Using Interventions to Improve Out-of-Distribution Generalization of Text-Matching Systems

Parikshit Bansal
Microsoft Research
parikshitb52@gmail.com

Yashoteja Prabhu
Microsoft Research
yprabhu@microsoft.com

Emre Kiciman
Microsoft Research
emrek@microsoft.com

Amit Sharma
Microsoft Research
amshar@microsoft.com

Abstract

Given a user's input text, *text-matching* recommender systems output relevant items by comparing the input text to available items' description, such as product-to-product recommendation on e-commerce platforms. As users' interests and item inventory are expected to change, it is important for a text-matching system to generalize to data shifts, a task known as out-of-distribution (OOD) generalization. However, we find that the popular approach of fine-tuning a large, base language model on paired item relevance data (e.g., user clicks) can be counter-productive for OOD generalization. For a product recommendation task, fine-tuning obtains *worse* accuracy than the base model when recommending items in a new category or for a future time period. To explain this generalization failure, we consider an *intervention-based* importance metric, which shows that a fine-tuned model captures spurious correlations and fails to learn the causal features that determine the relevance between any two text inputs. Moreover, standard methods for causal regularization do not apply in this setting, because unlike in images, there exist no universally spurious features in a text-matching task (the same token may be spurious or causal depending on the text it is being matched to). For OOD generalization on text inputs, therefore, we highlight a different goal: avoiding high importance scores for certain features. We do so using an intervention-based regularizer that constraints the *causal effect* of any token on the model's relevance score to be similar to the base model. Results on Amazon product and 3 question recommendation datasets show that our proposed regularizer improves generalization for both in-distribution and OOD evaluation, especially in difficult scenarios when the base model is not accurate. [1]

1 Introduction

In item-to-item recommendation [1], the goal is to output a list of relevant items given an input item. Many such recommender systems utilize the text content of an item, such as recommending relevant questions given a question in an online forum [23], suggesting relevant products given a product title [14], or predicting relevant ads given a search engine query [4]. A popular way for training these *text-matching* recommender systems is to fine-tune text embeddings from a *base* language model like BERT using supervised user feedback (such as clicks). For example, one may use a contrastive loss to ensure that given an input query, embedding of another user-labelled relevant item is closer [6, 28].

[1] arxiv version : https://arxiv.org/pdf/2210.10636.pdf

NeurIPS 2022Workshop on Causal Machine Learning for Real-World Impact (CML4Impact 2022).

However, the generalizability of the fine-tuned embeddings to item distributions beyond the user-labelled data has received little attention. Distribution shifts are common in recommendation systems, such as when new product categories are added to an e-commerce platform, the list of recommendable items is modified, or the popularity of items changes over time. As a result, after a recommender model is deployed, it is likely to encounter out-of-distribution data compared to its training set. For such out-of-distribution data, we find that fine-tuned models using relevance labels can be worse than the pre-trained base model that they start from. On a product recommendations dataset from Amazon, while fine-tuning always increases in-distribution accuracy, the accuracy of the fine-tuned model on unseen product categories is lower than that of the base model. We find a similar result on question-to-question recommendation on online forums (*sentence matching* task [17, 22]).

To understand why, we characterize two common shifts in recommender systems: change in input query's distribution and change in candidate items' distribution. Even without any change in the relevance function, these distribution shifts lead to a change in association between relevance score and the input tokens of a model. As a result, even though standard fine-tuned models obtain high validation accuracy, they overfit to the training distribution by capturing spurious associations (e.g., between brand and a product category). We characterize model spuriousness through an intervention-based importance score for any tokens subset, which can be interpreted as the *causal effect* of the token subset on the model's relevance function. Using the score, we find that the fine-tuned embedding on Amazon products learns a spuriously high importance for certain tokens while forgetting the rest.

To improve generalizability, we consider a causal graph for the relevance function's computation and highlight the difficulty of not having universally spurious tokens for the text-matching task. Instead, we argue that avoiding disproportionately high importance scores for tokens can be a viable way to regularize for OOD generalization. Specifically, we propose `ITVReg`, a method that constrains the importance of tokens to be similar to that in the base model, which has been trained on a larger and more diverse set and thus less likely to share the same spurious patterns. To do so, the method utilizes *interventions* on the training data (e.g., by masking certain tokens) to create OOD queries.

We evaluate `ITVReg` on OOD data for product title and question recommendation tasks and find that it improves OOD accuracy of naively finetuned models. Compared to a baseline of directly regularizing a model's predictions to the base model, `ITVReg` performs the best under reasonable OOD shifts, where exploiting training data is useful. Interestingly, `ITVReg` also helps accuracy on an IID test set: it increases accuracy over low-frequency, "tail" items. Our contributions include, **1)** We characterize how new items break correlation between tokens of the query and candidate items; as a result state-of-the-art models for text-matching can be worse than the base model on which they are fine-tuned. **2)** We propose a regularizer based on an interventional metric of token importance that improves OOD generalization.

2 Related Work

Our work connects recommender systems, sentence matching, and OOD generalization literature.

Text-matching recommendation systems and Sentence matching. Among item-to-item recommendation systems, text-matching is a popular technique since many items can be characterized through their text, e.g., product-to-product recommendation on e-commerce websites or question-to-question recommendation on online forums. Given an input query (such as a product's title or description), the goal is to find the most relevant items. State-of-the-art models use dense text retrieval techniques [28, 11], based on a pre-trained *base model* like BERT [7] for the initial encoding. Then, either a bi-encoder or cross-encoder architecture [22, 11] is trained for learning the similarity between any two items. The former is computationally efficient while the latter is more accurate but inefficient [17]. Since recommender systems often deal with a large number of items, we restrict ourselves to bi-encoders. Contrastive loss with negative mining [31, 6, 10] is a popular way to train a bi-encoder model because recommender systems usually have one-sided feedback on the relevant pairs of items. Closer to our work, [22] tackled the problem of domain adaptation while our work focuses on the harder domain generalization problem [32].

OOD generalization and fine-tuning. Domain generalization in the vision literature [32] aims to identify spurious features in image data and remove them from a model's representation using data augmentation on the spurious feature [29] or through regularization [5]. While recent work has attempted using data augmentations from vision [27], such augmentations are not always useful in text data [30] since it is difficult to obtain universal augmentations. For instance, in the recommendation scenario, certain tokens (e.g., brand of a product) may be both spurious and causal depending on the user intent (e.g., substitute or accessory for a product). Given the prevalance of pre-trained base models, another direction is to utilize the base models for OOD generalization since they are trained on larger, diverse data [9]. Recent work proposes fine-tuning-based OOD generalization on image classification [12, 25, 5], but this direction has not been explored for text models or contrastive loss, especially in the recommendations context.

3 Distribution shifts in recommendation

Let $\mathcal{X}$ be the set of input queries and $\mathcal{Z}$ be the set of candidate items. Let the train data be $(X_i, Z_i, R_i)_{1...N}$, where, X_i, Z_i correspond to the query and item text respectively and R_i is the relevance or similarity score between X_i, Z_i. We define $f_\theta : \mathcal{X} \to \mathbb{R}^N$ as our encoder, parameterised by θ. We use θ_0 as parameters for `Base` model. We assume that f outputs unit norm embeddings. For training, state-of-the-art methods [6, 28] embed both queries and items in a common space using a large, pre-trained base encoder like BERT that provides the initial embeddings. The embeddings are fine-tuned using a supervised loss on the paired relevance labels given below $\mathcal{L} = \max(0, 1 + f_\theta(X)^\top f_\theta(Z^-) - f_\theta(X)^\top f_\theta(Z^+))$ where X, Z^+ and Z^- denote a query, relevant item, and a non-relevant item respectively. When ratings or relevance scores are available, one may also use the MSE loss [17]. For a new input query, the model outputs a vector $\hat{R} \in \mathbb{R}^M$ where M is number of items, and $r_j = f_\theta(X)^\top f_\theta(Z_j)$ is its *relevance score* with jth item. The system is evaluated on its accuracy in predicting the "ground-truth" relevance labels (e.g., metric may be precision at rank k).

3.1 Types of distribution shift

We study three shifts: change in queries $P(X)$, items $P(Z)$, and both queries and items $P(X, Z)$. **Change in P(X).** The distribution of queries changes but the set of candidate recommendations may remain the same. For example, the popularity of certain queries can shoot up due to external events, or the system may be expanded to new queries (e.g., products on another website). **Change in P(Z).** The distribution of candidate items changes while the queries remain the same. For example, an e-commerce platform may alter the eligibility rules for an item to be recommended, or recommend items from a partner website. **Change in both P(X) and P(Z).** A final scenario is when the distribution of both queries and items changes. For example, introducing a new item category in an online store leads to new queries that can be accessed by user and also be recommended.

