0	49.4	71.8	52.0	17.9	58.8
CiteBERT (2021)	78.8	74.8	53.2	73.6	49.9	71.3	45.0	67.9	50.3	72.1	51.6	17.0	58.8
DeCLUTR (2021b)	81.2	88.0	63.4	80.6	60.0	78.6	57.2	77.4	62.9	80.9	52.0	17.4	66.6
SPECTER* (2020a)	**82.0**	86.4	83.6	91.5	84.5	92.4	88.3	94.9	88.1	94.8	**53.9**	**20.0**	80.0
Replicated SPECTER training data (w/ leakage):													
SciNCL (ours)	81.4	**88.7**	**85.3**	**92.3**	**87.5**	**93.9**	**93.6**	**97.3**	**91.6**	**96.4**	**53.9**	19.3	**81.8**
± σ w/ ten seeds	*.449*	*.422*	*.128*	*.08*	*.162*	*.118*	*.104*	*.054*	*.099*	*.066*	*.203*	*.356*	*.064*
Random S2ORC training data (w/o leakage):													
SPECTER	81.3	88.4	83.1	91.3	84.0	92.1	86.2	93.9	87.8	94.7	52.2	17.5	79.4
SciNCL (ours)	81.3	89.4	84.3	91.8	85.6	92.8	91.4	96.3	90.1	95.7	54.3	19.9	81.1

Table 5.1: Results on the SciDocs test set. With replicated SPECTER training data, SciNCL surpasses the previous best avg. score by 1.8 points and also outperforms the baselines in 9 of 12 task metrics. Our scores are reported as mean and standard deviation σ over ten random seeds. With training data randomly sampled from S2ORC, SciNCL outperforms SPECTER in terms of avg. score with 1.7 points. The scores with * are from Cohan et al. (2020a). *Oracle SciDocs* † is the upper bound of the performance with triplets from SciDocs's data.

We compare against the following baselines (details in Appendix 5.10.5): USE (Cer et al., 2018), BERT (Devlin et al., 2019b), BioBERT (Lee et al., 2019), SciBERT (Beltagy et al., 2019), CiteBERT (Wright and Augenstein, 2021), DeCLUTR (Giorgi et al., 2021b), the graph-convolution approach SGC (Wu et al., 2019), Citeomatic (Bhagavatula et al., 2018), and SPECTER (Cohan et al., 2020a).

Also, we compare against *Oracle SciDocs* which is identical to SciNCL except that its triplets are generated based on SciDocs's validation and test set using their gold labels. For example, papers with the same MAG labels are positives and papers with different labels are negatives. Similarly, the ground truth of the other tasks is used, i.e., clicked recommendations are considered as positives etc. In total, this procedure creates 106K training triplets for *Oracle SciDocs*. Moreover, we under-sample triplets from the classification tasks to ensure a balanced triplet distribution over the tasks. Accordingly, *Oracle SciDocs* represents an estimate for the performance upper bound that can be achieved with the current setting (triplet margin loss and SciBERT encoder).

5.5 Overall Results

Table 5.1 shows the results, comparing SciNCL with the best validation performance against the baselines. With replicated SPECTER training data (w/ leakage), SciNCL achieves an average performance of 81.8 across all metrics, which is a 1.8 point absolute improvement over SPECTER (the next-best baseline). When trained without leakage, the improvement of SciNCL over SPECTER is consistent with 1.7 points but generally lower (79.4 avg. score). In the following, we refer to the results obtained through training on the replicated SPECTER data (w/ leakage) if not otherwise mentioned.

We find the best validation performance based on SPECTER's data when positives and hard negative are sampled with kNN, whereby positives are $k^{+}{=}25$, and hard negatives are $k^{-}_{\text{hard}}{=}4000$ (Section 5.6). Easy negatives are generated through filtered random sampling. SciNCL's scores are reported as mean over ten random seeds (seed $\in [0, 9]$).

For MAG classification, SPECTER achieves the best result with 82.0 F1 followed by SciNCL with 81.4 F1 (-0.6 points). For MeSH classification, SciNCL yields the highest score with 88.7 F1 (+2.3 compared to SPECTER). Both classification tasks have in common that the chosen training settings lead to over-fitting. Changing the training by using only 1% training data, SciNCL yields 82.2 F1@MAG (Table 5.2). In all user activity and citation tasks, SciNCL yields higher scores than all baselines. Moreover, SciNCL outperforms SGC on direct citation prediction, where SGC outperforms SPECTER in terms of nDCG. On the recommender task, SPECTER yields the best P@1 with 20.0, whereas SciNCL achieves 19.3 P@1 (in terms of nDCG SciNCL and SPECTER are on par).

When training SPECTER and SciNCL without leakage, SciNCL outperforms SPECTER even in 11 of 12 metrics and is on par in the other metric. This suggests that SciNCL's hyperparameters have a low corpus dependency since they were only optimized on the corpus with leakage.

Regarding the LLM baselines, we observe that the general-domain BERT, with a score of 63.4, outperforms the domain-specific BERT variants, namely SciBERT (59.6) and BioBERT (58.8). LLMs without citations or contrastive objectives yield generally poor results. This emphasizes the anisotropy problem of embeddings directly extracted from current LLMs and highlights the advantage of combining text and citation information.

In summary, we show that SciNCL's triplet selection leads on average to a performance improvement on SciDocs, with most gains being observed for user activity and citation tasks. The gain from 80.0 to 81.8 is particularly notable given that even *Oracle SciDocs* yields with 83.0 an only marginally higher avg. score despite using test and validation data from SciDocs for the triplet selection.

5.6 Impact of Sample Difficulty

In this section, we present the optimization of SciNCL's sampling strategy (Section 5.3.3). We optimize the sampling for positives and hard or easy negatives with partial grid search on a random sample of 10% of the replicated SPECTER training data (sampling based on queries). Our experiments show that optimizations on this subset correlate with the entire dataset. The validation scores in Section 5.2 and Figure 5.3 are reported as the mean over three random seeds.

5.6.1 Positive Samples

Figure 5.2 shows the avg. scores on the SciDocs validation set depending on the selection of positives with the kNN strategy. We only change k^+, while negative sampling remains fixed to its best setting (Section 5.6.2). The performance is relatively stable for $k^+{<}100$ with peak at $k^+{=}25$, for $k^+{>}100$ the performance declines as k^+ increases. Wang and Isola (2020) state that positive samples should be semantically similar to each other, but not too similar to the query. For example, at $k^+{=}5$, positives may be a bit "too easy" to learn, such that they produce less informative gradients than the optimal setting $k^+{=}25$. Similarly, making k^+ too large leads to the *sampling induced margin* being too small, such that *positives collide with negative samples*, which creates contrastive label noise that degrades performance (Saunshi et al., 2019).

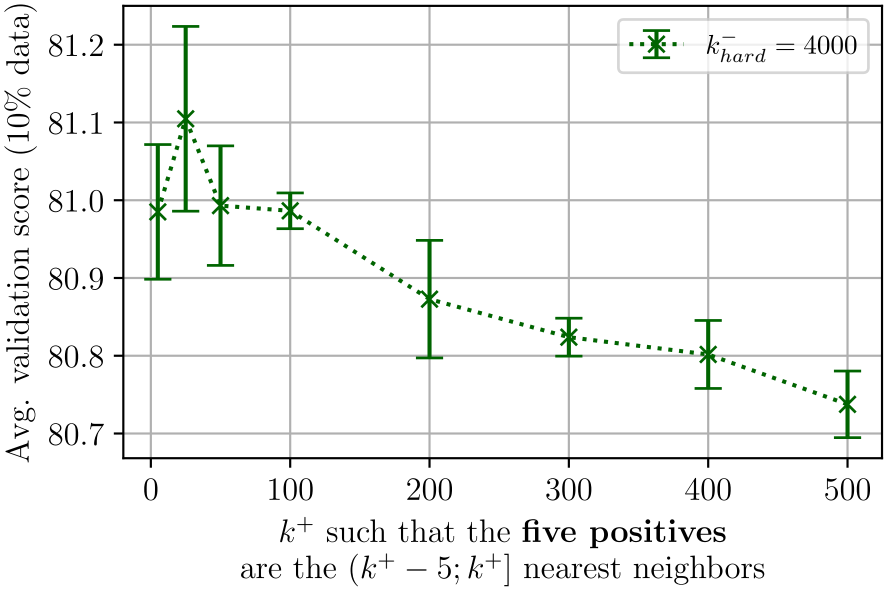


Figure 5.2: Results on the validation set w.r.t. positive sampling with kNN when using 10% training data.

Another observation is the standard deviation σ: One would expect σ to be independent of k^+ since random seeds affect only the negatives. However, positives and negatives interact with each other through the triplet margin loss. Therefore, σ is also affected by k^+. To account for the interaction of positives and negatives, one could sample simultaneously based on the distance to the query and the distance of positives and negatives to each other.

5.6.2 Hard Negative Samples

Figure 5.3 presents the validation results for different k^{-}_{hard} given the best setting for positives (k^+=25). The performance increases with increasing k^{-}_{hard} until a plateau between 2000<k^{-}_{hard}<4000 with a peak at k^{-}_{hard}=4000. This plateau can also be observed in the test set, where k^{-}_{hard}=3000 yields a marginally lower score of 81.7 (Table 5.2). For k^{-}_{hard}>4000, the performance starts to decline again. This suggests that for large k^{-}_{hard} the samples are not “hard enough” which confirms the findings of Cohan et al. (2020a).

5.6.3 Easy Negative Samples

Filtered random sampling of easy negatives yields the best validation performance compared pure random sampling (Table 5.2). However, the performance difference is marginal. When rounded to one decimal, their average test scores are identical. The marginal difference is caused by the large corpus size and the resulting small probability of randomly sampling one paper from the kNN results. But without

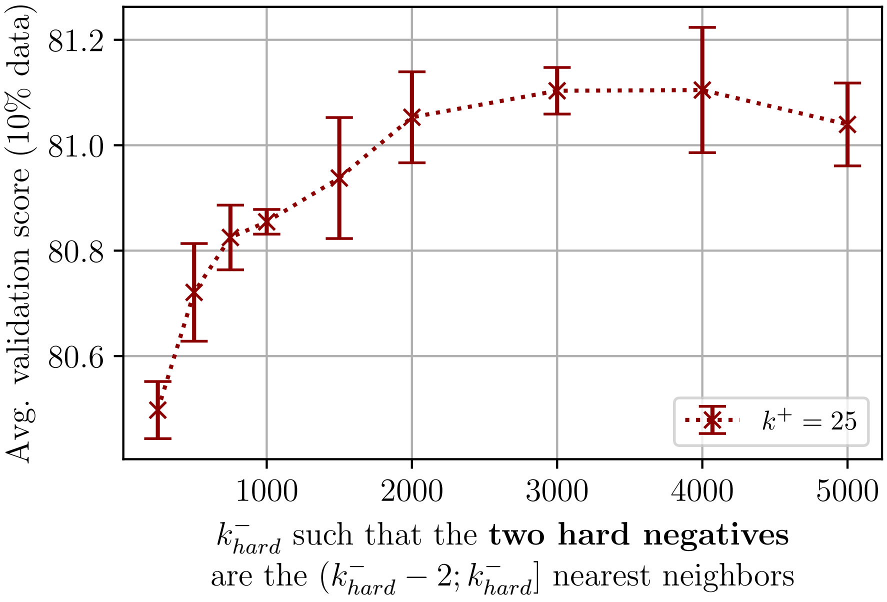


Figure 5.3: Results on the validation set w.r.t. hard negative sampling with KNN using 10% training data.

filtering, the effect of random seeds increases, since we find a higher standard deviation compared to the one with filtering.

As a potential way to decrease randomness, we experiment with other approaches like k-means clustering but find that they decrease the performance (Appendix 5.10.6).

5.6.4 Collisions

Similar to SPECTER, SciNCL's sampling based on graph embeddings could cause collisions when selecting positives and negatives from regions close to each other. To avoid this, we rely on a sample induced margin that is defined by the hyperparameter k^+ and k^-_{hard} (distance between red and green band in Figure 5.1). When the margin gets too small, positives and negatives are more likely to collide. A collision occurs when the paper pair (d_q, d_s) is contained in the training data as positive and as negative sample at the same time. Figure 5.4 demonstrates the relation between the number of collisions and the size of the sample induced margin. The number of collisions increases when the sample induced margin gets smaller. The opposite is the case when the margin is large enough ($k^-_{\text{hard}} > 1000$), i.e., then the number of collisions goes to zero. This relation also affects the evaluation performance as Figure 5.2 and Figure 5.3 show. Namely, for large k^+ or small k^-_{hard} SciNCL's performance declines and approaches SPECTER's performance.

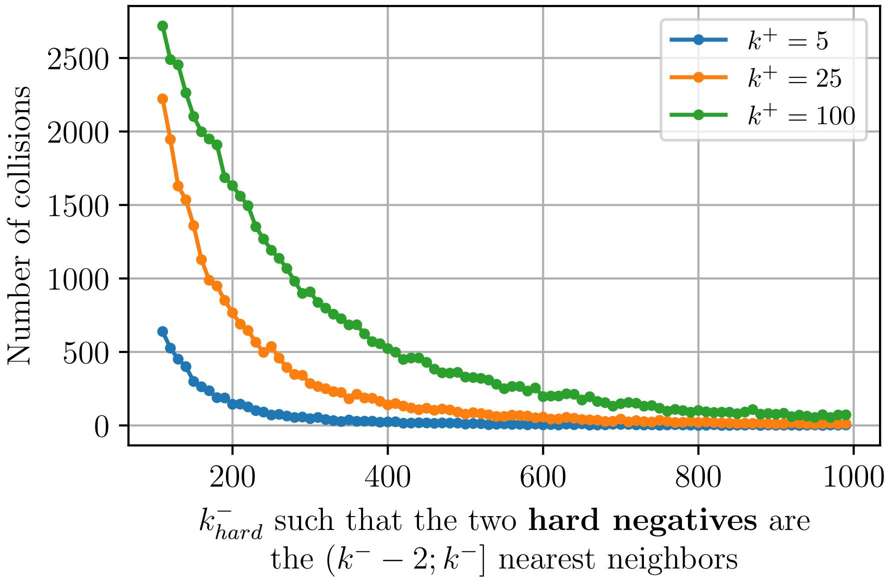


Figure 5.4: Number of collisions w.r.t. size of the sample induced margin as defined through k^+ and k^-_{hard}.

	CLS	USR	CITE	REC	Avg.	Δ
SciNCL	85.0	88.8	**94.7**	36.6	**81.8**	–
SPECTER	84.2	88.4	91.5	36.9	80.0	-1.8
k^-_{hard}=2000	84.9	88.8	**94.7**	36.1	81.6	-0.2
k^-_{hard}=3000	84.5	88.7	94.6	36.9	81.7	-0.1
easy neg. w/ random	85.1	88.8	**94.7**	36.6	**81.8**	0.0
undirected citations	84.6	88.8	**94.7**	36.6	81.7	-0.1
Init. w/ BERT-Base	83.4	88.4	93.8	37.5	81.2	-0.6
Init. w/ BERT-Large	84.6	88.7	94.1	36.4	81.4	-0.4
Init. w/ BioBERT	83.7	88.6	93.8	**37.7**	81.4	-0.4
1% training data	85.2	88.3	92.7	36.1	80.8	-1.0
10% training data	85.1	88.7	93.5	36.2	81.1	-0.6
BitFit training	**85.8**	88.6	93.7	35.3	81.2	-0.5

Table 5.2: Ablations. Numbers are averages over tasks of the SciDocs test set, average score over all metrics, and rounded absolute difference to SciNCL.

5.7 Ablation Analysis

Next, we evaluate the impact of language model initialization and number of parameters and triples.

5.7.1 Initial Language Models

Table 5.2 shows the effect of initializing the model weights not with SciBERT but with general-domain LLMs (BERT-Base and BERT-Large) or with BioBERT. The initialization with other LLMs decreases the performance. However, the decline is marginal (BERT-Base -0.6, BERT-Large -0.4, BioBERT -0.4) and all LLMs outperform the SPECTER baseline. For the recommendation task, in which SPECTER is superior over SciNCL, BioBERT outperforms SPECTER. This indicates that the improved triplet mining of SciNCL has a greater domain adaption effect than pretraining on domain-specific literature. Given that pretraining of LLMs requires a magnitude more resources than the fine-tuning with SciNCL, our approach can be a solution for resource-limited use cases.

5.7.2 Data and Computing Efficiency

The last three rows of Table 5.2 show the results regarding data and computing efficiency. When keeping the citation graph unchanged but training the language model with only 10% of the original triplets, SciNCL still yields a score of 81.1 (-0.6). Even with only 1% (6840 triplets), SciNCL achieves a score of 80.8 that is 1.0 points less than with 100% but still 0.8 points more than the SPECTER baseline. With this *textual* sample efficiency, one could manually create triplets or use existing supervised datasets as in Gao et al. (2021b).

Lastly, we evaluate BitFit training (Ben Zaken et al., 2022b), which only trains the bias terms of the model while freezing all other parameters. This corresponds to training only 0.1% of the original parameters. With BitFit, SciNCL yields a considerable score of 81.2 (-0.5 points). As a result, SciNCL could be trained on the same hardware with even larger (general-domain) language models Section (5.7.1).

5.8 Conclusion

We present a novel approach for contrastive learning of scientific document embeddings that addresses the challenge of selecting informative positive and negative samples. By leveraging citation graph embeddings for sample generation, SciNCL achieves a score of 81.8 on the SciDocs benchmark, a 1.8 point improvement over the previous best method SPECTER. This is purely achieved by introducing tunable sample difficulty and avoiding collisions between positive and negative samples, while existing LLM and data setups can be reused. This improvement over SPECTER can be also observed when excluding the SciDocs papers during training (see w/o leakage in Table 5.1). Furthermore, SciNCL's improvement from 80.0 to 81.8 is particularly notable given that even *oracle triplets*, which are generated with SciDocs's test and validation data, yield with 83.0 only a marginally higher score.

Our work highlights the importance of sample generation in a contrastive learning setting. We show that language model training with 1% of triplets is sufficient to outperform SPECTER, whereas the remaining 99% provide only 1.0 additional points (80.8 to 81.8). This sample efficiency is achieved by adding reasonable effort for sample generation, i.e., graph embedding training and KNN search. We also demonstrate that in-domain LLM pretraining (like SciBERT) is beneficial, while general-domain LLMs can achieve comparable performance and even outperform SPECTER. This indicates that controlling sample difficulty and avoiding collisions is more effective than in-domain pretraining, especially in scenarios where training an LLM from scratch is infeasible.

5.9 Limitations

SciNCL's strategy of selecting positive and negative samples requires additional computational resources for training the graph embedding model, performing the KNN search, and optimizing the hyperparameters $k^{+}, k^{-}_{\text{hard}}$ (Section 5.4.3). While some of the compute resources are offset by the sample-efficient language model training (Section 5.7.2), we still consider the increased compute effort as the major limitation of the SciNCL method.

Especially the training of the graph embedding model accounts for most of the additional compute effort. This is also the reason for us providing only a shallow of evaluation of the graph embeddings (Appendix 5.10.4). For example, we did not evaluate the effect of different graph embeddings on the actual SciDocs performance. Moreover, evaluations with smaller subsets of the S2ORC citation graph are missing. Such evaluations could indicate whether also less citation data can be sufficient, which would lower the compute requirements but would make SciNCL also applicable in domains where less graph data is available.

Acknowledgements

We would like to thank Christian Schulze and his team for providing the compute infrastructure that made our experiments possible. The research presented in this article is partially funded by the German Federal Ministry of Education and Research (BMBF) through the projects QURATOR (Rehm et al., 2020) (Unternehmen Region, Wachstumskern, no. 03WKDA1A) and PANQURA (no. 03COV03E).

5.10 Appendices

5.10.1 Mapping to S2ORC

Table 5.3: Mapping to S2ORC citation graph

S2ORC mapping	Success rate
SciDocs papers	
- with S2ORC IDs	220,815 / 223,932 (98.6%)
- in S2ORC graph	197,811 / 223,932 (88.3%)
SPECTER papers	
- with S2ORC IDs	311,094 / 311,860 (99.7%)
- in S2ORC graph	260,014 / 311,860 (83.3%)

Neither the SPECTER training data nor the SciDocs test data comes with a mapping to the S2ORC dataset, which we use for the training of the citation embedding model. However, to replicate SPECTER's training data and to avoid leakage of SciDocs test data such a mapping is needed. Therefore, we try to map the papers to S2ORC based on PDF hashes and exact title matches. The remaining paper metadata is collected through the Semantic Scholar API. Table 5.3 summarizes the outcome of mapping procedure. Failed mappings can be attributed to papers being unavailable through the Semantic Scholar API (e.g., retracted papers) or papers not being part of S2ORC citation graph.

5.10.2 SPECTER-SciDocs Leakage

When replicating SPECTER (Cohan et al., 2020a), we found a substantial overlap between the papers[3] used during the model training and the papers from their SciDocs benchmark[4]. In both datasets, papers are associated with Semantic Scholar IDs. Thus, no custom ID mapping as in Section 5.10.1 is required to identify papers that leak from training to test data. From the 311,860 unique papers used in SPECTER's training data, we find 79,201 papers (25.4%) in the test set of SciDocs and 79,609 papers (25.5%) in its validation set. When combining test and validation set, there is a total overlap of 126,176 papers (40.5%). However, this overlap affects only the 'unsupervised' paper metadata (title, abstract, citations, etc.) and not the gold labels used in SciDocs (e.g., MAG labels or clicked recommendations).

[3] https://github.com/allenai/specter/issues/2

[4] https://github.com/allenai/scidocs

5.10.3 Dataset Creation

As describe in Section 5.4.2, we conduct our experiments on two datasets. Both datasets rely on the citation graph of S2ORC (Lo et al., 2020). More specifically, S2ORC with the version identifier `20200705v1` is used. The full citation graph consists of 52.6M nodes (papers) and 467M edges (citations). Table 5.4 presents statistics on the datasets and their overlap with SPECTER and SciDocs. The steps to reproduce both datasets are:

Replicated SPECTER (w/ leakage)

In order to replicate SPECTER's training data and do not increase the leakage, we exclude all SciDocs papers which are not used by SPECTER from the S2ORC citation graph. This means that apart from the 110,538 SPECTER papers not a single other SciDocs paper is included. The resulting citation graph has 52.5M nodes and 463M edges and is used for training the citation graph embeddings.

For the SciNCL triplet selection, we also replicate SPECTER's query papers and its corpus from which positive and negatives are sampled. Our mapping and the underlying citation graph allows us to use 227,869 of 248,007 SPECTER's papers for training. Regarding query papers, we use 131,644 of 136,820 SPECTER's query papers. To align the number training triplets with the one from SPECTER, additional papers are randomly sampled from the filtered citation graph.

Random S2ORC subset (w/o leakage)

To avoid leakage, we exclude all successfully mapped SciDocs papers from the S2ORC citation graph. After filtering the graph has 52.3 nodes and 447M edges. The citation graph embedding model is trained on this graph.

Next, we reproduce triplet selection from SPECTER. Any random 136,820 query papers are selected from the filtered graph. For each query, we generate five positives (cited by the query), two hard negatives (citation of citation), and three random nodes from the filtered S2ORC citation graphs. This sampling produces 684,100 training triplets with 680,967 unique papers IDs (more compared to the replicated SPECTER dataset). Based on these triplets the SPECTER model for this dataset is trained with the same model settings and hyperparameters as SciNCL (second last row in Table 5.1).

