Domain-specific knowledge distillation yields smaller and better models for conversational commerce

Kristen Howell
LivePerson
khowell@liveperson.com

Jian Wang
LivePerson
jwang@liveperson.com

Akshay Hazare
LivePerson
ahazare@liveperson.com

Joseph Bradley
LivePerson
jbradley@liveperson.com

Chris Brew
LexisNexis
christopher.brew@google.com

Xi Chen
LivePerson
xchen@liveperson.com

Matthew Dunn
LivePerson
mdunn@liveperson.com

Beth Ann Hockey
LivePerson
bhockey@liveperson.com

Andrew Maurer
Amazon.com Inc
abmaurer@amazon.com

Dominic Widdows
IonQ
widdows@ionq.com

Abstract

In the context of conversational commerce, where training data may be limited and low latency is critical, we demonstrate that knowledge distillation can be used not only to reduce model size, but to simultaneously adapt a contextual language model to a specific domain. We use Multilingual BERT (mBERT; Devlin et al., 2019) as a starting point and follow the knowledge distillation approach of Sanh et al. (2019) to train a smaller multilingual BERT model that is adapted to the domain at hand. We show that for in-domain tasks, the domain-specific model shows on average 2.3% improvement in F1 score, relative to a model distilled on domain-general data. Whereas much previous work with BERT has fine-tuned the encoder weights during task training, we show that the model improvements from distillation on in-domain data persist even when the encoder weights are frozen during task training, allowing a single encoder to support classifiers for multiple tasks and languages.

1 Introduction

Encoders and language models such as BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019) and ELMo (Peters et al., 2017) are the backbone of many NLP technologies. They are typically trained on data from Wikipedia, CommonCrawl, or large homogeneous collections of text; however, language varies widely in real-world settings and the type of language used in some contexts is not well represented in the data used to train these models. In particular, the language used in e-commerce, and more specifically, conversational commerce, such as conversations pertaining to customer service in the context of online shopping or banking, exhibits both syntactic structures and vocabulary that are under-represented in the Wikipedia data used to train multilingual BERT.

At the same time, these models are too large to deploy in many industry settings, where computational resources and inference-speed are concerns. Model size is often reduced using methods such as quantization (Whittaker and Raj, 2001; Shen et al., 2020), pruning (Han et al., 2015, 2016) and knowledge distillation (Hinton et al., 2015; Sanh et al., 2019). However, even leveraging these techniques, the memory footprint of the typical encoder can easily be three orders of magnitude greater than that of the typical classifier, and it follows that encoding is much more time-intensive than classification.[1] In conversational commerce, a variety of classifiers are required to model different aspects of the conversation. In this case, it is beneficial for efficiency, to use a single encoder for all of the classifiers as illustrated in Figure 1 (left), rather than using a separate encoder for each classification task (Figure 1, right).[2] Thus text can be encoded only once and passed to any number of downstream classifiers.

Typically, domain adaptation with language models is accomplished using back-propagation during task training (see inter alia Devlin et al., 2019; Liu et al., 2019; Sanh et al., 2019). However, this approach requires a separate encoder for each classifier. Instead, we adapt the encoder to a particular domain before classifier training. We show that knowledge distillation, a common approach for reducing model size, is very adept for domain adaptation. This allows us to accomplish two goals,

[1] For example, a BERT encoder has hundreds of millions of parameters (see Table 7), while a self attention classifier like the one used in Section 8 has about 600,000.

[2] Houlsby et al. (2019), inter alios, have proposed similar architectures.

Proceedings of The Fifth Workshop on e-Commerce and NLP (ECNLP 5), pages 151 - 160

