Semantic matching for text classification with complex class descriptions

Brian M. de Silva Kuan-Wen Huang Gwang Gook Lee
Karen Hovsepian Yan Xu Mingwei Shen
Amazon
{slvbria, kuanwen, gglee, khhovsep, yanxuml, mingweis}@amazon.com

Abstract

Text classifiers are an indispensable tool for machine learning practitioners, but adapting them to new classes is expensive. To reduce the cost of new classes, previous work exploits class descriptions and/or labels from existing classes. However, these approaches leave a gap in the model development cycle as they support either zero- or few-shot learning but not both. Existing classifiers either do not work on zero-shot problems, or fail to improve much with few-shot labels. Further, prior work is aimed at concise class descriptions, which may be insufficient for complex classes. We overcome these shortcomings by casting text classification as a matching problem, where a model matches examples with relevant class descriptions. This formulation lets us leverage labels *and* complex class descriptions to perform zero- and few-shot learning on new classes. We compare this approach with numerous baselines on text classification tasks with complex class descriptions and find that it achieves strong zero-shot performance and scales well with few-shot samples, beating strong baselines by 22.48% (average precision) in the 10-shot setting. Furthermore, we extend the popular Model-Agnostic Meta-Learning algorithm to the zero-shot matching setting and show it improves zero-shot performance by 4.29%. Our results show that expressing text classification as a matching problem is a cost-effective way to address new classes. This strategy enables zero-shot learning for cold-start scenarios and few-shot learning so the model can improve until it is capable enough to deploy.

1 Introduction

Advances in supervised text classification have enabled the automation of countless classification pipelines, but such methods struggle when new classes are introduced. In the typical setting, labels are collected for a fixed number of classes, and a model is trained to assign examples to their respective classes. However, incorporating new classes into the model incurs substantial costs. The process often involves labeling new examples, relabeling existing examples, and retraining the model. To reduce these costs, we seek a method which rapidly adapts from seen classes to new/unseen ones, using few to no labeled examples.

We study a common scenario wherein we have labeled data for a multi-class or multi-label text classification problem and then new classes are introduced. Crucially, we assume access to *class descriptions*. Such information is not uncommon in applications with frequent class changes. For example, in product compliance, new classes are often accompanied by documentation describing the products covered by the new regulations. Concise descriptions may suffice for simple classes like *books*, but detailed descriptions are needed for more specific or complicated classes like *books from genre 1 or genre 2 or . . . or genre 10*. Given labeled data and descriptions for existing classes, and only descriptions for new classes, our goal is to build a model to classify examples into new classes with minimal additional labeling, making it deployable in read-world scenarios. We focus on modest sized language models rather than large ones since they more readily scale to the large problems common in real-world applications. This paper addresses three problems in this setting: how to (1) leverage both class descriptions and existing labeled data to generalize to new classes (in zero- and few-shot settings); (2) effectively use long, complex class descriptions; and (3) align training with zero-shot evaluation.

First, to generalize from seen classes to unseen ones, an ideal approach should exploit all available data sources and be applicable in both zero- and few-shot settings. Intuitively, a model which incorporates more information should be more accurate. Labels from existing classes help language models adapt from pre-training objectives to the problem domain, while class descriptions aid in generaliz-

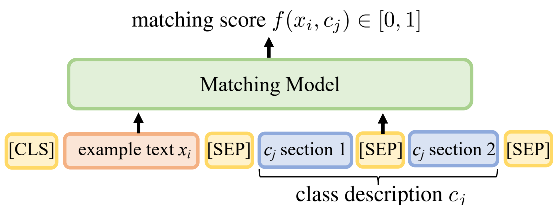

Figure 1: A matching model outputs a matching score $f(x_i, c_j)$ by assessing whether example text x_i "matches" class description c_j.

ing to new classes with fewer labels. Flexibility between zero- and few-shot learning is important for ensuring the model works when no or few labels are available. Additionally, zero-shot performance is often insufficient to meet real-world performance requirements, making it desirable for the model to significantly improve with more collected labels. Existing techniques for zero- and few-shot learning tend to satisfy only some of these desiderata. They either specialize in only zero- or few-shot settings, or they exploit only one of the aforementioned data sources. To address this, we follow Yin et al. 2019; Halder et al. 2020; Wang et al. 2021 and reformulate multi-class or multi-label text classification as a binary *matching* problem, where we train a model to match example texts with class descriptions based on semantic similarity. The model assigns each (example text, class description) pair a score, representing the likelihood of the example belonging to the respective class as in Figure 1. We use seen class labels to teach the model to match examples with applicable class descriptions. For the zero-shot setting we simply pass in descriptions for new classes, and for few-shot setting we continue to fine-tune the model on labels and descriptions from new classes. Therefore this approach uses (a) class descriptions and existing labels to perform zero-shot classification and (b) additional labels from new classes to accomplish few-shot learning.

Second, we seek a model which is effective for long, complex class descriptions, but this area is relatively unexplored. Previous work (Yin et al., 2019; Wang et al., 2021; Schick and Schütze, 2021; Pappas and Henderson, 2019; Pushp and Srivastava, 2017; Xiao et al., 2019; Zhang et al., 2019) has primarily focused on short inputs such as class names or brief sentences. They work when classes can be concisely summarized in a few words (e.g. *bicycle* or *socks*) but may be inadequate when class descriptions are necessarily long or specific (e.g. *Category I equipment*). To fully leverage the information in long, complex class descriptions, we use a cross-encoder architecture with a pre-trained transformer backbone. This approach facilitates rich interaction between example text and class descriptions by ingesting them together as a single concatenated input.

Third, the matching formulation described above unifies zero- and few-shot learning, but it still exhibits a discrepancy between training and evaluation. Matching model training follows the standard supervised learning paradigm: training loss is computed on *seen* classes and descriptions in the hope that the model will generalize to *unseen* descriptions. We propose a novel extension of Model-agnostic meta-learning (MAML) (Finn et al., 2017) to the matching setting, specifically targeting zero-shot learning. Similarly to how MAML was designed to align model training to few-shot evaluation, our extension augments the training process to better mimic zero-shot evaluation.

In this paper we investigate the efficacy of the matching approach using a cross-encoder architecture for zero- and few-shot text classification with long, complex class descriptions. Our **contributions** are as follows:

- We show that the matching model effectively leverages existing labels and complex class descriptions to reduce the label cost of adapting to new classes: it exhibits strong zero-shot performance (13.30% better than 10-shot BERT) and outperforms the baselines by 22.48% (macro average precision) in the 10-shot setting.

- We extend MAML from few- to zero-shot learning to improve zero-shot performance for matching models by 4.3%.

The remainder of this paper is structured as follows. We review related work in Section 2. Section 3 formalizes our problem statement, introduces the matching model and baselines, and details our extension of MAML to zero-shot learning. In Section 4 we discuss our experiment setup then present and interpret the results of our experiments. We give closing thoughts in Section 5.

2 Related work

Few-shot learning Few-shot learning was originally explored in the domain of image classification (Koch et al., 2015; Vinyals et al., 2016; Snell et al., 2017; Bateni et al., 2022) where a subset of classes—train classes—have sufficient examples, while the remaining classes—test classes—have relatively few examples. Such methods have since been repurposed for NLP tasks (Dopierre et al., 2021), and new techniques, such as in-context and prompt-based learning, have shown great promise for few-shot text classification. Large language models like GPT-3 and 4 (Brown et al., 2020) can solve NLP tasks by including few-shot examples in-context. However, their large parameter sizes and maximum token limitations hinder their ability to efficiently utilize more labels. Prompt-based methods like pattern exploiting training (PET) (Schick and Schütze, 2021, 2020) address these challenges by prompt-tuning language models and using them to annotate large unlabeled datasets for downstream tasks. However, PET's dependence on domain-specific data and handcrafted patterns limits its applicability. Methods like LM-BFF seek to automate prompt selection, but they are limited to cases when classes can be described in a few tokens (Gao et al., 2021a). SetFit (Tunstall et al., 2022) addresses the shortcomings of prompt-based methods by fine-tuning a sentence transformer in contrastive manner with a prompt-free framework. Finally, adapters (Houlsby et al., 2019; Pfeiffer et al., 2020a,b) have been shown to perform well in few-shot settings because of their ability to preserve the generalizability of pre-trained transformers while avoiding catastrophic forgetting and overfitting. In this work we test methods which do not require tailored prompts: prototypical networks, adapters, and SetFit.

Zero-shot learning Zero-shot learning for text classification involves classifying texts into classes which were not seen during the learning stage (unseen classes). To this end, zero-shot techniques often rely on transferring knowledge from seen to unseen classes by leveraging semantic attributes or descriptions of classes (Meng et al., 2020), inter-class correlations (Rios and Kavuluru, 2018), or joint embeddings of classes and examples (Nam et al., 2016; Schopf et al., 2022). Given the complexity of class descriptions in our datasets, we focus on methods that use semantic attributes. Matching models, one such approach, are of particular interest. These models take pairs of texts as input—in our case a class description and example text—and output a score. They differ from "matching networks" (Vinyals et al., 2016), which are instead aimed at few-shot learning. Matching models can be categorized into a taxonomy (Trabelsi et al., 2021; Khattab and Zaharia, 2020) based on the amount of interaction they allow between input pairs. In increasing order of interaction, performance, and computational complexity, we have (1) representation-based models (or bi-encoders) (Pappas and Henderson, 2019; Pushp and Srivastava, 2017), which embed the inputs separately before computing their matching score; (2) late interaction models (Trabelsi et al., 2021; Khattab and Zaharia, 2020), which allow more interplay between input pairs, but still have separate encoding stacks; and (3) all-to-all models (or cross-encoders) (Ye et al., 2020; Yin et al., 2019; Halder et al., 2020), which facilitate immediate interaction. Matching models also have close ties to information retrieval (see Appendix A).

Label-aware classification Label-aware classification techniques enhance classifiers by incorporating additional label information alongside standard input features. These methods use various types of label information, including label hierarchy (Nam, 2019), label dependency or correlation (Zhang et al., 2018; Yang et al., 2018; Xun et al., 2020), or label semantic information. The datasets in this paper lack label hierarchies, and label correlation is not practical in zero- and few-shot settings due to the need for data to learn the correlations. However, label semantic information is useful for zero- and few-shot applications. In the absence of labeled examples, models require some sort of information to scale to new classes. Early approaches gravitated toward representation-based or late interaction models and usually assumed short label descriptions (Xiao et al., 2019). Recent work (Schick and Schütze, 2021; Pappas and Henderson, 2019; Pushp and Srivastava, 2017; Nayak and Bach, 2020; Ye et al., 2021) has shown the promise of using textual entailment as a format for data-poor NLP tasks. These methods convert NLP tasks into entailment problems then leverage trained entailment models. This approach has yet to be explored for datasets with long class descriptions. Other label-aware classification methods use combinations of different types of label information, such as commonsense knowledge graphs for labels (Nayak and Bach, 2020); heterogeneous label graphs (Chen et al., 2017; Xu and Lapata, 2020); and word embeddings, class descriptions, class hierarchy, and knowledge graphs (Zhang et al., 2019).

3 Models

Matching differs from standard supervised learning because it uses class descriptions as model inputs, in addition to other features. We formalize the matching problem in Section 3.1. Next we describe our proposed solution in Section 3.2: a matching model that measures semantic similarity between class descriptions and examples. Finally, we discuss the baselines against which we compare the matching model in Section 3.3.

3.1 Problem formulation

Multi-label and multi-class classification problems can be converted to matching problems and vice-versa. For a multi-label classification task with p labels, we learn a function $h : X \to Y_S = \{0,1\}^p$ from a multi-label training set $D_{\text{train}} = \{(x_i, y_i)\}_{i=1}^N$, where x_i represents an input and y_j is a p-dimensional binary vector. To generalize to a matching problem, our objective shifts to learning a function $f : X \times C \to [0,1]$, where X denotes a text-based instance space and C a set of class descriptions. Here f maps (example, class description) pairs to probability scores. This change requires expanding the dataset D_{train} into a triplet format: $\tilde{D}_{\text{train}} = \{(x_i, c_j, y_{ij})\}$, where $(x_i, c_j, y_{i,j}) \in X \times C \times \{0,1\}$ for $i = 1, 2, \cdots, N$ and $j = 1, 2, \cdots, p$. In this formulation, instead of predicting the entire multi-label y_i, models predict a single binary label y_{ij} for class j. The probability scores $f(x_i, c_j)$ can be converted back into multi-label predictions via thresholding (Zhang and Zhou, 2013), i.e., $\hat{y}_i = \{\mathbb{1}[f(x_i, c_j) > \gamma_j] | j = 1, 2, \ldots, p\}$ for a set of thresholds $\{\gamma_j\}_{j=1}^p$, where $\mathbb{1}$ is the indicator function.