Lastly, the SciNCL triplets are generated based on the citation graph embeddings of the same 680,967 unique papers IDs, i.e, the FAISS index contains only these papers and not the remaining S2ORC papers. Also, the same 136,820 query papers are used.

Table 5.4: Statistics for our two datasets and their overlap with SPECTER and SciDocs respectively.

	Replicated	Random
	SPECTER	S2ORC subset
	(w/ leakage)	(w/o leakage)
Training triplets	684,100	684,100
Unique paper IDs	248,007	680,967
- in SPECTER	227,869	9,182
- in SciDocs	110,538	0
- in SciDocs and in SPECTER	110,538	0
Query paper IDs	136,820	136,820
- in SciDocs	69,306	0
- in SPECTER queries	131,644	463
Citation graph		
- Nodes	52,526,134	52,373,977
- Edges	463,697,639	447,697,727

5.10.4 Graph Embedding Evaluation

To evaluate the underlying citation graph embeddings, we experiment with a few of BigGraph's hyperparameters. We trained embeddings with different dimensions d=$\{128, 512, 768\}$ and different distance measures (cosine similarity and dot product) on 99% of the data and test the remaining 1% on the link prediction task. An evaluation of the graph embeddings with SciDocs is not possible since we could not map the papers used in SciDocs to the S2ORC corpus. All variations are trained for 20 epochs, margin m=0.15, and learning rate λ=0.1 (based on the recommended settings by Lerer et al. (2019)).

Table 5.5: Link prediction performance of BigGraph embeddings trained on S2ORC citation graph with different dimensions and distance measures.

Dim.	Dist.	MRR	Hits@1	Hits@10	AUC
128	Cos.	54.09	43.39	75.21	85.75
128	Dot	89.75	85.84	96.13	97.70
512	Dot	94.60	92.47	97.64	98.64
768	Dot	95.12	93.22	97.77	98.74

Table 5.5 shows the link prediction performance measured in MRR, Hits@1, Hits@10, and AUC. Dot product is substantially better than cosine similarity as distance measure. Also, there is a positive correlation between the performance and the size of the embeddings. The larger the embedding size the better link prediction performance. Graph

embeddings with d=768 were the largest possible size given our compute resources (available disk space was the limiting factor).

5.10.5 Baseline Details

If not otherwise mentioned, all BERT variations are used in their *base-uncased* versions.

The weights for BERT (*bert-base-uncased*), BioBERT (*biobert-base-cased-v1.2*), CiteBERT (*citebert*), DeCLUTR (*declutr-sci-base*) are taken from Huggingface Hub[5]. We use Universal Sentence Encoder (USE) from Tensorflow Hub[6]. For *Oracle SciDocs*, we use the SciNCL implementation and under-sample the triplets from the classification tasks to ensure a balanced triplet distribution over the tasks. The SPECTER version for the random S2ORC training data (w/o leakage) is also trained with the SciNCL implementation. Please see Cohan et al. (2020a) for additional baseline methods and their implementation details.

5.10.6 Negative Results

We investigated additional sampling strategies and model modification of which none led to a significant performance improvement.

5.10.6.1 Undirected Citations

Our graph embedding model considers citations as directed edges by default. We also train a SciNCL model with undirected citations by first converting a single edge (a, b) into the two edges (a, b) and (b, a). This approach yields a slightly worse performance (81.7 avg. score; -0.1 points) and, therefore, was discarded for the final experiments.

5.10.6.2 kNN with interval large than c

Our best results are achieved with kNN where the size of the neighbor interval $(k - c'; k]$ is equal to the number of samples c' that the strategy should generate. In addition to this, we also experimented with large intervals, e.g., $(1000; 2000]$, from which c' papers are randomly sampled. This approach yields comparable results but suffers from a larger effect of randomness and is therefore more difficult to optimize.

5.10.6.3 K-Means Cluster for Easy Negatives

Easy negatives are supposed to be far away from the query. Random sampling from a large corpus ensures this as our results show. As an alternative approach, we tried

[5] https://huggingface.co/models
[6] https://tfhub.dev/google/universal-sentence-encoder-large/5

k-means clustering whereby we selected easy negatives from the centroid that has a given distance to the query's centroid. However, this decreased the performance.

5.10.6.4 Sampling with Similarity Threshold

As alternative to kNN, we select samples based on cosine similarity in the citation embedding space. Take c' papers that are within the similarity threshold t of a query paper d^Q such that $s(f_c(d^Q), f_c(d_i)) < t$, where s is the cosine similarity function.

For example, given the similarity scores $S{=}\{0.9, 0.8, 0.7, 0.1\}$ (ascending order, the higher the similarity is the closer the candidate embedding to the query embedding is) with $c'{=}2$ and $t{=}0.5$, the two candidates with the largest similarity scores and larger than the threshold would be 0.8 and 0.7. The corresponding papers would be selected as samples. While the positive threshold t^+ should close to 1, the negative threshold t^- should be small to ensure samples are dissimilar from d^Q. However, the empirical results suggest that this strategy is inferior compared to kNN.

5.10.6.5 Hard Negatives with Similarity Threshold

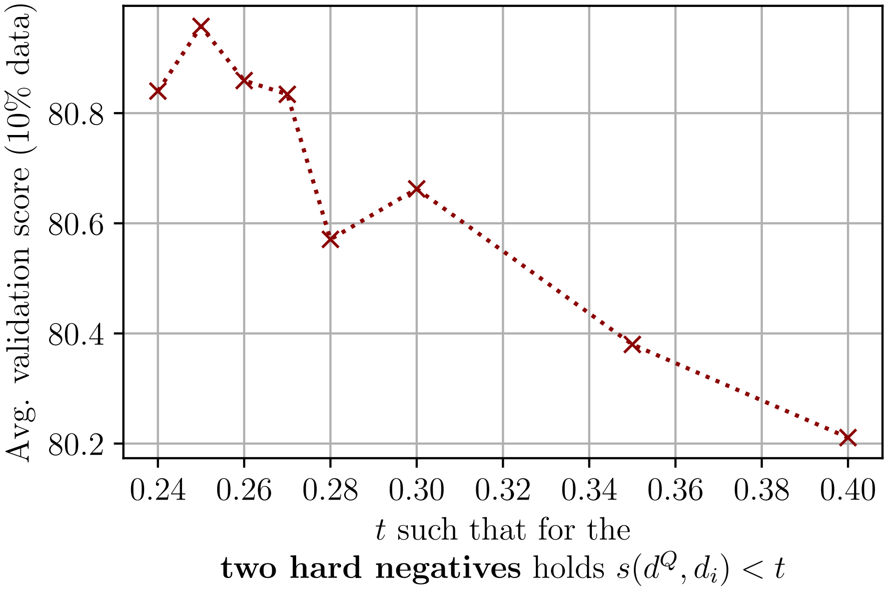


Figure 5.5: Results on the validation set w.r.t. hard negative sampling with Sim using 10% training data.

Selecting hard negatives based on the similarity threshold yields a test score of 81.7 (-0.1 points). Figure 5.5 show the validation results for different similarity thresholds. A similar pattern as in Figure 5.3 can be seen. When the negatives are closer to the query paper (larger similarity threshold t), the validation score decreases.

5.10.6.6 Positives with Similarity Threshold

Positive sampling with Sim performs poorly since even for small $t^+ < 0.5$ many query papers do not have any neighbors within this similarity threshold (more than 40%). Solving this issue would require changing the set of query papers which we omit for comparability to SPECTER.

5.10.6.7 Sorted Random

Simple random sampling does not ensure if a sample is far or close to the query. To integrate a distance measure in the random sampling, we first sample n candidates, then order the candidates according to their distance to the query, and lastly select the c' candidates that are the closest or furthest to the query as samples.

5.10.6.8 Mask Language Modeling

Giorgi et al. (2021b) show that combining a contrastive loss with a mask language modeling loss can improve text representation learning. However, in our experiments a combined function decreases the performance on SciDocs, probably due to the effects found by (Li et al., 2020).

5.10.6.9 Student-Teacher Learning

Student-teacher learning is effective in related work on cross-modal knowledge transfer (Kaur et al., 2021; Tian et al., 2020a). We also try to adopt this approach for our experiments, whereby the Transformer language model is the student, and the citation graph embedding model is the teacher. By directly learning from the citation embeddings, we could circumvent the positive and negative sampling needed for triplet loss learning, which introduces unwanted issues like collisions. Given a batch of document representations derived from text D_{Text} (through the language model) and the citation graph representations for the same documents D_{Graph}, we compute the pairwise cosine similarity for both sets S_{Text} and S_{Graph}. To transfer the knowledge from the citation embeddings into the language model, we devise the student-teacher loss $\mathcal{L}_{ST}$ based on a mean-squared-error loss (MSE) such that the difference between the cosine similarities is minimized:

$$\mathcal{L}_{ST} = \text{MSE}(S_{Text}, S_{Graph}) \tag{5.1}$$

Despite the promising results from Tian et al. (2020a), the student-teacher approach performs poorly in our experiments. We attribute this the overfitting to the citation data (the training loss approaches zero after a few steps while the validation loss remains high). The model trained with $\mathcal{L}_{ST}$ yields only a SciDocs average score of

64.7, slightly better than SciBERT but substantially worse than SciNCL with triplet loss.

Additionally, we experiment with a joint loss that is the sum of triplet margin loss $\mathcal{L}_{Triplet}$ (see Section 5.3.1) and the student-teacher loss $\mathcal{L}_{ST}$:

$$\mathcal{L}_{Joint} = \mathcal{L}_{Triplet} + \mathcal{L}_{ST} \tag{5.2}$$

Training with the joint loss $\mathcal{L}_{Joint}$ achieves an average score of 80.5. Even though the joint loss is not subject to overfitting, its SciDocs performance is slightly worse than the triplet loss $\mathcal{L}_{Triplet}$ alone. Given this outcome and that the computation of the cosine similarities adds additional complexity, we discard the student-teacher approach for the final experiments.

5.10.6.10 SPECTER & Bidirectional Citations

SPECTER (Cohan et al., 2020a) relies on unidirectional citations for their sampling strategy. While papers *cited by* the query paper are considered as positives samples, those *citing* the query paper (opposite citation direction) could be negative samples. We see this use of citations as a conceptional flaw in their sampling strategy.

To test the actual effect on the resulting document representation, we first replicate the original unidirectional sampling strategy from SPECTER with our training data (see w/ leakage in Section 5.4.2). The resulting SPECTER model achieves an average score of 79.0 on SciDocs.[7] When changing the sampling strategy from unidirectional to bidirectional ('citations to the query' are also treated as a signal for similarity), we observe an improvement of +0.4 points to 79.4. Consequently, the use of unidirectional citations is not only a conceptional issue but also degrades learning performance.

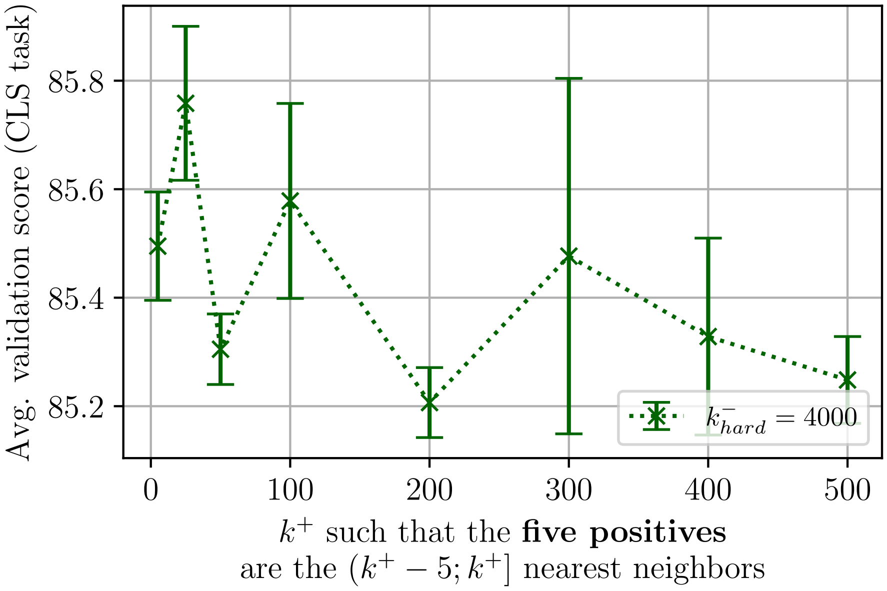
5.10.7 Task-specific Results

Figure 5.7 and 5.9 present the validation performance like in Section 5.6 but on a task-level and not as an average over all tasks. The plots show that the optimal k^+ and k^-_{hard} values are partially task dependent.

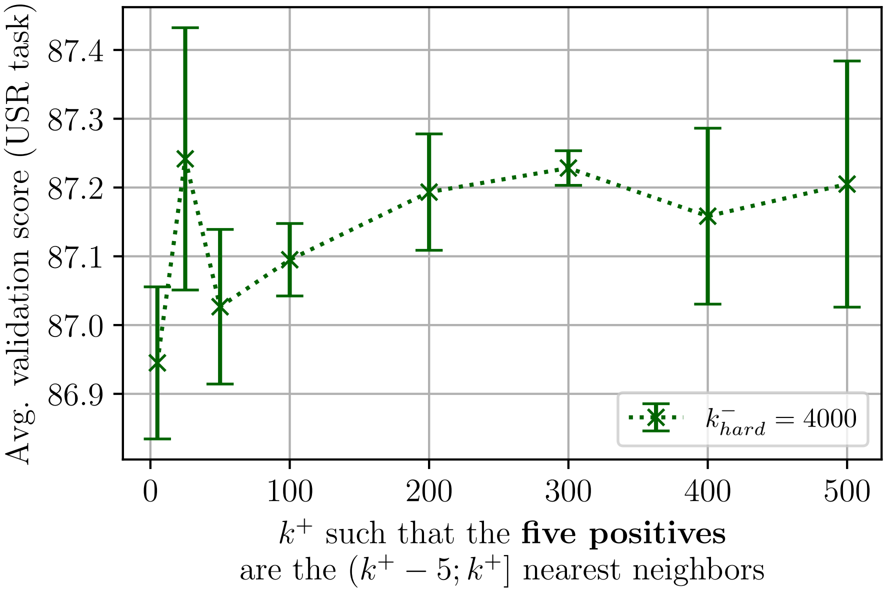
5.10.8 Examples

Table 5.6 lists three examples of query papers with their corresponding positive and negative samples. The complete set of triplets that we use during training is available in our code repository[1].

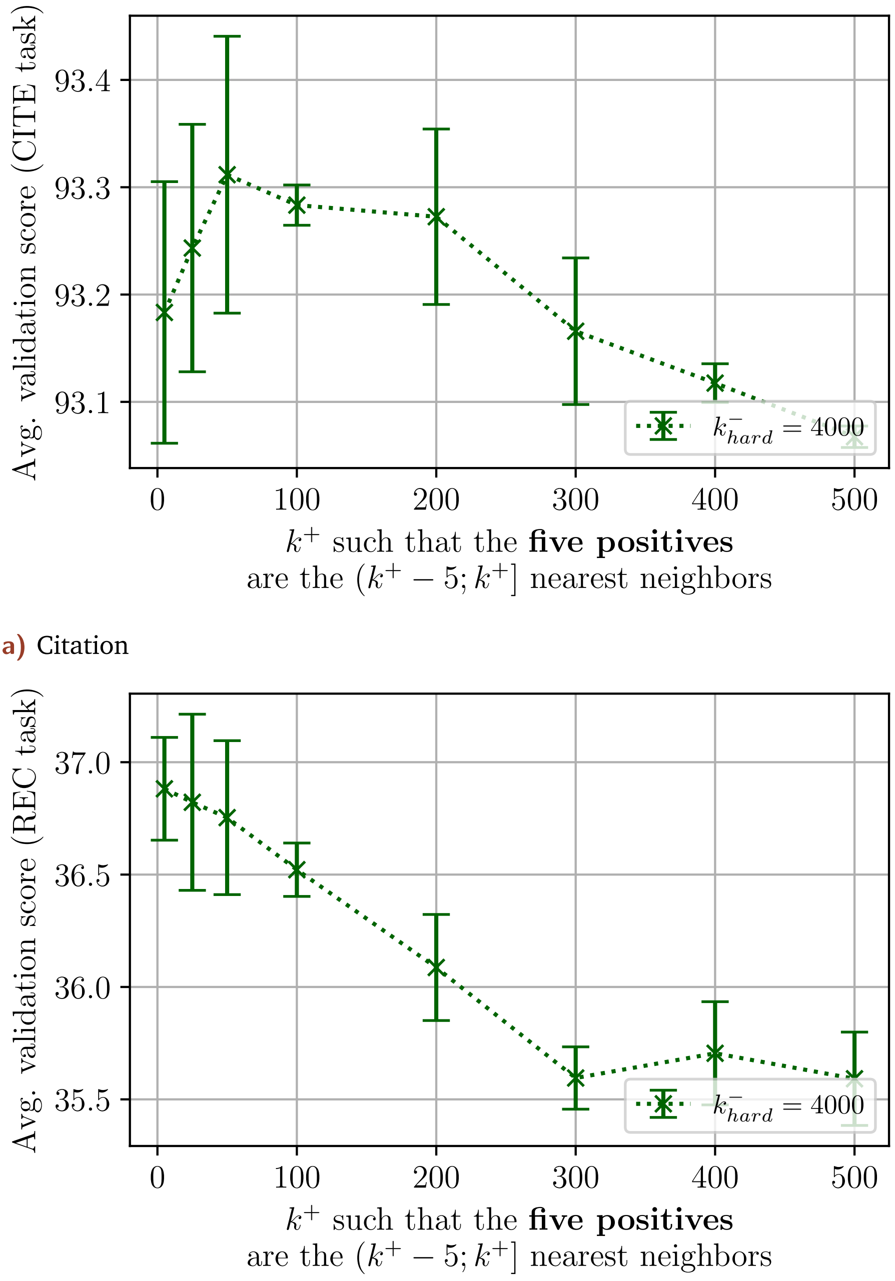
[7] The difference to the scores reported in Cohan et al. (2020a) is due to the difference in the underlying training data.



(a) Classification



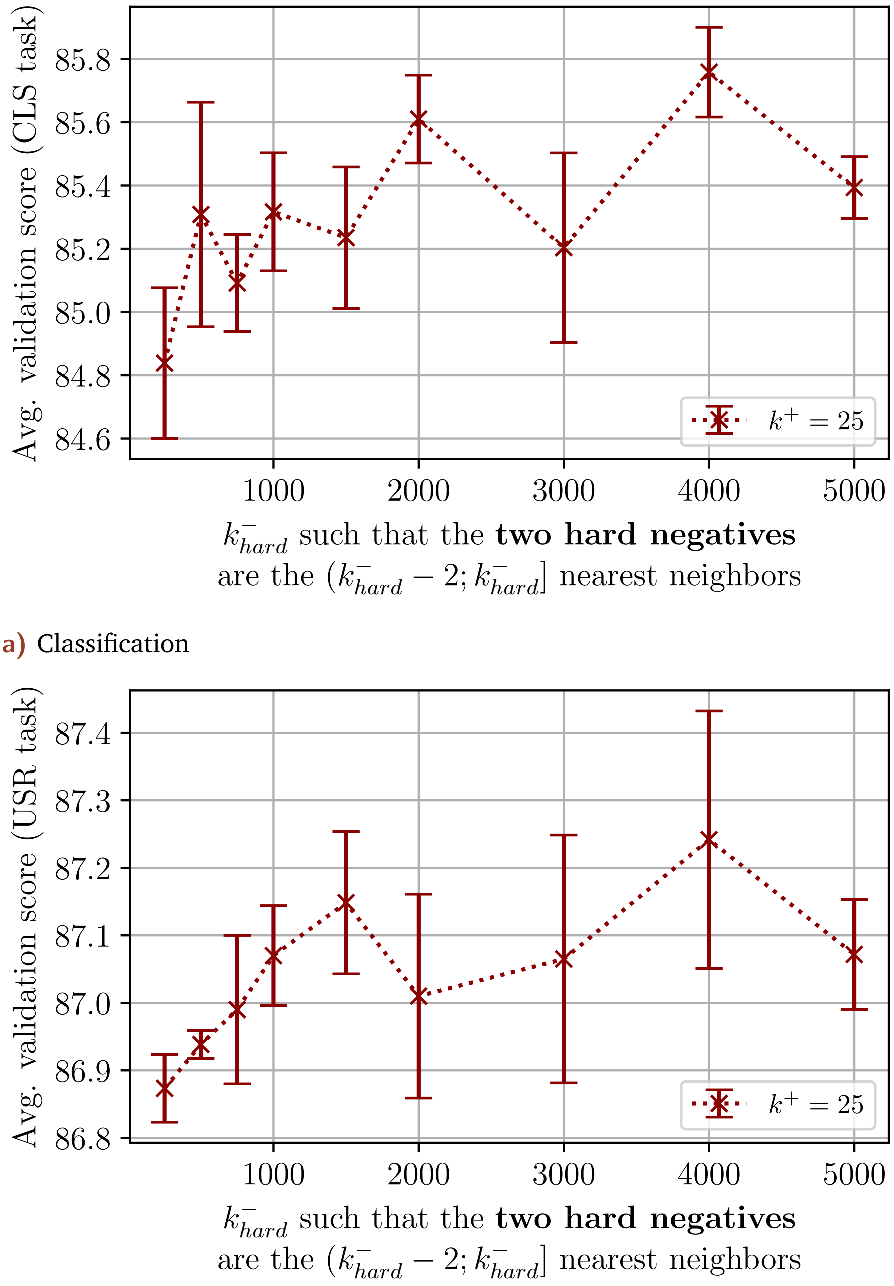
(b) User activities



(a) Citation

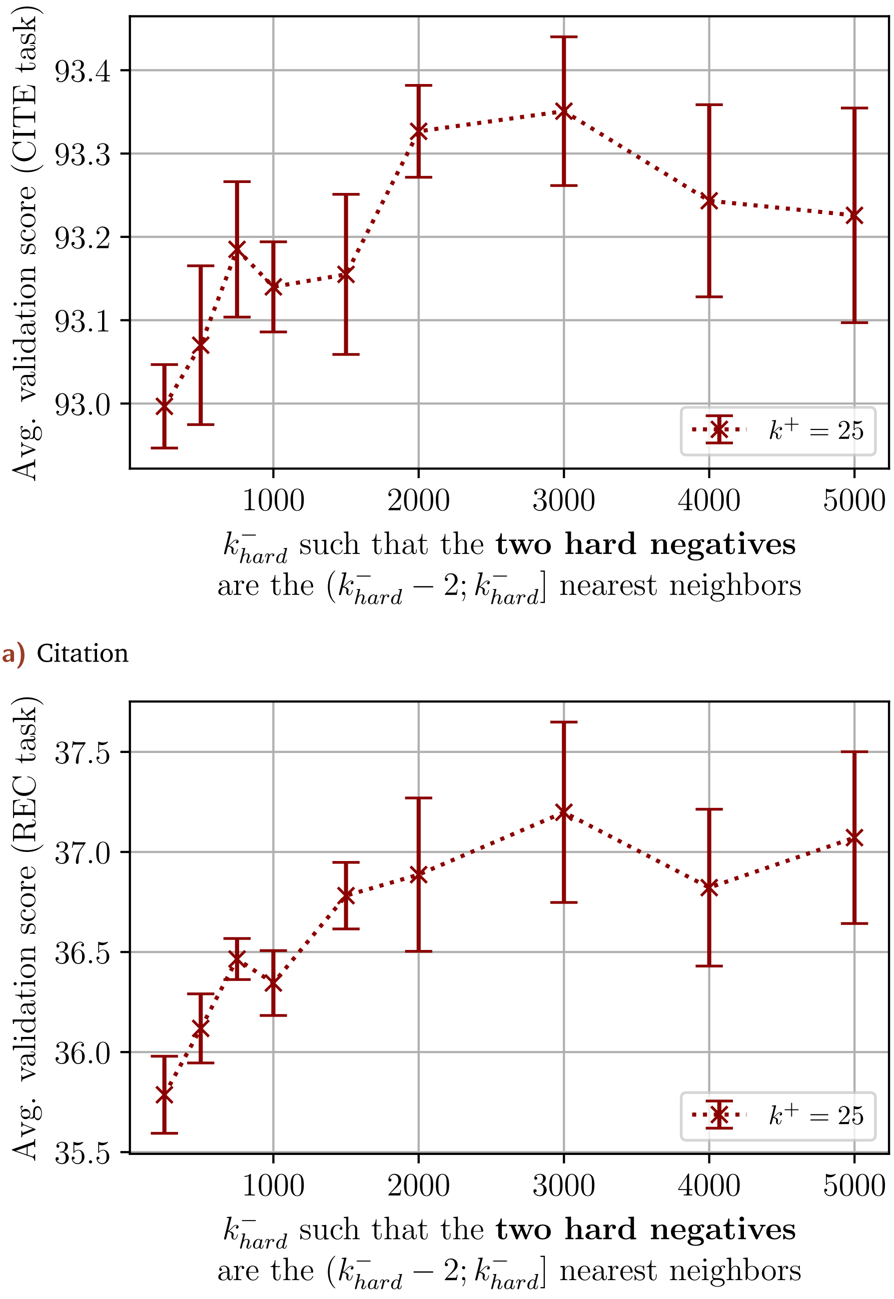
(b) Recommendation

Figure 5.7: Task-level validation performance w.r.t. k^+ with KNN strategy using 10% training data.



