Supplementary Material

A Details on supervised methods

A.1 ADA (Shah et al., 2018)

We use the implementation provided by the original authors, with default parameters unless stated otherwise.[18] As described in Section 4.3, we pair every low-resource (target) forum with the CQADupStack (source) forum with which it has the highest word trigram overlap (see Table 9).

ADA involves a bidirectional LSTM sentence encoder F and a domain discriminator D. F is trained to (a) discriminate between true duplicate-original question pairs and random pairs and (b) make source and target domain representations indistinguishable to D. Objective (a) is optimized by minimizing $\text{hinge}_m(\cos(F(q), F(c)), y)$, where (q, c) is a source domain question pair, y is its label and m is the hinge loss margin. Objective (b) is optimized by gradient reversal (Ganin et al., 2016). Objective (a) has access to all labeled duplicate-original pairs from the source domain, with 100 random pairs per true pair. Objective (b) has access to all unlabeled questions from both domains.

The original authors use heldout labeled target domain data for early stopping. Since individual target datasets are too small for a dev / test split, we take the (highly irregular) step of performing early stopping directly on the test sets. Since this works in favor of the baseline, we consider it acceptable for comparison with MV-DASE.

We do not use the word embeddings provided by the original authors, as they do not cover our vocabulary. Instead, we train FastText on the concatenation of source and target domain.[19]

[18] http://github.com/darsh10/qra_code

[19] We also tried GloVe, but did not achieve better results.

source	target
english	buddhism
gaming	chess
physics	cogsci
programmers	law
unix	networkengineering
gaming	outdoors
programmers	productivity
unix	reverseengineering
wordpress	sitecore
gaming	sports
programmers	sqa
android	windowsphone

Table 9: Source-target mappings for ADA.

	bud	che	cog	law	net	out
ADA	.229	.164	.161	.250	.132	.207
BERT-MAN	.096	.066	.175	.171	.051	.114
	pro	rev	sit	spo	sqa	win
ADA	.117	.147	.225	.299	.193	.218
BERT-MAN	.088	.111	.118	.178	.096	.112

Table 10: MAP of ADA and BERT-MAN on low-resource forums. *bud* and *che* are heldout forums.

A.2 MAN (Chen and Cardie, 2018)

We use and extend the implementation provided by the original authors, with default parameters unless stated otherwise.[20]

The Multinomial Adversarial Network (MAN) framework is designed for multi-source multi-target domain-adaptation scenarios, so we train a single model on all 24 forums. MAN involves a shared encoder F_s, a set of private encoders $\{F_{d_1} \ldots F_{d_N}\}$ (one per source domain, here: $N = 12$), a shared classifier C and a shared domain discriminator D. In the original framework, F_s and $\{F_d\}$ are implemented as LSTMs, Convolutional Neural Networks or Deep Averaging Networks; however, none of these worked well in initial experiments.[21] Since current State of the Art solutions for many sentence-pair tasks involve Transformer architectures (e.g., Liu et al. (2019b)), we instead instantiate F_s and $\{F_d\}$ as BERT modules (*bert-base-uncased*, downloaded from PyTorch-Transformers[22]). All BERT modules are initialized with the same pre-trained weights, but their parameters can diverge during training. We denote the resulting system as "BERT-MAN".

Our data points are question pairs $x = (q, c)$, which may either be true pairs ($y = 1$), random pairs ($y = 0$), or they may be unlabeled. They are encoded with the standard BERT tokenizer, using appropriate special tokens: [CLS], q, [SEP], c, [SEP]. Sentence pair embeddings $F(x)$ are derived from the topmost layer of the [CLS] token.

C is a feed-forward classifier that takes as input concatenated shared and private representations: $C([F_s(x^{(n)}); F_{d_n}(x^{(n)})])$, where $x^{(n)}$ is a labeled question pair from source domain n. C, F_s and

[20] github.com/ccsasuke/man

[21] In another experiment, we also tried replacing C with the cosine similarity hinge loss classifier from Shah et al. (2018) (see A.1).

[22] github.com/huggingface/pytorch-transformers

$\{F_d\}$ are trained to minimize negative log likelihood. F_s is also trained to make representations from different domains indistinguishable to D, using gradient reversal. This objective is optimized on unlabeled data from all domains, including the target domains.

Early stopping is performed on the heldout target forums (*buddhism* and *chess*, see Table 3). At test time, given query $q^{(n')}$ and a set of potential candidate questions $\{c^{(n')}\}$ from target domain n', we rank candidates by $C([F_s(x_i^{(n')}); \mathbf{0}])$, where $x_i^{(n')} = (q^{(n')}, c_i^{(n')})$.

Since training 13 BERT encoders in parallel is expensive, we use a reduced batch size of 4. We also have to restrict the number of subwords per question pair to 254, which results in truncation for about 40% of question pairs. We make sure to truncate both questions equally, i.e., we combine the first 128 subwords of q with the first 128 subwords of c. It is probable that BERT-MAN would have performed better with more resources during training, but this is beyond the scope of our work.

See Table 10 for a comparison of BERT-MAN and ADA.