By viewing a multi-class problem as a constrained multi-label problem, we can convert it to or from a matching problem. To do so we replace the multi-class target $y_i^{MC} \in \{1, 2, \ldots, p\}$ with a multi-label target $y_i = (\mathbb{1}[y_i^{MC} = 1], \mathbb{1}[y_i^{MC} = 2], \ldots, \mathbb{1}[y_i^{MC} = p])$. Then, to get a class prediction, we take the argmax of f over all classes, i.e., $\hat{y}_i = \text{argmax}_j f(x_i, c_j)$.

One drawback of the matching formulation is that the computational complexity of inference scales linearly with the number of examples N *and* the number of classes p: $\mathcal{O}(Np)$. Standard classifiers' computational complexities depend only on the number of examples: $\mathcal{O}(N)$.

We evaluate the proposed solution and baselines in zero-shot and few-shot settings. Zero-shot means that a classifier is only trained on the seen label set Y_S without seeing any examples from the unseen label set Y_U. That is to say, the training set D_{train} is a subset of $X \times Y_S$, whereas the test set D_{test} is a subset of $X \times Y$, where $Y = Y_S \oplus Y_U = \{0,1\}^{p+q}$ and $Y_U = \{0,1\}^q$ with $q > 0$. We refer to the q new classes as unseen classes since they are not seen during training. The others are seen classes. Note that in order to build a triplet version of this dataset we require class descriptions $\{c_{p+m}\}_{m=1}^q$ for classes that are introduced at evaluation time.

In Perez et al. 2021, the authors show that prior work tends to overestimate few-shot performance due to hyperparameters and prompt selections tuned on held-out samples. Therefore, we follow the "true" few-shot learning setting. This means that validations sets contain no unseen class labels. Few-shot learning involves first training a classifier on $D_{\text{train}} \subset X \times Y_S$, then updating the classifier using a few-shot dataset $D_{\text{few-shot}} \subset X \times Y$ before evaluating it on a test dataset $D_{\text{test}} \subset X \times Y$ with the new classes. Here, k-shot learning indicates that $D_{\text{few-shot}}$ contains k positive examples of each new class.

3.2 Matching model

A matching model (Figure 1) learns semantic relationships between examples and class descriptions. We interpret its output $f(x_i, c_j)$ as a matching score between example x_i and class description c_j. During training, we teach the model to relate class descriptions with examples so that, during testing, the model can classify examples into unseen classes by matching examples with corresponding (potentially new) class descriptions.

Our model design largely follows previous works (Schick and Schütze, 2021; Pappas and Henderson, 2019; Pushp and Srivastava, 2017). For comparability with prior research, we adopt a pre-trained BERT model (Devlin et al., 2018) as our base matching model, though other architectures can augment performance (see Appendix E.2, where we show larger models increase performance). We express the input to the model (x_i, c_j) as "[CLS] x_i [SEP] c_j [SEP]". When class descriptions have multiple sections we use [SEP] tokens to separate them, e.g. "[CLS] x_i [SEP] c_j section 1 [SEP] c_j section 2 [SEP]." Including the [SEP] token allows the model to distinguish between sections. We use the hidden vector for [CLS] in the final layer as the aggregate representation of the input: $\mathbf{g}(x_i, c_j)$. We then apply a dropout layer, a linear layer, and a softmax to obtain matching score $f(x_i, c_j)$. In the ranking model taxonomy of Siblini et al. 2020; Nam 2019 this is considered an *all-to-all interaction* or *cross-encoder* model since BERT's attention mechanism facilitates immediate interaction between example and class description texts.

For zero-shot classification of an example x_i we use the model trained on D_{train} to compute matching scores between x_i and unseen class descriptions $f(x_i, c_{p+1}), f(x_i, c_{p+2}), \ldots, f(x_i, c_{p+q})$ in addition to matching scores on the seen class descriptions $f(x_i, c_1), f(x_i, c_2), \ldots, f(x_i, c_p)$. To adapt a model trained on D_{train} to new classes with few-shot data, we fine-tune the model on $D_{\text{few-shot}}$ as in Wang et al. 2021. The matching model is applicable to both multi-class and multi-label problems. And it can generalize to any number of new classes in the zero-shot setting via additional class descriptions. However, there are likely practical constraints on the number of classes it can learn during fine-tuning. We leave the question of how many to future work.

This approach is closely related to the one proposed in Yin et al. 2019 (and, later, Wang et al. 2021; Halder et al. 2020), except for two key differences. We extend the maximum token length of 128 in Yin et al. 2019 to 512 to account for our datasets' longer class descriptions. We also skip pre-training on the MNLI dataset (Williams et al., 2018) since our initial experiments showed it harms performance on our datasets.

3.2.1 Meta-learning

Model-agnostic meta-learning (MAML) (Finn et al., 2017) is a popular algorithm for meta-learning neural networks. It is highly versatile, and has been applied to classification in vision and NLP, and to regression and reinforcement learning problems (Lee et al., 2022; van der Heijden et al., 2021). MAML attempts to learn ideal parameter initializations of arbitrary neural networks for rapid improvement on new tasks with gradient descent optimization, making it ideal for few-shot learning. This is accomplished by aligning the training objective with few-shot performance on randomly chosen downstream tasks. Training proceeds in a series of episodes; each episode simulating several few-shot learning problems. Starting from the current parameter initialization θ_i, several models are fine-tuned for a fixed number of iterations on one *support set* out of the batch of tasks selected from this episode: few-shot training samples from a subset of the classes. The fine-tuned model is then tested on the *query set*—test samples from the classes in the support set—and the parameter initialization is updated for the next episode: $\theta_i \to \theta_{i+1}$. Together, a support and query set make up a task. Typically episodes consist of batches of tasks, which are resampled at the start of each episode. Bringing training and testing into agreement results in a parameter initialization that is primed for few-shot learning. It is natural to wonder whether we can improve matching model performance further with MAML.

The standard MAML classification framework limits meta-learning to the encoder layers sitting below the classification head, which is task-specific. The classification head is replaced and randomly initialized for each new task in MAML's inner loop, and then frozen in the outer loop. By formulating text classification as a binary "class-to-text" matching task, we can use the same classification head across new tasks, and, therefore, expand meta-learning through the classification head, allowing the MAML framework to improve zero-shot performance. In particular, recall that the MAML training algorithm proceeds by sampling, at each episode, a set of n support classes S, a set of n query classes Q, and labeled samples for each class in S and Q. The few-shot application and the need to randomly initialize the classification head for each new task enforce $S = Q$ so that the n-class classification head learned during the inner loop (on classes in S) can be used during the outer loop (on classes in Q). In our matching MAML formulation, this restriction is relaxed – the query set can consist of new unseen classes, which corresponds to directly evaluating the performance under the zero-shot scenario. We note that this generalization is not dependent on the model architecture. It is dependent on our ability to formulate the text classification task as a binary matching task.

We propose two novel ways of sampling the query set classes in MAML: (1) *Local zero-shot*: the query and support set classes do not overlap within a given task but may overlap across tasks; (2) *Global zero-shot*: query set classes and support set classes have no overlap. Both are aimed at improving zero-shot performance by mimicking zero-shot evaluation during training. Each is a different way of enforcing $S \cap Q = \emptyset$, as opposed to the usual constraint $S = Q$.

3.3 Baselines

We consider four baselines: two standard supervised models and two few-shot approaches. Unlike the matching model, the baselines do not consider the class descriptions. Therefore, they tackle classification in the usual way rather than reformulating it as a matching problem. The supervised models consist of gradient boosted trees via **LightGBM** (Ke et al., 2017) and a **BERT** classifier (Devlin et al., 2018). Both are popular tools for text classification. We consider two few-shot classifiers: a BERT model with adapter modules (**BERT + Adapter**) (Houlsby et al., 2019) and **SetFit** (Tunstall et al., 2022). BERT + Adapter inserts lightweight adapter layers with bottleneck architectures in each transformer block of a pre-trained BERT model. When fine-tuning on a downstream task, we only train the weights of adapter layers. The small number of trainable parameters acts as a regularizer to prevent overfitting. SetFit (Tunstall et al., 2022) is a recently proposed method that uses a two-stage training approach to achieve strong few-shot performance without prompts or verbalizers. It first fine-tunes a pre-trained sentence transformer on contrastive pairs sampled from the few-shot set, then it trains a classification head on the resulting embeddings. See Appendix C for more information about each model.

Beyond the above baselines, we also generalized prototypical networks (Snell et al., 2017) to the multi-label setting (Multi-Label Prototypical Classifier—**MLPC**). MLPC uses labeled examples from the seen classes to learn a metric space in which to classify new examples by comparison against class prototypes. Each class has two prototypes—one for positive (in-class) examples and one for negative (out-of-class) examples—formed by averaging the embeddings of few-shot positive or negative instances of each class. As MLPC is novel, we give more details in Appendix B.

Besides prototypical networks, which necessitate labeled examples for class representations, there are also *unsupervised* similarity-based methods such as Haj-Yahia et al. 2019; Schopf et al. 2022. They perform classification based on similarity between class and example representations, relying on keywords to describe each class. In addition to the baselines above, we also compare Lbl2TransformerVec (Schopf et al., 2022) against a matching model in a zero-shot setting (see Appendix E.3).

4 Experiments

We conduct two experiments to test the models from Section 3 in zero- and few-shot scenarios. Our primary experiment compares the ability of each model to adapt to new classes. We then explore whether we can improve the zero- or few-shot performance of the matching model by incorporating meta-learning during training. Both experiments involve four text classification datasets: two public and two internal real-world e-commerce datasets. The experiment setup is similar in each case. We first select 20% of classes to be unseen, simulating new classes. Next we partition the data into training, few-shot, and test sets consisting of examples from seen, unseen, and seen and unseen classes, respectively. We train the models to predict seen classes on the train sets, update them to predict unseen classes using the few-shot sets, and evaluate them on the test sets. In order to measure the speed at which the models adapt to unseen classes we vary the number of labels available in the few-shot sets as 10, 50, 100, 500, and 1000. The few-shot samples are nested, e.g. a 100-shot dataset is a subset of the 500- and 1000- shot sets.

Section 4.1 describes the four datasets. Sections 4.2 and 3.2.1 present the results of our model comparison and meta-learning experiments, respectively, and discuss their implications. Appendix D gives details on how we constructed D_{train}, $D_{\text{few-shot}}$, D_{test} and their triplet counterparts.

4.1 Setup

We tested the matching model and baselines from Section 3 on four text classification datasets: Yahoo Answers (Yahoo) (Zhang et al., 2015), Amazon Review Data 2018 (Amazon) (Ni et al., 2019), Product Classification 1 (PC1), and Product Classification 2 (PC2). Yahoo and Amazon are public datasets and PC1 and PC2 are internal. Amazon, PC1, and PC2 are all e-commerce product classification datasets. We describe each dataset in more detail below and put additional information in Appendix D.

Few-shot Yahoo (Yahoo) The Yahoo Answers dataset involves a multi-class text classification task: to classify historical Q&A data from the Yahoo Answers website into one of 10 categories based on the question and top answer. The labels are noisy as they were assigned by the original question posters. And there is often ambiguity about which category a question belongs to. These factors make it hard for any classifier to reach high accuracy. This dataset has also been used for more constrained learning tasks. For example, Yin et al. 2019 adapts this dataset for zero-shot text classification and Schick and Schütze 2020 uses it for few-shot text classification.

Amazon Review Data 2018 (Amazon) Amazon review data 2018 (Ni et al., 2019) is a review rating classification dataset. The dataset comprises over 230 million reviews of 15 million Amazon products along with metadata. We extract product text (e.g. title, description) from the metadata and convert it into a multi-class text classification task. The task aims to classify products into one of the top 23 product categories based on product text.

PC1 and PC2 Product Classification 1 and 2 (PC1 and PC2) are multi-label classification datasets consisting of products sold on an e-commerce website and their associated regulatory classes. Inputs x_i are the concatenation of texts associated with the products, like item names, descriptions, and product types. Some classes are general in scope and others are narrow. PC1 has 50 classes, with each sample belonging to at least one class. PC2 can be viewed as a collection of 28 binary classification tasks; one task per class. Labels for both datasets are from human annotators. Each class description comes from a policy document describing the class. These documents are much more complex than class descriptions previously studied in the literature. They have multiple sections indicating, for example, specific negative examples for the class and general characteristics of positive examples.