Method	IID	OOD
Base Model	20.01	**30.61**
Fine-tuned Model	38.74	28.31.

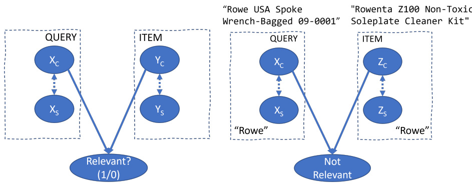
Table 1: Precision@1 for recommending product titles on Amazon. OOD denotes queries from five categories that were not included in the train data. Fine-tuning on training data results in lower precision than the base model.

While the criteria for relevance (i.e., user intent) can change over time, for simplicity, we assume that $P(R|Z, X)$ remains constant where R is the true relevance score. As we show below, even under this favorable assumption, text-matching models can learn patterns that do not generalize.

Example: OOD data on Amazon products. Let us use the *Amazon Titles* dataset (see Supp. D) to illustrate the problem with OOD generalization for fine-tuned models. Both the input query and candidate items are titles of products on Amazon.com. We simulate a scenario where $P(X)$ is changed by removing queries from five categories from the train data and evaluating the model on those removed queries using precision@1 metric.Table 1 compares the precision@1 of the base model (MSMarco [20]) and fine-tuned model that is initialized with the base model. As expected, the fine-tuned model improves significantly over the base model on IID test data, almost doubling the precision. However, on the queries from the unseen item categories, the finetuned model performs *worse* than the base model.

3.2 Explaining the OOD generalization failure: An intervention-based importance score

Explanation through causal graph. To understand the failure, consider the schematic causal diagram for the data-generating process of the training set. Figure 1 shows that each query can be broken down into its semantic (causal) component X_c and its spurious component X_s. Same for candidate items, denoted by Z_c and Z_s. The semantic components X_c and Z_c together cause the relevance label. An ideal encoder should only learn the semantic components. However, since the



(a) General graph. (b) Example graph.

Figure 1: Causal graph for the data-generating process for query, item and relevance score, (X, Z, R). Spurious tokens X_s or Z_s do not cause query-item relevance but are predictive of the relevance score.

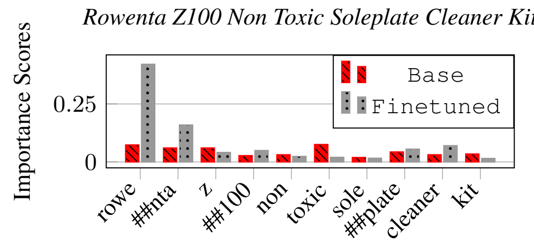


Figure 2: Token-wise importance scores for the *Rowenta ...* example query using the `Base` and `Finetuned` models. `Base` gives approximately equal importance scores to all tokens whereas `Finetuned` model gives dispropotionately high weights to the token *"rowe"*

non-semantic components are correlated with the semantic component and can be easier to learn, a fine-tuned encoder may learn the non-semantic components too.

Specifically, when new queries are introduced, the correlation between X_c and X_s can be broken, even as $P(R|X, Z)$ remains the same. Hence, X_s is no longer a good predictor of relevance and any model relying on X_s will fail to generalize. For example, for certain queries in the Amazon data, X_s may correspond to the brand of a query product and X_c the rest of its title . In the training data, the brand (e.g., "samsung") may be correlated with a single product category (e.g., "smartphone") and thus an encoder may learn a non-zero weight for the brand to determine the relevance score. However, the correlation may be broken in the evaluation data where the same brand is associated with another product category (e.g., "refrigerator"). We can reason analogously for a change in $P(Z)$.

Explanation through intervention-based importance score. OOD generalization failure is more intuitively understood through intervention-based analysis [18], where we change parts of a query and inspect the model's prediction. To understand which tokens a model learns as important, we define an *importance score* s of each token, that corresponds to the causal effect of a token on the model's relevance score. To compute s for a token, we mask that token (intervention) and compute the dot product similarity (relevance) of the masked input's embedding with the original input's embedding. [2] $s_j = 1 - f_\theta(X)^\intercal f_\theta(X'_{-j})$ where X'_{-j} is X with its jth token masked.

Consider a candidate item in the Amazon dataset, "*Rowenta Z100 Non Toxic Soleplate Cleaner Kit*". Figure 2 shows the importance scores. The base encoder gives almost equal importance to each token, whereas the finetuned encoder is biased towards the token *rowe*, possibly because Rowenta products tend to be relevant to other Rowenta products. This token obtains such a disproportionely high importance score that the item's representation (average of all tokens' representation) is defined by it. Here *rowe* $\in Z_s$ is a spurious token since relevance may depend on the brand *rowenta*, but not on *rowe* alone. When we consider an OOD query having an actual brand called "Rowe", "*Rowe USA Spoke Wrench - Bagged 09-0001*", the model matches it to the *Rowenta* item and other products by Rowenta (see Suppl. Table 9), exposing the spurious correlation learnt by the encoder. The query, "*Soleus Air MS-09 Oscillating Reflective Heater*" is another example. The importance scores are in Suppl. Figure 4. Again, we see the finetuned encoder is biased towards the token *reflective* because *reflective* is a strong matching signal in the train set: products with this token are often marked as relevant to products that have it too. As a result, the top predicted item by the finetuned model is *3M Scotchlite Reflective Tape, Silver, 1-Inch by 36-Inch* (not a relevant item, see Suppl. Table 10).

4 Regularization for OOD generalization

The above analysis indicates that fine-tuning encourages a model to learn high importance for certain tokens while forgetting the rest, which is undesirable if the tokens are spuriously associated with the relevance label. However, it is non-trivial to identify the spurious versus semantic tokens.

Building a Risk Invariance Predictor. As we mentioned in Section 3, in text-matching systems, we expect the distribution of queries $P(X)$, distribution of labels $P(Z)$, or both to change from train to test data. At the same time, we assumed that $P(R|X, Z)$, i.e., the relevance between a query and label remains invariant across distributions. That is, the relevance function $g(.|X, Z)$ (for bi-encoders it can be written as $f(X)^\top f(Z)$, see Sec 3) remains invariant. Then the goal is that the relevance

Fine-tuning Method	Temporal Shift		Categorical Shift	
	Amazon131K (IID)	**Amazon1.3M (OOD)**	**AmazonCatRemoved (IID)**	**AmazonCatOOD (OOD)**
`Base`	22.50	25.71	20.01	30.61
`Finetuned`	**39.71 ± 0.14**	26.02 ± 0.08	**38.74 ± 0.08**	28.31 ± 0.17
`MaskReg`	39.56 ± 0.01	26.65 ± 0.02	37.92 ± 0.11	29.09 ± 0.06
`SimCSE`	39.47 ± 0.11	26.05 ± 0.02	38.05 ± 0.56	28.52 ± 0.10
`OutReg`	38.03 ± 0.53	**27.60 ± 0.03**	37.66 ± 0.11	**31.21 ± 0.03**
`ITVReg`	**39.72 ± 0.10**	27.08 ± 0.01	**38.77 ± 0.02**	29.53 ± 0.04

Table 2: P@1 for Temporal and Categorical Shifts on *Amazon Titles*.

function has similar accuracy on train as well as test data. We now formally express this intuition of a OOD generalizable relevance function, using the definition of risk invariance from [16]. Let $\mathcal{P}$ be a set of distributions over R, X, Z such that $P(R|X, Z)$ remains invariant but the marginal distributions $P(X, Z)$ can vary across the distributions. Then, an optimal predictor is,

Definition 4.1. Optimal Risk Invariant Predictor Define the risk of predictor g on distribution $P_t \in \mathcal{P}$ as $R_{P_t}(g) = \mathbb{E}_{x,z,r\sim P_t} l(g(x, z), r)$ where l is the loss function. Then, the set of risk-invariant predictors obtain the same risk across all distributions $P_t \in \mathcal{P}$, and set of the optimal risk-invariant predictors is defined as the risk-invariant predictors that obtain minimum risk on all distributions.

From the causal graph of Figure 1, we assumed that each query can be broken down into two mutually exclusive subsets of tokens, causal X_c and spurious X_s where $X_c \bigcup X_s = X$, such that only X_c affects the relevance with a label. Similarly, for label Z we can get causal Z_c and spurious Z_s, such that only Z_c affects the relevance with a query. Thus, an intuitive way to achieve risk invariance is to build a predictor g for relevance that only depends on X_c and Z_c for any query and label, $P(R|X_c, Z_c)$. From OOD generalization literature on images [32, 24], we may use one of these methods to learn X_c, Z_c from observed data: regularization [15, 13], weighting [19, 29], or data augmentation [32]. But such a solution relies on assumptions that the spurious features (e.g., image background) **1)** are universal for all inputs; **2)** can change independently of the semantic features.