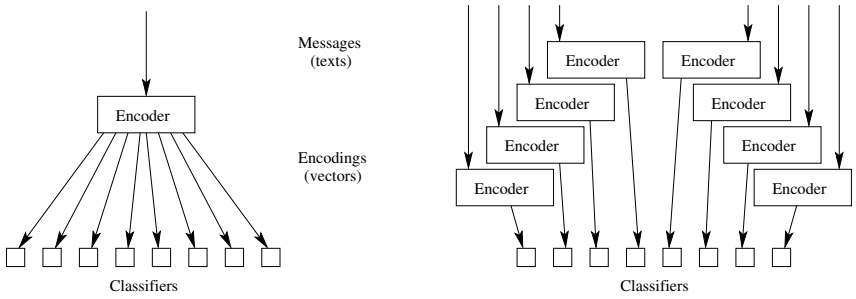

Figure 1: The difference in architecture between using one encoder for multiple tasks vs. one encoder per task

size reduction and domain adaptation, with a single training process. Evaluating on five languages and two domains, we show that distilling on unlabeled data from the domain of interest results in a smaller model that is domain-specific and outperforms the F1-score of a model distilled on domain-general data by 2.3% on average and the larger teacher model by an F1 of 1.2%. The improvement in performance persists even when relatively little training data is used. We show that the domain-adapted encoder performs better than the domain-general model both when encoder weights are fine-tuned, as in previous work, and when they are frozen, leaving them task agnostic. Furthermore, the boost in performance from distillation on in-domain data is greater than the improvement from fine-tuning the encoder during task training.

We begin with an overview of previous work in domain adaptation and knowledge distillation, highlighting the benefit of doing both at once (§2). This is followed by a description of the domains, data and tasks with which we evaluate domain adaptation through knowledge distillation (§3). We detail our approach in Section 4, investigating how much data is necessary (§5) and examining the impact of domain adaptation on sentence embeddings (§6). We evaluate on two domains and five languages in Section 8, considering both training scenarios where encoder weights are frozen for task training and where they are fine-tuned.

2 Related Work

2.1 Knowledge distillation

Many state-of-the-art NLP models have achieved high performance with increased parameters and layers, and in doing so have become too computationally expensive for some applications. Knowledge distillation addresses this problem with a "teacher-student" training approach in which a smaller "student" model learns to mimic a larger "teacher" model (Sanh et al., 2019) or an ensemble of models (Bucilă et al., 2006; Hinton et al., 2015).

In the context of reducing model size with BERT, task-specific distillation has been successful (Tang et al., 2019; Chatterjee, 2019) as has distillation of the encoder during pre-training (Sanh et al., 2019). Distillation of the pre-trained encoder is particularly beneficial as the distilled model can be applied to any number of downstream tasks. Sanh et al. (2019) released a distilled version of English BERT (DistilBERT), which is 40% smaller and 60% faster than the original model, while retaining 97% of its NLU capabilities. This was followed by DistilmBERT, distilled from mBERT using data from 104 languages, which is 24% smaller and 38% faster than its teacher. In both cases, the same data was used for knowledge distillation as for pre-training the original models. We adopt this approach, limiting our training data to the languages and domain of interest to demonstrate that less data can be used to distill a model for a specific setting.

2.2 Domain adaptation

When sufficient data is not available to train a model from scratch, a smaller amount of data can be used to adapt a domain-general model. In the context of BERT, domain adaption of the encoder through continued pre-training on in-domain data followed by task-specific fine-tuning has improved performance on domain-specific applications (Han and Eisenstein, 2019; Gururangan et al., 2020; Rietzler et al., 2020; Whang et al., 2020).

Previous work suggests that the teacher-student approach used for knowledge distillation is well suited to domain adaptation. In ASR, it has been applied to adapt models trained on clean speech to handle noisy speech, models for speech from headset mics to work for distant mics (Manohar et al., 2018), and for speaker adaptation (Yu et al., 2013). In neural machine translation, multidomain models

have been distilled from single-domain specialist models (see inter alia Currey et al. 2020; Mghabbar and Ratnamogan 2020). In the context of sentiment analysis, Ruder et al. (2017) use an ensemble of models to train a domain-adapted model on unlabeled in-domain data and Ryu and Lee (2020) combine distillation with adversarial domain adaptation to mitigate over-fitting from task fine-tuning, rather than to reduce model size.

We show that knowledge distillation can simultaneously reduce the size of the model and adapt it to a domain. While, some degree of performance loss during distillation is typical, we show that focusing the training objective on in-domain data can eliminate performance loss and even improve model performance in the domain of interest. Our training objective does not require labeled data, and because we do this before task fine-tuning, the resulting model can be used for any number of in-domain tasks.