Metric: macro average precision We measure model performance, relative to a 10-shot BERT baseline, via average precision (AP). To get a summary score, we compute the AP for each unseen class, then take an unweighted average (i.e. macro AP). Next, to obtain a relative score, we subtract the macro AP for a 10-shot BERT baseline. Finally, we further average across three seen/unseen class splits to reduce the impact of unseen class selection. We chose AP because we do not know *a priori* at which operating point a model will be deployed. AP summarizes performance across all operating points. Further, for imbalanced datasets like the PC1 and PC2 test sets, AP is a more indicative of performance than accuracy.

4.2 Model comparison

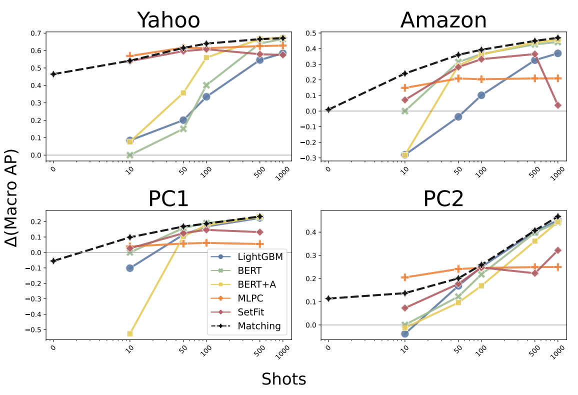

Figure 2: Absolute improvement in macro average precision (AP) over a BERT 10-shot baseline (gray lines), averaged across unseen classes and three seen/unseen class splits. Note the log scale of the x-axes and the different scales of the y-axes. See Table 1 for details.

No one approach does best on all four datasets across all amounts of few-shot data, but the matching model comes close (Figure 2). The way performance varies with additional labeled examples is model-dependent, but the matching model exhibits favorable performance in both the low- and high-data regimes. At low sample counts the matching model uses unseen class descriptions to achieve modest levels of performance, whereas the LightGBM, BERT, and BERT + Adapter baselines tend to struggle. With 10 shots the matching model outperforms these models by an average of 33.8%, 25.4% , and 44.1%, respectively, emphasizing the importance of class descriptions in enabling classification in zero-shot and k-shot scenarios for small k. With enough labels most baselines eventually learn class boundaries without the aid of descriptions: for larger label counts they tend to “catch up” with the matching model. However, the matching model remains competitive in such cases. This shows that, with a matching model, we may use class descriptions to buoy performance levels in the low-data regime and still make efficient use of extra labels in high-data scenarios.

MLPC and SetFit are most competitive when few labels are available, but they struggle to improve with more samples. For MLPC this is because labeled data from unseen classes do not improve its embedding. Instead it uses these data to build class

Dataset	Model	0 shot	10 shots	50 shots	100 shots	500 shots	1000 shots
Yahoo	LightGBM	-	8.40	20.07	33.47	54.57	58.43
	BERT	-	0.00	15.07	40.10	64.08	66.63
	BERT+Adapter	-	7.61	35.71	55.98	**66.59**	**67.33**
	MLPC	-	**56.88**	**61.65**	61.33	62.57	62.86
	SetFit	-	53.93	59.66	60.77	57.90	57.49
	Matching	**46.45**	54.17	61.49	**63.98**	66.56	67.11
Amazon	LightGBM	-	−28.06	-3.68	10.12	32.59	36.98
	BERT	-	0.00	31.45	36.52	42.75	44.22
	BERT+Adapter	-	−28.22	29.07	36.15	44.06	45.72
	MLPC	-	14.86	20.88	20.39	20.94	20.95
	SetFit	-	7.12	28.18	33.25	36.54	3.68
	Matching	**0.91**	**24.05**	**36.01**	**39.27**	**44.96**	**46.96**
PC1	LightGBM	-	−10.16	11.55	16.82	22.34	-
	BERT	-	0.00	15.83	**19.05**	22.68	-
	BERT+Adapter	-	−52.69	10.43	17.65	23.18	-
	MLPC	-	3.92	5.75	6.18	5.48	-
	SetFit	-	2.67	12.49	14.64	13.18	-
	Matching	**−5.51**	**9.82**	**16.83**	18.67	**23.32**	-
PC2	LightGBM	-	−3.82	16.84	24.98	40.22	45.13
	BERT	-	0.00	12.22	21.78	39.77	44.22
	BERT+Adapter	-	−1.22	9.60	16.87	36.12	44.37
	MLPC	-	**20.47**	**24.11**	24.58	24.90	24.95
	SetFit	-	7.28	17.70	24.76	22.26	32.14
	Matching	**11.36**	13.69	20.09	**25.89**	**40.70**	**46.75**
Average	LightGBM	-	−8.41	11.20	21.35	37.43	46.85
	BERT	-	0.00	18.64	29.36	42.32	51.69
	BERT+Adapter	-	−18.63	21.21	31.66	42.49	52.47
	MLPC	-	24.03	28.10	28.12	28.47	36.25
	SetFit	-	17.75	29.51	33.35	32.47	31.11
	Matching	**13.30**	**25.43**	**33.61**	**36.95**	**43.88**	**53.61**

Table 1: Average absolute improvement in macro AP (%) compared to 10-shot BERT. We average the scores across unseen classes and three seen/unseen class splits. The matching model achieves the best performance, on average, across each few-shot setting.

prototypes. If, after model training, the embedding is insufficient to separate positive and negative examples from unseen classes, improving the class prototypes can only help so much. For SetFit we hypothesize that sentence transformer fine-tuning is best suited to low-data scenarios and has diminishing returns with more samples. The other models, however, improve rapidly with additional data.

Matching model performance is improved further by using a more powerful backbone. Using DeBERTa (He et al., 2020) instead of BERT increases zero- and max-shot performance by 5.24% and 2.02%, respectively (see Appendix E.2).

There is a gap between the matching model's seen and unseen class metrics in the zero-shot setting, as one would expect (Table 2). With enough labels, however, performance on unseen classes exceeds the original performance on seen classes. This is likely because there are ~4 times fewer unseen classes for the model to learn during few-shot fine-tuning compared to seen class training.

Dataset	Seen	Unseen
Yahoo	52.83	46.45
Amazon	43.74	0.91
PC1	13.61	-5.51
PC2	53.74	11.36

Table 2: Zero-shot matching model performance (macro AP %) relative to 10-shot BERT. We average across seen or unseen classes and three seen/unseen class splits.

Meta-learning To test meta-learning applied to matching we follow the same experiment design as above but substitute standard fine-tuning with MAML when training on seen classes. Fine-tuning on few-shot data is unchanged. We use the MAML++ training algorithm, which stabilizes and improves on MAML (Antoniou et al., 2019), adapted from computer vision models to transformers as in van der Heijden et al. 2021. We test the three query set sampling strategies discussed above. See Appendix C for further details.

Model	0 shots	10 shots	50 shots	100 shots	500 shots	1000 shots
Standard training	−9.65	0.00	7.16	10.05	16.68	22.06
MAML++ few-shot	−8.83	−2.26	5.72	8.98	15.81	20.83
Local zero-shot	−5.36	0.36	6.40	8.74	15.14	21.14
Global zero-shot	−16.75	−7.45	1.67	5.14	13.76	18.94

Table 3: Average absolute improvement in macro AP (%) compared to a 10-shot matching model (BERT backbone) with standard training. We average the scores across unseen classes then across all four datasets.

Local zero-shot query sampling achieves a significant lift over standard training in the zero-shot setting (+4.29% on average) with negligible or harmful impact in the few-shot setting (Table 3). Hence we recommend this approach for zero-shot but not few-shot learning. Interestingly, we observe no improvement for few-shot learning with standard MAML++, perhaps due to the persistent binary classification layer of the matching setting. Global zero-shot query sampling significantly degrades performance in all settings.

5 Conclusion

This paper shows that casting text classification as a matching problem is an effective way to address new classes. Matching models have multiple benefits relative to other methods in this space. Namely, they (1) leverage both class descriptions and standard labels; and (2) can perform zero- and few-shot classification—both of which are often needed in practical scenarios. We show that they are able to exploit long, complex class descriptions, achieving 13.30% macro AP with zero-shots (relative to 10-shot BERT) and surpassing few-shot baselines by 22.48% with 10-shots. We also generalize MAML to the zero-shot setting and show that our proposed method improves zero-shot performance by a further 4.29%. This demonstrates that, when class descriptions are available, the proposed matching models are capable of handling new classes from the beginning (zero-shot, cold-start) to the end of the model development cycle (few- to many-shot).

Limitations

The proposed model leverages long text (class description) to measure similarity, however text length is limited to 512 tokens due to the underlying language model. This limitation may impact the model's performance on longer texts. We used four datasets in our study, but three of them (one public and two internal) are for e-commerce product classification. As a result, our model's performance may vary for other text classification tasks. An inherent disadvantage of the matching formulation is that the inference cost scales linearly with both the number of examples and the number of candidate classes, whereas for a traditional classifier the cost scales with the number of examples. Therefore, a matching formulation may be too computationally expensive for applications with many classes. Finally, our results may not generalize to much larger or smaller language models as we only considered modest-sized models with 66 to 336 million parameters.

Ethics Statement

This research was conducted in accordance with the ACL Ethics Policy.[1] We proposed a classification model that leverages class descriptions which could be the most beneficial to those doing zero- or few-shot text classification, especially in the product compliance space. In our study, we used two internal datasets, but they do not contain any personal or private information, which reduces privacy concerns. It is also important to note that as we used language models pre-trained on internet data (BERT, RoBERTa, DeBERTa, and DistilBERT), our model may inherit their biases. However, bias and fairness are not investigated in the paper.

References

Antreas Antoniou, Harri Edwards, and Amos Storkey. 2019. How to train your MAML. In *Seventh International Conference on Learning Representations*.

Peyman Bateni, Jarred Barber, Raghav Goyal, Vaden Masrani, Jan-Willem van de Meent, Leonid Sigal, and Frank Wood. 2022. Beyond simple meta-learning: Multi-purpose models for multi-domain, active and continual few-shot learning. *arXiv preprint arXiv:2201.05151*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind

[1]https://www.aclweb.org/portal/content/acl-code-ethics

Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879.

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*.

Thomas Dopierre, Christophe Gravier, and Wilfried Logerais. 2021. A neural few-shot text classification reality check. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 935–943.

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

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021a. Making pre-trained language models better few-shot learners. 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 3816–3830.

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*.

Hao Guo, Xiangyang Li, Lei Zhang, Jia Liu, and Wei Chen. 2021. Label-aware text representation for multi-label text classification. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7728–7732. IEEE.

Suchin Gururangan, Ana Marasović, 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*, pages 8342–8360.

Zied Haj-Yahia, Adrien Sieg, and Léa A Deleris. 2019. Towards unsupervised text classification leveraging experts and word embeddings. In *Proceedings of the 57th annual meeting of the Association for Computational Linguistics*, pages 371–379.

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.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*.

Yutai Hou, Yongkui Lai, Yushan Wu, Wanxiang Che, and Ting Liu. 2021. Few-shot learning for multi-label intent detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 13036–13044.

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 *International Conference on Machine Learning*, pages 2790–2799. PMLR.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30.

Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 39–48.

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

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196. PMLR.

Hung-Yi Lee, Shang-Wen Li, and Thang Vu. 2022. Meta learning for natural language processing: A survey. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 666–684.

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

Jimmy Lin, Rodrigo Nogueira, and Andrew Yates. 2021. Pretrained transformers for text ranking: Bert and beyond. *Synthesis Lectures on Human Language Technologies*, 14(4):1–325.

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

Sean MacAvaney, Andrew Yates, Arman Cohan, and Nazli Goharian. 2019. CEDR: Contextualized embeddings for document ranking. In *Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval*, pages 1101–1104.

Yu Meng, Yunyi Zhang, Jiaxin Huang, Chenyan Xiong, Heng Ji, Chao Zhang, and Jiawei Han. 2020. Text classification using label names only: A language model self-training approach. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9006–9017.

Jinseok Nam. 2019. Learning label structures with neural networks for multi-label classification.

Jinseok Nam, Eneldo Loza Mencía, and Johannes Fürnkranz. 2016. All-in text: Learning document, label, and word representations jointly. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30.