Problem: Spurious features depends on context. In text-based recommendations, it is difficult to find (a set of) tokens that are universally spurious, since *spuriousness* depends on context. For example, in our example with "samsung" products ("smartphone" and "refrigerator"), the brand was a spurious feature for learning the relevance score. However, when looking for a smartphone accessory, the brand is no longer spurious (in fact, it is the semantic feature). Thus, the subset X_c for a query may change based on the label, and similarly, the subset Z_c for a label may change based on different queries. The second assumption that tokens can be changed independently is also less plausible. Possibly spurious tokens like "samsung" can be associated with other parts of the query (e.g, product code or name); it is infeasible to remove them independently. Thus, while there are distribution shifts in text-based queries and items, it is difficult to find universal spurious features that can be modified to improve accuracy (as also observed empirically in prior work [8]). Therefore, fully removing the influence of any token can be counter-productive. Instead, we propose that our goal should be to ensure that no token is under-weighted such that it ceases to be important (ignored) for relevance prediction. Since finetuning on text-matching models tends to assign very high weights to certain tokens, we reframe our goal: to avoid learning large importance scores for some tokens that may lead to ignoring other tokens, and thus not generalize to new data. To do so, we regularize the importance scores to be similar to those from the base model, since the base model has been trained on larger, diverse data and is thus unlikely to encode the same spurious patterns.

We present two methods in this direction. The first is a brute-force attempt, that forces not just the importance scores but the entire prediction of a model to be the same as base model, subject to the strength of a regularization constant (see Supp. A for details). The second is described below.

Intervention-based regularization. Generalizing the importance score from Section 3.2, given a text input, we define an *intervention* as an independent change to a *subset* of tokens without affecting the rest of the input. Further, we consider any intervention that removes information about the token subset, such as masking, deleting, replacing the token subset, etc. By definition, an intervention deviates from the original distribution $P(X)$ that generated the data, thus creating an out-of-distribution sample. For each input X, we construct an interventional input X' using a transformation on a random subset X_{sub} of the tokens. The key idea behind our regularizer is that for any subset of tokens X_{sub}, its importance score using the finetuned model should be the same as the importance score using the base model. This is a relaxation of the *risk-invariance* goal of ensuring that the accuracy of the finetuned model remains the same for X and X'. Using the equation for importance score from Section 3.2, the regularizer can be written as,

$$[(1 - f_\theta(X)^\intercal f_\theta(X')) - (1 - f_{\theta_0}(X)^\intercal f_{\theta_0}(X'))]^2 = (f_\theta(X)^\intercal f_\theta(X') - f_{\theta_0}(X)^\intercal f_{\theta_0}(X'))^2 \quad (1)$$

We call this the *Interventional* regularizer (`ITVReg`). The full training loss (where L_{ERM} can be the contrastive loss from Sec 3) is, $L_{ERM} + \lambda \mathbb{E}_X[(f_\theta(X)^\intercal f_\theta(X') - f_{\theta_0}(X)^\intercal f_{\theta_0}(X'))^2]$

5 Evaluation

We study OOD generalization of text-matching models on product-to-product and question-to-question recommendation. Initial experiments on `ITVReg` indicated that masking as the intervention worked the best, so we use masking for all results. For hyperparameter tuning, see Supp. G. We use two product-to-product recommendation datasets, namely *AmazonTitles131K* collected in 2013 and *AmazonTitles1.3M* [3] collected in 2014. We also simulate a categorical shift setting in *AmazonTitles131K*, with OOD evaluation set termed *AmazonCatOOD* and the in-distribution evaluation set termed *AmazonCatRemoved*. *Precision@1* is used for evaluation (details in Supp. D)

Baselines. We compare `OutReg` and `ITVReg` to the `Base` model, standard `Finetuned` model, and two baselines from past work. **1)** `SimCSE`: We adapt SimCSE [8], an augmentation method for pretraining sentence matching models, for our fine-tuning task. During training, we use dropout (seeds s, s') to pass an input sentence twice through the model and obtain two embeddings, considered as a "relevant" pair. The regulariser is $(f(X,s)^\intercal f(X,s') - 1)^2$. **2)** `MaskReg`: [26] propose deleting a span of words in the input as data augmentation. To adapt it for comparison to `ITVReg`, we mask a portion of tokens instead. The regulariser is $(f(X)^\intercal f(X') - 1)^2$, where X' is masked input.

Results. Table 2 shows the IID and OOD P@1 metrics for different recommender models. We first analyze the temporal distribution shift: train on *Amazon131K* and test on *Amazon1.3M*. On IID evaluation, `ITVReg` and `Finetuned` both obtain the highest P@1 while `OutReg` obtains the lowest P@1. On OOD evaluation, however, `OutReg` obtains the highest P@1 followed by `ITVReg`.

To understand the tradeoff between `ITVReg` and `OutReg`, we study how the accuracy of these models would vary as different amounts of OOD data (from 1.3M dataset) is mixed with IID data (131K validation) for evaluation. This simulates the real-world scenario where new items are progressively added. Specifically, we progressively add 40K new items and their relevant queries from 1.3M to create multiple OOD datasets. As Suppl. Figure 5 shows, `ITVReg` is the only method that performs better than `Finetuned` (line $y = 0$) on P@1 on all OOD datasets. As the distribution shift becomes more extreme, (around 24% of 1.3M OOD data) `OutReg` surpasses `ITVReg` indicating that `OutReg` is more suitable for large distribution shifts. Next, we consider categorical distribution shift from Table 2. We obtain similar results as the temporal shift: on the OOD evaluation, `OutReg` followed by `ITVReg` are the best performing models. For P@1 on each category, see Suppl. Table 8. While we looked at adding new queries and items, in practice, OOD shifts are more commonly observed as reweighing of existing items. Hence, in addition to the average IID accuracy, it is important to analyze accuracy of a model at different levels of item popularity. We bin the items according to their frequency in five quantiles i.e. 0-20%, ..., 80-100%, where lower quantiles denote low-frequency *tail* and higher quantiles denote *head* items. Comparing the percentage gain in P@1 of methods wrt the `Finetuned` model (Figure 3), we find that `OutReg` helps on the tail items but suffers a huge drop in P@1 on head items. In comparison, `ITVReg` achieves best of both worlds, with the same gains on tail and significantly lower drop on head items. Qualitative results for all methods for the two queries from Section 3.2 are in Suppl. K.

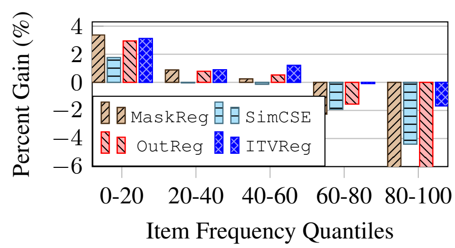


Figure 3: Percent gains in P@1 over `Finetuned` for different item frequency quantiles. `ITVReg` gains on tail (left) while not losing precision on head (right).

Importance Scores. We also evaluate importance scores for the two qualitative examples presented in Sec. 3.1 after `ITVReg` regularisation in Suppl. Fig 4. The importance scores for the problematic tokens *rowe* and *reflective* are now in a reasonable range of value guided by the `Base` model.

Question Recommendations We also use the setup of [21] for numbers on question recommendations task. More details are in Supp. E

6 Conclusion

We identified limitations of text-matching recommender systems on OOD generalization, showing that finetuned models often perform worse than the base model on which they were finetuned. We proposed two regularizers using a causal graph and an interventional measure of token importance.

References

[1] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE transactions on knowledge and data engineering*, 17(6):734–749, 2005.

[2] Jasmijn Bastings and Katja Filippova. The elephant in the interpretability room: Why use attention as explanation when we have saliency methods? *arXiv preprint arXiv:2010.05607*, 2020.

[3] K. Bhatia, K. Dahiya, H. Jain, P. Kar, A. Mittal, Y. Prabhu, and M. Varma. The extreme classification repository: Multi-label datasets and code, 2016.

[4] Andrei Z Broder, Peter Ciccolo, Marcus Fontoura, Evgeniy Gabrilovich, Vanja Josifovski, and Lance Riedel. Search advertising using web relevance feedback. In *Proceedings of the 17th ACM conference on information and knowledge management*, pages 1013–1022, 2008.

[5] Junbum Cha, Kyungjae Lee, Sungrae Park, and Sanghyuk Chun. Domain generalization by mutual-information regularization with pre-trained models. *arXiv preprint arXiv:2203.10789*, 2022.