(a) Classification

(b) User activities



(a) Citation

(b) Recommendation

Figure 5.9: Task-level validation performance w.r.t. k^-_{hard} with KNN strategy using 10% training data.

Table 5.6: Example query papers with their positive and negative samples.

Query:	**BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding**
Positives:	• A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference • Looking for ELMo's Friends: Sentence-Level Pretraining Beyond Language Modeling • GLUE : A MultiTask Benchmark and Analysis Platform for Natural Language Understanding • Dissecting Contextual Word Embeddings: Architecture and Representation • Universal Transformers
Negatives:	• Planning for decentralized control of multiple robots under uncertainty • Graph-Based Relational Data Visualization • Linked Stream Data Processing • Topic Modeling Using Distributed Word Embeddings • Adversarially-Trained Normalized Noisy-Feature Auto-Encoder for Text Generation
Query:	**BioBERT: a pre-trained biomedical language representation model for biomedical text mining**
Positives:	• Exploring Word Embedding for Drug Name Recognition • A neural joint model for entity and relation extraction from biomedical text • Event Detection with Hybrid Neural Architecture • Improving chemical disease relation extraction with rich features and weakly labeled data • GLUE : A MultiTask Benchmark and Analysis Platform for Natural Language Understanding
Negatives:	• Weakly Supervised Facial Attribute Manipulation via Deep Adversarial Network • Applying the Clique Percolation Method to analyzing cross-market branch banking ... • Perpetual environmentally powered sensor networks • Labelling strategies for hierarchical multi-label classification techniques • Domain Aware Neural Dialog System
Query:	**A Context-Aware Citation Recommendation Model with BERT and Graph Convolutional Networks**
Positives:	• Content-based citation analysis: The next generation of citation analysis • ScisummNet: A Large Annotated Dataset and Content-Impact Models for Scientific Paper ... • Citation Block Determination Using Textual Coherence • Discourse Segmentation Of Multi-Party Conversation • Argumentative Zoning for Improved Citation Indexing
Negatives:	• Adaptive Quantization for Hashing: An Information-Based Approach to Learning ... • Trap Design for Vibratory Bowl Feeders • Software system for the Mars 2020 mission sampling and caching testbeds • Applications of Rhetorical Structure Theory • Text summarization for Malayalam documents — An experience

Part IV

Data-efficient Representation Transfer and Adaptation

6 Paper 5: Long-Tail Zero and Few-Shot Learning via Contrastive Pretraining on and for Small Data

6.1 Introduction

Long-tail information has been found to be disproportionately affected during model compression, which has in turn been linked to reducing aspects of algorithmic fairness for minority information (Hooker et al., 2020a; Hooker, 2021). Additionally, real-world data is subject to long-tail learning challenges such as imbalances, few-shot learning, open-set recognition (Liu et al., 2019c), or feature and label noise (D'souza et al., 2021; Hu et al., 2021). Crucially, works by Hooker et al. (2020d); Zhuang et al. (2021) find that common long-tail evaluation measures like top-k metrics mask tail prediction performance losses. Current works on long-tail preservation in smaller models are focused on compressing large, supervised computer vision models (Liu et al., 2019c; Chang et al., 2019; Joseph et al., 2020; Blakeney et al., 2021; Jiang et al., 2021), while general long-tail learning methods only study supervised contrastive learning.

In this work, we extend the field of 'long-tail preservation in compact models' to (self-supervised) *pretrained language models* (PLMs), and investigate whether contrastive language modeling (CLM) can be used to train a small, long-tail preserving model which does not require compression or large pretrained models. In this context, large PLMs are an important point of reference since they are often assumed to be base models for use in arbitrary NLP downstream tasks, as a trade-off for their large pretraining costs. These models are pretrained over many text domains in the hopes of achieving partial in-domain pretraining that later overlaps with arbitrary downstream applications. This works well except in cases where fine-tuning data is limited (Rogers et al., 2020). Unfortunately, training data and sub-domains in the tail of a distribution are always limited and diverse by definition, which foreseeably increases the domain distribution mismatch between large PLMs and long-tail distributed end-task data. Hence, in order to train long-tail preserving models, it is useful to study small-scale, but in-domain pretraining, which ideally, is similarly or more compute efficient than fine-tuning a large PLM, while still achieving superior long-tail prediction performance.

Thus, we first evaluate a large PLM in a challenging long-tail tag prediction setup (see Section 6.4) and then move on to propose a small contrastive language model (CLM) to answer the following three research questions.

- RQ-1: Does a large pretrained language model, in this case, RoBERTa (Liu et al., 2019b), achieve good long-tail class prediction performance (Section 6.5.1)?
- RQ-2: Can we extend language models such that a small language model can retain accurate long-tail information, with overall training that is computationally cheaper than fine-tuning RoBERTa?
- RQ-3: What are the long-tail prediction performance benefits of small CLMs that unify self-supervised and supervised contrastive learning?

Contributions We address RQ-2 by proposing a contrastive language model objective that *unifies supervised learning with self-supervised pretraining* to produce a *small model, with strong long-tail retention* that is cheap to compute, thereby avoiding the need for compressing a large model. This takes inspiration from supervised contrastive learning, which is known to improve long-tail learning in NLP (Liu et al., 2017; Pappas and Henderson, 2019; Chang et al., 2019). However, we add *self-supervised contrastive learning* since its effect has not been studied in the context of language models for long-tail learning, especially not with the requirement of producing small models. We call this unified learning objective: Contrastive Long-tail Efficient Self-Supervision or CLESS. The method constructs pseudo-labels from input text tokens to use them for contrastive self-supervised pretraining. During supervised fine-tuning on real (long-tail) labels, the model directly reuses the self-supervision task head to predict real, human-annotated, text labels. Thus, we unify self-supervised and supervised learning regimes into a 'text-to-text' approach. This builds on ideas for large PLMs that use 'text-to-text' prediction like T5 (Raffel et al., 2020) and extends them to contrastive self-supervision to ensure long-tail retention in small language models that pretrain efficiently, even under strong data limitations. Using a 'text-to-text' prediction objective allows for modeling arbitrary NLP tasks by design, though in this work we focus exclusively on improving the under-studied field of long-tail language modeling. We evaluate RQ-1 and RQ-2 by comparing RoBERTa against CLESS regarding long-tail prediction in 6.5.1. To address RQ-3, we study three long-tail learning performance aspects. (RQ-3.1) We study how well our contrastive self-supervised pretraining generalizes to long-tail label prediction without using labeled examples, i.e. zero-shot, long-tail prediction in Section 6.5.2. (RQ-3.2) We evaluate how zero-shot performance is impacted by increased model size and pseudo-label amount during self-supervised pretraining (Section 6.5.2). (RQ-3.3) Finally, we investigate our models' few-shot learning capabilities during supervised long-tail fine-tuning and compare the results to the RoBERTa model in Section 6.5.3.

6.2 Related Work

In this section, we summarize related work and how it influenced our method design and evaluation strategy decisions.

6.2.1 Long-tail compression

Works by Hooker et al. (2020a,d) raised awareness of the disproportionate loss of long-tail information during model compression and the undesirable rise in algorithmic bias and fairness issues this may cause. Other works such as Liu et al. (2019c) pointed out that real-world learning is always long-tailed and that few-shot and zero-shot learning settings naturally arise in tailed, real-world distributions. To make matters worse, real-world long-tail data is highly vulnerable to noise, which creates drastic learning and evaluation challenges, especially for self-supervised learning methods. For example, D'souza et al. (2021) identify types of noise that especially impact long-tail data prediction and Zhuang et al. (2021) find that noise disproportionately affects long-tail metrics. In fact, all the aforementioned show that top-k metrics hide long-tail performances losses. This means that we need long-tail sensitive evaluation, which inspired us to use Average Precision as a measure. In addition, we split tail analysis into 5 buckets that all contain an equal amount of positive labels, where each bucket contains increasingly more and rarer classes – see Section 6.4. These label imbalances in long-tail tasks make manual noise treatment very cumbersome, but fortunately, contrastive objectives are naturally robust to label noise as we will detail in the paragraph below.

6.2.2 Contrastive learning benefits

Contrastive objectives like Noise Contrastive Estimation (NCE), have been shown to be much more robust against label noise overfitting than the standard cross-entropy loss (Graf et al., 2021b). Additionally, Zimmermann et al. (2021b) found that contrastive losses can "recover the true data distribution even from very limited learning samples". Supervised contrastive learning methods like Liu et al. (2017); Zhang et al. (2018); Pappas and Henderson (2019); Chang et al. (2019) have repeatedly demonstrated improved long-tail learning. Finally, Jiang et al. (2021) recently proposed contrastive long-tail compression into smaller models. However, this still leaves the research question (RQ-1), whether large models learn long-tail well enough in the first place, unanswered. These observations, learning properties and open research questions inspired us to forgo large model training and the subsequent compression by instead training small contrastive models and extending them with contrastive self-supervision to combine the benefits of language model pretraining and contrastive learning. This imbues a small (contrastive language) model with strong long-tail retention capabilities, as well as with data-efficient learning for better zero to few-shot learning – as is detailed in the results Section 6.5.

6.2.3 Long-tail learning

Long-tail learning has prolific subfields like extreme classification, which is concerned with supervised long-tail learning and top-line metric evaluation. The field provides varied approaches for different data input types like images (Liu et al., 2019c), categorical data, or text classification using small supervised (Liu et al., 2017) or large supervision fine-tuned PLMs like Chang et al. (2019) for supervised tail learning. However, these methods only explore *supervised* contrastive learning and limit their evaluation to *top-line metrics*, which, as mentioned above, mask long-tail performance losses. This naturally leads us to explore the effects of *self-supervised contrastive* learning (or pretraining) as one might expect such pretraining to enrich long-tail information before tail learning supervision. Additionally, as mentioned above, we use Average Precision over all classes, rather than top-k class, to *unmask long-tail performance losses*.

6.2.4 Negative and Positive Generation

As surveys like Musgrave et al. (2020a); Rethmeier and Augenstein (2021) point out, traditional contrastive learning research focuses on generating highly informative (hard) **negative samples**, since most contrastive learning objectives only use *a single positive learning sample* and b (bad) negative samples – Musgrave et al. (2020a) give an excellent overview. However, if too many negative samples are generated they can collide with positive samples, which degrades learning performance (Saunshi et al., 2019). More recent computer vision works like Khosla et al. (2020d); Ostendorff et al. (2022c) propose generating multiple **positive** samples to boost *supervised contrastive learning* performance, while Wang and Isola (2020) show that, when generating positive samples, the representations of positives should be close (related) to each other. Our method builds on these insights and extends them to *self-supervised contrastive learning* and to the language model domain using a straightforward extension to NCE. Instead of using only one positive example the like standard NCE by Mnih and Teh (2012), our method uses g good (positive) samples (see Section 6.3). To ensure that positive samples are representationally close (related) during self-supervised contrastive pretraining, we use words from a current input text as positive 'pseudo-labels' – i.e. we draw self-supervision pseudo-labels from a related context. Negative pseudo-labels (words) are drawn as words from other in-batch text inputs, where negative sample words are not allowed not appear in the current text to avoid the above-mentioned collision of positive and negative samples.

6.2.5 Data and parameter efficiency

Using CNN layers can improve data and compute efficiency over self-attention layers as found by various works (Rethmeier et al., 2020c; Kim et al., 2020; Tay et al., 2021). data-efficiency is paramount when pretraining while data is limited, which, for (rare) long-tail information, is by definition, always the case. Radford et al. (2021) find that replacing a Transformer language encoder with a CNN backbone increases zero-shot data-efficiency 3 fold. We thus use a small CNN text encoder, while for more data abundant or short-tail pretraining scenarios a self-attention encoder may be used instead. **Our method is designed to increase self-supervision signal, i.e. by sampling more positive and negatives, to compensate for a lack of large pretraining data (signal) – since rare and long-tailed data is always limited**. It is our goal to skip compression and still train small, long-tail prediction capable models. Notably, CLESS pretraining does not require special learning rate schedules, residuals, normalization, warm-ups, or a modified optimizer as do many BERT variations (Devlin et al., 2019a; Liu et al., 2019b, 2020a).

6.2.6 Label Denoising

Label dropout of discrete $\{0, 1\}$ labels has been shown to increase label noise robustness by (Szegedy et al., 2016). We use dropout on both the dense text and label embeddings. This creates a 'soft', but dense label noise during both self-supervised and supervised training, which is also similar to sentence similarity pretraining by Gao et al. (2021a), who used text embedding dropout rather than label embedding dropout to generate augmentations for contrastive learning.

6.3 CLESS: Unified Contrastive Self-supervised to Supervised Training and Inference

As done in natural language usage, we express labels as words, or more specifically as word embeddings, rather than as $\{0, 1\}$ label vectors. CLESS then learns to contrastively (mis-)match $<$text embedding, (pseudo/real) label embedding$>$ pairs as overviewed in Figure 6.1. For self-supervised pretraining, we in-batch sample g (good) positive and b (bad) negative $<$text, pseudo label$>$ embedding pairs per text instance to then learn good and bad matches from them. Positive pseudo labels are a sampled subset of words that appear in the current text instance. Negative pseudo labels are words sampled from the other texts within a batch. Crucially, negative words (pseudo labels) can not be the same words as positive words (pseudo labels) – i.e. $\mathbf{w}_i^+ \cap \mathbf{w}_j^- = \emptyset$.

This deceptively simple sampling strategy ensures that we fulfill two important criteria for successful *self-supervised contrastive learning*. One, using multiple positive labels improves learning if we draw them from a similar (related) context, as Wang and Isola (2020) proved. Two, we avoid collisions between positive and negative samples, which otherwise degrades learning when using more negatives as Saunshi et al. (2019) find. Similarly, for supervised learning, we use g positive, real labels and undersample b negative labels to construct <text, positive/negative real label> pairs. A text-2-label classifier ⑤ learns to match <text, label> embedding pairs using a noise contrastive loss (Ma and Collins, 2018), which we extend to use g positives rather than just one. This unifies self-supervised and supervised learning as contrastive 'text embedding to (label) text embedding matching' and allows direct transfer like zero-shot predictions of real labels after pseudo label pretraining– i.e. without prior training on other real labels as required by methods like (Zhang et al., 2018; Pappas and Henderson, 2019; Jiang et al., 2019b). Below, we describe our approach and link specific design choices to insights from existing research in steps ①-⑥.

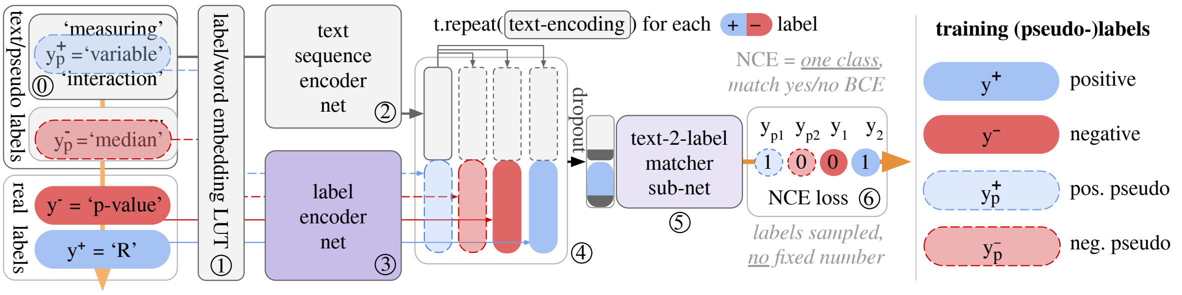


Figure 6.1: Contrastive <text, pseudo/real label> embedding pair matcher model: A word embedding layer E ① embeds text and real/pseudo labels, where labels are word IDs. CLESS embeds a text ('measuring variable interaction'), real positive (R) or negative (p-value) labels, and positive (variable) or negative (median) pseudo labels. A sequence encoder T ② embeds a single text, while a label encoder L ③ embeds c labels. Each text has multiple (pseudo) labels, so the text encoding $\mathbf{t}_i$ is repeated for, and concatenated with, each label encoding $\mathbf{l}_{i,l}^{\circ}$. The resulting batch of <text embedding, label embedding> pairs $[[\mathbf{t}_i, \mathbf{l}_{i,1}^{\circ}], \ldots, [\mathbf{t}_i, \mathbf{l}_{i,c}^{\circ}]]$ ④ are fed into a 'matcher' classifier ⑤ that is trained in ⑥ as a binary noise contrastive estimation loss L_B (Ma and Collins, 2018) over multiple label (mis-)matches $\{0, 1\}$ per text instance $\mathbf{t}_i$. Unlike older works, we add contrastive self-supervision over pseudo labels as a pretraining mechanism. Here, the word 'variable' is a positive self-supervision (pseudo) label for a text instance $\mathbf{t}_i$, while words from other in-batch texts, e.g. 'median', provide negative pseudo labels.

We give the model a text instance i of words $\mathbf{w}_i$ and a set of positive and negative label words $\mathbf{w}_i^{\circ} = \mathbf{w}_i^{+} \oplus \mathbf{w}_j^{-} \in \mathbb{R}^{c=g+b}$. We also construct a label indicator $\mathbb{I}_i$ as ground truth labels for the binary NCE loss in ⑥. This label indicator contains a g-sized vector of ones $\mathbf{1} \in \mathbb{N}_0^g$ to indicate positive (matching) <text, label> embedding pairs and a b-sized zero vector $\mathbf{0} \in \mathbb{N}_0^b$ to indicated mismatching pairs, resulting in the indicator

$$\mathbb{I}_i = \{\mathbf{1} \oplus \mathbf{0}\} \in \mathbb{N}_0^{c=g+b} \tag{⓪}$$

CLESS then encodes input text and labels in three steps ①-③. First, both the input text (words) $\mathbf{w}_i$ and the labels $\mathbf{w}_i^{\circ}$ are passed through a shared embedding layer ① to produce $E(\mathbf{w}_i)$ as text embeddings and $E(\mathbf{w}_i^{\circ})$ as label embeddings. Then, the text embeddings are encoded via a text encoder T ②, while labels are encoded by a label encoder L as follows:

$$\begin{align} & E(\mathbf{w}_i), E(\mathbf{w}_i^{\circ}) \tag{①} \\ \mathbf{t}_i &= T(E(\mathbf{w}_i)) \tag{②} \\ \mathbf{L}_i^{\circ} &= L(E(\mathbf{w}_i^{\circ})) = [\mathbf{l}_{i,1}^{+}, \ldots, \mathbf{l}_{i,g}^{+}, \mathbf{l}_{i,1}^{-}, \ldots, \mathbf{l}_{i,b}^{-}] \tag{③} \end{align}$$

To make model learning more data-efficient we initialize the embedding layer E with fastText word embeddings that we train on the 60MB of *in-domain text data*. Such word embedding training only computes a few seconds, while enabling one to make the text encoder architecture small, but well initialized. The text encoder T consists of a single, k-max-pooled CNN layer followed by a fully connected layer for computation speed and data-efficiency (Simoncelli and Olshausen, 2001; Dosovitskiy et al., 2020; Radford et al., 2021). As a label encoder L, we average the embeddings of words in a label and feed them through a fully connected layer – e.g. to encode a label ‘p-value’ we simply calculate the mean word embedding for the words ‘p’ and ‘value’.

To learn whether a text instance embedding $\mathbf{t}_i$ matches any of the c label embeddings $\mathbf{l}_{i,\cdot}^{\circ} \in \mathbf{L}_i^{\circ}$, we repeat the text embedding $\mathbf{t}_i$, c times, and concatenate text and label embeddings to get a matrix $\mathbf{M}_i$ of <text, label> embedding pairs:

$$\mathbf{M}_i = [[\mathbf{t}_i, \mathbf{l}_{i,1}^{+}], \ldots, [\mathbf{t}_i, \mathbf{l}_{i,c}^{-}]] \tag{④}$$

This text-label paring matrix $\mathbf{M}_i$ is then passed to the matcher network M ⑤, which first applies dropout to each text-label embedding pair and then uses a three layer MLP to produce a batch of c label match probabilities:

$$\mathbf{p}_i = \{\sigma(M(\mathbf{M}_{i,1})), \ldots, \sigma(M(\mathbf{M}_{i,c}))\} \tag{⑤}$$

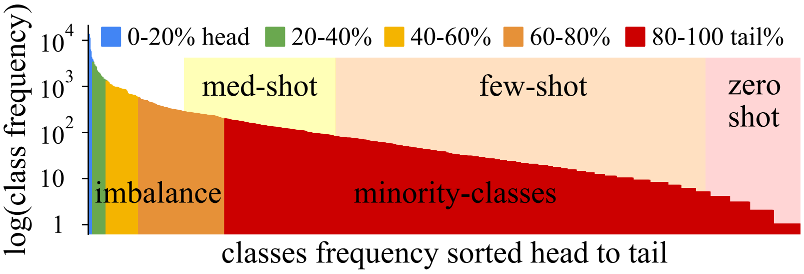


Figure 6.2: Head to long-tail as 5 balanced class bins: We bin classes by label frequency. Each bin contains equally many active label occurrences. Classes within a bin are imbalanced and become few-shot or zero-shot towards the tail, especially after train/dev/test splitting. Class frequencies are given in log scale – task data details in Section 6.4.