3 Use-cases and datasets

3.1 The conversational commerce use-case

Our first use case is in conversational commerce (hereafter CC), which involves messaging between customers and agents (human or automated) in a commercial customer service setting. Within CC, there are sub-domains for commercial industries, such as retail, financial services, airlines, etc.

Unlike the Wikipedia data used to train mBERT and DistilmBERT, CC is marked by questions, first and second person phrases, short responses, frequent typos and other textual and linguistic features that are more common in typed conversation. In addition to structural variation, CC data contains many product and service names that may not be common in Wikipedia data. These differences make CC a strong candidate for domain adaptation.

Our test-case is to classify customer messages as *intentful*, meaning that the message contains some actionable request, or *not intentful*. In CC, this is an important triage step that can be applied across sub-domains before sending messages to downstream classifiers. Because this classification task is applied to different sub-domains and customers say some surprising things, this task is rather challenging. Intentful messages can vary widely from requests for information, attempts to place orders or change account details, and disputes or complaints. Non-intentful messages include greetings, pleasantries, slot information that relies on a previous message for context, etc. After this triage step, intentful messages can be sent to downstream classifiers, which are specific to the industry or company, that predict specific intents such as "check order status" or "schedule appointment".

Table 1: Total amount of data used to distil each domain-adapted model in GB of uncompressed text

model	Data per language (GB) eng	esp	jap	por	rus	Total (GB)
CC-Distil -mBERT	0.95	0.60	0.44	0.35	0.00	2.34
TD-Distil -mBERT	1.58	0.34	0.06	0.34	0.75	3.07

Table 2: # natural (N) and translated (T) messages per split for the CC classification task

Split	Data	eng	esp	jap	por
Train	N	4951	1364	1810	1845
	T		4306	4306	4306
Val	N	1236	255	286	323
	T		1020	1020	1020
Test	N	10078	719	908	909

3.1.1 Dataset

We use a proprietary dataset from a variety of companies that use a particular conversational commerce platform.[3,4] We distilled the encoder using 2.3GB of unlabeled text from English [ISO 639-3: eng], Spanish [esp], Japanese [jap] and Portuguese [por] as detailed in Table 1. This data came from 25 companies that span the retail, telecommunications, financial and airlines sub-domains. To verify that the encoder generalizes beyond these companies, we sampled data for the classification task from an additional 14 companies that were not used in encoder training as well as 12 that were.

Annotations classification were provided by native speakers of each language, who were trained on the task. Due to limited access to data and annotators for Japanese, Portuguese and Spanish, we supplemented natural language training data with machine-translated data from English. For evaluation, we used only naturally produced data from each language (see Table 2). The complete breakdown by class for evaluation is in Table 3.

The majority of the English, Portuguese and Spanish data is from the Americas and the remainder from Australia and Europe, while the Japanese data is primarily from Japan. Because data-use

[3]To be clarified after the anonymity period.

[4]For customer privacy, all personally identifiable information is masked before we use the data, but even after masking we cannot make the data or models publicly available.

Table 3: # messages per label in the CC evaluation set

label	eng	esp	jap	por
intentful	4050	423	475	537
not intentful	6030	298	435	373

agreements and laws vary by company and country, we could not sample evenly across regions; still, we sampled from as diverse a range of countries and company types as possible, in an effort to maximize representation of different speaker communities.[5]

3.2 The technical discussion use-case

Because our first dataset is proprietary, we repeat the experiments using data from online forums for technical discussions about programming (hereafter TD). This domain is marked by technical jargon, which includes many words that have non-technical homonyms, as we describe in Section 6. These forums also include code and urls, which can be useful for classification but are not common in the Wikipedia data used to train mBERT.

For evaluation we used a multi-label prediction task to automatically label posts with the appropriate tag or tags for the topic.[6] Because StackOverflow uses hundreds of tags, the task is limited to the ten most common, which are listed in Table 4.