[6] Kunal Dahiya, Ananye Agarwal, Deepak Saini, K Gururaj, Jian Jiao, Amit Singh, Sumeet Agarwal, Purushottam Kar, and Manik Varma. Siamesexml: Siamese networks meet extreme classifiers with 100m labels. In *International Conference on Machine Learning*, pages 2330–2340. PMLR, 2021.

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

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

[9] Dan Hendrycks, Xiaoyuan Liu, Eric Wallace, Adam Dziedzic, Rishabh Krishnan, and Dawn Song. Pretrained transformers improve out-of-distribution robustness. *arXiv preprint arXiv:2004.06100*, 2020.

[10] Trong Nghia Hoang, Anoop Deoras, Tong Zhao, Jin Li, and George Karypis. Learning personalized item-to-item recommendation metric via implicit feedback. *arXiv preprint arXiv:2203.12598*, 2022.

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

[12] Ananya Kumar, Aditi Raghunathan, Robbie Jones, Tengyu Ma, and Percy Liang. Fine-tuning can distort pretrained features and underperform out-of-distribution. *arXiv preprint arXiv:2202.10054*, 2022.

[13] Yoonho Lee, Huaxiu Yao, and Chelsea Finn. Diversify and disambiguate: Learning from underspecified data. *arXiv preprint arXiv:2202.03418*, 2022.

[14] Greg Linden, Brent Smith, and Jeremy York. Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing*, 7(1):76–80, 2003.

[15] Divyat Mahajan, Shruti Tople, and Amit Sharma. Domain generalization using causal matching. In *International Conference on Machine Learning*, pages 7313–7324. PMLR, 2021.

[16] Maggie Makar, Ben Packer, Dan Moldovan, Davis Blalock, Yoni Halpern, and Alexander D'Amour. Causally motivated shortcut removal using auxiliary labels. In *International Conference on Artificial Intelligence and Statistics*, pages 739–766. PMLR, 2022.

[17] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.

[18] Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. Beyond accuracy: Behavioral testing of nlp models with checklist. *arXiv preprint arXiv:2005.04118*, 2020.

[19] Shiori Sagawa, Pang Wei Koh, Tatsunori B Hashimoto, and Percy Liang. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. *arXiv preprint arXiv:1911.08731*, 2019.

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

[21] Darsh J Shah, Tao Lei, Alessandro Moschitti, Salvatore Romeo, and Preslav Nakov. Adversarial domain adaptation for duplicate question detection. *arXiv preprint arXiv:1809.02255*, 2018.

[22] Nandan Thakur, Nils Reimers, Johannes Daxenberger, and Iryna Gurevych. Augmented sbert: Data augmentation method for improving bi-encoders for pairwise sentence scoring tasks. *arXiv preprint arXiv:2010.08240*, 2020.

[23] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

[24] Jindong Wang, Cuiling Lan, Chang Liu, Yidong Ouyang, Tao Qin, Wang Lu, Yiqiang Chen, Wenjun Zeng, and Philip Yu. Generalizing to unseen domains: A survey on domain generalization. *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[25] Mitchell Wortsman, Gabriel Ilharco, Mike Li, Jong Wook Kim, Hannaneh Hajishirzi, Ali Farhadi, Hongseok Namkoong, and Ludwig Schmidt. Robust fine-tuning of zero-shot models. *arXiv preprint arXiv:2109.01903*, 2021.

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

[27] Xu Xie, Fei Sun, Zhaoyang Liu, Shiwen Wu, Jinyang Gao, Bolin Ding, and Bin Cui. Contrastive learning for sequential recommendation. *arXiv preprint arXiv:2010.14395*, 2020.

[28] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N Bennett, Junaid Ahmed, and Arnold Overwijk. Approximate nearest neighbor negative contrastive learning for dense text retrieval. In *International Conference on Learning Representations*, 2020.

[29] Huaxiu Yao, Yu Wang, Sai Li, Linjun Zhang, Weixin Liang, James Zou, and Chelsea Finn. Improving out-of-distribution robustness via selective augmentation. *arXiv preprint arXiv:2201.00299*, 2022.

[30] Junliang Yu, Hongzhi Yin, Xin Xia, Tong Chen, Lizhen Cui, and Quoc Viet Hung Nguyen. Are graph augmentations necessary? simple graph contrastive learning for recommendation, 2021.

[31] Junliang Yu, Hongzhi Yin, Xin Xia, Tong Chen, Jundong Li, and Zi Huang. Self-supervised learning for recommender systems: A survey. *arXiv preprint arXiv:2203.15876*, 2022.

[32] Kaiyang Zhou, Ziwei Liu, Yu Qiao, Tao Xiang, and Chen Change Loy. Domain generalization in vision: A survey. *arXiv preprint arXiv:2103.02503*, 2021.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section **??**.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [No]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Output-based regularization

Since both the fine-tuned model and the base model have the same architecture, a straightforward way to avoid very high importance scores is to make the neural network output be the same as the base model. In a bi-encoder architecture for text-matching, we can implement this by enforcing that the finetuned and base encoders output exactly the same representation for an input. Given the finetuning $f_\theta(.)$ and base $f_{\theta_0}(.)$ encoders and an input query X, the output regularizer (`OutReg`) can be written as, $[f_\theta(X) - f_{\theta_0}(X)]^2$.

This simple regularizer is expected to do well for extreme distribution shifts where learning from IID data may not be that useful, since it constrains the encoder's representation to be closer to the base encoder. However, the same property is the biggest weakness of `OutReg`. As a regularization, it is too strong and discourages learning from train data: with a high enough penalty term, the final encoder may be exactly equal to the base encoder.

B Qualitative Analysis

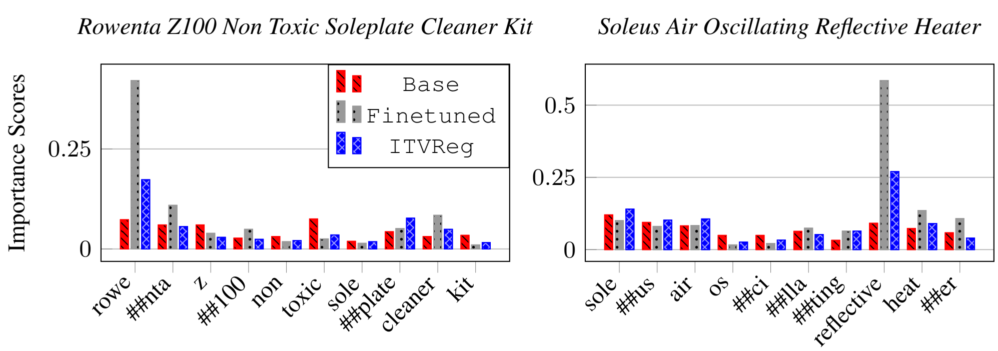


Figure 4: Token-wise importance scores for two example queries using the `Base`, `Finetuned` and `ITVReg`. `ITVReg` is able to normalise the importance scores for *"rowe"* (as in Fig. 2) and *"reflective"* tokens to be closer to `Base` model.

C Temporal Evolution

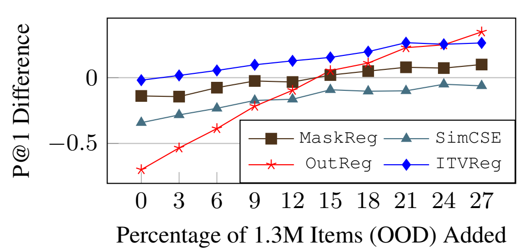


Figure 5: To simulate temporal evolution, we start with *Amazon131K* and add 39K new items from *Amazon1.3M* dataset at each tick. `ITVReg` is the only method that is consistently better than `Finetuned` on P@1.

D Amazon Titles Recommendations

Dataset. We use two product-to-product recommendation datasets, namely *AmazonTitles131K* and *AmazonTitles1.3M* [3]. These datasets contain titles of products that were recommended on a focal product's page. Only the product title can be used for recommendation. There is a time shift though: 131K is collected from the year 2013 and 1.3M is collected from 2014. As both new queries and labels are added while moving from 131K to 1.3M, this simulates the $P(X, Y)$ distribution shift setup. For each dataset, the same candidate items (131K and 1.3M items respectively) are shared across train and validation datasets. Each query in the train set has a labelled set of positive items with relevance 1. The validation set contains unseen queries and we need to recommend the most relevant items for them.