Here, applying dropout to label and text embeddings induces a *dense version of label noise*. Discrete {0,1} label dropout has been shown to improve robustness to label noise in Szegedy et al. (2016); Lukasik et al. (2020). Because we always predict correct pseudo labels in pretraining, this forces the classifier to learn to correct dropout induced label noise.

Finally, we use a binary noise contrastive estimation loss as in (Ma and Collins, 2018), but extend it to use g positives, not one.

$$L_B = -\frac{1}{c}\sum_{l=1}^{g+b=c} \mathbb{I}_{i,l} \cdot log(\mathbf{p}_{i,l}) + (1-\mathbb{I}_{i,l}) \cdot log(1-\mathbf{p}_{i,l}) \qquad ⑥$$

Here, L_B is the mean binary cross-entropy loss of g positive and b negative labels – i.e. it predicts $c{=}b{+}g$ label probabilities $\mathbf{p}_i$, where the label indicators $\mathbb{I}_i$ from ① are used as ground truth labels.

Though we focus on evaluating CLESS for long-tail prediction in this work, other NLP tasks such as question answering or recognizing textual entailment can similarly be modeled as contrast pairs $<X{=}\text{'}text\ 1\ [sep]\ text\ 2\text{'}, Y{=}\text{'}is\ answer\text{'}>$. Unlike T5 language models (Raffel et al., 2020), this avoids translating back and forth between discrete words and dense token embeddings. Not using T5s' softmax objective, also allows for predicting unforeseen (unlimited) test classes (label). We provide details on hyperparameter tuning of CLESS for self-supervised and supervised learning in the Appendix.

6.4 Data: resource constrained, long-tail, multi-label, tag prediction

To study efficient, small model, long-tail learning for ‘text-to-text’ pretraining models, we choose a multi-label question tag prediction dataset as a testbed. We use the “Questions from Cross Validated” dataset, where machine learning concepts are tagged per question – – https://www.kaggle.com/stackoverflow/statsquestions, accessed on 30th of Aug 2021. This dataset is small (80MB of text), and entails solving a challenging ‘text-to-text’ long-tailed prediction task. The dataset has 85k questions with 244k positive labels, while we do not use answer texts. As with many real-world problems, labels are vague, since tagging was crowd-sourced. This means that determining the correct amount of tags per question (label density) is hard, even for humans. The task currently has no prior state-of-the-art. As seen in Figure 6.2, the datasets’ class occurrence frequencies are highly long-tailed, i.e. the 20% most frequently occurring classes result in 7 ‘head’ classes, while the 20% least frequent (rightmost) label occurrences cover 80% or 1061/1315 of classes. Tags are highly sparse – at most 4 out of 1315 tags are labeled per question. We pretrain fastText word embeddings on the unlabeled text data to increase learning efficiency, and because fastText embeddings only take a few seconds to pretrain. The full details regarding preprocessing can be found in the Appendix.

Long-tail evaluation metrics and challenges: Long-tail, multi-label classification is challenging to evaluate because (i) top-k quality measures mask performance losses on long-tailed minority classes as Hooker et al. (2020d) point out. Furthermore, (ii) measures like ROC_{AUC} overestimate performance under class imbalance (Davis and Goadrich, 2006; Fernández et al., 2018), and (iii) discrete measures like F-score are not scalable, as they require discretization threshold search under class imbalance. Fortunately, the Average Precision score $AP = \sum_n (R_n - R_{n-1}) P_n$ addresses issues (i-iii), where P_n and R_n are precision and recall at the nth threshold. We choose AP_{micro} weighting as this score variant is the hardest to improve.

6.5 Results

In this section, we analyze the three research questions: (RQ-1) Does RoBERTa learn long-tail tag prediction well? (RQ-2) Can a 12.5x smaller CLESS model achieve good long-tail prediction, and at what cost? (RQ-3) How does CLESS compare in zero to few-shot prediction and does its model size matter. We split the dataset into 80/10/10 for training, development, and test set. *Test scores or curves are reported for models that have the best development set average precision score* AP_{micro} *over all 1315*

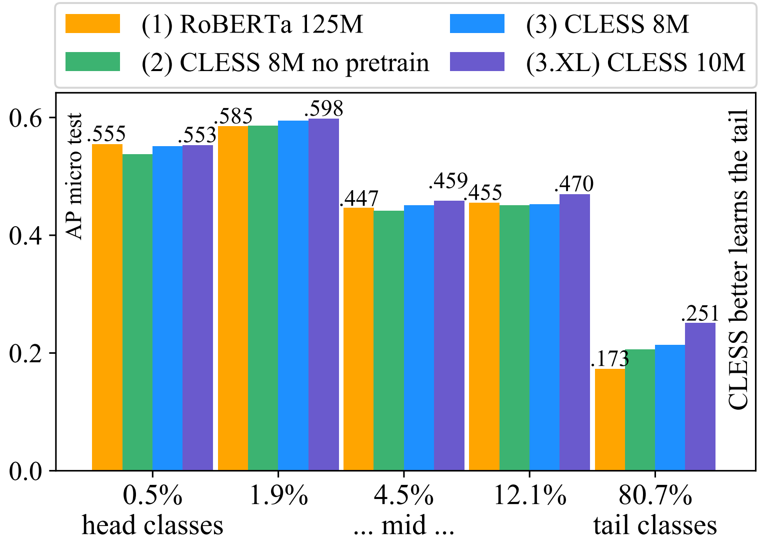


Figure 6.3: Long-tail performance (RQ-1, RQ-2), over all five head to tail class bins – see Figure 6.2. The tail class bin contains 80.7% or 1062/1315 of classes. The non-pretrained CLESS (2) underperforms, while RoBERTa performs the worst on the 80.7% of tail classes. The largest pretrained CLESS model (3.XL) outperforms RoBERTa in tail and mid class prediction, while performing nearly on par for the $7/1315{=}0.5\%$ (most common) head classes.

classes. RoBERTa has 125 million parameters and is pretrained on 160GB of text data. CLESS has 8-10 million parameters and is pretrained on just 60MB of in-domain text data. We use a ZeroR classifier, i.e. predicting the majority label per class, to establish imbalanced guessing performance. The ZeroR AP_{micro} on this dataset is 0.002 since a maximum of 4 in 1315 classes are active per instance – i.e. which underlines the challenge of the task.

6.5.1 (RQ-1+2): Long-tail Capture of RoBERTa vs. CLESS

Here we compare the long-tail prediction performance of RoBERTa (1) vs. CLESS setups that, either were pretrained (3, 3.XL), or not pretrained (2). Plotting individual scores for 1315 classes is unreadable. Instead, we sort classes from frequent to rare and assign them to one of five '20% of the overall class frequency' bins, such that all bins are balanced. This means all bins contain the same amount of positive real labels (label occurrences) and are directly comparable. As seen in Figure 6.2, this means that the head bin (left) contains the most frequent $7/1315{=}0.5\%$ classes, while the tail contains the most rarely occurring $1061/1315{=}80.7\%$ classes.

6.5.1.1 RoBERTa: a large pretrained model does not guarantee long-tail capture

Figure 6.3 shows how a tag prediction fine-tuned RoBERTa performs over the five class bins as described above or in Section 6.4. RoBERTa learns the most common ($.5\%$ head) classes well, but struggles with mid to tail classes. On the tail class bin, i.e. on $1061/1315{=}80.7\%$ of classes, RoBERTa performs worse than a CLESS model that did not use contrastive pretraining (2). This allows multiple insights. One, *a large PLM should not implicitly be assumed to learn long-tail information well*. Two, large-scale pretraining data should not be expected to contain enough (rare) long-tailed domain information for an arbitrary end-task, since in the tail-domain, data is always limited. Three, even a small supervised contrastive model, without pretraining, can improve long-tail retention (for 80.7% of classes). Together these results indicate that compressing a large PLM may not be the optimal approach to training a small, long-tail prediction capable model.

6.5.1.2 CLESS: contrastive pretraining removes the need for model compression

Model (3) and (3.XL) use our contrastive pretraining on the end-tasks' 60MB of unlabeled text data before supervised fine-tuning. We see that models with contrastive pretraining (3, 3.XL) noticeably outperform RoBERTa (1) and the non-pretrained contrastive model (2), on all non-head class bins, but especially on the 80.7% tail classes. We also see that the pretraining model parameter amount impacts CLESS performance as the 10 million parameter model (3.XL) outperforms the 8M parameters model (3) over all class bins and especially the tail bin. The above observations are especially encouraging as they tell us that contrastive in-domain pretraining can produce small, long-tail learning capable models without the need for compressing large models. It also tells us that model capacity matters in long-tail information retention, but not in the common sense that large PLMs are as useful as they have proven to be for non-long-tail learning applications. This also means that contrastive self-supervised LM pretraining can help reduce algorithmic bias caused by long-tail information loss in smaller models, the potential fairness impact of which was described by (Hooker et al., 2020a,d; Hooker, 2021).

6.5.1.3 Practical computational efficiency of contrastive language modeling

Though the long-tail performance results of CLESS are encouraging, its computational burden should ideally be equal or less than that of fine-tuning RoBERTa. When we analyzed training times we found that RoBERTa took 126 GPU hours to fine-tune for 48 epochs, when using 100% of fine-tuning labels. For the same task we found that CLESS (3.XL) took 7 GPU hours for self-supervised pretraining (without labels) and 5 GPU hours for supervised fine-tuning over 51 epochs – To bring CLESS to the same GPU compute load as RoBERTa ($\approx 96\%$) we parallelized our data generation – otherwise our training times double and the GPU load is only $\approx 45\%$. As a result, pretraining plus fine-tuning takes CLESS (3.XL) 12 hours compared to 126 for fine-tuning RoBERTa. This means that the proposed contrastive in-domain pretraining has both qualitative and computational advantages, while remaining applicable in scenarios where large collections of pretraining data are not available – which may benefit use cases like non-English or medical NLP. Additionally, both methods benefit from parameter search, but since CLESS unifies self-supervised pretraining and supervised fine-tuning as one objective we can reuse pretraining hyperparameters during fine-tuning. A more in-depth account of computational trade-offs is given in the Appendix , while details of hyperparameter tuning are given in the Appendix.

It is of course possible to attempt to improve the long-tail performance of RoBERTa, e.g. via continued pretraining on the in-domain data (Gururangan et al., 2020) or by adding new tokens (Pörner et al., 2020; Hoover et al., 2020). However, this further increases the computation and memory requirements of RoBERTa, while the model still has to be compressed – which requires even more computation. We also tried to further improve the embedding initialization of CLESS using the method described in (Rethmeier and Plank, 2019), to further boost its learning speed. While this helped learning very small models (<2M parameters), it did not meaningfully impact the performance of contrastive pretraining or fine-tuning.

6.5.2 (RQ-3.1-2): contrastive zero-shot long-tail learning

Thanks to the unified learning objective for self-supervised and supervised learning, CLESS enables zero-shot prediction of long-tail labels after self-supervised pretraining, i.e. without prior training on any labels. Therefore, in this section, we analyze the impact of using more model parameters (RQ-3.1) as well as using more pseudo labels (RQ-3.2) during self-supervised contrastive pretraining.

6.5.2.1 (RQ-3.1): More self-supervision and model size improve zero-shot long-tail capture

In Section 6.4, we study how CLESSs' zero-shot long-tail retention ability is impacted by: (left) using more pseudo labels (learning signal) during pretraining; and (right) by using only portions of unlabeled text data for pretraining. To do so, we pretrain CLESS variants on pseudo labels and evaluate each variant's zero-shot AP_{micro} performance

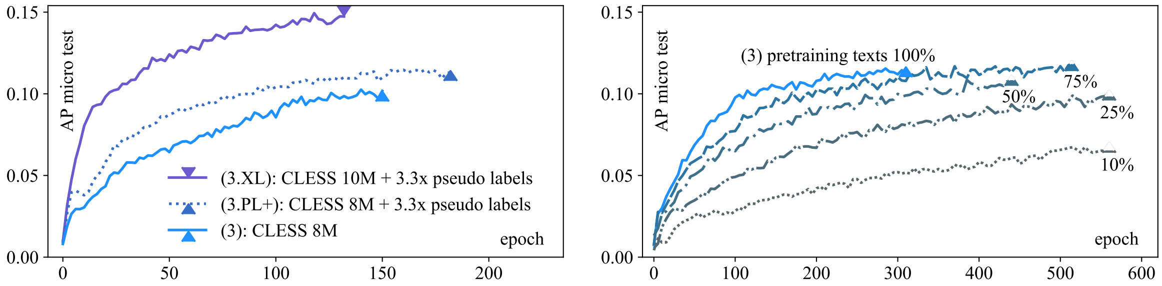


Figure 6.4: Zero-shot pretraining data-efficiency: by model size, pseudo label amount and pretraining text amount. Left: The zero-shot (data-efficiency) performance of the self-supervised pretraining base model (3) is increased when, adding more self-supervision pseudo labels (3.PL+) and when increasing model parameters (3.XL). Right: When only using only a proportion of the pretraining input data texts to pretrain model (3), its zero-shot learning is slowed down proportionally, but *still converges towards the 100% for all but the most extreme pretraining data reductions*.

over all 1315 classes of the real-label test set from Section 6.4. As before, we show test score curves for the models with the best AP_{micro} dev set performance.

The left plot of Figure 6.4, shows the effect of increasing the number of self-supervision pseudo label and model parameters. The CLESS 8M model (3), pretrained with 8 million parameters and 150 pseudo labels, achieves around $.10AP_{micro}$ on the test real labels as zero-shot long-tail performance. When increasing the pseudo label number to 500 in model (3.PL+), the model gains zero-shot performance (middle curve), without requiring more parameters. When additionally increasing the model parameters to 10M in (3.XL), the zero-shot performance increases substantially (top curve). Thus, both increasing self-supervision signal amount and model size boost zero-shot performance.

6.5.2.2 RQ-3.2: Contrastive pretraining leads to data-efficient zero-shot long-tail learning

Further, in the *right plot* of Figure 6.4 we see the CLESS 8M model (3) when trained on increasingly smaller portions $(100\%, \ldots, 10\%)$ of pretraining text. For all *but the smallest pretraining data portions ($< 25\%$)* the model still converges towards the original 100% performance. However, as expected, its convergence slows proportionally with smaller pretraining text portions since each data reduction implies seeing less pseudo label self-supervision per epoch. As a result, the data reduced setups need more training epochs, so we allowed $5x$ more waiting-epochs for early stopping than in the *left* plot. Thus, our contrastive self-supervised objective can pretrain data-effectively from very limited data. Similar data-efficiency gains from using contrastive objectives were previously only observed in computer vision applications by Zimmermann et al. (2021b), which confirms our initial intuition that contrastive self-supervision is generally useful for self-supervised learning from limited data.

Methods like Pappas and Henderson (2019); Jiang et al. (2019b); Augenstein et al. (2018) required supervised pretraining on real labels to later predict other, unseen labels in a zero-shot fashion. CLESS instead uses self-supervised pretraining to enable zero-shot prediction without training on real labels. This 'text-to-text' prediction approach is intentionally reminiscent of zero-shot prediction approaches in large PLMs like GPT-3 (Brown et al., 2020), but is instead designed to maximize zero-shot, long-tail prediction for use cases that strongly limit pretraining data amounts and model size. Hooker et al. (2020d) hypothesized that long-tail prediction depends on the model capacity (parameter amount). Additionally, Brown et al. (2020) found that zero-shot prediction performance depends on model capacity, but (Frankle and Carbin, 2019b; Rethmeier et al., 2020b) experimentally showed or visualized how inefficiently model capacity is used by common models, especially after fine-tuning. From the above observations, we can confirm the impact of model size for the doubly challenging task of *long-tail, zero-shot prediction*, but we can also confirm that contrastive pretraining allows a model to much more efficiently use its capacity for long-tail capture, i.e. requiring 12.5x fewer parameters (capacity) than common the RoBERTa model. Perhaps more encouragingly, we also observed that cheap, contrastive in-domain pretraining boosts zero-shot prediction, even when pretraining data is very limited – i.e. either by lack of large domain text data or due to data limitations caused by a long-tail distribution.

6.5.3 (RQ-3.3): few-shot long-tail learning

Since CLESS models allow direct transfer (reuse) of the pretrained prediction head for supervised label prediction one would also expect the models' few-shot long-tail prediction performance to benefit from self-supervised pretraining. We thus study the few-shot learning performances of both CLESS and RoBERTa, to understand differences in large pretrained language models (PLMs) and small contrastive language model (CLM) pretraining in more detail. For the few-shot setup, we use 100%, 50% and 10% of labeled text instances for supervised training or fine-tuning of all models. This implies that if labels were common in the 100% setup, they now become increasingly rare or few-shot in the 10% setup, since the smaller label sets are still long-tail distributed. We again use AP_{micro} test set performance over all 1315 classes to compare models.

In Figure 6.5, we see that when using full supervision (100%), all models perform similarly, with CLESS (3.XL) slightly outperforming RoBERTa (.493 vs. .487) AP_{micro_test}. For few-shot learning (10%, 50%), we see that CLESS 3.XL retrains

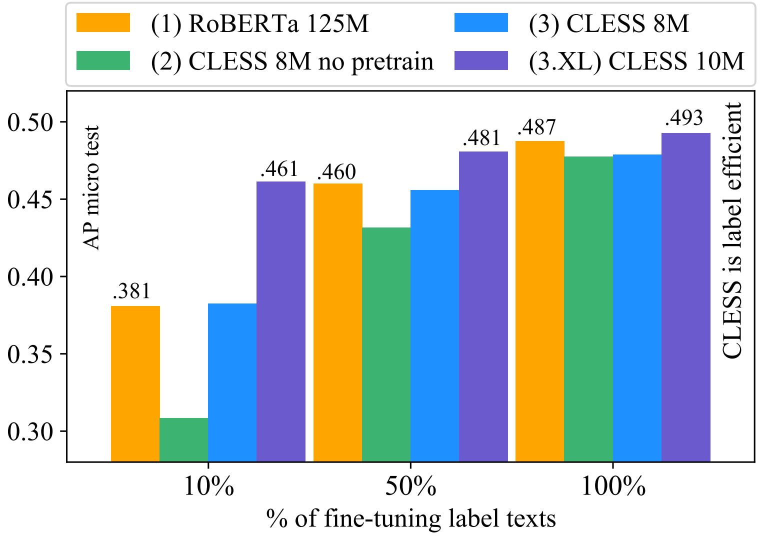


Figure 6.5: (RQ-3.3) Few-shot label-efficiency: (1) RoBERTa. (2) CLESS without pretraining. (3) CLESS with pretraining. (3.XL) CLESS pretrained with more pseudo labels and model parameter as described in (Section 6.5.2). AP_{micro_test} scores for few-shot portions: 100%, 50%, 10% of training samples with real labels. CLESS 10M outperforms RoBERTa, and retrains 93.5% of its long-tail performance using only 10% of fine-tuning label texts.

$.461/.493{=}0.935\%$ of its original performance when using only 10% of fine-tuning labels, while RoBERTa and CLESS 8M each retain around 77%. This demonstrates that even a sightly larger contrastive pretraining model, with increased self-supervision signal (3.XL), not only improves zero-shot learning performance as was seen in Figure 6.4, but also markedly boosts few-shot performance. Noticeably, the only non-pretrained model (2), performs much worse than the others in the more restricted few-shot scenarios. Since models (2) and (3) use the same hyperparameters and only differ in being pretrained (3) or not being pretrained (2), this demonstrates that contrastive self-supervised pretraining largely improves label efficient learning.

6.6 Conclusion

We introduce CLESS, a contrastive self-supervised language model (CLM), that unifies self-supervised pretraining and supervised fine-tuning into a single contrastive 'text embedding to text embedding' matching objective. Through three research questions (RQ-1 to RQ-3) we demonstrate that this model learns superior zero-shot, few-shot, and fully supervised long-tail retention in *small models without needing to compress large models*. In RQ-1, we first show that a fine-tuned, large pretrained language model like RoBERTa should not implicitly be expected to learn long-tail information well. Then, in RQ-2, we demonstrate that our contrastive self-supervised pretraining objective enables very text data-efficient pretraining, which also results in markedly improved (label efficient) few-shot or zero-shot long-tail learning. Finally, in RQ-3, we find that using more contrastive self-supervision signals and increasing model parameter capacity play important roles in boosting zero to few-shot long-tail prediction performance when learning from very limited in-domain pretraining data. We also find that the very low compute requirements of our method make it a viable alternative to large pretrained language models, especially for learning from limited data or in long-tail learning scenarios, where tail data is naturally limited. In future work, we envision applying CLESS to low-data domains like medicine (Rethmeier et al., 2020c) and fact-checking (Augenstein et al., 2019), or to tasks where new labels emerge at test time, e.g. hashtag prediction (Ma et al., 2014).

Acknowledgments

This research was conducted within the Cora4NLP project, which is funded by the German Federal Ministry of Education and Research (BMBF) under funding code 01IW20010.

Table 6.1: Time complexity O(Layer), data-efficiency, number of trainable parameters, number of all parameters. The data-efficiency of Convolutions (*) is reported in various works to be superior to that of self-attention models (Liu et al., 2020b; Yogatama et al., 2019a; Merity et al., 2017a; Wang et al., 2020a; Kim et al., 2020; Bender et al., 2021a; Radford et al., 2021). d is the input embedding size and its increase slows down convolutions. n is the input sequence length and slows down self-attention the most (Vaswani et al., 2017a). There exist optimizations for both problems.

Layer Type	$O(Layer)$	Reported Data Requirements	Trainable Parameters
Convolution	$O(n \cdot d^2)$	small (*)	8M-10M (CLESS)
Self-Attention	$O(n^2 \cdot d)$	large to web-scale (*)	125M (RoBERTa)

6.7 Appendices

6.7.1 Text Processing Details

We decompose tags such as 'p-value' as 'p' and 'value' and split latex equations into command words, as they would otherwise create many long, unique tokens. In the future, character encodings may be better for this specific dataset, but that is out of our current research scope. Words embedding are pretrained via fastText on the training corpus text. 10 tag words are not in the input vocabulary and thus we randomly initialize their embeddings. Though we never explicitly used this information, we parsed the text and title and annotated them with 'HTML-like' title, paragraph, and sentence delimiters, i.e. *</title>, </p>, and </s>*.

6.7.2 Complexity

Here we will discuss the time and transfer complexity of CLESS vs. Self-attention models. We do so since time complexity is only meaningful if the data-efficiency of two methods is the same, because the combination of convergence speed, computation speed, and end-task performance makes a model effective and efficient.

6.7.2.1 Time complexity:

Our text encoder uses a single 1D CNN encoder layer which has a complexity of $O(n \cdot k \cdot d \cdot f)$ vs. $O(n^2 \cdot d)$ for vanilla self-attention as outlined in Vaswani et al. (2017a). Here n is the input sequence length, k is the convolution filter size, d is the input embedding dimension [$d = 512$ in (Vaswani et al., 2017a) vs. $d = 100$ for us], and f is the number of convolution filters (at maximum $f = 3 \cdot 100$ for our (3.XL) pretraining model). Since we use kernel sizes $\{1, 2, 3\}$ we get for the largest configuration (3.XL) an $O(n \cdot k = 6 \cdot d = 1 \cdot f = 3d) \approx O(n \cdot 3d^2)$ vs. $O(n^2 \cdot 5d)$ in a vanilla (2017) self-attention setup where d=512. Furthermore Transformer self-attention runs an n-way soft-max computation at every layer (e.g. 16 layers), while we run $g \cdot b$

single-class predictions at the final output layer using a noise contrastive objective NCE. We use NCE to undersample both: true negative learning labels (label=0) as well as positive and negative pseudo labels (input words). If the goal is to learn a specific supervised end-task, more informed sampling of positive and negative pseudo labels can be devised. However, we did not intend to overfit the supervised task by adding such hand-crafted human biases. Instead we use random sampling to pretrain a model for arbitrary downstream tasks (generalization), which follows a similar logic as random masking does in masked language modeling.