Nihal V Nayak and Stephen H Bach. 2020. Zero-shot learning with common sense knowledge graphs. *arXiv preprint arXiv:2006.10713*.

Jianmo Ni, Jiacheng Li, and Julian McAuley. 2019. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. 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 188–197.

Nikolaos Pappas and James Henderson. 2019. Gile: A generalized input-label embedding for text classification. *Transactions of the Association for Computational Linguistics*, 7:139–155.

Hyunji Park, Yogarshi Vyas, and Kashif Shah. 2022. Efficient classification of long documents using transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 702–709.

Ethan Perez, Douwe Kiela, and Kyunghyun Cho. 2021. True few-shot learning with language models. *Advances in neural information processing systems*, 34:11054–11070.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2020a. Adapterfusion: Non-destructive task composition for transfer learning. *arXiv preprint arXiv:2005.00247*.

Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. 2020b. Adapterhub: A framework for adapting transformers. *arXiv preprint arXiv:2007.07779*.

Pushpankar Kumar Pushp and Muktabh Mayank Srivastava. 2017. Train once, test anywhere: Zero-shot learning for text classification. *arXiv preprint arXiv:1712.05972*.

Natraj Raman, Sameena Shah, and Manuela Veloso. 2022. Structure and semantics preserving document representations. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 780–790.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Anthony Rios and Ramakanth Kavuluru. 2018. Few-shot and zero-shot multi-label learning for structured label spaces. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing. Conference on Empirical Methods in Natural Language Processing*, volume 2018, page 3132. NIH Public Access.

Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Timo Schick and Hinrich Schütze. 2020. Exploiting cloze questions for few shot text classification and natural language inference. *arXiv preprint arXiv:2001.07676*.

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*, pages 2339–2352.

Tim Schopf, Daniel Braun, and Florian Matthes. 2022. Evaluating unsupervised text classification: zero-shot and similarity-based approaches. In *Proceedings of the 6th International Conference on Natural Language Processing and Information Retrieval, 2022*, pages 6–15.

Konstantinos Sechidis, Grigorios Tsoumakas, and Ioannis Vlahavas. 2011. On the stratification of multi-label data. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2011, Athens, Greece, September 5-9, 2011, Proceedings, Part III 22*, pages 145–158. Springer.

Wissam Siblini, Mohamed Challal, and Charlotte Pasqual. 2020. Delaying interaction layers in transformer-based encoders for efficient open domain question answering. *arXiv preprint arXiv:2010.08422*.

Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30.

Mohamed Trabelsi, Zhiyu Chen, Brian D Davison, and Jeff Heflin. 2021. Neural ranking models for document retrieval. *Information Retrieval Journal*, 24:400–444.

Lewis Tunstall, Nils Reimers, Unso Eun Seo Jo, Luke Bates, Daniel Korat, Moshe Wasserblat, and Oren Pereg. 2022. Efficient few-shot learning without prompts. *arXiv preprint arXiv:2209.11055*.

Niels van der Heijden, Helen Yannakoudakis, Pushkar Mishra, and Ekaterina Shutova. 2021. Multilingual and cross-lingual document classification: A meta-learning approach. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1966–1976.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. 2016. Matching networks for one shot learning. *Advances in neural information processing systems*, 29.

Marilyn Walker, Heng Ji, and Amanda Stent. 2018. Proceedings of the 2018 conference of the north american chapter of the association for computational linguistics: Human language technologies, volume 1 (long papers). In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*.

Sinong Wang, Han Fang, Madian Khabsa, Hanzi Mao, and Hao Ma. 2021. Entailment as few-shot learner. *arXiv preprint arXiv:2104.14690*.

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL HLT 2018*, pages 1112–1122. Association for Computational Linguistics (ACL).

Lin Xiao, Xin Huang, Boli Chen, and Liping Jing. 2019. Label-specific document representation for multi-label text classification. 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 466–475.

Yumo Xu and Mirella Lapata. 2020. Coarse-to-fine query focused multi-document summarization. In *Proceedings of the 2020 Conference on empirical methods in natural language processing (EMNLP)*, pages 3632–3645.

Guangxu Xun, Kishlay Jha, Jianhui Sun, and Aidong Zhang. 2020. Correlation networks for extreme multi-label text classification. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1074–1082.

Pengcheng Yang, Xu Sun, Wei Li, Shuming Ma, Wei Wu, and Houfeng Wang. 2018. Sgm: Sequence generation model for multi-label classification. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3915–3926.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32.

Chenchen Ye, Linhai Zhang, Yulan He, Deyu Zhou, and Jie Wu. 2021. Beyond text: Incorporating metadata and label structure for multi-label document classification using heterogeneous graphs. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3162–3171.

Zhiquan Ye, Yuxia Geng, Jiaoyan Chen, Jingmin Chen, Xiaoxiao Xu, SuHang Zheng, Feng Wang, Jun Zhang, and Huajun Chen. 2020. Zero-shot text classification via reinforced self-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3014–3024.

Wenpeng Yin, Jamaal Hay, and Dan Roth. 2019. Benchmarking zero-shot text classification: Datasets, evaluation and entailment approach. 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 3914–3923.

Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Min Zhang, and Shaoping Ma. 2020. An analysis of BERT in document ranking. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1941–1944.

Jingqing Zhang, Piyawat Lertvittayakumjorn, and Yike Guo. 2019. Integrating semantic knowledge to tackle zero-shot text classification. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1031–1040.

Min-Ling Zhang and Zhi-Hua Zhou. 2013. A review on multi-label learning algorithms. *IEEE transactions on knowledge and data engineering*, 26(8):1819–1837.

Wenjie Zhang, Junchi Yan, Xiangfeng Wang, and Hongyuan Zha. 2018. Deep extreme multi-label learning. In *Proceedings of the 2018 ACM on international conference on multimedia retrieval*, pages 100–107.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28.

A Additional related work

Information retrieval The goal for an information retrieval (IR) system is to return objects—from a large pool of candidates—which are most relevant to a given query. We will assume the objects to be text-based documents. We may view classification with class descriptions as an IR problem; the example text provides the query and the class descriptions are the documents to be searched. IR systems are often used as components of other systems: QA (Chen et al., 2017), summarization (Xu and Lapata, 2020), recommendation (Walker et al., 2018), and search and navigation (Raman et al., 2022). Modern architectures can generally be broken into two stages: a fast, high-recall retrieval stage (e.g. BM25 (Robertson et al., 2009)) seeking to quickly pare down the list of possibly relevant documents, and a slower but more accurate ranker which ranks the pruned list of documents by relevance to the query. Many document ranking models compute a score for each document with respect to the query, much like a matching model. Indeed, neural language models like BERT (Devlin et al., 2018) are common choices for ranking models (Zhan et al., 2020; MacAvaney et al., 2019; Lin et al., 2021; Li et al., 2020). One drawback of such architectures, however, is the difficulty with which they scale to longer documents. Researchers have proposed various techniques (Lin et al., 2021) for ingesting longer input texts such as aggregating scores across different document sections (MacAvaney et al., 2019; Park et al., 2022) and using networks with larger maximum token lengths (Yang et al., 2019). Though recent work has begun to question the effectiveness of such methods for text classification (Park et al., 2022).

B Multi-label prototypical classifier

Our multi-label prototypical classifier (MLPC) adapts prototypical networks (Snell et al., 2017) from multi-class image classification to the multi-label text classification setting. In essence, MLPC uses examples from seen classes to learn a metric space in which it can classify new examples via comparison against seen and unseen class prototypes. Each class has two prototypes—one for positive (in-class) examples and one for negative (out-of-class) examples—formed by averaging the embeddings of few-shot positive and negative instances of each class. Hence our objective is to learn a d-dimensional metric space with an embedding transformation $g_\theta : X \to \mathbb{R}^d$ with trainable parameters θ which maps samples from the text-based instance space X to the embedding space. To handle text input we follow Dopierre et al. 2021 and implement this map with a transformer (BERT (Devlin et al., 2018)).

Episodic training To simulate the few-shot testing environment (few labels per class), we train the transformer g_θ with a series of few-shot episodes. In each episode, we sample support sets $S_{j,0}, S_{j,1}$, and a query set T_j from triplet training dataset $\tilde{D}_{\text{train}}\{(x_i, c_j, y_{ij})\}_{I\times J}$ for each class j, where $I = \{1, \dots, N\}$ and $J = \{1, \dots, p\}$ are the index sets of text samples and classes. From the samples (x_i, c_j, y_{ij}) of the support sets, we compute class prototypes $\mathbf{c}_{j,0}$ and $\mathbf{c}_{j,1}$ for class j with the embedding transformation g_θ, and negative and positive text samples in $S_{j,0}, S_{j,1}$, respectively.

$$\mathbf{c}_{j,k} = \frac{1}{|S_{j,k}|} \sum_{x \in S_{j,k}} g_\theta(x), \quad j \in J, k \in \{0, 1\}. \tag{1}$$

For every text sample x in query set T_j, we compute the Euclidean distances between its embedding $\mathbf{x} = g_\theta(x)$ and class prototypes $\mathbf{c}_{j,0}$ and $\mathbf{c}_{j,1}$ and then we define the output distribution of prototypical classifier given an input x as

$$\begin{aligned} f(x_i, c_j) &= \Pr(y_{i,j} = 1 | x, c_j) \\ &= \frac{\exp(-d(\mathbf{x}, \mathbf{c}_{j,1}))}{\exp(-d(\mathbf{x}, \mathbf{c}_{j,0})) + \exp(-d(\mathbf{x}, \mathbf{c}_{j,1}))}, \end{aligned} \tag{2}$$

where $d(\mathbf{x}, \mathbf{c}) = \|\mathbf{x} - \mathbf{c}\|_2$ is the Euclidean distance. The output distribution is essentially a softmax over the distances between embedding $\mathbf{x}$ and class prototypes $\mathbf{c}_{j,0}$ and $\mathbf{c}_{j,1}$. Given a labeled example $(x, c_j, y_{i,j} = k)$, $k \in \{0, 1\}$ from query set T_j, the sample cross-entropy loss between ground truth label $y_{i,j}$ and prediction $f(x_i, c_j)$ is

$$\begin{aligned} \text{CE Loss} &= -\log(\Pr(y_{i,j} = k | x, c_j)) \\ &= d(\mathbf{x}, \mathbf{c}_{j,k}) + \log\left(\sum_{k'=0}^{1} \exp(-d(\mathbf{x}, \mathbf{c}_{j,k'}))\right). \end{aligned} \tag{3}$$

We learn θ via SGD by minimizing the cross-entropy loss summed over labeled examples, repeating the process multiple times for each class with randomly generated support and query sets. See Algorithm 1 in Appendix C for pseudocode.

Inference The test datasets can be expressed in triplet format as $\tilde{D}_{\text{test}} = \{(x_i, c_j, y_{ij})\}_{I_{\text{test}} \times J_{\text{test}}}$, where I_{test} and J_{test} are the index sets of test samples and test classes. The classifier produces probability score $f(x_i, c_j)$ of x_i belonging to class j for all $(i, j) \in I_{\text{test}} \times J_{\text{test}}$ given few-shot support sets $S_{j,0}, S_{j,1}, j \in J_{\text{test}}$ as

$$\begin{aligned} f(x_i, c_j) &= \Pr(y_{i,j} = 1 | x_i, c_j) \\ &= \frac{\exp(-d(\mathbf{x}_i, \mathbf{c}_{j,1}))}{\exp(-d(\mathbf{x}_i, \mathbf{c}_{j,0})) + \exp(-d(\mathbf{x}_i, \mathbf{c}_{j,1}))}, \\ &\quad \forall (i, j) \in I_{\text{test}} \times J_{\text{test}}. \end{aligned} \tag{4}$$

Note that MLPC does not use the class descriptions, only the few-shot samples. We found that incorporating class descriptions as in (Hou et al., 2021) (using them to anchor class prototypes) hurt performance.

C Model training

This section provides details of how each model was trained, including model backbones and hyperparameter choices.

C.1 Matching model

We used the following hyperparameter configuration when training the zero-shot version of the matching model across all datasets:

- Base model: *bert-base-uncased*
- Dropout: 0.1
- Epochs: 2
- Batch size: 32
- Learning rate schedule: linear warmup for 10% of an epoch to a maximum value of 5e-6, then linear decay to 0

When tuning the zero-shot matching models on few-shot data we changed the following hyperparameters:

- Epochs: 3
- Learning rate schedule: start at 5e-6 with cosine decay to 0

For the few-shot experiments we randomly selected 5% of the Yahoo training set to use as validation data and 1% of the Amazon, PC1, and PC2 training sets. For the sanity check experiment on the Yahoo dataset we used the validation set given in (Yin et al., 2019).