3.2.1 Dataset

The data for this task comes from the anonymized dumps for the StackOverflow topics in English [eng], Japanese [jap], Portuguese [por], Russian [rus] and Spanish [esp].[7]We created classification datasets by sampling posts that contain one or more of the ten tags targeted by the task. For our validation and held-out evaluation sets for English, Spanish, Portuguese and Russian, we sampled messages such that each tag occurred in the set at least 100 times for a total of 1000 posts per set. As each post can have multiple tags, tags can occur more than 100 times. The total posts per tag for evaluation are in Table 4.[8] We then randomly selected 8000 training samples that contained at least one of the tags. Because the Japanese dump is much smaller, we created splits of half the size (500/500/4000).

[5]We do not have access to demographic data for the users who produced the data, and cannot make any claims about how well the models generalize across speaker communities of various ages, genders, ethnicities or socio-economic groups.

[6]This task comes from https://github.com/theRajeshReddy/StackOverFlow-Classification

[7]https://archive.org/details/stackexchange

[8]The number of posts per tag for the train and validation sets can be found in the dataset's readme.

Table 4: # posts per label in the TD evaluation set

label	eng	esp	jap	por	rus
c#	111	110	56	120	109
java	118	127	57	148	174
php	126	138	57	144	168
javascript	168	206	108	197	182
android	110	115	71	115	108
jquery	132	114	53	136	106
python	107	106	51	104	101
html	109	103	50	110	118
c++	104	104	51	101	116
ios	110	100	53	102	102

For encoder training, we sampled data from the remaining messages, including posts that did not contain the tags of interest. We assembled a 3GB training set (based on the results in Section 5) using all of the data for Japanese (0.055GB, 2% of the total), Portuguese (0.34GB, 11%), Spanish (0.34GB, 11%) and Russian (0.75GB. 24%), and 1.58GB (52%) for English to reach 3GB total.

4 Knowledge Distillation Method

For knowledge distillation, we use the established and open-source approach of Sanh et al. (2019),[9][10] which follows Liu et al. (2019)'s proposed best practices for BERT training. These include dynamic masking, large batches to leverage gradient accumulation and training on the masked language modeling task but not next sentence prediction.

Sanh et al.'s implementation of knowledge distillation trains the student model using the distillation loss of the soft target probabilities of the teacher. Because of this, the student model benefits from the the teacher model's full distribution during training. Due to this rich input, we expect that high performance can be achieved with less training data.

We use mBERT[11] as the teacher model and our student models have the same general architecture, hidden-size dimension and number of word embeddings. We reduce the model size by removing the token-embeddings and pooling and reducing the number of layers from 12 to 6. This reduces the total number of parameters by 43 million or 24% and increases the inference speed by 38% (Table 5).[12]

[9]Detailed instructions for training with Huggingface's Distil* module can be found at https://github.com/huggingface/transformers/blob/783d7d2629e97c5f0c5f9ef01b8c66410275c204/examples/research_projects/distillation/README.md.

[10]Here we discuss the most relevant training details, but Ibid. provides a full account of the training procedure.

[11]https://github.com/google-research/bert/blob/master/multilingual.md

[12]The change in model size and speed is equivalent to that

Table 5: Model size and and average inference speed on single-thread CPU with a batch size of 1

Model	# Params (millions)	Inf time per message (milliseconds)
mBERT	178	305
DistilmBERT	135	188
TD-DistilmBERT	135	189
CC-DistilmBERT	135	189

Whereas Sanh et al. (2019) used the same training data as the teacher model, we use only data from the domain we are adapting to. Intuitively, a model that will be deployed in a single domain does not need to learn everything the base model can do — it only needs to learn what it can do for the domain at hand. This allows us to reduce the training data and time needed for distillation.

5 Data requirements

Because knowledge distillation takes advantage of an existing model, which was already trained on a large amount of data, we expect that distillation training will be relatively economical in its use of data. Furthermore, many research objectives focus on a single domain and do not require the breadth of NLU capability of a domain-general model, but instead benefit from a depth of capability in one domain. Here we attempt to establish how much data is enough for knowledge distillation for a single domain and where we reach diminishing returns.