6.7.2.2 Transfer complexity

: Traditional transfer NLP approaches like RoBERTa (Liu et al., 2019b) need to initialize a new classification head per task which requires either training a new model per task or a joint multi-task learning setup. CLESS however can train multiple tasks, even if they arrive sequentially over time, while reusing the same classifier head from prior pretraining or fine-tuning. Thus, there is no need to retrain a separate model each time as in current Transformer transfer models. Once pretrained a CLESS model can zero-shot transfer to any new task since the match classifier is reused.

6.7.3 Hyperparameters

In this section, we describe the data and memory efficiency of the proposed method as well as the hyperparameter tuning we conducted.

6.7.3.1 Data, sample and memory efficiency:

We analyzed input data and label efficiency in the main documents zero and few-shot learning sections. Regarding data-efficiency and model design choices we were guided by the existing research and optimized for data-efficient learning with inherent self-supervised zero-shot capabilities in order to facilitate and study supervision-free generalization to unforeseen tasks. We explain the origins of these design choices in more detail below. As mentioned in the related research section, Transformers rely on large to Web-scale pretraining data collections 'end-task external pretraining data' (Liu et al., 2020b; Yogatama et al., 2019a), which results in extensive pretraining hardware resources (Hooker, 2020; Dodge et al., 2020), concerns about environmental costs (Strubell et al., 2019a; Bender et al., 2021a) and unintended contra-minority biases (Mitchell et al., 2020; Waseem et al., 2020; Bender et al., 2021a). CNNs have been found to be more data-efficient than Transformers, i.e. train to better performance with less data, several works. For example in OPENAI's CLIP model, see Figure 2 in (Radford et al., 2021), the authors find that replacing a Transformer language model backbone with a CNN backbone increased the zero-shot data-efficiency 3 fold, which they further increased by adding a *supervised* contrastive learning objective. (Dosovitskiy et al.,

2020) showed that adding a CNN component to a vision Transformer model helps with data and computational efficiency, see Figure 5 and text in (Dosovitskiy et al., 2020). When comparing works on small-scale data pretraining capabilities between (Merity et al., 2017a) (CNN, LSTM) with recent Transformer models Wang et al. (2020a), one can see that Transformer encoders struggle to learn from small pretraining collections. They also struggle to fine-tuning on smaller supervised collections (Liu et al., 2020a; Dodge et al., 2020; Rogers et al., 2020). For CLESS, tuning the embedding layer made little difference to end-task performance, when starting training with pretrained fastText word embedding. Thus embedding tuning the embedding layer can be turned off to reduce gradient computation and memory. For example, when not tuning embeddings, the CLESS 10M model has only 3.2M trainable parameters.

Filter size: num filters	{1: 57, 2: 29, 3: 14}, {**1**:100, **2**:100, **1**:100},{1: 285, 2: 145, 3: 70}, {1:10, 10:10, 1:10}, {1:15, 2:10, 3:5}, {1:10}, {1:100}, {10:100}
lr	0.01, 0.0075, 0.005, ***0.001***, 0.0005, 0.0001
bs (match size)	**1024**, *1536*, 4096
max-k	1, 3, *7*, **10**
match-classifier	***two_layer_classifier***, 'conf':[{'do': None|**.2**, 'out_dim': **2048**|4196|*1024*}, {'do':***None***|0.2}]}, one_layer_classifier, 'conf':[{'do':.2}]}
label encoder	*one_layer_label_enc, 'conf':[{'do':* None|*.2, 'out_dim': 100*}, one_layer_label_enc, **'conf':[{'do': .2, 'out_dim': 300}**
seq encoder	*one_layer_label_enc, 'conf':[{'do':* None|*.2, 'out_dim': 100*}, one_layer_label_enc, **'conf':[{'do': .2, 'out_dim': 300}**
tune embedding:	True, False
#real label samples:	20, *150*, **500** (g positives (as annotated in dataset), b random negative labels – 20 works well too)
#pseudo label samples:	20, 150, **500** (g positives input words, b negative input words) – used for self-superv. pretraining
optimizer:	**ADAM** – default params, except lr

Table 6.2: **Explored parameters**. We conducted a random grid search over the following hyperparameters while optimizing important parameters first to largely limit trials. We also pre-fit the filter size, lr, and filters on a 5k training subset of samples to further reduce trails. Then, to further reduce the number of trials, we tuned in the following order: learning rate lr, filter sizes f, max-k pooling, tuning embeddings, batch size bs, and finally the depth of the matching-classifier MLP. This gave us a baseline model, (2) CLESS 8M, that does not use pretraining to save trials and compute costs, but could be used to build up into the self-supervised pretraining models (3) and (3.XL) by increasing self-supervision and model size. Fortunately, RoBERTa has established default parameters reported in both its code documentation (https://github.com/pytorch/fairseq/tree/master/examples/roberta) (accessed on 30 Sept. 2021) and the https://simpletransformers.ai (accessed on 30 Sept. 2021) version, where we varied batch size, warmup, and learning rate around the default setting of these sources. Below we give the search parameters for CLESS. For CLESS 8M (2,3) the best params are *italic* and for CLESS 10M (3.XL) the best params are **bold**

6.7.3.2 Parameter tuning + optima (2)-(3.XL)

We provide detailed parameter configurations as python dictionaries for reproducibility in the code repository within the `/confs` folder. In 6.2 we see how the hyperparameters explored in CLESS – the optimal CLESS 3.XL parameters are marked in bold. The baseline CLESS configuration (2) hyperparameters were found as explained in the table, using the non-pretraining CLESS 8M (2) model – its best parameters are *italic*. We found these models by exploring hyperparameters that have been demonstrated to increase generalization and performance in (Jiang et al., 2020a; He et al., 2019). To find optimal hyperparameter configurations for the baseline model (2) we ran a random grid search over the hyperparameter values seen in 6.2. For the baseline CLESS 8M model (2), without pretraining, we found optimal hyperparameters to be: $lr = 0.001$ (lr=0.0005 works too), $filter\_sizes\_and\_number = \{1 : 100, 2 : 100, 3 : 100\}$, $match\_classifier$=two_layer_classifier, 'conf':[{'do': None|.2, 'out_dim': 2048 | 4196 | 1024}, $max\_k_pooling$=7, bs=1536, etc. – see table 6.2. Increasing the filter size, classifier size, its depth, or using larger k in k-max pooling decreased dev set performance of the non-pretrained model (i.e., CLESS 8M) due to increased overfitting. The largest pretrained CLESS 10M (3.XL) model was able to use more: 'max-k=10', a larger 'label' and 'text sequence encoder'= one_layer_label_enc, 'conf':[{'do': .2, 'out_dim': 300} while the batch size shrinks to 1024 due to increased memory requirements of label matching. Note that label and text encoder have the same output dimension in all settings – so text and label embeddings remain in the same representation dimensionality $\mathbb{R}^{300}$. The label encoder averages word embeddings (average pooling), while the text encoder uses a CNN with filters as in 6.2. The model receives text word ids and label-word ids, that are fed to the 'text encoder' and 'label-encoder'. These encoders are sub-networks that are configured via dictionaries to have fully connected layers and dropout, with optimal configurations seen in the table. As the match-classifier, which learns to contrast the (text embedding, label embedding) pairs, we use a two_layer_MLP which learns a similarity (match) function between text embedding to label embedding combinations (concatenations).

During self-supervised pretraining, the models (3) and (3.XL) optimize for *arbitrary unforeseen long-tail end-tasks*, which allows zero-shot prediction without ever seeing real labels, but also uses a very diverse learning signal by predicting sampled positive and negative input word embeddings. If the goal is to solely optimize for a specific end-task, this self-supervision signal can be optimized to pretrain much faster, e.g. by only sampling specific word types like nouns or named entities. With specific end-task semantics in mind, the pseudo label and input manipulations can easily be adjusted. This allows adding new self-supervision signals without a need to touch the model's network code directly, which helps ease application to new tasks and for less experienced machine learning practitioners. Finally, we mention implementation features, that can safely be avoided to reduce computation and optimization effort, so that following research needs not explore this option. When training the supervised and self-supervised loss at the same time (jointly), CLESS rescales both batch losses to be of the same loss value as using a single loss. This makes it easy to balance (weight) the two loss contributions in learning, and allows transferring hyperparameters between self-supervised and supervised pretraining. We also allow re-weighting the loss balance by a percentage, so that one loss can dominate. However, we found that in practice: (a) using the self-supervised loss along with the supervised one does not improve quality, but slows computation (2 losses). (b) We also found that, if one decides to use joint self and supervised training, loss re-weighting had no marked quality effects, and should be left at 1.0 (equal weighting), especially since it otherwise introduces further, unnecessary hyperparameters. For pretraining research, hyperparameter search is very involved, because we deviate in common practice by introducing a new architecture, a new loss variation, an uncommon optimization goal and metrics as well as a new dataset. Thus we ended up with 205 trails for small test set, RoBERTa, CLESS variants, zero-shot and few shot hyperparameter search. On the herein reported dataset, we have not yet tested further scaling up model parameters for pretraining as this goes against the goal of the paper and is instead investigated in followup work. Furthermore, when we ran such parameter scale-up experiments, to guarantee empirical insights, these created a significant portion of trails, meaning that, now that sensible parameters are established, we can use much fewer trials, as is the case with pretrained transformers. The work at hand suggest that, once sensible parameters are established, they are quite robust, such that doubling the learning rate, batch size and loss weighting only cause moderate performance fluctuations. Finally, the above reported pretraining hyperparameters seem to work well on currently developed followup research, that uses other, even much larger, datasets. This makes the 205 hyperparameter trials a one time investment for initial pretraining hyperparameter (re)search for this contrastive language model (CLESS).

7 Paper 6: MoRTy: Unsupervised Learning of Task-specialized Word Embeddings by Autoencoding

7.1 Introduction

Word embeddings are ubiquitous in Natural Language Processing. They provide a low-effort, high pay-off way to improve the performance of a specific supervised end-task by transferring knowledge. However, recent works indicate that universally best embeddings are not yet possible (Bollegala and Bao, 2018; Kiela et al., 2018a; Dingwall and Potts, 2018), and that they instead need to be tuned to fit specific end-tasks using inductive bias – i.e., semantic supervision for the unsupervised embedding learning process (Conneau et al., 2018; Perone et al., 2018). This way, embeddings can be tuned to fit a specific single-task (ST) or multi-task (MT: set of tasks) semantic (Xiong et al., 2018).

Fine-tuning requires labeled data, which is often either too small, not available or of low quality and creating or extending labeled data is costly and slow. Word embeddings are typically induced from huge unlabeled corpora with billions of tokens, but for limited-resource domains like biology or medicine, it becomes less clear whether there is still transfer. We set out to create task-specified embeddings cheaply, with self-supervision, that are able to provide consistent improvements, even in limited resource settings.

We evaluate the impact of our method, named MoRTy, on 18 publicly available benchmark tasks developed by Jastrzebski et al. (2017)[1] using two ways to induce embeddings, Fasttext and GloVe. We test them in two setups corresponding to two different overall aims: (a) to specialize embeddings to better fit a *single* supervised task or, (b) to generalize embeddings for *multiple* supervised end-tasks, i.e., to optimize MoRTys for *single* or *multi-task* settings. Since most embeddings are pre-trained on large corpora, we also investigate whether our method further improves embeddings trained on small corpus setups.

[1] https://github.com/kudkudak/word-embeddings-benchmarks

Hence, we demonstrate the method's application for single-task, multi-task, small, medium and web-scale (common crawl) corpus-size settings (Section 7.4). *Learning to scale-up* by pretraining on more (un-)labeled data is both: (a) not always possible in low-resource domains due to lack of such data, and (b) heavily increases the compute requirements of comparatively small supervised down-stream task. This not only leads to high per model-instance costs but also limits *learning to scale-out*, i.e., when combining many smaller models into a larger dynamic model as is desirable in continual learning settings, where models, inputs and objectives may emerge or disappear over time. To provide an alternative in such settings we design MoRTy as a *learning-to-scale-down* approach, that uses less data and compute to achieve a performance improvement despite *forgoing (un-)supervised fine tuning* on target domain data. Consequently, MoRTy uses very little resources,[2] producing a low carbon footprint, especially regarding recent, compute intensive, *scale-up* approaches like ELMo or BERT (Peters et al., 2018; Devlin et al., 2018) which have high hardware and training time requirements and a large carbon footprint as recently demonstrated by Strubell et al. (2019b). As a result, we demonstrate a simple, unsupervised scale-down method, that allows further pretraining exploitation, while requiring minimum extra effort, time and compute resources. As in standard methodology, optimal post-processed embeddings can be selected according to multiple proxy-tasks for overall improvement or using a single end-task's development split—e.g., on a fast baseline model for further time reduction.

7.2 MoRTy embeddings

Our proposed post-processing method provides a Menu **o**f Reconstructing Transformations to yield improved end-task performance (MoRTy).

Approach: The key idea of MoRTy is to create a family of embeddings by learning to reconstruct the original pre-trained embeddings space via autoencoders.

The resulting family or representations (post-processed embeddings) gives a "menu" which can be picked from in two ways: (a) standard development set tuning, to gain performance at a *single* supervised task (ST), or (b) via benchmark tasks, to boost performance of *multiple tasks* (MT). The first is geared towards optimizing embeddings for a single specific task (specialization), the latter aims at embedding generalization, that works well across tasks.

In more details, the overall MoRTy recipe is: **(1) Train (or take)**: an original (pre-trained) embedding space E_{org} using embedding method f. **(2) Reconstruct** E_{org}:

[2] $<$ 1GB memory including the whole dataset, computes fast on GPU and CPU and inherits FastText's dynamic out-of-vocabulary token embedding generation, which is useful in handling unforeseen words in down-stream tasks.

compute multiple randomly initialized representations of E_{org} using a reconstruction loss (mean square error, cf. below). **(3) Pick:** performance-optimal representation for the end-task(s) via a task's development split(s) or proxy tasks, depending on the end-goal, i.e., specialization or generalization. **(4) Gain:** use optimal MoRTy (E_{post}) to push relative performance on end task(s).

Which autoencoder variant? For step (2), we found the following autoencoder recipe to work best: A linear autoencoder with one hidden layer, trained via bRMSE (batch-wise root mean squared error), the same hidden layer size as the original embedding model and half of its learning rate[3]– i.e., *a linear, complete autoencoder*, trained for a single epoch (cf. end of Section 7.3).

We experimented with alternative autoencoders: sparse (Ranzato et al., 2007), denoising, discrete (Subramanian et al., 2018), and undercomplete autoencoders, but found the simple recipe to work best. In the remainder of the paper, we test this 'imitation-scheme' setup recipe.

7.3 Experiments

With the aim of deriving a simple yet effective 'best practice' usage recipe, we evaluate MoRTy as follows: a) using two word embedding methods f; b) corpora of different sizes to induce E_{org}, i.e., small, medium and web-scale; c) evaluation across 18 semantic benchmark tasks spanning three semantic categories to broadly examine MoRTy's impact, while assessing both single and multi-task end goals; and finally e) evaluate 1-epoch setups in relation to different corpus sizes.

Embeddings and Corpus Size: We evaluate embeddings trained on small, medium (millions of tokens) and large (billions of tokens) corpus sizes. In particular, we train 100-dimensional embeddings with Fasttext (Bojanowski et al., 2016)[4] and GloVe (Pennington et al., 2014)[5] on the 2M and 103M WikiText created by Merity et al. (2016). We complement them with off-the-shelf web-scale Fasttext and GloVe embeddings (trained on 600B and 840B tokens, respectively). This results in the following vocabulary sizes for Fasttext and GloVe embeddings, respectively: on 2M 25,249 and 33,237 word types. For 103M we get 197,256 and 267,633 vocabulary words. Public, off-the-shelf – common-crawl trained – Fasttext and GloVe embeddings have very large vocabularies of 1,999,995 and 2,196,008 words.

[3] Original Fasttext and GloVe used $lr = 0.05$, so $lr \approx 0.025$ is a 'careful' rate and used throughout the experiments in this paper.

[4] To train Fasttext we used `https://fasttext.cc`

[5] To train GloVe we used the python glove_python wheel

To account for variation in results, we train both embedding methods *five times each*[6] on the two WikiText corpus sizes. We observed only minor variations, $< 0.5\%$ between runs for both Fasttext and GloVe, in overall performance Σ – i.e., when summing the scores of all benchmark tasks.

Semantic benchmark tasks: We use a publicly available word embedding benchmark implementation developed by Jastrzebski et al. (2017) – chosen for reproducibility and breadth. The 18 tasks span three semantic categories: (a) word similarity (6 tasks), (b) word analogy (3 tasks), and (c) word and sentence categorization (9 tasks).[7]

embedder model	Fasttext base performance			MT % change by 1 overall Morty			ST % change by 18 single Mortys			GloVe base performance			MT % change by 1 overall Morty			ST % change by 18 single Mortys		
train size	2M	103M	600B	2M	103M	600B	2M	103M	600B	2M	103M	840B	2M	103M	840B	2M	103M	840B
AP	0.31	0.59	0.68	-6.1	-0.9	-1.5	8.2	5.2	4	0.2	0.43	0.61	2.7	5.6	9.3	13.2	9.2	12.2
BLESS	0.3	0.73	0.84	-2.2	3.8	-3	13	9.7	5.4	0.27	0.51	0.85	1.6	-1.6	-1.8	7.9	7.9	4.7
Battig	0.14	0.32	0.48	-3.6	0.1	-3.7	7	4	0.5	0.1	0.19	0.46	3.5	2	1.9	7.4	5.4	8.5
ESSLI 1a	0.48	0.76	0.77	2.2	4.3	17.6	27.5	10.2	17.6	0.46	0.63	0.75	0	3.1	9.1	8	8.9	12.1
ESSLI 2b	0.63	0.75	0.78	9.2	2.7	0	26.5	11.3	12.9	0.51	0.74	0.75	19.9	-0.5	6.7	23.7	11.7	16.7
ESSLI 2c	0.54	0.54	0.62	-3.7	10.7	-10.7	11	19.7	10.7	0.46	0.54	0.62	2.1	2.7	0	16.9	16.7	10.7
Google	0.06	0.04	0.12	33.6	293.8	187.3	45.3	319.3	217.2	0	0.05	0.58	42.7	13.8	2.8	60.4	18.6	5.9
SEval 12 2	0.11	0.16	0.24	1.6	4.3	-2.8	18.1	14.1	4.8	0.11	0.15	0.2	6.5	2.2	1	11.4	5	2.4
MSR	0.28	0.08	0.18	18.8	246.2	117.1	27.5	267.3	137	0	0.09	0.57	45.6	30.9	-2.4	100.7	38.1	10.1
MTurk	0.24	0.52	0.73	65.6	5.1	1.1	98	12.6	1.5	0.3	0.46	0.69	-22.4	2.6	0.5	1.6	4.2	2.6
RG65	0.29	0.71	0.86	65.2	0.7	2.1	104.7	5.3	5.6	0.15	0.44	0.77	11.6	3.9	-1.3	30.8	10	4
RW	0.21	0.38	0.59	-17.1	-0.8	-2	4.1	2.4	0.9	0.2	0.21	0.46	-2.1	11.8	2	4	19.8	10.3
MEN	0.36	0.71	0.84	13	0.4	-0.4	22	2.3	0.3	0.16	0.51	0.8	3.6	5.6	0.5	15.1	7	7.7
SimLex999	0.18	0.31	0.5	-23.2	3.7	-1.2	7.3	9	3.1	0.03	0.22	0.41	147.8	7.3	3.1	228.3	11.7	9.3
TR9856	0.1	0.13	0.18	2.8	-4.1	-37.1	20.5	17.3	-2.5	0.09	0.08	0.1	13.9	8.9	-4.7	19.8	47.3	36.7
WS353	0.46	0.69	0.79	3.9	1	-1.7	10	2.9	0.6	0.16	0.45	0.74	31.5	7.2	0.7	36.8	8.2	5.6
WS353R	0.35	0.63	0.74	16.4	1.7	-2.8	24.3	4.1	1.6	0.08	0.4	0.69	53.1	6.5	1.1	62	8.2	2.7
WS353S	0.52	0.77	0.84	3.2	0.4	0.6	13.3	3	1.9	0.27	0.58	0.8	15.1	6.5	0.3	20.2	7.6	5.9
∑ tasks	5.55	8.83	10.79	8.9	5.8	3.4	8.9	5.8	3.6	3.56	6.68	10.84	7.8	4.3	1.9	7.8	4.3	1.9
category	2.39	3.7	4.17	-2.1	-0.2	1.8	11.4	4.5	3.1	2	3.04	4.05	3.5	-0.8	2.4	7.3	3.3	5.5
analogy	0.45	0.28	0.55	15.5	115	72.2	24.6	125.2	92.7	0.11	0.29	1.34	7.4	4.2	1.3	12.3	15.8	6.5
similarity	2.71	4.85	6.07	6.2	-0.6	-4.7	17.3	2.2	-0.3	1.45	3.35	5.45	9.2	1.8	0	11	6.3	2.9

legend	<50%	50%	>50%	< -10%	no change	> +10%	<50%	50%	>50%	< -10%	no change	> +10%

Table 7.1: **MORTY on Fasttext and GloVe**: Above are scores for: 18 individual tasks (AP-WS353S), the sum of 18 scores Σ, and scores grouped by semantic: similarity (AP-ESSLI2c), analogy (Google-MSR), classification (MTurk-WS253S). **Left column:** shows absolute scores of the original embedder. **Middle column:** shows % score change after fine-tuning with the MORTY that has the *highest overall score* Σ – i.e., 1 MORTY for all tasks (`multi-task`). **Right column:** shows % score change after applying 18 individually best MORTYs per `single-task` – i.e., 18 MORTYs . Each column is further split by corpus size – 2M, 103M(illion) and 600/840B(illion) tokens. All scores are averages over 5 original embedder scores and respective MORTY changes.

[6] Fasttext was trained using the implementation's (`fasttext.cc`) default parameters. GloVe was trained with the same parameters as in (Pennington et al., 2014) – Figure 4b. Though, 4a gave the same results.

[7] Jastrzebski et al. (2017) use measures form the dataset literature: Spearman correlation for similarity, 3CosAdd for analogy and accuracy and cluster purity for categorization.

Evaluation and Experimental Details For the single-task setup we show MORTY'S relative, percentual performance change (`ST % change`) produced by choosing the best MORTY embedding per task – 18 MORTYS. Correspondingly, for multi-task results we show `MT % change` obtained by choosing the MORTY embedding with the best score over all tasks Σ – i.e., one MORTY for all tasks. Performances in Table 7.1 are averaged over 5 runs each of Fasttext and GloVe per corpus size. To maximize MORTY'S usability we evaluate a **1-epoch** training scheme. We test its robustness – *particularly for limited resource use* – by training 1 epoch on three corpus sizes (small to web-scale), using the best multi-task (MT/ Σ) base embedder – see Fasttext Table 7.1. We again account for variation by using 3 randomly initialized MORTY runs, each over the 5 respective runs per corpus size. In this experiment, a single epoch yielded very stable boosts, that are comparable to multi-epoch training.