C.2 Multi-label prototypical classifier

We used the following hyperparameter configuration to learn the parameters of embedding transformation g_θ via episodic training with prototypical network:

- Base model: *bert-base-uncased*
- Dropout: 0.1
- Batch size: 64
- Episodes: 5000
- Maximum learning rate: {1e-6, 2e-6, 5e-6, 1e-5, 2e-5}
- Learning rate schedule: linear warmup to maximum value at 6% of training steps and then linear decay to 0
- Early-stopping: True

Our experiments show that few-shot Yahoo and PC1 datasets favor lower learning rates (best at 1e-6) while we get the best performance on the PC2 dataset at 1e-5.

Algorithm 1 gives an overview of the episodic training for MLPC.

C.3 LightGBM

For LightGBM we used the default model parameters with `objective='binary'` for the PC1 and PC2 datasets and `objective='multiclass'` for the Yahoo dataset. We trained the model on the few-shot datasets:

- Yahoo - train a single multi-class model using samples in the few-shot dataset (rather than training separate binary classifiers for each class then taking the argmax of probability scores)
- PC1 - train a separate binary model for each unseen class using all samples in the few-shot dataset
- PC2 - train a separate binary model for each unseen class using all samples in the few-shot dataset

We applied the same text preprocessing as for the other models and used a simple Bag-of-Words text featurization. The PC1 and PC2 datasets each have four categorical features associated with them,

Algorithm 1: Episodic training of MLPC. E denotes the number of training episodes, N_S and N_T denote the number of support set and query set, respectively. RandomSample$(D, y_{ij} = k, N)$ denotes a uniformly sampled subset of D with N samples satisfying the condition of $y_{ij} = k$.

```
Data: Triplet training data
      $\tilde{D}_{\text{train}} = \{(x_i, c_j, y_{ij})\}_{I \times J}$
1  Initialize $g_\theta$            // with an off-the-shelf pre-trained transformer
2  for $e$ in $\{1, \dots, E\}$ do
3      for $j$ in $J$ do
4          Obtain a slice of dataset $\tilde{D}_j = \{(x_i, c_j, y_{ij})\}_{i \in I}$ for class $c_j$
5          $S_{j,0} =$ RandomSample$(\tilde{D}_j, y_{ij} = 0, N_S)$
6          $S_{j,1} =$ RandomSample$(\tilde{D}_j, y_{ij} = 1, N_S)$
7          $T_j =$ RandomSample$(\tilde{D}_j \setminus S_{j,0} \setminus S_{j,1}, y_{ij} = 0$ or $1, N_T)$
8          Compute $\{\mathbf{c}_{j,0}, \mathbf{c}_{j,0}\}$ from $S_{j,0}, S_{j,1}$
9          Compute loss $L$ from $T_j$ and $\{\mathbf{c}_{j,0}, \mathbf{c}_{j,0}\}$
10         Update $\theta \leftarrow \theta - \eta \nabla_\theta L$
                                            // iterate through each class
                                            // iterate through episodes
```

three of which we converted to text and appended to the example text: product type, minimum age, and maximum age. The LightGBM model, however, can ingest categorical features directly. Therefore we also passed these four features to LightGBM as categorical variables using an ordinal encoding.

C.4 BERT

We trained the BERT models on text data with the same preprocessing steps as with the matching model and MLPC, using the following hyperparameters:

- Base model: *bert-base-uncased*
- Epochs: 10
- Learning rate schedule: linear warmup for 10% of an epoch to a maximum value of 1e-6, then linear decay to 0
- Dropout: 0.1

C.5 BERT + Adapter

We followed the adapter framework proposed in Pfeiffer et al. 2020b to insert adapter module with a 768×48 down-projection layer and a 48×768 up-projection layer in every BERT transformer block. We then train BERT + Adapter models using the following hyperparameters:

- Base model: *bert-base-uncased*
- Epochs: 10
- Learning rate schedule: linear warmup for 10% of an epoch to a maximum value of 5e-5, 1e-4, or 2e-4, then linear decay to 0
- Dropout: 0.1

We have found that training a BERT + Adapter model with a higher maximum learning rate works better since the adapter weights are initialized randomly without pre-training.

C.6 SetFit

We used the official SetFit package[2] to train the SetFit models. Wherever possible, we chose default or recommended parameters. We list these parameters below.

- Sentence transformer backbone: `all-mpnet-base-v2`. We chose this pre-trained backbone because it gave the best all-around performance of the available sentence transformers[3].
- Contrastive loss: cosine similarity
- Epochs: 1 for contrastive training, 25 for tuning the classification head
- Learning rates: 2e-5 for contrastive training, 1e-2 for tuning the classification head
- We froze the sentence transformer while training the classification head as otherwise we faced CUDA OOM issues.
- The number of iterations (text pairs to generate for contrastive learning) differed based on the number of few-shot samples k. Using a fixed number of iterations across all values of k results in an unwieldy number of samples for large k. We set the number of iterations to: 5 if $k < 500$, 2 if $k = 500$, and 1 if $k = 1000$.

[2] https://github.com/huggingface/setfit

[3] https://www.sbert.net/docs/pretrained_models.html#sentence-embedding-models/

We trained a separate binary (one-vs-all) model for each unseen class on each few-shot dataset before evaluating on the test set. We used the same text processing steps as for the other models.

C.7 Meta-learning

The training and evaluation process for the meta-learning experiments closely followed that for the matching model. We simply substituted the standard method of training the zero-shot model with a MAML++ (Antoniou et al., 2019) training routine. The model inputs, few-shot fine-tuning process, and evaluation are all unchanged. We used a BERT model architecture.

Meta-training consists of a fixed number of epochs, each with the same number of training episodes. At the end of each epoch we estimate the loss on the validation with a fixed number of episodes. We use the model checkpoint with the lowest validation loss. When drawing samples for the support and query sets we balance the number of positive and negative examples. The matching formulation naturally induces an imbalance between positives and negatives for datasets with complete label information and large numbers of classes (all datasets but PC2). Without enforcing this balance the model sometimes becomes trapped in local minima; assigning every sample a score close to 0. Samples for the support and query sets are drawn with replacement.

Below we give the important hyperparameters from the meta-learning experiments.

- Batch size: 1
- Max token length: 512
- Minimum learning rate: 1e-8
- Classes per task: 4
- Support samples per class: 3
- Query samples per class: 7
- Inner and outer loop learning rates: 5e-5
- Total epochs: 50
- Episodes per epoch: 200
- Multi-step loss epochs: 10
- Number of evaluation tasks during validation: 200

Slightly different hyperparameters (shown below) worked best for each query sampling method. We selected the hyperparameters based on performance on loss on the validation set.

- MAML++: 60 evaluation tasks
- Local zero-shot query sampling: inner and outer loop learning rates were 1e-5
- Global zero-shot query sampling: 50% of classes were reserved for the query set

D Few-shot datasets

In this section we provide additional details about the Yahoo, PC1, and PC2 few-shot datasets. We repeated the procedures below five times, selecting different seen and unseen classes in each case to test the models' robustness. However in this paper we only report results for the first three splits: 0, 1, and 2.

D.1 Yahoo

To align the Yahoo Answers dataset from Pushp and Srivastava 2017; Gururangan et al. 2020 with our few-shot experiments, we pool all the examples from the original train, test, and dev sets and build our own train, few-shot, and test sets. We first randomly select two of the ten classes to act as unseen classes (classes 3 and 4 for the experiments that follow). From each of the remaining eight seen classes we then sample 80% of the examples to serve as D_{train}, with the other 20% going to D_{test}. We randomly select 1000 examples from each of the two unseen classes to build $D_{\text{few-shot}}$. We allocate the remaining examples from the unseen classes to D_{test}. In order to ease the computational burden of evaluating on the test set, we further downsample D_{test} to form $D_{test-small}$ and evaluate on that set instead. Finally, we convert D_{train}, $D_{\text{few-shot}}$, and D_{test} into triplet format so that each example $(x_i, y_i) \in D_{\text{train}}$ is mapped to $p = 8$ examples $\{(x_i, c_j, \mathbb{1}[y_i = j])\}_{j=1}^{8}$ and each example $(x_i, y_i) \in D_{\text{few-shot}} \cup D_{test-small}$ is mapped to $p + q = 10$ examples $\{(x_i, c_j, \mathbb{1}[y_i = j])\}_{j=1}^{10}$. See D for details about the datasets. We use the same class descriptions described in Section E.1. We measure performance by averaging the AP of the model across unseen classes, i.e. we compute a AP score for each unseen class, then take the unweighted average as the final score.

Table 4 shows the distribution of samples across the different splits.

Split	Total samples	Samples per seen class	Samples per unseen class
Train	934.4k	116.8k	0
Test	523.6k	29.2k	145k
Test (small)	100k	6k	26k
n-shot	$2n$	0	n

Table 4: Number of samples in the Yahoo few-shot data splits. $n \in \{10, 50, 100, 500, 1000\}$.

D.2 Amazon

We build a multi-class e-commerce product classification dataset by extracting the metadata from Ni et al. 2019. We take 'asin' as the unique identifier for product examples, 'category' as the target class, and the concatenation of 'title', 'description', and 'feature' as example text. To build a balanced classification dataset, we first choose the top-23 product categories which have more than 4000 product examples. We then downsample and obtain 4000 product examples per class. For each class, we need class description which describes the product category. We manually search on Wikipedia and Google and combine text from difference sources with minor modifications, e.g. removing irrelevant information like citations, pronunciation, root, etc.

Next, we construct D_{train}, $D_{\text{few-shot}}$, and D_{test} by selecting 5 unseen classes and 18 seen classes. For each seen class, we put 2000 examples in D_{train}, 1000 examples in $D_{\text{few-shot}}$, and 1000 examples in D_{test}; while for each unseen classes, we only put 1000 examples in $D_{\text{few-shot}}$ and 1000 examples in D_{test}. Table 5 shows the number of positive examples for each class.

D.3 Product Classification 1 (PC1)

Product Classification 1 (PC1) dataset contains 50 classes which have between 57 and 4,952 examples with a mean of 1,319. Examples can belong to multiple class (multi-label classification problem) with an average of 1.35 classes and a maximum of seven. Labels were assigned by manual reviewers. The labels are dense in the sense that every example is a positive example for some classes and a negative example for all the others. As a result, after we convert these multi-label datasets into their triplet versions, there are many more negative examples than positive examples; the modal example becomes one positive and 49 negative examples.

We construct D_{train}, $D_{\text{few-shot}}$, and D_{test} by first selecting $q = 10$ unseen classes (20%), building a pool of examples falling into any of the 10 classes.[4] From this pool we sample 500 examples from each class to form $D_{\text{few-shot}}$. We do not have enough samples for 1000-shot learning. Note that because examples can belong to multiple classes, it is possible for $D_{\text{few-shot}}$ to contain more than 500 samples from a given class. We assign the remaining examples from unseen classes to D_{test}. We then split the data from seen classes between D_{train} and D_{test} at an 80/20 ratio, using iterative stratification (Sechidis et al., 2011) to ensure different label combinations are distributed evenly across the train and test sets. In the end $|D_{\text{train}}| = 29,917$ and $|D_{\text{test}}| = 13,782$. Finally, we convert the splits into triplet format.

For compatibility with MLPC, $D_{\text{few-shot}}$ must also have 500 positive examples from each *seen* class. We sample these from D_{train}, upsampling for classes with fewer than 500 positive examples.

Table 6 shows the number of positive examples for each class in D_{train} and D_{test}.

D.4 Product Classification 2 (PC2)

Unlike with the PC1 dataset, here the labels are sparse. With the PC1 data we had full knowledge of the multi-label for each example, e.g. $y_i^{PC1} = (0, 1, 0, 0, 1, 0, \dots, 0)$, but with the PC2 data we have only partial knowledge, e.g. $y_i^{PC2} = (?, ?, 1, ?, 0, \dots, ?)$. It is still possible to convert such labels into a triplet format, however: (x_i, y_i^{PC2}) is mapped to $\{(x_i, c_j, y_{ij}^{PC2}) | 1 \le j \le p, y_{ij}^{PC2} \neq ?\}$ rather than $\{(x_i, c_j, y_{ij}^{PC2}) | 1 \le j \le p\}$ (replacing p with $p + q$ for the test set). For the PC2 dataset we perform this conversion before constructing the train, few-shot, and test sets.