As a case-study, we use increasing quantities of StackOverflow English data for knowledge distillation and compare the performance of these models to both the teacher model (mBERT) and HuggingFace's multilingual distilled BERT model (DistilmBERT), which was distilled using the same approach. To measure the impact of domain adaptation from knowledge distillation alone, we freeze the encoder weights during task training and present the results in Table 6 and Figure 2.

We distilled models with English StackOverflow data, using increments of 0.3GB. We found that a minimum of 1.5GB was needed for convergence, but 2.1GB was enough to outperform DistilmBERT and perform on par with the teacher mBERT. Improvements stop after 3GB. We conclude that 2.1 GB is sufficient and 3GB is optimal for adaptation to the TD domain, while more data increases training time without improving performance.

We contextualize this finding by considering the data quantity used to train mBERT and DistilmBERT. While it is hard to ascertain the exact amount of data used to train these models, we estimated by following the data sampling procedure used by the creators of those models.[14] By our best estimate, roughly 222GB of uncompressed text was used.[15] In contrast, only 2.1GB of uncompressed

Table 6: Macro Precision, Recall and F1 on TD evaluation task for models distilled with increasing data quantities. The number in each TD model name corresponds to the GB of uncompressed text used for training.[13]

Model	Precision	Recall	F1
mBERT *	0.796	0.626	0.692
DistilmBERT *	0.796	0.602	0.679
Wiki3.0*	0.778	0.572	0.651
TD1.5	0.789	0.530	0.627
TD1.8	0.788	0.500	0.605
TD2.1	0.808	0.629	0.700
TD2.4	0.809	0.651	0.718
TD2.7	0.807	0.629	0.705
TD3.0	**0.817**	**0.664**	**0.727**
TD3.3	0.823	0.658	0.728
TD3.6	0.814	0.661	0.724
TD3.9	0.821	0.636	0.710
TD4.2	0.817	0.648	0.718
TD4.5	0.816	0.647	0.714

*=baseline model

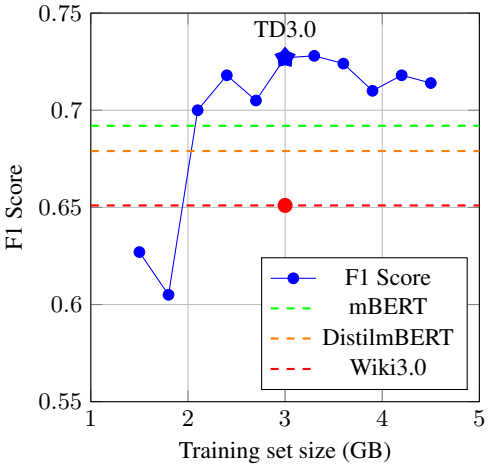

Figure 2: Training set size vs Macro F1 (see Table 6).

in Sanh et al. 2019, however, that paper considers only the English BERT model, while we use multilingual BERT. Because the multilingual model has a significantly larger vocabulary (or number of word embeddings), which is not reduced by this distillation process, the proportionate difference in model size is for distilled mBERT models is less than for distilled BERT.

[13] Here and throughout the paper, reported results are the mean of 10 random initializations.

[14] The procedure for mBERT is detailed at https://github.com/google-research/bert/blob/master/multilingual.md and DistilmBERT used data sampled in the same way (Sanh, pc).

[15] The original numbers may have been smaller as our estimate is based on the wiki data dumps on Oct. 1, 2020 and the models were trained before that time.

text was needed to outperform DistilmBERT on our in-domain task. Thus, for distillation for a single domain and language, the required amount of training data is reduced by two orders of magnitude.[16]

In this experiment, the new data matches on both domain and language. To test whether the language match is responsible for the improvement, we distilled a model on 3GB of English Wikipedia data. We sampled this data by randomly selecting 3000 1MB chunks of text from the English Wikipedia dump. This model (Wiki3.0) under-performs the one distilled on the same amount of StackOverflow data, showing that the language match alone is insufficient to explain the improved performance and suggests that the domain match is more important.