7.4 Results

The main results are provided in Table 7.1 and Figure 7.1. There are several take-aways.

f: **Fasttext and GloVe:** First, regarding the base embeddings (cf. per-category base performance scores in Table 7.1): i) we notice that Fasttext performs overall better than GloVe; ii) classification and similarity results improve the larger the corpus; consistently over f; and iii) GloVe is better for the analogy tasks on web-scale data.[8]

MORTY for multi-task application: Second, the `MT % change` columns show that a single best MORTY improves overall performance Σ (black row)[9] – the sum of 18 tasks – by 8.9, 5.8 and 3.4 percent compared to Fasttext base. As corpus size increases, there is less space for MORTY to improve Σ scores. What is interesting to note is that MORTY is able to recover analogy performance on 103M (to more than 2M level). This is also reflected in the Google and MSR analogy scores doubling and tripling (middle column). On 2M we also see a modest improvement (6.2) for similarity tasks, while classification on 2M slightly dropped. Regarding GloVe (3 rightmost columns) we notice lower overall performance (black column), which is consistent with findings by Levy et al. (2015). MORTY on GloVe produces lower but more stable improvements for the MT setting (middle column), with analogy and similarity performance noticeably increasing for the small 2M dataset. Generally, we see both performance increases and drops for individual task, especially on 2M and Fasttext, indicating that, a single overall best MORTY specializes the base Fasttext embedding

[8] GloVe 3CosAdd matches (Levy and Goldberg, 2014).

[9] Note that, % change for Σ is not the average of the individual task changes, but the % change of the sum of 18 individual scores.

to better fit a specific subset of the 18 tasks, while still beating the base embedders f in overall score (Σ).

MoRTy for single-task application: In the `ST % change` columns we see best single task (ST) results for task-specific optimal MoRTy embeddings. Both embedders get consistent boosts, with Fasttext exhibiting significantly higher improvement from MoRTy on 2M and 103M, despite already starting out at a higher base performance.

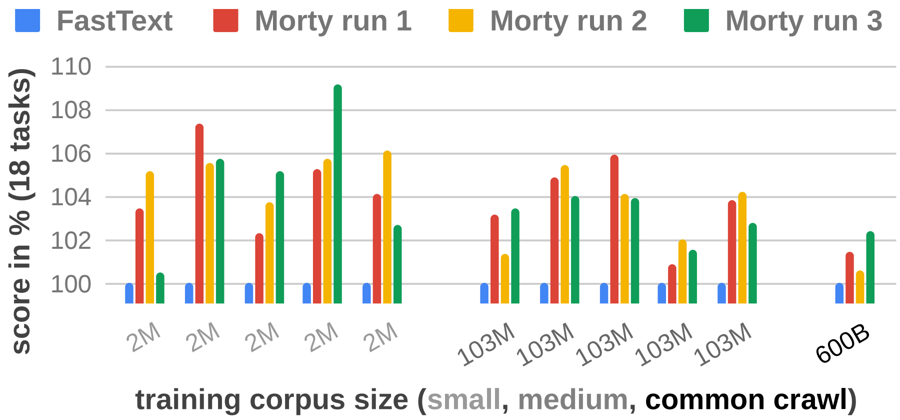


Figure 7.1: **1-epoch MoRTy (MT %) performance change over Fasttext:** Blue bars show Fasttext baseline performance (100%). 3 Morty runs: trained on Fasttext for **1 epoch** (2x5 Fasttext for corpus sizes 2M and 103M and 1x for 600B). Detailed description on next page.

Applying the MoRTy 1-epoch recipe So far, we saw MoRTys potential for overall (ST/MT/Σ) performance improvements, but will we observe the same *in the wild*? To answer this question for the MT use-case, we apply a *1-epoch* training only recipe. That is, we train *1-epoch* using a linear, complete autoencoder using *half of the base embedders learning rate* on three randomly initialized MoRTys, and then test them on the 18 task (MT) setup. Figure 7.1 shows consistent MT/Σ score improvements for each of the 3 MoRTy-over-Fasttext runs (red, yellow, green) on 2M, 103M, and 600B vs. base Fasttext (blue 100).

We see that, for practical application, this allows MoRTy to boost supervised MT performance even without using a supervised development split or proxy task(s), while also *eliminating multi-epoch tuning*. Both Figure 7.1 and Table 7.1 show similar overall (MT) improvements per corpus size, which suggests that 1-epoch training is sufficient and that **MoRTy is especially beneficial on smaller corpora** – i.e., in limited resource settings.

7.5 Related Work

There is a large body of work on information transfer between supervised and unsupervised tasks. First and foremost **unsupervised-to-supervised** transfer includes using embeddings for supervised tasks. However, transfer also works vice versa, in a **supervised-to-unsupervised** setup to (learn to) specialize embeddings to better fit a specific supervised signal (Ruder and Plank, 2017; Ye et al., 2018). This includes injecting generally relevant semantics via retrofitting or auxiliary multi-task supervision (Faruqui et al., 2015; Kiela et al., 2018b). **Supervised-to-supervised** methods provide knowledge transfer between supervised tasks which is exploited successively (Kirkpatrick et al., 2017), jointly (Kiela et al., 2018b) and in joint-succession (Hashimoto et al., 2017).

Unsupervised-to-unsupervised transfer is less studied. Dingwall and Potts (2018) proposed a GloVe *model-modification* that retrofits publicly available GloVe embeddings to produce specialized domain embeddings, while Bollegala and Bao (2018) propose *meta-embeddings* via denoising autoencoders to *merge* diverse (Fasttext and GloVe) embeddings spaces. The later, is also a low-effort approach and closest to ours. However, it focuses on embedding merging that they *tuned on a single* semantic similarity task, while MoRTy provides an overview of *tuning for 19 different settings*. Furthermore, MoRTy requires only a single embedding space, which contributes to the literature by outlining that meta-embedding improvements may partly stem from re-encoding rather than only from semantic merging.

7.6 Conclusion

We demonstrated a low-effort, self-supervised, *learning scale-down* method to *construct task-optimized* word embeddings from existing ones to gain performance on a (set of) supervised end-task(s) without direct domain adaptation. Despite its simplicity, MoRTy is able to produce significant performance improvements for *single* and *multi-task* supervision settings as well as for a variety of desirable word encoding properties while forgoing building and tuning complex model architectures and labeling.[10] Perhaps most importantly, MoRTy shows considerable benefits for low-resource settings and thus provides a *learning-to-scale-down* alternative to recent *scale-up* approaches.

Acknowledgements

This work was supported by the German Federal Ministry of Education and Research (BMBF) through the project DEEPLEE (01IW17001) and by the European Union's Horizon 2020 research and innovation programme under grant agreement No 780495 (BigMedilytics). We also thank Philippe Thomas and Isabelle Augenstein for helpful discussions.

[10] Source code at https://github.com/NilsRethmeier/MoRTy

References

Aviad Aberdam, Ron Litman, Shahar Tsiper, Oron Anschel, Ron Slossberg, Shai Mazor, R. Manmatha, and Pietro Perona. 2021. Sequence-to-sequence contrastive learning for text recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15302–15312.

Julius Adebayo, Justin Gilmer, Michael Muelly, Ian J. Goodfellow, Moritz Hardt, and Been Kim. 2018. Sanity Checks for Saliency Maps. In *Proceedings of NeurIPS*.

Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. 2019. FLAIR: An Easy-to-Use Framework for State-of-the-Art NLP. In *NAACL Demos)*, Minneapolis, Minnesota. ACL.

Arjun Akula, Spandana Gella, Yaser Al-Onaizan, Song-Chun Zhu, and Siva Reddy. 2020. Words aren't enough, their order matters: On the robustness of grounding visual referring expressions. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6555–6565, Online. Association for Computational Linguistics.

Omer Antverg and Yonatan Belinkov. 2021. Are all neurons created equal? interpreting and controlling BERT through individual neurons. In *eXplainable AI approaches for debugging and diagnosis.*

Stéphane Aroca-Ouellette and Frank Rudzicz. 2020. On Losses for Modern Language Models. In *EMNLP*, Online. Association for Computational Linguistics.

Leila Arras, Franziska Horn, Grégoire Montavon, Klaus-Robert Müller, and Wojciech Samek. 2017. "what is relevant in a text document?": An interpretable machine learning approach. *PLOS ONE*, 12:1–23.

Leila Arras, Ahmed Osman, Klaus-Robert Müller, and Wojciech Samek. 2019. Evaluating Recurrent Neural Network Explanations. In *ACL Workshop BlackboxNLP*, Florence, Italy. ACL.

Pepa Atanasova, Jakob Grue Simonsen, Christina Lioma, and Isabelle Augenstein. 2020. Generating Fact Checking Explanations. In *Proceedings of the ACL*.

Pepa Atanasova, Jakob Grue Simonsen, Christina Lioma, and Isabelle Augenstein. 2022. Fact checking with insufficient evidence. *Transactions of the Association for Computational Linguistics*, 10:746–763.

Isabelle Augenstein, Christina Lioma, Dongsheng Wang, Lucas Chaves Lima, Casper Hansen, Christian Hansen, and Jakob Grue Simonsen. 2019. Multifc: A real-world multi-domain dataset for evidence-based fact checking of claims. 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, Hong Kong, China, November 3-7, 2019*, pages 4684–4696. Association for Computational Linguistics.

Isabelle Augenstein, Sebastian Ruder, and Anders Søgaard. 2018. Multi-task learning of pairwise sequence classification tasks over disparate label spaces. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 1896–1906. Association for Computational Linguistics.

Anthony Bau, Yonatan Belinkov, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James R. Glass. 2019. Identifying and controlling important neurons in neural machine translation. In *ICLR*, New Orleans, LA, USA.

Yonatan Belinkov, Sebastian Gehrmann, and Ellie Pavlick. 2020. Interpretability and analysis in neural NLP. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts*, pages 1–5, Online. Association for Computational Linguistics.

Yonatan Belinkov and James Glass. 2019. Analysis Methods in Neural Language Processing: A Survey. *Transactions of ACL*.

Iz Beltagy, Arman Cohan, Robert Logan IV, Sewon Min, and Sameer Singh. 2022. Zero- and few-shot NLP with pretrained language models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts*, pages 32–37, Dublin, Ireland. Association for Computational Linguistics.

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: A Pretrained Language Model for Scientific Text. 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 3613–3618, Stroudsburg, PA, USA. Association for Computational Linguistics.

Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. 2022a. BitFit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1–9, Dublin, Ireland. Association for Computational Linguistics.

Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. 2022b. BitFit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1–9, Dublin, Ireland. Association for Computational Linguistics.

Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021a. On the dangers of stochastic parrots: Can language models be too big? In *FAccT '21: 2021 ACM Conference on Fairness, Accountability, and Transparency, Virtual Event / Toronto, Canada, March 3-10, 2021*, pages 610–623. ACM.

Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021b. On the dangers of stochastic parrots: Can language models be too big? New York, NY, USA. Association for Computing Machinery.

Chandra Bhagavatula, Sergey Feldman, Russell Power, and Waleed Ammar. 2018. Content-based citation recommendation. *NAACL HLT 2018 - 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, 1:238–251.

Cody Blakeney, Nathaniel Huish, Yan Yan, and Ziliang Zong. 2021. Simon says: Evaluating and mitigating bias in pruned neural networks with knowledge distillation.

P Bojanowski, E Grave, A Joulin, and T Mikolov. 2017. Enriching word vectors with subword information. *TACL*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *CoRR*, abs/1607.04606.

Danushka Bollegala and Cong Bao. 2018. Learning word meta-embeddings by autoencoding. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1650–1661. Association for Computational Linguistics.

Malik Boudiaf, Jérôme Rony, Imtiaz Masud Ziko, Eric Granger, Marco Pedersoli, Pablo Piantanida, and Ismail Ben Ayed. 2020. A unifying mutual information view of metric learning: Cross-entropy vs. pairwise losses. In *ECCV*, volume 12351 of *Lecture Notes in Computer Science*. Springer.

Robin Brochier, Adrien Guille, and Julien Velcin. 2019. Global Vectors for Node Representations. In *The World Wide Web Conference on - WWW '19*, volume 2, pages 2587–2593, New York, New York, USA. ACM Press.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners.

Maxime Bucher, Stéphane Herbin, and Frédéric Jurie. 2016. Hard negative mining for metric learning based zero-shot classification. In *Computer Vision – ECCV 2016 Workshops*, pages 524–531, Cham. Springer International Publishing.

Tiffany Tianhui Cai, Jonathan Frankle, David J. Schwab, and Ari S. Morcos. 2020. Are all negatives created equal in contrastive instance discrimination?

Shan Carter, Zan Armstrong, Ludwig Schubert, Ian Johnson, and Chris Olah. 2019a. Activation atlas. *Distill*. Https://distill.pub/2019/activation-atlas.

Shan Carter, Zan Armstrong, Ludwig Schubert, Ian Johnson, and Chris Olah. 2019b. Activation Atlas. *Distill*.

Daniel Matthew Cer, Yinfei Yang, Sheng yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Universal sentence encoder. *arXiv:1803.11175*.

Wei-Cheng Chang, Hsiang-Fu Yu, Kai Zhong, Yiming Yang, and Inderjit Dhillon. 2019. X-BERT: eXtreme Multi-label Text Classification with using Bidirectional Encoder Representations from Transformers. *NeurIPS*.

Jiaao Chen, Zichao Yang, and Diyi Yang. 2020a. MixText: Linguistically-informed interpolation of hidden space for semi-supervised text classification. In *ACL*, Online. Association for Computational Linguistics.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. 2020b. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, pages 1597–1607.

Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey E. Hinton. 2020c. Big self-supervised models are strong semi-supervised learners. In *NeurIPS*.

Zewen Chi, Li Dong, Furu Wei, Nan Yang, Saksham Singhal, Wenhui Wang, Xia Song, Xian-Ling Mao, Heyan Huang, and Ming Zhou. 2021. InfoXLM: An information-theoretic framework for cross-lingual language model pre-training. In *NAACL-HLT*, Online. Association for Computational Linguistics.

E Choi, MT Bahadori, E Searles, C Coffey, M Thompson, J Bost, J Tejedor-Sojo, and J Sun. 2016. Multi-layer Representation Learning for Medical Concepts. In *KDD*. ACM.

E Choi, MT Bahadori, L Song, WF Stewart, and J Sun. 2017. GRAM: graph-based attention model for healthcare representation learning. In *Proc. of the 23rd ACM SIGKDD*.

Kevin Clark, Minh-Thang Luong, Quoc Le, and Christopher D. Manning. 2020. Pre-training transformers as energy-based cloze models. In *EMNLP*, Online. Association for Computational Linguistics.

Arman Cohan, Waleed Ammar, Madeleine van Zuylen, and Field Cady. 2019. Structural Scaffolds for Citation Intent Classification in Scientific Publications. In *Proceedings of the 2019 Conference of the North*, volume 1, pages 3586–3596, Stroudsburg, PA, USA. Association for Computational Linguistics.

Arman Cohan, Sergey Feldman, Iz Beltagy, Doug Downey, and Daniel Weld. 2020a. SPECTER: Document-level Representation Learning using Citation-informed Transformers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2270–2282, Stroudsburg, PA, USA. Association for Computational Linguistics.

Arman Cohan, Sergey Feldman, Iz Beltagy, Doug Downey, and Daniel Weld. 2020b. SPECTER: Document-level representation learning using citation-informed transformers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2270–2282, Online. Association for Computational Linguistics.

Alexis Conneau and Douwe Kiela. 2018. SentEval: An Evaluation Toolkit for Universal Sentence Representations. In *Proceedings of LREC*, Miyazaki, Japan.

Alexis Conneau, Germán Kruszewski, Guillaume Lample, Loïc Barrault, and Marco" Baroni. 2018. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2126–2136. Association for Computational Linguistics.

Fahim Dalvi, Nadir Durrani, Hassan Sajjad, Yonatan Belinkov, Anthony Bau, and James R. Glass. 2019. What is one grain of sand in the desert? analyzing individual neurons in deep NLP models. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 6309–6317. AAAI Press.

Jesse Davis and Mark Goadrich. 2006. The relationship between Precision-Recall and ROC curves. In *Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006), Pittsburgh, Pennsylvania, USA, June 25-29, 2006*, pages 233–240.

Cyprien de Masson d'Autume, Sebastian Ruder, Lingpeng Kong, and Dani Yogatama. 2019. Episodic Memory in Lifelong Language Learning. In *Advances of NeurIPS*, Montréal, Canada.

Yuntian Deng, Anton Bakhtin, Myle Ott, Arthur Szlam, and Marc'Aurelio Ranzato. 2020. Residual energy-based models for text generation. In *ICLR*.

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019a. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019b. 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*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Nicholas Dingwall and Christopher Potts. 2018. Mittens: an extension of glove for learning domain-specialized representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human*

Language Technologies, Volume 2 (Short Papers), pages 212–217. Association for Computational Linguistics.

Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah A. Smith. 2020. Fine-Tuning Pretrained Language Models: Weight Initializations, Data Orders, and Early Stopping. *CoRR*, abs/2002.06305.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2020. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale.

D. D'souza, Zach Nussbaum, Chirag Agarwal, and Sara Hooker. 2021. A tale of two long tails. *ArXiv*, abs/2107.13098.

Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, Barret Zoph, Liam Fedus, Maarten P Bosma, Zongwei Zhou, Tao Wang, Emma Wang, Kellie Webster, Marie Pellat, Kevin Robinson, Kathleen Meier-Hellstern, Toju Duke, Lucas Dixon, Kun Zhang, Quoc Le, Yonghui Wu, Zhifeng Chen, and Claire Cui. 2022. GLaM: Efficient scaling of language models with mixture-of-experts. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 5547–5569. PMLR.

Xiangyu Duan, Hongfei Yu, Mingming Yin, Min Zhang, Weihua Luo, and Yue Zhang. 2019. Contrastive attention mechanism for abstractive sentence summarization. In *EMNLP-IJCNLP*, Hong Kong, China. Association for Computational Linguistics.

Nadir Durrani, Hassan Sajjad, Fahim Dalvi, and Yonatan Belinkov. 2020. Analyzing individual neurons in pre-trained language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4865–4880, Online. Association for Computational Linguistics.

Yanai Elazar, Shauli Ravfogel, Alon Jacovi, and Yoav Goldberg. 2021. Amnesic probing: Behavioral explanation with amnesic counterfactuals. *Transactions of the Association for Computational Linguistics*, 9:160–175.

Aaron Elkiss, Siwei Shen, Anthony Fader, Güneş Erkan, David States, and Dragomir Radev. 2008. Blind men and elephants: What do citation summaries tell us about a research article? *Journal of the American Society for Information Science and Technology*, 59(1):51–62.

D. Erhan, Yoshua Bengio, Aaron C. Courville, and Pascal Vincent. 2009a. Visualizing higher-layer features of a deep network.

Dumitru Erhan, Yoshua Bengio, Aaron C. Courville, and Pascal Vincent. 2009b. Visualizing Higher-Layer Features of a Deep Network. In *University of Montreal publications*.

Hongchao Fang, Sicheng Wang, Meng Zhou, Jiayuan Ding, and Pengtao Xie. 2020a. Cert: Contrastive self-supervised learning for language understanding.

Hongchao Fang, Sicheng Wang, Meng Zhou, Jiayuan Ding, and Pengtao Xie. 2020b. CERT: Contrastive Self-supervised Learning for Language Understanding. *arXiv:2005.12766*, pages 1–16.

Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard Hovy, and Noah A Smith. 2015. Retrofitting word vectors to semantic lexicons. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1606–1615. Association for Computational Linguistics.

Alberto Fernández, Salvador García, Mikel Galar, Ronaldo C. Prati, Bartosz Krawczyk, and Francisco Herrera. 2018. *Learning from Imbalanced Data Sets*. Springer.

Jonathan Frankle and Michael Carbin. 2019a. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*.

Jonathan Frankle and Michael Carbin. 2019b. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*.

Jonathan Frankle and Michael Carbin. 2019c. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. In *ICLR*, New Orleans, LA, USA.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021a. Simcse: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 6894–6910. Association for Computational Linguistics.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021b. SimCSE: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6894–6910, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Sebastian Gehrmann, Hendrik Strobelt, Robert Krüger, Hanspeter Pfister, and Alexander M. Rush. 2019. Visual Interaction with Deep Learning Models through Collaborative Semantic Inference. *IEEE TVCG*.

Sebastian Gehrmann, Hendrik Strobelt, Robert Krüger, Hanspeter Pfister, and Alexander M. Rush. 2020. Visual interaction with deep learning models through collaborative semantic inference. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):884–894.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Leilani H. Gilpin, David Bau, Ben Z. Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. 2018. Explaining Explanations: An Overview of Interpretability of Machine Learning. In *IEEE DSAA*, Turin, Italy.

John Giorgi, Osvald Nitski, Bo Wang, and Gary Bader. 2021a. DeCLUTR: Deep contrastive learning for unsupervised textual representations. In *ACL*, Online. Association for Computational Linguistics.

John Giorgi, Osvald Nitski, Bo Wang, and Gary Bader. 2021b. DeCLUTR: Deep Contrastive Learning for Unsupervised Textual Representations. 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 879–895, Stroudsburg, PA, USA. Association for Computational Linguistics.

Bela Gipp and J Beel. 2009. Citation Proximity Analysis (CPA) - A new approach for identifying related work based on Co-Citation Analysis. *Birger Larsen and Jacqueline Leta, editors, Proceedings of the 12th International Conference on Scientometrics and Informetrics (ISSI'09)*, 2(July):571–575.

Mario Giulianelli, Jack Harding, Florian Mohnert, Dieuwke Hupkes, and Willem H. Zuidema. 2018. Under the Hood: Using Diagnostic Classifiers to Investigate and Improve how Language Models Track Agreement Information. In *EMNLP Workshop BlackboxNLP*, Brussels, Belgium.

Florian Graf, Christoph Hofer, Marc Niethammer, and Roland Kwitt. 2021a. Dissecting supervised constrastive learning. In *ICML*, volume 139 of *PMLR*. PMLR.

Florian Graf, Christoph Hofer, Marc Niethammer, and Roland Kwitt. 2021b. Dissecting supervised constrastive learning. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 3821–3830. PMLR.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, Bilal Piot, koray kavukcuoglu, Remi Munos, and Michal Valko. 2020. Bootstrap your own latent - a new approach to self-supervised learning. In *NeurIPS*, volume 33. Curran Associates, Inc.

Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable Feature Learning for Networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16*, pages 855–864, New York, New York, USA. ACM Press.

Beliz Gunel, Jingfei Du, Alexis Conneau, and Veselin Stoyanov. 2021. Supervised contrastive learning for pre-trained language model fine-tuning. In *9th ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Suchin Gururangan, Ana Marasovic, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 8342–8360. Association for Computational Linguistics.

Michael Gutmann and Aapo Hyvärinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *AISTATS*, volume 9 of *PMLR*, Chia Laguna Resort, Sardinia, Italy. JMLR Workshop and Conference Proceedings.