The train set $\hat{D}_{\text{train}}$ has, wherever possible, 50,000 positive and 50,000 negative samples from each seen class and no samples from unseen classes. Therefore the PC2 train set is much more balanced than the PC1 one. 9 of 22 seen classes have fewer

[4]We only consider classes with at least 700 examples as viable candidates for unseen classes to ensure there are enough to split between $D_{\text{few-shot}}$ and D_{test}

Split	Total samples	Samples per seen class	Samples per unseen class
Train	36,000	2,000	0
Test	23,000	1,000	1,000
n-shot	$2n$	n	n

Table 5: Number of samples in the Amazon few-shot data splits, where $n \in \{10, 50, 100, 500, 1000\}$. In each split, there are 5 unseen classes and 18 seen classes. For split 0, the unseen classes are: Home and Kitchen; Tools and Home Improvement; CDs and Vinyl; Movies and TV; Musical Instruments, while the seen classes are: Clothing, Shoes and Jewelry; Books; Automotive; Sports and Outdoors; Electronics; Toys and Games; Cell Phones and Accessories; Kindle Store; Arts, Crafts and Sewing; Grocery and Gourmet Food; Patio, Lawn and Garden; Office Products; Pet Supplies, Industrial and Scientific; Video Games; Appliances; Software; Collectibles and Fine Art.

than 100,000 samples due to a lack of labels (with a minimum of 7,223). We equip $\hat{D}_{\text{few-shot}}$ with 1,000 positive and 1,000 negative samples. When we discuss k-shot learning with respect to the PC2 dataset, we use k positive and k negative samples. As with the PC1 dataset, the few-shot data are nested. The test set $\hat{D}_{\text{test}}$ consists of 1,000 positive examples and 19,000 negative examples for each class.[5] We intentionally created an imbalanced test set to mimic a production environment, in which examples from any given class are expected to be rare.

Table 7 gives an overview of the number of samples (counting both positive and negative) for each of the classes in the *triplet versions* of the PC2 datasets.

D.5 Dataset difficulty

All models perform better on the PC1 and Amazon datasets compared to Yahoo and PC2 (Table 1). This is likely because the Yahoo and PC2 problems are harder than PC1. The labels in the Yahoo dataset are noisy. They were assigned by users of Yahoo! Answers and are often wrong. Furthermore, it is common for questions to plausibly fall under multiple categories even though the labels are multi-class. Section E.1 shows that our results are in line with those previously published in (Yin et al., 2019). The PC2 dataset is challenging by design. Its list of classes is not exhaustive, but rather consists of product classes which proved especially difficult for existing ML models. Therefore we expect lower scores on this dataset at small label counts. However, most models appear to scale well on this dataset, suggesting that high performance is achievable with more labels.

[5]In the PC2 test set two classes have only 9,000 negative examples and one has 4,000 due to data availability.

E Additional experiments

E.1 A sanity check

Aiming to verify the correctness of our implementations of the methods from 3 we conducted a sanity check by replicating an experiment from (Yin et al., 2019) using our models and comparing the results. The experiment involves zero-shot multi-class text classification using class descriptions. Our multi-label prototypical classifier is not a zero-shot model, as it requires few-shot support samples for each class. In this experiment, we take 1000 support samples from the dev set (separated from train and test sets). The comparison between multi-label prototypical classifier and other zero-shot models is not fair; however, this experiment can still show us whether our implementations are reasonable.

E.1.1 Setup

For this experiment we use a Yahoo Answers dataset introduced in Zhang et al. 2015, with the data split and experiment setup used in Yin et al. 2019. The objective for this dataset is to use the concatenation of a Yahoo Answers question and its top answer to predict which of 10 categories the poster filed the question under: Society & Culture, Science & Mathematics, Health, Education & Reference, Computers & Internet, Sports, Business & Finance, Entertainment & Music, Family & Relationships, Politics & Government.

We use the class descriptions (Yin et al., 2019) found worked best for this dataset: c_j = "this text describes something about w_j", where w_j is the 1-3 word name of the category (e.g. "Sports"). The dev and test sets contain 6k and 10k examples from each of the 10 classes, respectively (60k and 100k samples total). There are two non-overlapping train sets, each with a different set of five seen classes with 130k samples per class. The class

Class	Seen	Train samples (+)	Test samples (+)
0	✓	999	315
1	✓	867	226
2	✓	748	192
3		100	419
4	✓	847	216
5	✓	829	207
6	✓	499	126
7	✓	793	206
8	✓	890	223
9	✓	1,071	280
10		100	1,393
11	✓	760	438
12	✓	1,026	263
13	✓	853	219
14	✓	1,323	364
15	✓	1,756	469
16	✓	1613	449
17		107	714
18	✓	1,241	528
19	✓	1,503	554
20	✓	810	202
21	✓	388	111
22	✓	792	199
23	✓	2,676	1,569
24	✓	791	198
25	✓	818	218
26	✓	986	246
27	✓	522	130
28	✓	2,371	1,089
29	✓	1,026	579
30	✓	759	190
31	✓	1,107	305
32	✓	860	218
33		100	437
34		100	454
35	✓	216	54
36	✓	775	222
37		106	672
38	✓	545	286
39		100	669
40	✓	944	268
41	✓	990	250
42	✓	382	101
43	✓	45	11
44		100	965
45	✓	227	567
46	✓	822	208
47		100	416
48	✓	1,332	350
49		100	476

Table 6: Summary of the (non-triplet) versions of the PC1 dataset (split 0). We report the number of *positive* samples for each class.

Class	Seen	Train samples	Test samples
0	✓	100,000	20,000
1	✓	100,000	20,000
2		0	20,000
3	✓	100,000	20,000
4	✓	100,000	20,000
5		0	10,000
6	✓	100,000	20,000
7	✓	48,790	20,000
8	✓	100,000	20,000
9	✓	63,334	20,000
10	✓	100,000	20,000
11	✓	19,842	20,000
12		0	20,000
13	✓	100,000	20,000
14	✓	42,009	20,000
15	✓	100,000	20,000
16		0	10,000
17	✓	7,223	5,000
18	✓	24,253	20,000
19		0	20,000
20	✓	100,000	20,000
21		0	20,000
22	✓	100,000	20,000
23	✓	25,899	20,000
24	✓	100,000	20,000
25	✓	100,000	20,000
26	✓	81,253	20,000
27	✓	15,885	20,000

Table 7: Summary of the triplet versions of the PC2 dataset (split 0).

index sets for the two train sets are $\{1, 3, 5, 7, 9\}$ and $\{2, 4, 6, 8, 10\}$, respectively. Converting these datasets to from a multi-class format to a triplet format increases the size of the dev and test sets by a factor of 10 and the train set by a factor of 5. Each example is mapped to a positive example for one class and negative examples for the remaining classes. The presence of unseen classes in the validation set means that this is not a "true" zero-shot problem.

We compare our models against two models proposed in (Yin et al., 2019). The "Binary-BERT" model is equivalent to our matching model, but uses a max token length of 128 (as opposed to 512 for the matching model). The "MNLI-entailment" model is the same as the Binary-BERT model, except it is pre-trained on MNLI. All models are fine-tuned on the training data, with hyperparameters tuned on the dev set, then evaluated on the test set. We measure performance via multi-class accuracy, as in (Yin et al., 2019). The Binary-BERT, MNLI-entailment, and matching models use an additional hyperparameter which dampens the matching scores for seen classes by a fixed amount because others (Pushp and Srivastava, 2017; Guo et al., 2021) have noted that matching scores for unseen classes tend to be lower than for seen classes in the zero-shot setting.

E.1.2 Results

The results in Table 8 confirm that our implementations are sound. We find that we are able to get similar or better performance on unseen classes when comparing our approaches against those in Yin et al. 2019. On unseen classes our MLPC approach improves significantly on the accuracy scores from Yin et al. 2019. This is not surprising given that MLPC gets to use few-shot samples from unseen classes, whereas the other models are zero-shot.

E.2 Matching model backbones

While we used BERT-base as the backbone for the matching model in the main paper, other language models could lead to different levels of performance. In this experiment we replaced BERT-base with other transformer models with sizes ranging from 66 million to 336 million parameters (Table 9). The model weights came from Hugging Face:[6]. We replicated the experiments from the main paper on seen/unseen class split 0, changing only the backbone models. To understand the trade-offs between compute cost and performance we also tracked the training time and inference latency for each backbone (Table 11).

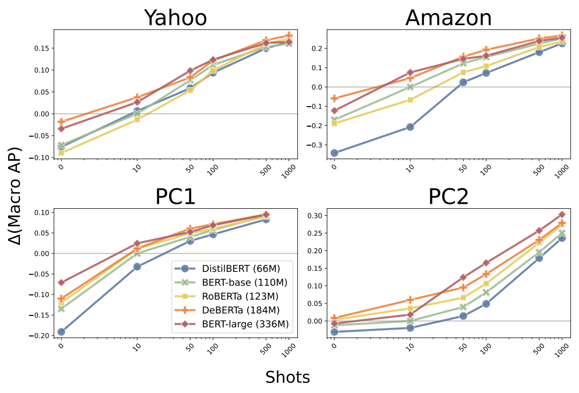

Figure 3: A comparison of different matching model backbones. The baseline in this case is a matching model with a BERT backbone trained on 10 few-shot samples for each class (gray line). Note the log scale of the x-axes and the different scales of the y-axes. See Table 10 for details.

[6]`https://huggingface.co/`. In particular, we used the following models: `distilbert-base-uncased`, `bert-base-uncased`, `roberta-base`, `microsoft/deberta-v3-base`, and `bert-large-uncased`

Increasing the size of the backbone generally leads to improved performance. (see Figure 3 and Table 10). The larger DeBERTa and BERT-large models consistently outperform RoBERTa and BERT-base, which in turn tend to outperform DistilBERT. Despite BERT-large having almost twice as many parameters as DeBERTa, both models have comparable performance. This shows the benefit of improved pre-training strategies.

The downside to larger models is their increased training and inference cost. The train and inference times for the backbones are proportional to the parameter counts (Table 11). DeBERTa and BERT-large are about 1.7 and 2.6 times slower than BERT, respectively, while DistilBERT takes about 0.6 the time of BERT. Which backbone is appropriate will depend on the accuracy and compute constraints of the problem at hand, but DeBERTa is a good all-around choice.

E.3 Zero-shot comparison against unsupervised similarity-based approach

In this section, we compare the matching model and Lbl2TransformerVec (Schopf et al., 2022), an unsupervised similarity-based approach. Lbl2TransformerVec operates by relying on manually defined keywords for each class, and employs a pre-trained embedding transformation, such as Doc2Vec (Le and Mikolov, 2014), S-BERT (Reimers and Gurevych, 2019), or SimCSE (Gao et al., 2021b). The transformation embeds keywords, example texts, and classes into the same space. Notably, the class representation is constructed as the average embedding of a set of most relevant examples to given keywords. In our experiment, we observe stronger results with S-BERT among the three embedding transformations. To perform zero-shot classification, Lbl2TransformerVec leverages class keywords, while the matching model uses class descriptions. Lbl2TransformerVec is aimed at zero-shot learning, and lacks a clear way to enhance performance with additional few-shot labels, so we focus on the zero-shot setting in this experiment.

The matching model significantly outperforms Lbl2TransformerVec across all four datasets (Table 12). The lower performance of Lbl2TransformerVec can be attributed, in part, to the process of extracting class keywords from long class descriptions. These descriptions often involve complex elements such as negations and

	Split 0		Split 1	
Model	Seen classes	Unseen classes	Seen classes	Unseen classes
Binary-BERT	72.60%	44.30%	80.60%	34.90%
MNLI-entailment	70.90%	52.10%	77.30%	45.30%
MLPC	71.41%	**58.21%**	77.11%	**57.67%**
Matching	66.54%	56.50%	74.84%	44.52%

Table 8: Comparison of models with entailment approach on the original Yahoo Answers dataset (zero shots). We measure multi-class accuracy. We obtained metrics for the first two models from Yin et al. 2019.