While going from 131K to 1.3M dataset represents a temporal shift where both queries and items' distributions change (and possibly other factors), we simulate a more controlled setup using category information.We exclude queries from the 5 most popular second-level categories from the train set, and use them to construct the OOD evaluation set (*AmazonCatOOD*). The in-distribution validation set also has the categories excluded (*AmazonCatRemoved*). Thus, the item distribution $P(Z)$ remains the same while the query distribution $P(X)$ changes across train and evaluation. Since X, Z have been treated symmetrically in our setup, this setups results can also be extrapolated to a case where the label distribution $P(Z)$ changes while the query distribution $P(X)$ remains the same.

Fine-tuning. We follow the modeling procedure used in a recent state-of-the-art work [6]. We initialize with MSMARCO-DistilBERT-v4 [20] and use contrastive loss with hard negative mining. Details are in Supp H.1.

All experiments are run for 3 random seeds and the mean and standard deviation are reported.

Evaluation metric. We use the $Precision@1$ metric for evaluation. Given predicted similarities $\hat{R} \in \mathbb{R}^M$ and ground truth similarities R (where r_i is 1 if its a relevant item, otherwise 0), $P@k$ is defined as $P@k = \frac{1}{k} \sum_{l \in rank_k(\hat{R})} R_l$.

E Question Recommendation Evaluation

Datasets. We use the setup of [21] for our experiments. The *AskUbuntu* and *SuperUser* data comes from Stack Exchange, a family of technical community support forums. Both datasets contain around one million pairs of sentences, labeled either zero or one denoting a negative and a positive pair. Simulating a cold-start scenario with limited items, we present main results on 10% subsample of these datasets. Results on the complete datasets are in Suppl L. To study generalization under an extreme distribution shift, we additionally evaluate on a non-technical forum, *Quora-Question Pairs* dataset [23].

Fine-tuning. We adopt the same modeling procedure as [22], training with a MSE loss between the ground truth and the predicted relevance score. We initialize the models using two different `Base` models: NLI-DistilBERT [20] MSMARCO-DistilBERT-v4 [20] the latter having a substantially higher accuracy on our task. For more details on training, refer Supp H.2.

Evaluation metric. We use AUC(0.05) as the evaluation metric as in [21]. AUC(0.05) is the area under the ROC curve when the false positive rate (fpr) ranges from 0 to 0.05.

Results. Table 3 shows the IID and OOD performance of models trained on the two forum datasets. Let us first consider AUC on evaluation within the technical forums and using the less accurate NLI-DistilBERT `Base` model. On both IID and OOD evaluation, `ITVReg` obtains the highest AUC for both datasets. On OOD evaluation, `MaskReg` and `ITVAug` have the second-best AUC for SuperUser and AskUbuntu evaluation sets respectively. `OutReg` has comparatively lower AUC because the base model is not accurate: `ITVReg` obtains $>$ 10 points higher AUC on both OOD datasets.

Using the more accurate MSMARCO-DistilBERT-v4, accuracy of both `OutReg` and `ITVReg` increases. P@1 on OOD data of `OutReg` is comparable to `ITVReg` on *AskUbuntu* evaluation and 1 point higher than `ITVReg` on *SuperUser* evaluation. In comparison, `MaskReg` and `SimCSE` fail to utilise the better `Base` model. When evaluated on an extreme distribution shift (*Quora*), `Base` model performs the best on OOD evaluation (Table 3) , followed by `OutReg`. Finetuning on *AskUbuntu* or *SuperUser* adds no information.

In OOD setup (training on AskUbuntu or Superuser) we can see that base model does the best and fine-tuning brings no gains.

Base Model	Method	Train : AskUbuntu			Train : Superuser		
		AskUbuntu	**Superuser**	**Quora**	**AskUbuntu**	**Superuser**	**Quora**
NLI	`Base`	31.80	48.50	**14.00**	31.80	48.50	**14.00**
	`Finetuned`	65.68 ± 0.54	72.70 ± 0.73	12.74 ± 0.43	**47.98 ± 6.63**	75.70 ± 1.49	12.82 ± 0.49
	`MaskReg`	**71.51 ± 1.16**	77.55 ± 0.57	12.44 ± 0.48	33.83 ± 3.07	81.97 ± 0.33	13.13 ± 0.17
	`SimCSE`	67.31 ± 1.28	73.46 ± 2.24	12.87 ± 0.36	45.03 ± 4.22	76.80 ± 0.68	13.41 ± 0.20
	`OutReg`	56.86 ± 0.69	66.28 ± 0.66	**14.09 ± 0.18**	36.52 ± 1.53	71.11 ± 0.86	**14.44 ± 0.28**
	`ITVAug`	70.10 ± 0.80	76.98 ± 0.33	13.74 ± 0.12	45.86 ± 4.10	81.34 ± 0.11	14.22 ± 0.45
	`ITVReg`	**71.24 ± 0.62**	**78.89 ± 0.59**	13.57 ± 0.32	**47.00 ± 1.58**	**83.40 ± 0.54**	14.67 ± 0.18
MSMARCO	`Base`	54.01	80.73	**18.39**	54.01	80.73	**18.39**
	`Finetuned`	68.11 ± 1.09	75.33 ± 0.74	14.05 ± 0.19	59.27 ± 2.16	79.28 ± 0.94	14.35 ± 0.20
	`MaskReg`	72.59 ± 0.41	79.00 ± 0.08	12.63 ± 0.36	32.05 ± 6.53	83.53 ± 0.53	13.65 ± 0.20
	`SimCSE`	70.07 ± 0.32	76.54 ± 1.43	14.98 ± 0.08	48.97 ± 3.28	80.51 ± 0.70	14.70 ± 0.05
	`OutReg`	73.04 ± 0.77	**84.09 ± 0.28**	17.16 ± 0.13	**60.75 ± 0.84**	**86.24 ± 0.13**	17.64 ± 0.25
	`ITVAug`	73.17 ± 0.46	80.53 ± 0.23	15.36 ± 0.48	54.85 ± 3.94	84.29 ± 0.35	15.64 ± 0.32
	`ITVReg`	**74.64 ± 0.56**	82.86 ± 0.65	16.11 ± 0.36	**60.07 ± 2.42**	**86.24 ± 0.24**	16.39 ± 0.31

Table 3: AUC (0.05) using two different base models, NLI-DistilBERT-Base and MSMARCO-DistilBERT-Base. First three columns correspond to training on *AskUbuntu* and the last three training on *SuperUser*. When evaluated on technical forums with a weaker base model (NLI), `ITVReg` obtains the best AUC on both IID and OOD evaluation.

Summary. Compared to other methods, `ITVReg` obtains high IID accuracy and OOD accuracy, thus exploiting the best of the `Base` model and the training data. However, under extreme distribution shifts, finetuning training data is not useful and thus `OutReg` or the `Base` model is more suitable.

E.1 Ablations

while with our methods `ITVReg` you get "*Channellock 804 4.5-Inch Adjustable Wrench, Chrome*" as the top predicted label See Figure **??** for more details and Table 10,9 for qualitative samples

F Dataset Details

Quora, SuperUser, AskUbuntu The ratio of positives to negatives in *SuperUser, AskUbuntu* is 1:100, while in *Quora* is 4:7.

Amazon131K In *Amazon131K* we have categorical information for 99K labels. The categorical shift experiments are conducted on this filtered data.

G Hyper-parameter Tuning

For `SimCSE` the only choice of hyperparameter is the λ regularisation coefficient. We search over 3 values of λ i.e. 0.01,0.1,1.0 and select 0.1 as the best value. For other methods also we fix λ as 0.1. See Table 4

Method	**Quora**	**AskUbuntu**	**Superuser**
SimCSE 0.01	54.98	73.70	84.91
SimCSE 0.1	55.40	74.34	86.38
SimCSE 1.0	54.89	74.07	85.41

Table 4: `SimCSE` IID numbers finetuned with base model as MSMARCO-DistilBERT-v4 on complete *Quora*,*AskUbuntu*,*SuperUser* datasets. This shows that 0.1 is the optimal hyperparameter for `SimCSE`

For `MaskReg` and `ITVReg` another hyperparameter choice is the masking fraction of the input. We search over 2 choices of masking namely masking 50% of input, or 15% of input. We find that 15% works best for `MaskReg` while 50% gives best results for `ITVReg`. We use these parameters for all experiments. For `MaskReg` results are reported in Table 5, while for `ITVReg` results can be seen in Table 11,13,12.

We also try running with lower learning rates but higher learning rate gives better numbers and hence we work with 1e-4. See Table 6 for reference.