Kishaloy Halder, Alan Akbik, Josip Krapac, and Roland Vollgraf. 2020. Task-aware representation of sentences for generic text classification. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3202–3213, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Jialong Han, Yan Song, Wayne Xin Zhao, Shuming Shi, and Haisong Zhang. 2018. hyperdoc2vec: Distributed Representations of Hypertext Documents. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2384–2394, Stroudsburg, PA, USA. Association for Computational Linguistics.

Momchil Hardalov, Arnav Arora, Preslav Nakov, and Isabelle Augenstein. 2021. Few-shot cross-lingual stance detection with sentiment-based pre-training.

H Harutyunyan, H Khachatrian, DC Kale, G Ver Steeg, and A Galstyan. 2017. Multitask learning and benchmarking with clinical time series data. *Scientific Data*.

Kazuma Hashimoto, caiming xiong, Yoshimasa Tsuruoka, and Richard Socher. 2017. A joint many-task model: Growing a neural network for multiple nlp tasks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1923–1933. Association for Computational Linguistics.

Fengxiang He, Tongliang Liu, and Dacheng Tao. 2019. Control batch size and learning rate to generalize well: Theoretical and empirical evidence. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 1141–1150.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 9726–9735. Computer Vision Foundation / IEEE.

Ernst Hellinger. 1909. Neue Begründung der Theorie Quadratischer Formen von Unendlichvielen Veränderlichen. *Journal für die reine und angewandte Mathematik*, 136.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531.

R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. 2019a. Learning deep representations by mutual information estimation and maximization. In *International Conference on Learning Representations*.

R. Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Philip Bachman, Adam Trischler, and Yoshua Bengio. 2019b. Learning deep representations by mutual information estimation and maximization. In *7th ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katherine Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Oriol Vinyals, Jack William Rae, and Laurent Sifre. 2022. An empirical analysis of compute-optimal large language model training. In *Advances in Neural Information Processing Systems*.

Fred Hohman, Haekyu Park, Caleb Robinson, and Duen Horng Chau. 2020. Summit: Scaling Deep Learning Interpretability by Visualizing Activation and Attribution Summarizations. *IEEE TVCG*.

Andreas Nugaard Holm, Barbara Plank, Dustin Wright, and Isabelle Augenstein. 2022. Longitudinal Citation Prediction using Temporal Graph Neural Networks. In *AAAI 2022 Workshop on Scientific Document Understanding (SDU 2022)*.

Sara Hooker. 2020. The Hardware Lottery.

Sara Hooker. 2021. Moving beyond “algorithmic bias is a data problem”. *Patterns*, 2(4):100241.

Sara Hooker, Aaron Courville, Gregory Clark, Yann Dauphin, and Andrea Frome. 2020a. What Do Compressed Deep Neural Networks Forget?

Sara Hooker, Aaron Courville, Yann Dauphin, and Andrea Frome. 2020b. What doe pruned deep neural networks forget? In *Bridging AI and Cognitive Science ICLR Workshop*.

Sara Hooker, Nyalleng Moorosi, Gregory Clark, Samy Bengio, and Emily Denton. 2020c. Characterizing and mitigating bias in compact models. In *ICML Workshop on Human Interpretability in Machine Learning*.

Sara Hooker, Nyalleng Moorosi, Gregory Clark, Samy Bengio, and Emily Denton. 2020d. Characterizing and Mitigating Bias in Compact Models. In *WHI Workshop at ICML Jul 12, 2020 – Jul 18, 2020*. ICML.

Benjamin Hoover, Hendrik Strobelt, and Sebastian Gehrmann. 2020. exbert: A visual analysis tool to explore learned representations in transformer models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations, ACL 2020, Online, July 5-10, 2020*, pages 187–196. Association for Computational Linguistics.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR.

Jeremy Howard and Sebastian Ruder. 2018. Universal Language Model Fine-tuning for Text Classification. In *Proceedings of the ACL*, Melbourne, Australia.

Niel Teng Hu, Xinyu Hu, Rosanne Liu, Sara Hooker, and Jason Yosinski. 2021. When does loss-based prioritization fail?

Dan Iter, Kelvin Guu, Larry Lansing, and Dan Jurafsky. 2020. Pretraining with contrastive sentence objectives improves discourse performance of language models. In *ACL*, Online. Association for Computational Linguistics.

Ashish Jaiswal, Ashwin Ramesh Babu, Mohammad Zaki Zadeh, Debapriya Banerjee, and Fillia Makedon. 2021a. A survey on contrastive self-supervised learning. *Technologies*, 9(1).

Ashish Jaiswal, Ashwin Ramesh Babu, Mohammad Zaki Zadeh, Debapriya Banerjee, and Fillia Makedon. 2021b. A survey on contrastive self-supervised learning. *Technologies*, 9(1).

Ashish Jaiswal, Ashwin Ramesh Babu, Mohammad Zaki Zadeh, Debapriya Banerjee, and Fillia Makedon. 2021c. A survey on contrastive self-supervised learning. *Technologies*, 9(1).

Stanislaw Jastrzebski, Damian Lesniak, and Wojciech Marian Czarnecki. 2017. How to evaluate word embeddings? on importance of data efficiency and simple supervised tasks. *CoRR*, abs/1702.02170.

Chanwoo Jeong, Sion Jang, Hyuna Shin, Eunjeong Lucy Park, and Sungchul Choi. 2020. A context-aware citation recommendation model with bert and graph convolutional networks. *Scientometrics*, pages 1–16.

Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc V. Le, Yun-Hsuan Sung, Zhen Li, and Tom Duerig. 2021. Scaling up visual and vision-language representation learning with noisy text supervision. In *ICML*, volume 139 of *PMLR*. PMLR.

Huajie Jiang, Ruiping Wang, Shiguang Shan, and Xilin Chen. 2019a. Transferable contrastive network for generalized zero-shot learning. In *IEEE/CVF ICCV*.

Huajie Jiang, Ruiping Wang, Shiguang Shan, and Xilin Chen. 2019b. Transferable Contrastive Network for Generalized Zero-Shot Learning. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 9764–9773.

Yiding Jiang, Behnam Neyshabur, Hossein Mobahi, Dilip Krishnan, and Samy Bengio. 2020a. Fantastic Generalization Measures and Where to Find Them. In *ICLR*.

Ziyu Jiang, Tianlong Chen, Ting Chen, and Zhangyang Wang. 2020b. Robust pre-training by adversarial contrastive learning. In *NeurIPS*, volume 33. Curran Associates, Inc.

Ziyu Jiang, Tianlong Chen, Bobak J Mortazavi, and Zhangyang Wang. 2021. Self-damaging contrastive learning. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 4927–4939. PMLR.

AEW Johnson, TJ Pollard, L Shen, HL Li-wei, M Feng, M Ghassemi, B Moody, P Szolovits, LA Celi, and RG Mark. 2016. MIMIC-III, a freely accessible critical care database. *Scientific data*.

Jeff Johnson, Matthijs Douze, and Herve Jegou. 2021. Billion-Scale Similarity Search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547.

Vinu Joseph, Shoaib Ahmed Siddiqui, Aditya Bhaskara, Ganesh Gopalakrishnan, Saurav Muralidharan, Michael Garland, Sheraz Ahmed, and Andreas Dengel. 2020. Reliable model compression via label-preservation-aware loss functions. *arXiv preprint arXiv:2012.01604*.

Minsuk Kahng, Pierre Y. Andrews, Aditya Kalro, and Duen Horng Chau. 2017. Activis: Visual exploration of industry-scale deep neural network models. *IEEE Transactions on Visualization and Computer Graphics*, 24:88–97.

Parminder Kaur, Husanbir Singh Pannu, and Avleen Kaur Malhi. 2021. Comparative analysis on cross-modal information retrieval: A review. *Computer Science Review*, 39:100336.

M. M. Kessler. 1963. Bibliographic coupling between scientific papers. *American Documentation*, 14(1):10–25.

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. 2020a. Supervised contrastive learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 18661–18673. Curran Associates, Inc.

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. 2020b. Supervised contrastive learning. In *NeurIPS*.

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. 2020c. Supervised contrastive learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 18661–18673. Curran Associates, Inc.

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. 2020d. Supervised contrastive learning. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Douwe Kiela, Changhan Wang, and Kyunghyun Cho. 2018a. Context-attentive embeddings for improved sentence representations. *CoRR*, abs/1804.07983.

Douwe Kiela, Changhan Wang, and Kyunghyun Cho. 2018b. Dynamic meta-embeddings for improved sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1466–1477. Association for Computational Linguistics.

Kang-Min Kim, Bumsu Hyeon, Yeachan Kim, Jun-Hyung Park, and SangKeun Lee. 2020. Multi-pretraining for Large-scale Text Classification. In *Findings of EMNLP*, pages 2041–2050. ACL.

Sehoon Kim, Amir Gholami, Zhewei Yao, Michael W. Mahoney, and Kurt Keutzer. 2021a. I-BERT: integer-only BERT quantization. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 5506–5518. PMLR.

Taeuk Kim, Kang Min Yoo, and Sang-goo Lee. 2021b. Self-guided contrastive learning for BERT sentence representations. 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 2528–2540, Online. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Lei Ba. 2015. Adam: A method for stochastic optimization. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pages 1–15.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526.

Tassilo Klein and Moin Nabi. 2020. Contrastive self-supervised learning for commonsense reasoning. In *ACL*, Online. Association for Computational Linguistics.

Lingpeng Kong, Cyprien de Masson d'Autume, Lei Yu, Wang Ling, Zihang Dai, and Dani Yogatama. 2020. A mutual information maximization perspective of language representation learning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey E. Hinton. 2019. Similarity of neural network representations revisited. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 3519–3529. PMLR.

Nikolaus Kriegeskorte, Marieke Mur, and Peter Bandettini. 2008. Representational similarity analysis - connecting the branches of systems neuroscience. *Frontiers in Systems Neuroscience*, 2.

BC Kwon, M Choi, JT Kim, E Choi, Young B Kim, S Kwon, J Sun, and J Choo. 2019a. RetainVis: Visual analytics with interpretable and interactive recurrent neural networks on electronic medical records. *IEEE TVCG*, 25.

Bum Chul Kwon, Min-Je Choi, Joanne Taery Kim, Edward Choi, Young Bin Kim, Soonwook Kwon, Jimeng Sun, and Jaegul Choo. 2019b. RetainVis: Visual Analytics with Interpretable and Interactive Recurrent Neural Networks on Electronic Medical Records. *IEEE TVCG*, 25(1).

Yann LeCun, Sumit Chopra, Raia Hadsell, Aurelio Ranzato, and Fu Jie Huang. 2006a. *A Tutorial on Energy-Based Learning*. Predicting Structured Data. The MIT Press.

Yann LeCun, Sumit Chopra, Raia Hadsell, Marc Aurelio Ranzato, and Fu Jie Huang. 2006b. *A tutorial on energy-based learning*. MIT Press.

Yann LeCun and Fu Jie Huang. 2005a. Loss functions for discriminative training of energy-based models. In *Proc. of the 10-th International Workshop on Artificial Intelligence and Statistics (AIStats'05)*.

Yann LeCun and Fu Jie Huang. 2005b. Loss functions for discriminative training of energy-based models. In *AISTATS*.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2019. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, pages 1–8.

Adam Lerer, Ledell Wu, Jiajun Shen, Timothee Lacroix, Luca Wehrstedt, Abhijit Bose, and Alex Peysakhovich. 2019. PyTorch-BigGraph: A Large-scale Graph Embedding System. In *Proceedings of The Conference on Systems and Machine Learning*.

Omer Levy and Yoav Goldberg. 2014. Linguistic regularities in sparse and explicit word representations. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 171–180. Association for Computational Linguistics.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225.

Bohan Li, Hao Zhou, Junxian He, Mingxuan Wang, Yiming Yang, and Lei Li. 2020. On the Sentence Embeddings from Pre-trained Language Models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9119–9130, Stroudsburg, PA, USA. Association for Computational Linguistics.

Junnan Li, Ramprasaath R. Selvaraju, Akhilesh Deepak Gotmare, Shafiq R. Joty, Caiming Xiong, and Steven C. H. Hoi. 2021. Align before fuse: Vision and language representation learning with momentum distillation. *CoRR*, abs/2107.07651.

Carolyn E. Lipscomb. 2000. Medical subject headings (mesh). *Bulletin of the Medical Library Association*, 88 3:265–6.

Jingzhou Liu, Wei-Cheng Chang, Yuexin Wu, and Yiming Yang. 2017. Deep Learning for Extreme Multi-label Text Classification. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Shinjuku, Tokyo, Japan, August 7-11, 2017*, pages 115–124.

Liyuan Liu, Xiaodong Liu, Jianfeng Gao, Weizhu Chen, and Jiawei Han. 2020a. Understanding the Difficulty of Training Transformers. In *EMNLP*, Online. Association for Computational Linguistics.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019a. Linguistic knowledge and transferability of contextual representations. In *NAACL-HLT*, Minneapolis, MN, USA.

Qi Liu, Matt J. Kusner, and Phil Blunsom. 2020b. A Survey on Contextual Embeddings. *CoRR*, abs/2003.07278.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Ziwei Liu, Zhongqi Miao, Xiaohang Zhan, Jiayun Wang, Boqing Gong, and Stella X. Yu. 2019c. Large-Scale Long-Tailed Recognition in an Open World. In *IEEE CVPR, Long Beach, CA, USA, June 16-20, 2019*, pages 2537–2546.

Kyle Lo, Lucy Lu Wang, Mark Neumann, Rodney Kinney, and Daniel Weld. 2020. S2ORC: The Semantic Scholar Open Research Corpus. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4969–4983, Stroudsburg, PA, USA. Association for Computational Linguistics.

Lajanugen Logeswaran and Honglak Lee. 2018. An efficient framework for learning sentence representations. In *ICLR*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. *7th International Conference on Learning Representations, ICLR 2019*.

Q Lu, Y Li, N de Silva, S Kafle, J Cao, D Dou, TH Nguyen, P Sen, B Hailpern, and B Reinwald. 2019. Learning electronic health records through hyperbolic embedding of medical ontologies. In *BCB '19*. ACM Press.

Michal Lukasik, Srinadh Bhojanapalli, Aditya Menon, and Sanjiv Kumar. 2020. Does label smoothing mitigate label noise? In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 6448–6458. PMLR.

Ronny Luss, Pin-Yu Chen, Amit Dhurandhar, Prasanna Sattigeri, Yunfeng Zhang, Karthikeyan Shanmugam, and Chun-Chen Tu. 2021. Leveraging latent features for local explanations. In *KDD SIGKDD*. ACM.

Kelvin Luu, Xinyi Wu, Rik Koncel-Kedziorski, Kyle Lo, Isabel Cachola, and Noah A. Smith. 2021. Explaining Relationships Between Scientific Documents. 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 2130–2144, Stroudsburg, PA, USA. Association for Computational Linguistics.

L Ma, J Gao, Y Wang, C Zhang, J Wang, W Ruan, W Tang, X Gao, and X Ma. 2019a. AdaCare: Explainable Clinical Health Status Representation Learning via Scale-Adaptive Feature Extraction and Recalibration. In *34th AAAI*. AAAI.

L Ma, C Zhang, Y Wang, W Ruan, J Wang, W Tang, X Ma, X Gao, and J Gao. 2019b. ConCare: Personalized Clinical Feature Embedding via Capturing the Healthcare Context. In *34th AAAI*. AAAI.

T Ma, C Xiao, and F Wang. 2018. Health-ATM: A deep architecture for multifaceted patient health record representation and risk prediction. In *SIAM International Conference on Data Mining*.

Zhuang Ma and Michael Collins. 2018. Noise Contrastive Estimation and Negative Sampling for Conditional Models: Consistency and Statistical Efficiency. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 3698–3707.

Zongyang Ma, Aixin Sun, Quan Yuan, and Gao Cong. 2014. Tagging your tweets: A probabilistic modeling of hashtag annotation in twitter. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM 2014, Shanghai, China, November 3-7, 2014*, pages 999–1008. ACM.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Ilaria Manco, Emmanouil Benetos, Elio Quinton, and György Fazekas. 2022. Contrastive audio-language learning for music. *CoRR*, abs/2208.12208.

Tom McCoy, Ellie Pavlick, and Tal Linzen. 2019a. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448, Florence, Italy. Association for Computational Linguistics.

Tom McCoy, Ellie Pavlick, and Tal Linzen. 2019b. Right for the Wrong Reasons: Diagnosing Syntactic Heuristics in Natural Language Inference. In *Proceedings of the ACL*, Florence, Italy.

Gábor Melis, Tomás Kociský, and Phil Blunsom. 2020. Mogrifier LSTM. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Yu Meng, Chenyan Xiong, Payal Bajaj, Saurabh Tiwary, Paul Bennett, Jiawei Han, and Xia Song. 2021. COCO-LM: correcting and contrasting text sequences for language model pretraining. *CoRR*, abs/2102.08473.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018. Regularizing and Optimizing LSTM Language Models. In *ICLR*, Vancouver, BC, Canada.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *CoRR*, abs/1609.07843.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017a. Pointer Sentinel Mixture Models. In *ICLR*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017b. Pointer Sentinel Mixture Models. In *ICLR*, Toulon, France.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 14014–14024.

Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *ICLR Workshop Track Proceedings*.

Tomás Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 3111–3119.

Tomás Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013c. Distributed representations of words and phrases and their compositionality. In *NeurIPS*.

Margaret Mitchell, Dylan Baker, Nyalleng Moorosi, Emily Denton, Ben Hutchinson, Alex Hanna, Timnit Gebru, and Jamie Morgenstern. 2020. Diversity and inclusion metrics in subset selection. In *AIES '20: AAAI/ACM Conference on AI, Ethics, and Society, New York, NY, USA, February 7-8, 2020*, pages 117–123. ACM.

Andriy Mnih and Yee Whye Teh. 2012. A fast and simple algorithm for training neural probabilistic language models. In *ICML*, ICML'12, Madison, WI, USA. Omnipress.

Christoph Molnar. 2022. *Interpretable Machine Learning*, 2 edition. Independently published (February 28, 2022).

Ari S. Morcos, Maithra Raghu, and Samy Bengio. 2018. Insights on representational similarity in neural networks with canonical correlation. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 5732–5741.

Kevin Musgrave, Serge J. Belongie, and Ser-Nam Lim. 2020a. A Metric Learning Reality Check. In *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XXV*, pages 681–699.

Kevin Musgrave, Serge J. Belongie, and Ser-Nam Lim. 2020b. A metric learning reality check. In *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XXV*, pages 681–699.

Lukas Muttenthaler, Isabelle Augenstein, and Johannes Bjerva. 2020. Unsupervised evaluation for question answering with transformers. In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 83–90, Online. Association for Computational Linguistics.

Lukas Muttenthaler, Jonas Dippel, Lorenz Linhardt, Robert A. Vandermeulen, and Simon Kornblith. 2022. Human alignment of neural network representations.

Toan Q. Nguyen and Julian Salazar. 2019. Transformers without tears: Improving the normalization of self-attention. In *Proceedings of the 16th International Conference on Spoken Language Translation*, Hong Kong. Association for Computational Linguistics.

Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. 2017a. Feature visualization. *Distill*. Https://distill.pub/2017/feature-visualization.

Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. 2017b. Feature Visualization. *Distill*. Https://distill.pub/2017/feature-visualization.

Malte Ostendorff, Nils Rethmeier, Isabelle Augenstein, Bela Gipp, and Georg Rehm. 2022a. Neighborhood contrastive learning for scientific document representations with citation embeddings. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, page 11670–11688, Online. Association for Computational Linguistics.

Malte Ostendorff, Nils Rethmeier, Isabelle Augenstein, Bela Gipp, and Georg Rehm. 2022b. Neighborhood contrastive learning for scientific document representations with citation embeddings. *CoRR*, abs/2202.06671.

Malte Ostendorff, Nils Rethmeier, Isabelle Augenstein, Bela Gipp, and Georg Rehm. 2022c. Neighborhood contrastive learning for scientific document representations with citation embeddings.

Malte Ostendorff, Terry Ruas, Till Blume, Bela Gipp, and Georg Rehm. 2020. Aspect-based Document Similarity for Research Papers. In *Proceedings of the 28th International Conference on Computational Linguistics (COLING 2020)*.

Nikolaos Pappas and James Henderson. 2019. GILE: A Generalized Input-Label Embedding for Text Classification. *Trans. Assoc. Comput. Linguistics*, 7.

Advait Parulekar, Liam Collins, Karthikeyan Shanmugam, Aryan Mokhtari, and Sanjay Shakkottai. 2023. Infonce loss provably learns cluster-preserving representations.

Frank Pasquale. 2015. *The black box society: The secret algorithms that control money and information*. Harvard University Press.

Simon Pasternack. 1969. The scientific enterprise: Public knowledge. an essay concerning the social dimension of science. *Science*, 164(3880):669–670.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543. Association for Computational Linguistics.

Christian S. Perone, Roberto Silveira, and Thomas S. Paula. 2018. Evaluation of sentence embeddings in downstream and linguistic probing tasks.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '14*, pages 701–710, New York, New York, USA. ACM Press.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *CoRR*, abs/1802.05365.

Matthew E. Peters, Sebastian Ruder, and Noah A. Smith. 2019. To Tune or Not to Tune? Adapting Pretrained Representations to Diverse Tasks. In *ACL Workshop RepL4NLP*, Florence, Italy.

Maria Petroni, Letizia Petroni, Francesca Marchignoli, Anna Simona Sasdelli, Paolo Caraceni, Giulio Marchesini, and Federico Ravaioli. 2020. Nutrition in patients with type 2 diabetes: Present knowledge and remaining challenges. In *Nutrients vol. 13*.

Nicola Pezzotti, Thomas Höllt, Jan C. van Gemert, Boudewijn P. F. Lelieveldt, Elmar Eisemann, and Anna Vilanova. 2018. Deepeyes: Progressive visual analytics for designing deep neural networks. *IEEE Transactions on Visualization and Computer Graphics*, 24:98–108.

T Pham, T Tran, D Phung, and S Venkatesh. 2016. DeepCare: A Deep Dynamic Memory Model for Predictive Medicine. In *Advances in Knowledge Discovery and Data Mining*. Springer International Publishing.

Jason Phang, Haokun Liu, and Samuel R. Bowman. 2021a. Fine-tuned transformers show clusters of similar representations across layers. In *Proceedings of the Fourth BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 529–538, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jason Phang, Haokun Liu, and Samuel R. Bowman. 2021b. Fine-tuned transformers show clusters of similar representations across layers. In *Proceedings of the Fourth BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 529–538, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Nina Poerner, Benjamin Roth, and Hinrich Schütze. 2018. Interpretable textual neuron representations for NLP. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 325–327, Brussels, Belgium. Association for Computational Linguistics.

Nina Pörner, Ulli Waltinger, and Hinrich Schütze. 2020. Inexpensive domain adaptation of pretrained language models: Case studies on biomedical NER and covid-19 QA. In *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*, volume EMNLP 2020 of *Findings of ACL*, pages 1482–1490. Association for Computational Linguistics.

S Purushotham, C Meng, Z Che, and Y Liu. 2017. Benchmarking deep learning models on large healthcare datasets. *Journal of Biomedical Informatics*.

Yanru Qu, Dinghan Shen, Yelong Shen, Sandra Sajeev, Weizhu Chen, and Jiawei Han. 2021. CoDA: Contrast-enhanced and diversity-promoting data augmentation for natural language understanding. In *ICLR*.