Model	Parameters	Paper
DistilBERT	66 million	Sanh et al. 2019
BERT-base	110 million	Devlin et al. 2018
RoBERTa	123 million	Liu et al. 2019
DeBERTa	184 million	He et al. 2020
BERT-large	336 million	Devlin et al. 2018

Table 9: Transformer backbones tested for the matching model.

keywords or keyphrases to be excluded from the class. Such information cannot be fully captured by a small number of keywords, potentially leading to suboptimal results. Furthermore, we mostly relied on default values for Lbl2TransformerVec hyperparameters (besides keyword extraction and the pre-trained embedding transformation). It is likely that hyperparameter optimization would improve the metrics reported here.

E.4 Full meta-learning results

This section reports the full results of the meta-learning experiments from Section 3.2.1. Figure 4 shows a summary plot and Table 13 provides the raw numbers.

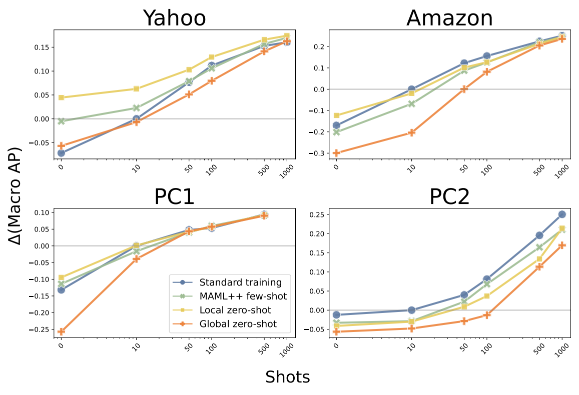

Figure 4: Absolute improvement in macro AP of meta-learning approaches over a 10-shot matching model with a BERT backbone (gray lines), averaged across unseen classes. Note the log scale of the x-axes and the different scales of the y-axes. See Table 13 for details.

F Future work

There are myriad avenues for building upon the work described in this paper. Pre-training the matching model on either an entailment or classification dataset could improve performance significantly. For instance one might consider pre-training on one or more of the datasets in this paper, then fine-tuning on the other. The zero-shot capabilities of the matching model create the possibilities of employing active learning to select few-shot samples or pseudo-labeling to bootstrap additional training samples. It would also be interesting to investigate how the models scale to even more labeled samples. The class descriptions considered in this work were, for the most part, short enough to fit within the maximum token length of the transformer-based models. Additional work is needed to generalize these approaches to even longer class descriptions. Analyzing the attention layers or token salience scores could provide valuable insights into what information matching models are using to make predictions. There are also other ways one might exploit the structure inherent in policy documents besides separating them [SEP] tokens (Li et al., 2020). Finally, there are "free" means of further improving model performance, which we did not explore in this work. These include, but are not limited to, applying domain adaptive and/or task adaptive pre-training (Gururangan et al., 2020), performing additional hyperparameter optimization, or incorporating additional features beyond text (categorical, relevance features from simple retrieval models like BM25 (Robertson et al., 2009), etc.) (Siblini et al., 2020; Chen et al., 2017).

G Example inputs

Here we give examples of the class descriptions and example texts fed into our models, after preprocessing. Note that only the matching model used the class descriptions.

Dataset	Backbone	0 shot	10 shots	50 shots	100 shots	500 shots	1000 shots
Yahoo	DistilBERT	-7.58	0.63	5.85	9.39	14.97	16.42
	BERT-base	-7.17	0.00	7.66	11.16	15.27	16.03
	RoBERTa	-8.97	-1.35	5.37	9.87	16.03	17.10
	DeBERTa	-1.83	3.76	8.30	12.19	16.82	17.89
	BERT-large	-3.43	2.65	9.85	12.36	16.18	16.43
Amazon	DistilBERT	-34.20	-20.75	2.43	7.24	18.03	22.57
	BERT-base	-17.01	0.00	12.24	15.58	22.46	25.11
	RoBERTa	-18.90	-6.65	7.60	10.80	20.68	23.52
	DeBERTa	-5.92	4.59	15.74	19.25	25.15	26.67
	BERT-large	-12.27	7.53	14.55	16.15	23.85	25.53
PC1	DistilBERT	-19.12	-3.24	3.06	4.65	8.30	-
	BERT-base	-13.48	0.00	4.04	5.72	9.18	-
	RoBERTa	-11.99	1.14	4.92	6.02	9.02	-
	DeBERTa	-11.02	1.19	6.06	7.12	9.51	-
	BERT-large	-7.10	2.46	5.23	6.87	9.46	-
PC2	DistilBERT	-3.11	-1.98	1.41	4.87	17.89	23.58
	BERT-base	-1.24	0.00	3.99	8.12	19.55	25.05
	RoBERTa	0.36	3.59	6.64	10.62	22.23	27.40
	DeBERTa	0.85	6.00	9.52	13.32	23.06	27.91
	BERT-large	-0.73	1.82	12.41	16.53	25.71	30.32
Average	DistilBERT	-16.00	-6.33	3.19	6.54	14.80	20.86
	BERT-base	-9.72	0.00	6.98	10.15	16.62	22.06
	RoBERTa	-9.87	-0.82	6.13	9.33	16.99	22.67
	DeBERTa	**-4.48**	**3.88**	9.91	12.97	18.64	**24.16**
	BERT-large	-5.88	3.61	**10.51**	**12.98**	**18.80**	24.09

Table 10: Comparison of backbones for the matching model, measured as absolute improvement in macro AP (%) over 10-shot BERT. Results are averaged across unseen classes.

Dataset	Backbone	Train time (s)	Inference time / sample (ms)
Yahoo	DistilBERT	66,601	2.469
	BERT	133,523	3.912
	RoBERTa	142,860	3.914
	DeBERTa	241,303	6.590
	BERT-large	336,248	10.127
Amazon	DistilBERT	5,521	2.406
	BERT	9,832	3.724
	RoBERTa	10,027	3.796
	DeBERTa	17,089	6.073
	BERT-large	26,314	9.813
PC1	DistilBERT	10,613	2.474
	BERT	18,812	3.757
	RoBERTa	19,113	3.839
	DeBERTa	33,117	6.537
	BERT-large	49,581	9.883
PC2	DistilBERT	14,451	2.497
	BERT	25,742	3.663
	RoBERTa	26,725	3.891
	DeBERTa	45,435	6.405
	BERT-large	68,481	9.851

Table 11: Train and inference times for different matching model backbones on a p3.16xlarge ec2 instance. We measured inference latency by dividing the total inference time (not including time to instantiate the model or preprocess data) by the number of samples.

Dataset	Model	0 shot
Yahoo	Lbl2TransformerVec	23.68
	Matching	46.45
Amazon	Lbl2TransformerVec	-29.14
	Matching	0.91
PC1	Lbl2TransformerVec	-31.29
	Matching	-5.51
PC2	Lbl2TransformerVec	-0.29
	Matching	11.36
Average	Lbl2TransformerVec	-9.26
	Matching	13.30

Table 12: Zero-shot performance (macro AP %) of Lbl2TransformerVec and matching model on four datasets relative to a supervised 10-shot BERT model. We average the scores across unseen classes and three seen/unseen data splits.

Dataset	Model	0 shot	10 shots	50 shots	100 shots	500 shots	1000 shots
Yahoo	Standard training	-7.17	0.00	7.66	11.16	15.27	16.03
	MAML++ few-shot	-0.51	2.27	7.89	10.58	15.69	16.96
	Local zero-shot	4.45	6.28	10.29	12.93	16.58	17.41
	Global zero-shot	-5.67	-0.68	5.11	7.95	14.13	16.25
Amazon	Standard training	-17.01	0.00	12.24	15.58	22.46	25.11
	MAML++ few-shot	-20.15	-6.85	8.73	12.57	22.00	24.45
	Local zero-shot	-12.31	-1.97	10.16	12.66	21.32	24.60
	Global zero-shot	-29.96	-20.39	0.07	8.17	20.52	23.61
PC1	Standard training	-13.19	0.00	4.77	5.34	9.43	-
	MAML++ few-shot	-11.38	-1.60	4.04	5.97	9.12	-
	Local zero-shot	-9.46	0.14	4.24	5.66	9.27	-
	Global zero-shot	-25.72	-3.92	4.36	5.73	9.04	-
PC2	Standard training	-1.24	0.00	3.99	8.12	19.55	25.05
	MAML++ few-shot	-3.31	-2.88	2.21	6.81	16.44	21.07
	Local zero-shot	-4.13	-3.00	0.88	3.69	13.40	21.41
	Global zero-shot	-5.65	-4.80	-2.86	-1.31	11.35	16.96
Average	Standard training	-9.65	0.00	7.16	10.05	16.68	22.06
	MAML++ few-shot	-8.83	-2.26	5.72	8.98	15.81	20.83
	Local zero-shot	-5.36	0.36	6.40	8.74	15.14	21.14
	Global zero-shot	-16.75	-7.45	1.67	5.14	13.76	18.94

Table 13: Average absolute improvement in macro AP (%) of different meta-learning strategies compared to a 10-shot matching model (BERT backbone) with standard training. We average the scores across unseen classes.

G.1 Yahoo

For the Yahoo Answers dataset the class descriptions were of the form “this text has to do with <class label>” where class label is one of Society & Culture, Science & Mathematics, Health, Education & Reference, Computers & Internet, Sports, Business & Finance, Entertainment & Music, Family & Relationships, Politics & Government". Table 14 shows examples of input texts and their labels (selected from the first 40 examples in the test set). Notice that some labels are questionable (see the last two rows).

G.2 Amazon

Tables 15, 16, and 17 show example texts of products on `amazon.com` and their labels of the Amazon e-commerce product classification dataset. The class labels are the assigned product categories. This classification task is challenging since the dataset is a web crawl data, and hence the example texts may contain misleading, irrelevant, or noisy information.

Example text	label
why does n t an optical mouse work on a glass table or even on some surfaces optical mice use an led and a camera to rapidly capture images of the surface beneath the mouse the infomation from the camera is analyzed by a dsp digital signal processor and used to detect imperfections in the underlying surface and determine motion some materials such as glass mirrors or other very shiny uniform surfaces interfere with the ability of the dsp to accurately analyze the surface beneath the mouse since glass is transparent and very uniform the mouse is unable to pick up enough imperfections in the underlying surface to determine motion mirrored surfaces are also a problem since they constantly reflect back the same image causing the dsp not to recognize motion properly when the system is unable to see surface changes associated with movement the mouse will not work properly	Computers & Internet
what is the best off road motorcycle trail long distance trail throughout ca i hear that the mojave road is amazing search for it online	Sports
what is trans fat how to reduce that i heard that tras fat is bad for the body why is that where can we find it in our daily food trans fats occur in manufactured foods during the process of partial hydrogenation when hydrogen gas is bubbled through vegetable oil to increase shelf life and stabilize the original polyunsatured oil the resulting fat is similar to saturated fat which raises bad ldl cholesterol and can lead to clogged arteries and heart disease until very recently food labels were not required to list trans fats and this health risk remained hidden to consumers in early july fda regulations changed and food labels will soon begin identifying trans fat content in processed foods	Health
how many planes fedex has i heard that it is the largest airline in the world according to the www fedex com web site air fleet 670 aircraft including 47 airbus a300 600s 17 boeing dc10 30s 62 airbus a310 200 300s 36 boeing md10 10s 2 atr 72s 5 boeing md10 30s 29 atr 42s 57 boeing md11s 18 boeing 727 100s 10 cessna 208as 94 boeing 727 200s 246 cessna 208bs 30 boeing dc10 10s 17 fokker f 27s	Business & Finance
why do people blush when they are embarrassed why do people blush when they are embarrassed from ask yahoo http ask yahoo com ask 20040113 html blushing is a unique blend of evolutionary and social behavior it s an involuntary reaction of the sympathetic nervous system which is responsible for our fight or flight response but blushing is solely triggered by social cues people generally blush when they re feeling embarrassed scared or stressed as a result of the fight or flight response the capillaries that carry blood to the skin widen and the increased blood flow lends the face as well as sometimes the chest neck or even the body or legs a reddened color excessive facial blushing or erythrophobia is caused by overactivity of the sympathetic nervous system the condition can cause a lot of psychological duress and has engendered several support groups it s common knowledge that animals do n t blush so while there are some evolutionary cues behind blushing it s also linked to something uniquely human moral consciousness	Science & Mathematics
is lin qingxia aka brigitte lin the most beautiful woman in chinese cinema this is according to stephen chow http www hkentreview com 2005 features kfh kfhprem html is it true who is the best looking male star did they make any movies together well everyone has different description on what beauty is i like lin qingxia but i think many girls are prettier than she was she is more than 40 years old now if lin qingxia is the most beautiful woman in the chinese cinema the most handsome man in chinese cinema should be chin han because they always made movies together however a male movie star once was asked his girlfriend in real life or the girlfriend in movie is more beautiful he gave a very good answer i think my mother is the most beautiful woman in the world	Entertainment & Music
what s the best way to create a bootable windos dos cd i do n t use floppies any more and need to boot from something other than my hard disk well the best way is to look at whatever program you have for burning cds and see if it has an option to create a bootable cd if you ca n t find it or use windows itself to burn cds then it s a little more complicated note if you find that booting from cd does n t work you may have to adjust your bios setting to allow your machine to boot from cd if you want to boot to windows the easiest way is probably to go here http www nu2 nu bootcd and download a utility that will do it for you there are instructions there depending on what type of boot you want if you do n t trust using a third party utility microsoft has some instructions here http support microsoft com kb 167685 en us this process is not very straightforward though if you just need to get into your filesystem and poke around you might consider booting a different os for example http www freedos org freedos and http www knoppix net knoppix linux may do what you want for these you can download iso images and burn bootable cds	Computers & Internet
what is the best riddle that you know i m trying to have a library of the best riddles that people encountered so a good riddle would be a riddle that has no ugly tricks in it pure logic answer ,Äì no tricks no funky solutions if you were standing in front of a door one leading to heaven and one leading to hell neither of which are labeled and guarding that door is a guard one who always lies and one who always tells the truth but you do n t know who s who what question would you ask to definitively know which one is the door to heaven versus hell answer what would the other guard say that you would say is the door to heaven hell then you know the opposite door is the door to heaven hell	Business & Finance
why would big ten keep its name inspite of adding a 11th school colleges univs xd br br upto 1980s and early 90s big ten conference had only 10 schools but then somehow it was decided to add psu as the 11th big ten school xd br br any thoughts insight on why the name of the conference not changed i would think to keep the brand recognition and the history of the big ten the conference has invested a lot in building the big ten name as well as any merchandising and corporate sponsorship	Education & Reference