Base Model	Method	Train : AskUbuntu		Train : Superuser	
		AskUbuntu	**Superuser**	**AskUbuntu**	**Superuser**
NLI	`Base`	31.80	48.50	31.80	48.50
	`MaskReg` (0.15)	71.51 ± 1.16	77.55 ± 0.57	33.83 ± 3.07	81.97 ± 0.33
	`MaskReg` (0.50)	64.57 ± 0.68	69.87 ± 0.36	19.85 ± 4.14	77.17 ± 0.98
MSMARCO	`Base`	54.01	80.73	54.01	80.73
	`MaskReg` (0.15)	72.59 ± 0.41	79.00 ± 0.08	32.05 ± 6.53	83.53 ± 0.53
	`MaskReg` (0.50)	64.42 ± 1.44	71.56 ± 0.56	20.36 ± 5.69	77.64 ± 0.94

Table 5: AUC (0.05) for `MaskReg` with 15% and 50% input token masking. 15% is the optimal masking for `MaskReg`. The setup is same as in Table 3.

Learning Rate	2 Epochs	4 Epochs	10 Epochs
1e-5	39.15	45.05	53.40
5e-5	50.67	54.87	57.20
1e-4	52.14	55.54	57.80

Table 6: AUC(0.05) on IID complete *Quora* with `Finetuned` method on NLI-DistilBERT-Base model. Higher learning rate and more epochs help in training. For computational efficiency we hence take 4 epochs for complete setting and 20 epochs for 10% *Quora* subset.

Since we deal with short sentences, the max token length is fixed at 32 which allows for bigger batch sizes (900 for Amazon Titles, 250 for Question Recommendation).

For `ITVReg` and `OutReg` we report numbers for various values of hyperparameter λ in Table 7. We can see that for higher values of λ behaves more like the `Base` model but the numbers for both `ITVReg` and `OutReg` are stable across this wide range of λ.

H Training Details

H.1 AmazonTitle Recommendations Training Details

We follow the modeling procedure used in a recent state-of-the-art work [6]. For training, We use contrastive loss with hard negative mining. we initialise the model as MSMARCO-DistilBERT model, and fine-tune it for 200 epochs with a batch size of 900. The only difference from [6]'s setup is that we train only till 200 epochs instead of convergence (300-400 epochs) due to computational constraints. We use Adam optimizer with learning rate of 1e-4.

H.2 Question Recommendations Training Details

We train all the models for 5 epochs with a batch size of 250. We use learning rate of 1e-4 to fine-tune the model. We train the model with a MSE loss between the ground truth and the predicted similarities, as done in [22]. For *Quora* on the 10% subset, we train the model for 20 epochs with lr of 1e-4.

Base Model	Method	Train : Quora		
		Quora	**AskUbuntu**	**Superuser**
MSMARCO-DistilBERT	`Base`	0.184 ± 0.000	0.540 ± 0.000	0.807 ± 0.000
	`ITVReg` 0.01	0.550 ± 0.031	0.165 ± 0.023	0.293 ± 0.018
	`ITVReg` 0.1	0.577 ± 0.002	0.175 ± 0.001	0.286 ± 0.011
	`ITVReg` 1.0	0.577 ± 0.002	0.175 ± 0.001	0.286 ± 0.011
	`OutReg` 0.01	0.570 ± 0.006	0.176 ± 0.023	0.278 ± 0.002
	`OutReg` 0.1	0.558 ± 0.005	0.297 ± 0.022	0.438 ± 0.006
	`OutReg` 1.0	0.491 ± 0.015	0.456 ± 0.001	0.646 ± 0.008

Table 7: AUC (0.05) for different values of the hyperparameter λ. We can see that both `ITVReg` and `OutReg` behave well in these reasonable range of λ. `OutReg` on higher values of hyperparameter imitates the `Base` model while `ITVReg` still learns useful information from the data

Fine-tuning Method	Automobiles	Kitchen and Dining	Health and Personal Care	Electronics	Tools and Home Imp.
No Finetune	32.66	30.68	30.42	32.07	31.67
Std Finetune	31.41 ± 0.28	29.70 ± 0.27	29.17 ± 0.25	30.60 ± 0.13	30.59 ± 0.25
`OutReg`	34.16 ± 0.04	32.25 ± 0.16	31.75 ± 0.11	33.14 ± 0.07	33.18 ± 0.10
`MaskReg`	32.16 ± 0.23	30.30 ± 0.18	29.85 ± 0.24	31.24 ± 0.11	31.22 ± 0.23
SimCSE	31.45 ± 0.12	29.84 ± 0.04	29.38 ± 0.06	30.70 ± 0.19	30.61 ± 0.16
`ITVReg`	32.61 ± 0.02	30.76 ± 0.10	30.26 ± 0.18	31.73 ± 0.14	31.59 ± 0.08

Table 8: Category wise numbers OOD numbers for *AmazonCatOOD* (Table 2). All categories have a similar trend as observed in Table 2. This shows that our results are agnostic to the choice of categories and hold true for any category in general.

I Category-wise P@1 for AmazonCatOOD

We report category wise numbers OOD numbers for AmazonCatOOD (Table 2). All categories have a similar trend. Results can be seen in Table 8.

J Evaluation Metrics

J.1 Precision@1

Given a query, the model outputs a vector $\hat{R} \in \mathbb{R}^M$, where M is the number of items and $\hat{r}_j$ denotes the similarity between the query and the j th item. We use the $Precision@1$ metric for evaluation, interpreted as the fraction of queries where the top-predicted item is in the query's ground-truth relevant items. Given predicted similarities $\hat{r}$ and ground truth similarities r (where r_i is 1 if its a relevant item, otherwise 0), $P@k$ is defined as $P@k = \frac{1}{k} \sum\limits_{l \in rank_k(\hat{r})} r_l$.

K Qualitative Samples

We give top 5 predicted labels as qualitative samples for "*Soleus Air Oscillating Reflective Heater*" query in Table 10 and for query " *Rowenta Z100 Non Toxic Soleplate Cleaner*" in Table 9.

L Complete Dataset Results

For the complete dataset results, we follow the same setup as described before, with the only difference being number of epochs. We run *Quora* for 4 epochs and *SuperUser, Askubuntu* for 1 epoch. Results can be seen in Table 11,13,12.

M 10% Dataset Subset Results

Results on 10% subset of the data (most of them are redundant as they overlap with results in main paper) can be seen here: Table 14,16,15.

N Experimental Budget

Computing Infrastructure We use 16GB V100 GPUs for all our experiments

Training Time Our training time is around 1 hour for each run on Question Recommendation. Amazon dataset takes 8 hrs for training.

Parameters Of Model We use DistilBERT model for all our experiments.

Method	Top 5 Predicted Items
`Base`	Ridgid 31105 24-Inch Aluminum Pipe **Wrench**
	Ridgid 31115 48-Inch Aluminum Pipe **Wrench**
	Ridgid 31110 36-Inch Aluminum Pipe **Wrench**
	Ridgid 31100 18-Inch Aluminum Pipe **Wrench**
	Craftsman 9-41796 Ratcheting Ready Bit Screwdriver
`Finetuned`	Rowenta ZD100 Non-Toxic Soleplate Cleaner Kit
	Rowenta DR5015 800 Watt Ultra Steam Brush with Travel Pouch
	Rowenta(R) Stainless Steel Soleplate Cleaning Kit ZD-110
	Rowenta DR6015 Ultrasteam Hand-Held Steam Brush with Travel Pouch, 800-watt
	Rowenta DR6050 Ultrasteam Hand-Held Steam Brush Dual-Voltage with Travel Pouch, 800-watt
`OutReg`	Rowenta DR6015 Ultrasteam Hand-Held Steam Brush with Travel Pouch, 800-watt
	Rowenta DW4060 Auto Steam Iron 1700W with Airglide Stainless Steel Soleplate Auto-off Anti-Scale, Blue
	Rowenta DR5015 800 Watt Ultra Steam Brush with Travel Pouch
	Rowenta VU2531 Turbo Silence 4-Speed Oscillating Desk Fan, 12-Inch, Bronze
	Rowenta(R) Stainless Steel Soleplate Cleaning Kit ZD-110
`MaskReg`	Rowenta(R) Stainless Steel Soleplate Cleaning Kit ZD-110
	Rowenta ZD100 Non-Toxic Soleplate Cleaner Kit
	Rowenta DG8430 Pro Precision Steam Station with 400 hole Stainless Steel soleplate 1800 Watt, Purple
	Rowenta DR5015 800 Watt Ultra Steam Brush with Travel Pouch
	Rowenta DR6015 Ultrasteam Hand-Held Steam Brush with Travel Pouch, 800-watt
`SimCSE`	Rowenta RH8559 Delta Force 18V Cordless Bagless Energy Star Rated Stick Vacuum Cleaner …
	Rowenta ZD100 Non-Toxic Soleplate Cleaner Kit
	Rowenta(R) Stainless Steel Soleplate Cleaning Kit ZD-110
	Rowenta DR6015 Ultrasteam Hand-Held Steam Brush with Travel Pouch, 800-watt
	Rowenta DR6050 Ultrasteam Hand-Held Steam Brush Dual-Voltage with Travel Pouch, 800-watt
`ITVReg`	**Wrench** Set, Open End Metric 4mm-6mm - SCR-913.00
	Craftsman 6 pc. Universal **Wrench** Set - Metric
	Tusk Spoke **Wrench** Set
	Crescent RD12BK 3/8-Inch Ratcheting Socket **Wrench**
	Allen **Wrench** Set, 10 Pc. Heavy Duty, Extra Long 9 T-handle, Metric Sizes