Marissa Radensky, Doug Downey, Kyle Lo, Zoran Popovic, and Daniel S. Weld. 2022. Exploring the role of local and global explanations in recommender systems. In *CHI '22: CHI Conference on Human Factors in Computing Systems, New Orleans, LA, USA, 29 April 2022 - 5 May 2022, Extended Abstracts*, pages 290:1–290:7. ACM.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR.

Jack W. Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, H. Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, Eliza Rutherford, Tom Hennigan, Jacob Menick, Albin Cassirer, Richard Powell, George van den Driessche, Lisa Anne Hendricks, Maribeth Rauh, Po-Sen Huang, Amelia Glaese, Johannes Welbl, Sumanth Dathathri, Saffron Huang, Jonathan Uesato, John Mellor, Irina Higgins, Antonia Creswell, Nat McAleese, Amy Wu, Erich Elsen, Siddhant M. Jayakumar, Elena Buchatskaya, David Budden, Esme Sutherland, Karen Simonyan, Michela Paganini, Laurent Sifre, Lena Martens, Xiang Lorraine Li, Adhiguna Kuncoro, Aida Nematzadeh, Elena Gribovskaya, Domenic Donato, Angeliki Lazaridou, Arthur Mensch, Jean-Baptiste Lespiau, Maria Tsimpoukelli, Nikolai Grigorev, Doug Fritz, Thibault Sottiaux, Mantas Pajarskas, Toby Pohlen, Zhitao Gong, Daniel Toyama, Cyprien de Masson d'Autume, Yujia Li, Tayfun Terzi, Vladimir Mikulik, Igor Babuschkin, Aidan Clark, Diego de Las Casas, Aurelia Guy, Chris Jones, James Bradbury, Matthew Johnson, Blake A. Hechtman, Laura Weidinger, Iason Gabriel, William S. Isaac, Edward Lockhart, Simon Osindero, Laura Rimell, Chris Dyer, Oriol Vinyals, Kareem Ayoub, Jeff Stanway, Lorrayne Bennett, Demis Hassabis, Koray Kavukcuoglu, and Geoffrey Irving. 2021. Scaling language models: Methods, analysis & insights from training gopher. *CoRR*, abs/2112.11446.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Alessandro Raganato, Yves Scherrer, and Jörg Tiedemann. 2019. The MuCoW test suite at WMT 2019: Automatically harvested multilingual contrastive word sense disambiguation test sets for machine translation. In *WMT (Volume 2: Shared Task Papers, Day 1)*, Florence, Italy. Association for Computational Linguistics.

Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. 2017. Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6078–6087, Red Hook, NY, USA. Curran Associates Inc.

Vinay Venkatesh Ramasesh, Ethan Dyer, and Maithra Raghu. 2021. Anatomy of catastrophic forgetting: Hidden representations and task semantics. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Marc' Aurelio Ranzato, Y-Lan Boureau, and Yann LeCun. 2007. Sparse feature learning for deep belief networks. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*, NIPS'07, pages 1185–1192, USA. Curran Associates Inc.

Gabrielle Ras, Ning Xie, Marcel van Gerven, and Derek Doran. 2022. Explainable deep learning: A field guide for the uninitiated. *Journal of Artificial Intelligence Research*, 73:329–397.

General Data Protection Regulation. 2016. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46. *Official Journal of the European Union (OJ)*, 59(1-88):294.

Georg Rehm, Peter Bourgonje, Stefanie Hegele, Florian Kintzel, Julán Moreno Schneider, Malte Ostendorff, Karolina Zaczynska, Armin Berger, Stefan Grill, Sören Räuchle, Jens Rauenbusch, Lisa Rutenburg, André Schmidt, Mikka Wild, Henry Hoffmann, Julian Fink, Sarah Schulz, Jurica Seva, Joachim Quantz, Joachim Böttger, Josefine Matthey, Rolf Fricke, Jan Thomsen, Adrian Paschke, Jamal Al Qundus, Thomas Hoppe, Naouel Karam, Frauke Weichhardt, Christian Fillies, Clemens Neudecker, Mike Gerber, Kai Labusch, Vahid Rezanezhad, Robin Schaefer, David Zellhöfer, Daniel Siewert, Patrick Bunk, Lydia Pintscher, Elena Aleynikova, and Franziska Heine. 2020. QURATOR: Innovative Technologies for Content and Data Curation. In *Proceedings of QURATOR 2020 – The conference for intelligent content solutions*, Berlin, Germany. CEUR Workshop Proceedings, Volume 2535. 20/21 January 2020.

Nils Reimers and Iryna Gurevych. 2019a. 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 2019, Hong Kong, China, November 3-7, 2019*, pages 3980–3990. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2019b. 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 3980–3990, Stroudsburg, PA, USA. Association for Computational Linguistics.

Xuanchi Ren, Tao Yang, Yuwang Wang, and Wenjun Zeng. 2021. Do generative models know disentanglement? contrastive learning is all you need. *CoRR*, abs/2102.10543.

Nils Rethmeier and Isabelle Augenstein. 2020. Long-tail zero and few-shot learning via contrastive pretraining on and for small data.

Nils Rethmeier and Isabelle Augenstein. 2021. A Primer on Contrastive Pretraining in Language Processing: Methods, Lessons Learned and Perspectives.

Nils Rethmeier and Isabelle Augenstein. 2022a. Long-tail zero and few-shot learning via contrastive pretraining on and for small data. *Computer Sciences & Mathematics Forum*, 3(1).

Nils Rethmeier and Isabelle Augenstein. 2022b. Long-Tail Zero and Few-Shot Learning via Contrastive Pretraining on and for Small Data. *Computer Sciences; Mathematics Forum*, 3(1).

Nils Rethmeier and Isabelle Augenstein. 2022c. A primer on contrastive pretraining in language processing: Methods, lessons learned & perspectives. *ACM Comput. Surv.* Just Accepted.

Nils Rethmeier and Isabelle Augenstein. 2022d. A primer on contrastive pretraining in language processing: Methods, lessons learned & perspectives. *ACM Comput. Surv.*

Nils Rethmeier and Barbara Plank. 2019. MoRTy: Unsupervised learning of task-specialized word embeddings by autoencoding. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 49–54, Florence, Italy. Association for Computational Linguistics.

Nils Rethmeier, Vageesh Kumar Saxena, and Isabelle Augenstein. 2020a. Tx-ray: Quantifying and explaining model-knowledge transfer in (un-)supervised NLP. In *Proceedings of the Thirty-Sixth Conference on Uncertainty in Artificial Intelligence, UAI 2020, virtual online, August 3-6, 2020*, volume 124 of *Proceedings of Machine Learning Research*, pages 440–449. AUAI Press.

Nils Rethmeier, Vageesh Kumar Saxena, and Isabelle Augenstein. 2020b. TX-Ray: Quantifying and Explaining Model-Knowledge Transfer in (Un-)Supervised NLP. In *Proceedings of the Thirty-Sixth Conference on Uncertainty in Artificial Intelligence, UAI 2020, Toronto, Canada, August 03-06, 2020*. AUAI Press.

Nils Rethmeier, Necip Oguz Serbetci, Sebastian Möller, and Roland Roller. 2020c. Efficare: Better prognostic models via resource-efficient health embeddings. In *AMIA 2020, American Medical Informatics Association Annual Symposium, Virtual Event, USA, November 14-18, 2020*. AMIA.

Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. Model-agnostic interpretability of machine learning. *In ICML Workshop on Human Inter-pretability in Machine Learning*, abs/1606.05386.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. A primer in bertology: What we know about how BERT works. *Trans. Assoc. Comput. Linguistics*, 8:842–866.

Alexis Ross, Ana Marasović, and Matthew Peters. 2021. Explaining NLP models via minimal contrastive editing (MiCE). In *Findings of ACL-IJCNLP*, Online. Association for Computational Linguistics.

Sebastian Ruder and Barbara Plank. 2017. Learning to select data for transfer learning with bayesian optimization. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 372–382. Association for Computational Linguistics.

Laura Ruis, Akbir Khan, Stella Biderman, Sara Hooker, Tim Rocktäschel, and Edward Grefenstette. 2022. Large language models are not zero-shot communicators. *CoRR*, abs/2210.14986.

T Saito and M Rehmsmeier. 2015. The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. *PloS one*.

JIF Salluh and M Soares. 2014. ICU severity of illness scores: APACHE, SAPS and MPM. *Current opinion in critical care*.

Wojciech Samek, Grégoire Montavon, Andrea Vedaldi, Lars Kai Hansen, and Klaus-Robert Müller, editors. 2019. *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, volume 11700 of *Lecture Notes in Computer Science*. Springer.

Wojciech Samek and Klaus-Robert Müller. 2019. Towards explainable artificial intelligence. In Wojciech Samek, Grégoire Montavon, Andrea Vedaldi, Lars Kai Hansen, and Klaus-Robert Müller, editors, *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, volume 11700 of *Lecture Notes in Computer Science*, pages 5–22. Springer.

Victor Sanh, Thomas Wolf, and Alexander M. Rush. 2020. Movement Pruning: Adaptive Sparsity by Fine-Tuning. *CoRR*, abs/2005.07683.

Naomi Saphra and Adam Lopez. 2018. Language Models Learn POS First. In *EMNLP Workshop BlackboxNLP*, Brussels, Belgium.

Nikunj Saunshi, Orestis Plevrakis, Sanjeev Arora, Mikhail Khodak, and Hrishikesh Khandeparkar. 2019. A theoretical analysis of contrastive unsupervised representation learning. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5628–5637. PMLR.

Timo Schick and Hinrich Schütze. 2021. It's not just size that matters: Small language models are also few-shot learners. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 2339–2352. Association for Computational Linguistics.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. Facenet: A unified embedding for face recognition and clustering. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 815–823.

Robert Schwarzenberg, Marc Hübner, David Harbecke, Christoph Alt, and Leonhard Hennig. 2019. Layerwise Relevance Visualization in Convolutional Text Graph Classifiers. In *TextGraphs-13*, Hong Kong. ACL.

Lei Shi, Kai Shuang, Shijie Geng, Peng Gao, Zuohui Fu, Gerard de Melo, Yunpeng Chen, and Sen Su. 2021. Dense contrastive visual-linguistic pretraining. In *MM '21: ACM Multimedia Conference, Virtual Event, China, October 20 - 24, 2021*, pages 5203–5212.

Connor Shorten, Taghi M. Khoshgoftaar, and Borko Furht. 2021. Text data augmentation for deep learning. *J. Big Data*, 8(1):101.

Eero P Simoncelli and Bruno A Olshausen. 2001. Natural image statistics and neural representation. *Annual review of neuroscience*, 24(1):1193–1216.

Antoine Simoulin and Benoit Crabbé. 2021. Contrasting distinct structured views to learn sentence embeddings. In *EACL Student Research Workshop*, Online. Association for Computational Linguistics.

Abhinav Singh, Adrien Peyrache, and Mark D Humphries. 2019. Medial prefrontal cortex population activity is plastic irrespective of learning. *Journal of Neuroscience*, 39(18).

Arnab Sinha, Zhihong Shen, Yang Song, Hao Ma, Darrin Eide, Bo-June Paul Hsu, and Kuansan Wang. 2015. An overview of microsoft academic service (mas) and applications. *Proceedings of the 24th International Conference on World Wide Web*.

Leon Sixt, Maximilian Granz, and Tim Landgraf. 2019. When Explanations Lie: Why Modified BP Attribution Fails.

Henry Small. 1973. Co-citation in the scientific literature: A new measure of the relationship between two documents. *Journal of the American Society for Information Science*, 24(4):265–269.

Richard Socher, Milind Ganjoo, Christopher D Manning, and Andrew Ng. 2013. Zero-shot learning through cross-modal transfer. In *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc.

H Song, D Rajan, JJ Thiagarajan, and A Spanias. 2018a. Attend and diagnose: Clinical time series analysis using attention models. In *32nd AAAI*. AAAI press.

Huan Song, Deepta Rajan, Jayaraman Thiagarajan, and Andreas Spanias. 2018b. Attend and diagnose: Clinical time series analysis using attention models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1).

Yang Song and Diederik P. Kingma. 2021. How to train your energy-based models. *CoRR*, abs/2101.03288.

Karolina Stanczak, Lucas Torroba Hennigen, Adina Williams, Ryan Cotterell, and Isabelle Augenstein. 2022a. A latent-variable model for intrinsic probing. *CoRR*, abs/2201.08214.

Karolina Stanczak, Edoardo Ponti, Lucas Torroba Hennigen, Ryan Cotterell, and Isabelle Augenstein. 2022b. Same neurons, different languages: Probing morphosyntax in multilingual pre-trained models. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1589–1598, Seattle, United States. Association for Computational Linguistics.

Karolina Stańczak and Isabelle Augenstein. 2021. A survey on gender bias in natural language processing.

Ilia Stepin, Jose M. Alonso, Alejandro Catala, and Martín Pereira-Fariña. 2021. A survey of contrastive and counterfactual explanation generation methods for explainable artificial intelligence. *IEEE Access*, 9:11974–12001.

Hendrik Strobelt, Sebastian Gehrmann, Michael Behrisch, Adam Perer, Hanspeter Pfister, and Alexander M. Rush. 2019. Seq2seq-Vis: A Visual Debugging Tool for Sequence-to-Sequence Models. *IEEE TVCG*, 25(1).

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019a. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 3645–3650. Association for Computational Linguistics.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019b. Energy and policy considerations for deep learning in nlp. In *Proceedings of ACL 2019, Florence, Italy, July 28 - August 2, 2019*. Association for Computational Linguistics.

YuSheng Su, Xu Han, Yankai Lin, Zhengyan Zhang, Zhiyuan Liu, Peng Li, and Maosong Sun. 2021. CSS-LM: A contrastive framework for semi-supervised fine-tuning of pre-trained language models. *CoRR*, abs/2102.03752.

Anant Subramanian, Danish Pruthi, Harsh Jhamtani, Taylor Berg-Kirkpatrick, and Eduard Hovy. 2018. Spine: Sparse interpretable neural embeddings. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Siqi Sun, Zhe Gan, Yuwei Fang, Yu Cheng, Shuohang Wang, and Jingjing Liu. 2020. Contrastive distillation on intermediate representations for language model compression. In *EMNLP*, Online. Association for Computational Linguistics.

M Sundararajan, A Taly, and Q Yan. 2017a. Axiomatic attribution for deep networks. In *34th ICML*. JMLR.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017b. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 3319–3328. PMLR.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017c. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 3319–3328. PMLR.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 2818–2826. IEEE Computer Society.

Alon Talmor, Yanai Elazar, Yoav Goldberg, and Jonathan Berant. 2020. oLMpics-on what language model pre-training captures. *Transactions of the Association for Computational Linguistics*, 8:743–758.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. 2022. Efficient transformers: A survey. *ACM Computing Surveys*, 55(6).

Yi Tay, Mostafa Dehghani, Jai Prakash Gupta, Vamsi Aribandi, Dara Bahri, Zhen Qin, and Donald Metzler. 2021. Are pretrained convolutions better than pretrained transformers? In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 4349–4359. Association for Computational Linguistics.

Simone Teufel, Advaith Siddharthan, and Dan Tidhar. 2006. Automatic classification of citation function. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing - EMNLP '06*, page 103, Morristown, NJ, USA. Association for Computational Linguistics.

Yonglong Tian, Dilip Krishnan, and Phillip Isola. 2019. Contrastive multiview coding. In *European Conference on Computer Vision*.

Yonglong Tian, Dilip Krishnan, and Phillip Isola. 2020a. Contrastive representation distillation. In *International Conference on Learning Representations*.

Yonglong Tian, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola. 2020b. What makes for good views for contrastive learning? In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

N Tomašev, X Glorot, JW Rae, M Zielinski, H Askham, A Saraiva, A Mottram, C Meyer, S Ravuri, I Protsyuk, et al. 2019. A clinically applicable approach to continuous prediction of future acute kidney injury. *Nature*.

Marcos V. Treviso, Tianchu Ji, Ji-Ung Lee, Betty van Aken, Qingqing Cao, Manuel R. Ciosici, Michael Hassid, Kenneth Heafield, Sara Hooker, Pedro Henrique Martins, André F. T. Martins, Peter A. Milder, Colin Raffel, Edwin Simpson, Noam Slonim, Niranjan Balasubramanian, Leon Derczynski, and Roy Schwartz. 2022. Efficient methods for natural language processing: A survey. *CoRR*, abs/2209.00099.

Michael Tschannen, Josip Djolonga, Paul K. Rubenstein, Sylvain Gelly, and Mario Lucic. 2020. On mutual information maximization for representation learning. In *International Conference on Learning Representations*.

Yui Uehara, Tatsuya Ishigaki, Kasumi Aoki, Hiroshi Noji, Keiichi Goshima, Ichiro Kobayashi, Hiroya Takamura, and Yusuke Miyao. 2020. Learning with contrastive examples for data-to-text generation. In *COLING*, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Aäron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *CoRR*, abs/1807.03748.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017a. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017b. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017c. Attention Is All You Need. *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6000–6010.

Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart Shieber. 2020. Investigating gender bias in language models using causal mediation analysis. In *Advances in Neural Information Processing Systems*, volume 33, pages 12388–12401. Curran Associates, Inc.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, koray kavukcuoglu, and Daan Wierstra. 2016. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing Multi-Head Self-Attention: Specialized Heads Do the Heavy Lifting, the Rest Can Be Pruned. In *Proceedings of the ACL*, Florence, Italy. ACL.

David Wadden, Shanchuan Lin, Kyle Lo, Lucy Lu Wang, Madeleine van Zuylen, Arman Cohan, and Hannaneh Hajishirzi. 2020. Fact or Fiction: Verifying Scientific

Claims. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7534–7550, Stroudsburg, PA, USA. Association for Computational Linguistics.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *ICLR*, New Orleans, LA, USA.

Chenguang Wang, Zihao Ye, Aston Zhang, Zheng Zhang, and Alexander J. Smola. 2020a. Transformer on a Diet. *CoRR*, abs/2002.06170.

Chenguang Wang, Zihao Ye, Aston Zhang, Zheng Zhang, and Alexander J. Smola. 2020b. Transformer on a Diet.

Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. 2022a. Interpretability in the wild: a circuit for indirect object identification in gpt-2 small.

Kexin Wang, Nandan Thakur, Nils Reimers, and Iryna Gurevych. 2022b. GPL: Generative pseudo labeling for unsupervised domain adaptation of dense retrieval. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2345–2360, Seattle, United States. Association for Computational Linguistics.

Tongzhou Wang and Phillip Isola. 2020. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *ICML*, volume 119 of *PMLR*. PMLR.

X. Wang, Yufei Ye, and Abhinav Kumar Gupta. 2018. Zero-shot recognition via semantic embeddings and knowledge graphs. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6857–6866.

Zeerak Waseem, Smarika Lulz, Joachim Bingel, and Isabelle Augenstein. 2020. Disembodied Machine Learning: On the Illusion of Objectivity in NLP. Anonymous preprint under review.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2020. Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45.

Dustin Wright and Isabelle Augenstein. 2021. CiteWorth: Cite-worthiness detection for improved scientific document understanding. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1796–1807, Online. Association for Computational Linguistics.

Chao-yuan Wu, R. Manmatha, Alexander J Smola, and Philipp Krahenbuhl. 2017. Sampling Matters in Deep Embedding Learning. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2859–2867. IEEE.

Felix Wu, Tianyi Zhang, Amauri Holanda de Souza, Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. 2019. Simplifying Graph Convolutional Networks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 6861-6871, pages 815–826. PMLR.

John M. Wu, Yonatan Belinkov, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James R. Glass. 2020a. Similarity analysis of contextual word representation models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 4638–4655. Association for Computational Linguistics.

Xing Wu, Chaochen Gao, Liangjun Zang, Jizhong Han, Zhongyuan Wang, and Songlin Hu. 2021. Smoothed Contrastive Learning for Unsupervised Sentence Embedding. *arXiv:2109.04321*.

Zhaofeng Wu, Robert L. Logan IV, Pete Walsh, Akshita Bhagia, Dirk Groeneveld, Sameer Singh, and Iz Beltagy. 2022. Continued pretraining for better zero- and few-shot promptability. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Zhuofeng Wu, Sinong Wang, Jiatao Gu, Madian Khabsa, Fei Sun, and Hao Ma. 2020b. Clear: Contrastive learning for sentence representation.

Zhuofeng Wu, Sinong Wang, Jiatao Gu, Madian Khabsa, Fei Sun, and Hao Ma. 2020c. CLEAR: Contrastive Learning for Sentence Representation. *arXiv:2012.15466*.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. 2020. Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval. In *International Conference on Learning Representations*, pages 1–16.

Shufeng Xiong, Hailian Lv, Weiting Zhao, and Donghong Ji. 2018. Towards twitter sentiment classification by multi-level sentiment-enriched word embeddings. *Neurocomputing*, 275:2459–2466.

Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Y. Chang. 2015. Network representation learning with rich text information. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, page 2111–2117. AAAI Press.

Zhe Ye, Fang Li, and Timothy Baldwin. 2018. Encoding sentiment information into word vectors for sentiment analysis. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 997–1007. Association for Computational Linguistics.

Dani Yogatama, Cyprien de Masson d'Autume, Jerome Connor, Tomás Kociský, Mike Chrzanowski, Lingpeng Kong, Angeliki Lazaridou, Wang Ling, Lei Yu, Chris Dyer, and Phil Blunsom. 2019a. Learning and Evaluating General Linguistic Intelligence. *CoRR*.

Dani Yogatama, Cyprien de Masson d'Autume, Jerome T. Connor, Tomás Kociský, Mike Chrzanowski, Lingpeng Kong, Angeliki Lazaridou, Wang Ling, Lei Yu, Chris Dyer, and Phil Blunsom. 2019b. Learning and evaluating general linguistic intelligence. *CoRR*, abs/1901.11373.

Tal Zarsky. 2016. The trouble with algorithmic decisions: An analytic road map to examine efficiency and fairness in automated and opaque decision making. *Science, Technology, & Human Values*, 41(1):118–132.

Jure Zbontar, Li Jing, Ishan Misra, Yann Lecun, and Stephane Deny. 2021. Barlow twins: Self-supervised learning via redundancy reduction. In *ICML*, volume 139 of *PMLR*. PMLR.

Dejiao Zhang, Shang-Wen Li, Wei Xiao, Henghui Zhu, Ramesh Nallapati, Andrew O. Arnold, and Bing Xiang. 2021. Pairwise supervised contrastive learning of sentence representations. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5786–5798, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Honglun Zhang, Liqiang Xiao, Wenqing Chen, Yongkun Wang, and Yaohui Jin. 2018. Multi-Task Label Embedding for Text Classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, November 2 – November 4, 2018*, pages 4545–4553, Brussels, Belgium. Association for Computational Linguistics.

Ningyu Zhang, Luoqiu Li, Xiang Chen, Shumin Deng, Zhen Bi, Chuanqi Tan, Fei Huang, and Huajun Chen. 2022. Differentiable prompt makes pre-trained language models better few-shot learners. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Donglin Zhuang, Xingyao Zhang, Shuaiwen Leon Song, and Sara Hooker. 2021. Randomness in neural network training: Characterizing the impact of tooling.

Roland S. Zimmermann, Yash Sharma, Steffen Schneider, Matthias Bethge, and Wieland Brendel. 2021a. Contrastive learning inverts the data generating process.

Roland S. Zimmermann, Yash Sharma, Steffen Schneider, Matthias Bethge, and Wieland Brendel. 2021b. Contrastive learning inverts the data generating process. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 12979–12990. PMLR.