Table 14: Examples from the Yahoo Answers dataset.

Example text	label
speedo essential endurance jammers aw18mix inc polyamide and elastane machine washable endurance fabric protects against fading and retains the garments original shape chlorine resistant ensures your swimwear wont degrade integral support hugs the contours of the body speedo essential endurance plus jammers sporty secure and comfortable the timelessly stylish essential endurance plus jammers are perfect for fitness training constructed using endurance fabric the speedo essential jammers are hardwearing and comfortable the fabric protects against fading allowing your swimwear to look newer for longer it also clings to the body and retains its shape meaning you don't have to worry about baggy shorts after multiple uses a 100 chlorine resistant coating ensures your swimwear won't degrade in the swimming pool adding to the overall durability of the product integral support is designed to hug the contours of the body to create a hydrodynamic profile in the water whilst four way stretch technology allows for a wider range of motion in all directions lastly a drawstring waist provides an enhanced fit and added security	Clothing, Shoes & Jewelry
a guide to the wildflowers of south carolina a most comprehensive state plant field guide and an excellent resource for the natural history of plants in south carolina and surrounding states choice magazine richard dwight porcher is a professor of biology and director of the herbarium at the citadel in charleston south carolina an authority on the flora of south carolina he is the author of wildflowers of the carolina lowcountry and lower pee dee and a co author of lowcountry the natural landscape he was born in berkeley county south carolina and received his b s from the college of charleston and ph d from the university of south carolina porcher trained under dr wade t batson and serves on the south carolina heritage trust advisory board and the scientific advisory board of the south carolina nature conservancy porcher lives in mount pleasant	Books
home 2 pack value round square swirl shaped silicone mold for chocolate jelly and candy 15 piece per mold chocmoldc1301 100 food grade silicone 15 round cavities each safe to use in the oven microwave freezer and dishwasher can be used in temperatures from 76 degrees f to as high as 446 degrees f now with this chocolate mold you can design your very own chocolates for gift boxes or party trays size of each cavity inch 1 2'create luscious and delicious chocolate treats from the comfort of your own home now with this chocolate mold you can design your very own chocolates for gift boxes or party trays this is easy to use flexible non toxic and long lasting safe to use in the oven microwave freezer and dishwasher	Home & Kitchen
storm trooper side white vinyl car laptop window wall decal decal automatically comes in white approximate size as shown 4 2 w x 5 5 h additional sizes available upon request made with premium high quality indoor outdoor vinyl lasts for several years of interior and exterior use without fading cracking or peeling discounts available with larger quantities purchased easy to apply instructions included free tester decal included with every orderthe decal is a single color white and does not have a background decals can be applied to any smooth clean surface car windows home windows interior walls laptop covers cell phone cases boats etc	Automotive
navy 550lb 8 strands cores reflective paracord parachute cord lanyard 50ft 100ftitem 550 lb 8 cores strands reflective paracord breaking strength 550 lb diameter 4 mm length 50ft 100ft inner strand core 8 tightly twisted inner strands coresthe reflective tracers appear to be light gray with normal light but in no or low light conditions the tracers will reflect back any light on them this makes the cord ideal for low light visibility conditions	Sports & Outdoors
audiopipe studio z 15 quot loudspeaker 8 ohm 350w wireless stream w remoteaudiopipe dzc1540ub studio z 15 loudspeaker 8 ohm 350w wireless stream w remoteprofessional abs loudspeakerwoofer 15 voice coil 2 magnet 40 oztweeter 1 titanium diaphragm drivermax power 350 wattsvoltage 110 220vimpedance 8 ohmsensitivity 98 dbfrequency response 20 hz 20khzlcd screen usb sd player wireless music stream fm radio remote controlspeaker output for a passive enclosureaux in for multimedia devicesblue led power indicatorwheels and handle for easy transportation	Electronics
disney goofy goof troop goofybrand new in original packagingbrand new in original packaging	Toys & Games
smartseries screen protector 3pk for the samsung r760 d710 gsii epic touchanti glare and scratch resistant washable and dust free easy to apply and remove made from a durable self adhesive polymer includes a cleaning cloth and an applicator cardmade from a tough self adhesive polymer the screen protector prevents fingerprints dirt dust and scratches from marking up your device the screen protector is both easy to apply and remove	Cell Phones & Accessories

Table 15: Examples from the Amazon dataset, label index 0 to 7.

Example text	label
spy games lethal limits kindle edition	Kindle Store
seekingtag plastic beach towel clips towel holder pack of 6pcs100 brand new made of durable plastic with zinc galvanized steel rust resistant also great for hanging towels to dry won't blow away or fold up includes 6 clips in bright colors 3pcs in blue 3pcs in yellow size about 12 3 5 cmseekingtag plastic beach towel clips towel holder pack of 6pcs	Tools & Home Improvement
geminigemini is a feel good cd with a mixture of r b hip hop jazz gospel for all to enjoy	CDs & Vinyl
tarnish resistant craft wire 20 gauge gold colortarnish resistant craft wire 20 gauge with a gold color on a tarnish resistant spool 15 yards per spool ideal for use as an embellishment wirethis tarnish resistant craft wire is ideal for use as an embellishment wire it is 20 gauge and gold in color there are 15 yards per spool	Arts, Crafts & Sewing
seattle's best blend decaf 12 ounce pack of 2 seattle's best coffee decaf level three seattle's best blend decaf ground coffee decaffeinated ground coffee balanced smooth full flavored character profile two parts great flavor one part relaxation 3 how to enjoy measure one rounded tablespoon 7g of coffee per 6 fl oz 180ml of water add more or less coffee to achieve the perfect cup for you statements regarding dietary supplements have not been evaluated by the fda and are not intended to diagnose treat cure or prevent any disease or health condition	Grocery & Gourmet Food
greenworks 15 inch 5 5 amp corded string trimmer 21272lightweight easy to use design only 7 05 lbs 15 inch cut path with pivoting head allows for edging and trimming capability with edging wheel 065 dual line auto feed with electric start gets you going within seconds no gas hassle ideal for small to medium size yards compatible greenworks replacment spool model 29082green works 5 5 amp corded 15 inch cutting width auto feed 0 065 single line electric string trimmer model no 21272	Patio, Lawn & Garden
2016 monthly mini wall calendar paths to god by tf publishing2016 paths to god mini calendar all calendar pages are printed on fsc certified paper and use environmentally safe inks size 7 x 7 in open size 7 x 14 in package quantity 1let the words of god instill peace throughout the year this paths to god mini wall calendar offers some of the most inspiring and spiritual teachings of the bible as well as images of nature's paths all calendar pages are printed on fsc certified paper and use environmentally safe inks	Office Products
american war generalsfor the first time national geographic gathers the nation s 10 leading war generals for an unprecedented look at the history of the u s army from the vietnam war to america s war on al qaeda american war generals reveals many never before heard stories and opinions from the legendary leaders of the modern u s army their accounts reveal the big changes that have transformed the u s military from the first troops to enter vietnam to the last combat troops to exit afghanistan explaining the critical personal experiences that shaped their lives and the way they approached modern warfare	Movies & TV

Table 16: Examples from the Amazon dataset, label index 8 - 15.

Example text	label
hooded alpaca wool dog cat knit hoodie sweater orange amp ivoryhooded striped alpaca sweater made of warm alpaca yarn perfect for dogs cats handmade by skilled artisans in peru exclusive design by alpaca warehouse made in peruthis is a brand new handmade dog and cat hooded sweater made of soft alpaca yarn it is very comfortable and warm and features a gorgeous color combination small size measures length 14 chest girth 16 medium size measures length 15 chest girth 18 large size measures length 16 chest girth 20 xlarge size measures length 17 chest girth 22	Pet Supplies
wall mount disposable glove dispenser rackunique open design allows you to see what size or type of glove is being dispensed ideal for use in patient rooms doctor's offices labs kitchens morethe open design allows you to see what size or type of glove is being dispensed the holders accommodate most glove boxes and are universal enough to be used for many tissue boxes as well warning this product can expose you to chemicals which is are known to the state of california to cause cancer and birth defects or other reproductive harm	Industrial & Scientific
wedgie wrpr50m 5 0mm medium wedgie rubber pick refill 18 piecesa revolutionary new sound for your acoustic or classical guitar the special elastomer material nearly eliminates pick noise leaving you with clean warm tones this pick sounds like your fingers yet plays like a pick rubbers come in 2 thicknesses and 3 levels of stiffness soft medium hard so that you can find your perfect sound mediumwedgie rubber pick refill 5 0mm med 18pcs	Musical Instruments
200 pcs lot fancy colorful butterfly sticker cover for ps3 controller playstation 3 skins games 1 skinunit type lot 200 pieces lot package weight 0 200kg 0 44lb package size 10cm x 6cm x 10cm 3 94in x 2 36in x 3 94in unit type lot 200 pieces lot package weight 0 200kg 0 44lb package size 10cm x 6cm x 10cm 3 94in x 2 36in x 3 94in	Video Games
2 pack replacement maytag mfd2561hes refrigerator water filter compatible maytag ukf8001 fridge water filter cartridgereplacement maytag mfd2561hes refrigerator water filter quantity 2 replaces ukf8001 fridge water filter cartridge refrigerator water filter retains beneficial nutrients in water while removing the taste of chlorine and odor leaving you with a fresh and clean tasting water reduces waterborne contaminants including cysts asbestos particulates lead mercury without removing beneficial minerals restricting the flow rate of the water cut down water bottle waste and save money while also helping the environment by using water filters for the highest quality water and best contaminant reduction replace the filter every 6 months on sale for a limited time please note this is an denali pure brand replacement part not an oem product this product is not affiliated with any oem brands and is not covered under any warranties offered by the original manufacturers any warranties for this product are offered solely by denali pure all mentions of brand names or model descriptions are made strictly to illustrate compatibility all brand names and logos are registered trademarks of their respective owners	Appliances
professional business card workshop 2 0create your own templates or choose a layout from 20 000 professional quality templates premium collection of clipart images photos and heart warming sentiments to finish each project new tools for creating animations and web pages for your projects select from over 250 000 premium images and 150 creative fonts including editable clipart stunning photography and more design your own photo calendars flip books collages slide shows and photo bouquets	Software
2015 topps baseball card 83 david wright nm mt2015 topps 83 david wright new york mets baseball cards one single 2015 topps series 1 trading card single card ships in top load and or soft sleeve cards combined card condidtion is near mint mint nm mt mint note stock image used2015 topps 83 david wright new york mets baseball cards	Collectibles & Fine Art

Table 17: Examples from the Amazon dataset, label index 16 - 22