Table 9: Top 5 predicted items for the query *Rowe USA Spoke Wrench - Bagged 09-0001* given by various methods sorted by relevance. Correct items should be about *wrench* and `ITVReg` and `Base` model both give the same. Other models rely on spurious feature i.e. *Rowe* for predicting items, which leads to wrong results

Method	Top 5 Predicted Items
`Base`	Camco 57723 Dust Cover for Portable Wave 6 Olympian **Heater**
	Sylvania SA200 10 Amp Outdoor Timer with Light Sensor
	Scosche CRAB Chrysler/Jeep Antenna Adapter
	Lasko 6435 Designer Series Ceramic Oscillating **Heater** with Remote Control
	Reusable Angel Ice Sculpture Mold
`Finetuned`	3M Scotchlite Reflective Tape, Silver, 2-Inch by 36-Inch
	3M Scotchlite Reflective Tape, Red, 2-Inch by 36-Inch
	Reflective Band - Made With Genuine Reflexite in America - By Jogalite (Pair of Two)
	Sunlite 4 Piece Bicycle Reflector Set with Brackets
	Road ID - Reflective Shoe Laces
`OutReg`	3M Scotchlite Reflective Tape, Silver, 1-Inch by 36-Inch
	3M Scotchlite Reflective Tape, Red, 2-Inch by 36-Inch
	Reflective Band - Made With Genuine Reflexite in America - By Jogalite (Pair of Two)
	Casual Canine Reflective Jacket
	Aspects 264 Weather Dome
`MaskReg`	Reflective Band - Made With Genuine Reflexite in America - By Jogalite (Pair of Two)
	3M Scotchlite Reflective Tape, Red, 2-Inch by 36-Inch
	Lasko 6435 Designer Series Ceramic Oscillating Heater with Remote Control
	3M Scotchlite Reflective Tape, Silver, 1-Inch by 36-Inch
	Skylink PS-101 AAA+ Motion Sensor
`SimCSE`	3M Scotchlite Reflective Tape, Silver, 2-Inch by 36-Inch
	3M Scotchlite Reflective Tape, Red, 2-Inch by 36-Inch
	Reflective Band - Made With Genuine Reflexite in America - By Jogalite (Pair of Two)
	Gates T274 Timing Belt
	Bell Automotive 22-5-00106-8 Heavy Duty Tubeless Tire Repair Kit
`ITVReg`	Reflective Band - Made With Genuine Reflexite in America - By Jogalite (Pair of Two)
	Broan 679 Ventilation Fan and Light Combination
	3M Scotchlite Reflective Tape, Silver, 1-Inch by 36-Inch
	3M Scotchlite Reflective Tape, Red, 2-Inch by 36-Inch
	Roadpro 12V **Heater** and Fan with Swing-out Handle

Table 10: Top 5 predicted items for the query *Soleus Air Oscillating Reflective Heater* given by various methods sorted by relevance. Correct items should be about *Heater* and `ITVReg` and `Base` model both give the same. Other models rely on spurious feature i.e. *Reflective* for predicting items, which leads to wrong results

Pre-trained Model	**Fine-tuning Method**	**Quora**	**AskUbuntu**	**Superuser**
NLI-DistilBERT-Base	`Base`	14.00	31.80	48.50
	`Finetuned`	54.84 ± 0.82	17.50 ± 2.41	29.25 ± 1.10
	`MaskReg`	56.00 ± 1.15	19.79 ± 0.97	29.53 ± 1.59
	`SimCSE`	56.24 ± 0.55	21.05 ± 1.57	29.04 ± 2.01
	`OutReg`	52.69 ± 0.41	17.41 ± 1.16	27.33 ± 0.90
	`ITVReg` (0.15)	57.19 ± 0.57	19.70 ± 1.33	28.67 ± 0.26
	`ITVReg` (0.50)	56.64 ± 0.74	20.56 ± 1.07	28.57 ± 1.04
MSMARCO-DistilBERT-v4	`Base`	18.39	54.01	
	`Finetuned`	54.50 ± 0.31	19.79 ± 3.55	30.48 ± 1.33
	`MaskReg`	56.00 ± 0.60	19.55 ± 3.12	33.71 ± 0.62
	`SimCSE`	56.29 ± 1.62	19.57 ± 1.54	31.71 ± 0.93
	`OutReg`	54.41 ± 0.04	32.01 ± 1.72	46.80 ± 1.75
	`ITVReg` (0.15)	56.63 ± 0.65	20.09 ± 1.46	33.06 ± 1.65
	`ITVReg` (0.50)	55.98 ± 0.68	19.50 ± 1.77	33.51 ± 0.62

Table 11: Trained on Complete *Quora* for 4 epochs (same hyperparameters as in paper)

Pre-trained Model	Fine-tuning Method	Quora	AskUbuntu	Superuser
NLI-DistilBERT-Base	`Base`	14.00	31.80	48.50
	`Finetuned`	11.02 ± 0.43	71.09 ± 1.05	74.90 ± 0.27
	`MaskReg`	13.36 ± 0.38	77.62 ± 0.31	79.09 ± 0.21
	`SimCSE`	12.41 ± 0.34	74.73 ± 0.79	77.58 ± 0.11
	`OutReg`	14.75 ± 0.41	68.18 ± 0.64	72.88 ± 1.14
	`ITVReg` (0.15)	13.90 ± 0.51	77.54 ± 0.33	79.70 ± 0.31
	`ITVReg` (0.50)	13.77 ± 0.63	77.75 ± 0.49	81.06 ± 0.32
MSMARCO-DistilBERT-v4	`Base`	18.39	54.01	80.73
	`Finetuned`	12.01 ± 0.33	70.70 ± 1.16	74.94 ± 0.22
	`MaskReg`	13.66 ± 0.28	78.04 ± 0.44	79.94 ± 0.26
	`SimCSE`	13.89 ± 0.37	75.39 ± 0.99	78.53 ± 0.15
	`OutReg`	17.34 ± 0.26	78.84 ± 0.49	85.01 ± 1.41
	`ITVReg` (0.15)	14.84 ± 0.53	78.10 ± 0.55	81.19 ± 0.67
	`ITVReg` (0.50)	15.20 ± 0.41	78.47 ± 0.17	82.27 ± 0.36

Table 12: Trained on Complete *AskUbuntu* for 1 epoch (same hyperparameters as in paper)

Pre-trained Model	Fine-tuning Method	Quora	AskUbuntu	Superuser
NLI-DistilBERT-Base	`Base`	14.00	31.80	48.50
	`Finetuned`	12.68 ± 0.37	50.90 ± 2.95	83.50 ± 0.97
	`MaskReg`	13.65 ± 0.31	40.56 ± 2.66	87.79 ± 0.67
	`SimCSE`	14.39 ± 0.64	53.64 ± 1.67	85.85 ± 0.58
	`OutReg`	15.60 ± 0.13	35.41 ± 2.91	80.68 ± 0.73
	`ITVReg` (0.15)	15.19 ± 0.20	44.38 ± 1.10	88.31 ± 0.65
	`ITVReg` (0.50)	14.30 ± 0.18	45.83 ± 5.42	88.78 ± 0.39
MSMARCO-DistilBERT-v4	`Base`	18.39	54.01	80.73
	`Finetuned`	12.61 ± 0.46	52.32 ± 3.07	84.60 ± 0.77
	`MaskReg`	14.20 ± 0.22	38.32 ± 4.16	88.63 ± 0.44
	`SimCSE`	14.08 ± 0.39	49.04 ± 3.56	86.55 ± 0.32
	`OutReg`	17.42 ± 0.40	57.83 ± 1.31	89.94 ± 0.23
	`ITVReg` (0.15)	16.03 ± 0.66	52.77 ± 2.07	88.85 ± 0.43
	`ITVReg` (0.50)	16.24 ± 0.31	56.33 ± 1.91	89.90 ± 0.30

Table 13: Trained on Complete *SuperUser* for 1 epoch (same hyperparameters as in paper)

Pre-trained Model	Fine-tuning Method	Quora	AskUbuntu	Superuser
NLI-DistilBERT-Base	`Base`	14.00	31.80	48.50
	`Finetuned`	33.22 ± 0.82	10.95 ± 1.45	18.56 ± 1.53
	`MaskReg`	35.02 ± 0.85	12.94 ± 1.05	21.48 ± 1.65
	`SimCSE`	32.52 ± 0.29	10.82 ± 1.19	20.99 ± 1.13
	`OutReg`	33.41 ± 0.70	14.91 ± 1.26	25.63 ± 1.67
	`ITVReg` (0.15)	33.94 ± 1.47	13.73 ± 0.66	22.74 ± 0.59
	`ITVReg` (0.50)	32.93 ± 0.50	12.33 ± 0.89	20.91 ± 0.52
MSMARCO-DistilBERT-v4	`Base`	18.39	54.01	80.73
	`Finetuned`	34.15 ± 0.30	13.31 ± 1.29	23.73 ± 1.63
	`MaskReg`	36.49 ± 0.70	11.89 ± 1.70	25.88 ± 2.49
	`SimCSE`	34.85 ± 1.32	11.34 ± 1.99	25.63 ± 0.64
	`OutReg`	38.50 ± 0.29	31.96 ± 3.23	55.98 ± 1.49
	`ITVReg` (0.15)	38.41 ± 1.09	15.91 ± 2.22	30.95 ± 1.49
	`ITVReg` (0.50)	35.26 ± 1.09	12.28 ± 1.58	24.61 ± 0.95

Table 14: Trained on 10% *Quora*

Pre-trained Model	Fine-tuning Method	Quora	AskUbuntu	Superuser
NLI-DistilBERT-Base	`Base`	14.00	31.80	48.50
	`Finetuned`	12.74 ± 0.43	65.68 ± 0.54	72.70 ± 0.73
	`MaskReg`	12.44 ± 0.48	71.51 ± 1.16	77.55 ± 0.57
	`SimCSE`	12.87 ± 0.36	67.31 ± 1.28	73.46 ± 2.24
	`OutReg`	14.09 ± 0.18	56.86 ± 0.69	66.28 ± 0.66
	`ITVReg` (0.15)	13.62 ± 0.25	71.45 ± 1.17	78.32 ± 0.58
	`ITVReg` (0.50)	13.57 ± 0.32	71.24 ± 0.62	78.89 ± 0.59
MSMARCO-DistilBERT-v4	`Base`	18.39	54.01	80.73
	`Finetuned`	14.05 ± 0.19	68.11 ± 1.09	75.33 ± 0.74
	`MaskReg`	12.63 ± 0.36	72.59 ± 0.41	79.00 ± 0.08
	`SimCSE`	14.98 ± 0.08	70.07 ± 0.32	76.54 ± 1.43
	`OutReg`	17.16 ± 0.13	73.04 ± 0.77	84.09 ± 0.28
	`ITVReg` (0.15)	15.67 ± 0.22	74.11 ± 0.58	81.78 ± 0.28
	`ITVReg` (0.50)	16.11 ± 0.36	74.64 ± 0.56	82.86 ± 0.65

Table 15: Trained on 10 % *AskUbuntu*

Pre-trained Model	Fine-tuning Method	Quora	AskUbuntu	Superuser
NLI-DistilBERT-Base	`Base`	14.00	31.80	48.50
	`Finetuned`	12.82 ± 0.49	47.98 ± 6.63	75.70 ± 1.49
	`MaskReg`	13.13 ± 0.17	33.83 ± 3.07	81.97 ± 0.33
	`SimCSE`	13.41 ± 0.20	45.03 ± 4.22	76.80 ± 0.68
	`OutReg`	14.44 ± 0.28	36.52 ± 1.53	71.11 ± 0.86
	`ITVReg` (0.15)	14.35 ± 0.28	40.10 ± 3.68	82.27 ± 0.38
	`ITVReg` (0.50)	14.67 ± 0.18	47.00 ± 1.58	83.40 ± 0.54
MSMARCO-DistilBERT-v4	`Base`	18.39	54.01	
	`Finetuned`	14.35 ± 0.20	59.27 ± 2.16	79.28 ± 0.94
	`MaskReg`	13.65 ± 0.20	32.05 ± 6.53	83.53 ± 0.53
	`SimCSE`	14.70 ± 0.05	48.97 ± 3.28	80.51 ± 0.70
	`OutReg`	17.64 ± 0.25	60.75 ± 0.84	86.24 ± 0.13
	`ITVReg` (0.15)	15.94 ± 0.15	53.02 ± 5.42	85.44 ± 0.23
	`ITVReg` (0.50)	16.39 ± 0.31	60.07 ± 2.42	86.24 ± 0.24

Table 16: Trained on 10% *SuperUser*

Base Model	Method	Train : Quora			Train : AskUbuntu			Train : Superuser		
		Quora	**AskUbuntu**	**Superuser**	**Quora**	**AskUbuntu**	**Superuser**	**Quora**	**AskUbuntu**	**Superuser**
NLI	`Base`	0.140 ± 0.000	0.318 ± 0.000	0.486 ± 0.000	0.140 ± 0.000	0.318 ± 0.000	0.486 ± 0.000	0.140 ± 0.000	0.318 ± 0.000	0.486 ± 0.000
	`Finetuned`	0.559 ± 0.016	0.181 ± 0.018	0.257 ± 0.021	0.092 ± 0.001	0.775 ± 0.000	0.762 ± 0.002	0.122 ± 0.014	0.632 ± 0.029	0.863 ± 0.005
	`MaskReg`	0.529 ± 0.046	0.231 ± 0.014	0.331 ± 0.031	0.119 ± 0.010	0.810 ± 0.008	0.803 ± 0.010	0.132 ± 0.000	0.417 ± 0.057	0.901 ± 0.007
	`SimCSE`	0.571 ± 0.003	0.176 ± 0.023	0.259 ± 0.004	0.118 ± 0.012	0.792 ± 0.001	0.794 ± 0.012	0.137 ± 0.003	0.587 ± 0.024	0.876 ± 0.005
	`OutReg`	0.553 ± 0.002	0.149 ± 0.015	0.225 ± 0.005	0.146 ± 0.003	0.691 ± 0.009	0.737 ± 0.003	0.155 ± 0.000	0.383 ± 0.014	0.777 ± 0.019
	`ITVReg`	0.566 ± 0.013	0.186 ± 0.011	0.295 ± 0.018	0.136 ± 0.001	0.820 ± 0.001	0.815 ± 0.001	0.151 ± 0.000	0.563 ± 0.015	0.903 ± 0.005
MSMARCO	`Base`	0.184 ± 0.000	0.540 ± 0.000	0.807 ± 0.000	0.184 ± 0.000	0.540 ± 0.000	0.807 ± 0.000	0.184 ± 0.000	0.540 ± 0.000	0.807 ± 0.000
	`Finetuned`	0.564 ± 0.001	0.176 ± 0.005	0.265 ± 0.001	0.104 ± 0.007	0.776 ± 0.013	0.757 ± 0.011	0.144 ± 0.011	0.601 ± 0.020	0.862 ± 0.009
	`MaskReg`	0.526 ± 0.008	0.225 ± 0.011	0.363 ± 0.017	0.108 ± 0.004	0.813 ± 0.004	0.810 ± 0.000	0.131 ± 0.005	0.434 ± 0.007	0.908 ± 0.005
	`SimCSE`	0.574 ± 0.019	0.187 ± 0.006	0.301 ± 0.017	0.123 ± 0.004	0.797 ± 0.005	0.787 ± 0.007	0.125 ± 0.004	0.575 ± 0.050	0.887 ± 0.000
	`OutReg`	0.558 ± 0.005	0.297 ± 0.022	0.438 ± 0.006	0.166 ± 0.001	0.805 ± 0.002	0.851 ± 0.001	0.173 ± 0.004	0.614 ± 0.013	0.903 ± 0.001
	`ITVReg`	0.577 ± 0.002	0.175 ± 0.001	0.286 ± 0.011	0.134 ± 0.004	0.831 ± 0.004	0.826 ± 0.000	0.159 ± 0.002	0.575 ± 0.000	0.911 ± 0.008

Table 17: AUC (0.05) using two different base models, NLI-DistilBERT-Base and MSMARCO-DistilBERT-Base. First three columns correspond to training on *Quora*, middle three to *AskUbuntu* and the last three training on *SuperUser*. When evaluated on technical forums with a weaker base model (NLI), `ITVReg` obtains the best AUC on both IID and OOD evaluation. These are different from Table 3, they run with lower learning rate of 1e-5 and higher number of epochs with best model numbers reported