6 Vector changes under domain adaptation

To better understand the differences between an encoder trained on domain-general data versus in-domain data, we compare sentence embeddings produced by the encoder that we adapted to the technical domain (TD3.0) and the encoder distilled on the same amount of domain-general data (Wiki3.0). The TD domain has lots of homonyms like 'python' and 'float' that have both a technical word-sense and a non-technical one. We expect models trained on the TD domain to pay attention to the dominant technical word-senses, and models trained on Wikipedia to pay greater attention to the non-technical word-senses. By extension, a distance function derived from a TD model is expected to be more sensitive to technical word-senses than a distance function derived from a Wikipedia model. Thus we expect the distance function for 'python' and a non-technical synonym (i.e., 'snake') to be closer when derived from a domain-general model and the distance function between 'python' and another programming language (i.e., 'PHP') to be closer when derived from a model trained on technical data.

Because BERT embeddings are contextual, we provide a context for each word pair by creating sentences for each, such that either word may appear in the sentence. Sentences designed for technical word pairs are biased towards a technical context, and sentences for non-technical word pairs are biased towards a non-technical sense.[17] As an example, the technical sentences used to compare Java and C# are given below in list items 1 and 2 and the non-technical sentences used for java and coffee are given in list items 3 and 4.

1. I can't find any code or post on how to get traffic data in **Java** for Windows Phone 8.
2. I can't find any code or post on how to get traffic data in **C#** for Windows Phone 8.
3. Jerry can't start his day without a cup of **java**.
4. Jerry can't start his day without a cup of **coffee**.

Substituting each word from a pair into the sentence, we have a pair of sentences like items 1 and 2 and embed each with both the TD model and the domain-general model. We take the mean of the token embeddings as a representation of the sentence[18] and then take the cosine similarities between the sentence embedding produced by the TD model and the embedding from the domain-general model. These cosine similarities are given for both the technical and non-technical pairs in Table 7.

We find that cosine distance is smaller for sentences that capture the general word-sense when they are encoded by the general model. Similarly, it is smaller for sentences that capture the technical word-sense when they are encoded with the TD model. This suggests that the TD model has adapted its representations for these homonyms to their technical meanings.

7 Experiments

We compare the performance of two domain-adapted encoders that were trained using the method described in Section 4. CC-DistilmBERT was trained with data from four languages from the CC domain (§3.1.1) and TD-DistilmBERT with data from five languages from the TD domain (§3.2.1). The amount of data used to distil each model is summarized in Table 1 and determined primarily based on availability, though informed by the findings in Section 5. For each language and domain, we used as much data as was available for distillation (see §3 for details), except in the case of English TD data, in which case we had more than enough data and used only as much as was necessary to reach approximately 3GB in total.

For classifier training, we train a structured self-

[16] Training with 3 GB took $<$ 2 days using 8 A100 GPUs.

[17] The full collection of sentences is in the appendix.

[18] The embedding of the <CLS> token is often used as a sentence embedding when using BERT. However, following (Liu et al., 2019), we distill models without using the next sentence prediction task, so the embedding for <CLS> is less likely to be a good representation of the sentence.

[18] Using Euclidean distance yielded similar results.

Table 7: Each word in the first column has at least one technical and non-technical sense (e.g., 'Java') and is paired with two terms, one technical and one non-technical that can be used in the same context (e.g., 'coffee' and 'C#'). This table shows the cosine similarity between the embedding for a sentence containing the ambiguous word and the same sentences containing its technical and non-technical alternatives instead, using both the general encoder (Gen. Cosine) and domain-adapted encoder (Tech. Cosine). We show that in most cases the non-technical pairs have a greater cosine similarity when encoded with the general model and the technical pairs have a grater cosine similarity when encoded with the technical model.

Word	General Neighbor	Tech. Cosine	Gen. Cosine	Technical Neighbor	Tech. Cosine	Gen. Cosine
Java	coffee	0.01453	**0.02816**	C#	**0.01350**	0.00424
Python	snake	0.01008	**0.04663**	PHP	0.00535	**0.00687**
floats	rafts	0.00672	**0.01210**	doubles	0.00830	**0.00401**
terminal	ending	0.00527	**0.00759**	command line	**0.01341**	0.00453
classes	lectures	0.01124	**0.01512**	objects	**0.00969**	0.00447
Oracle	Prophet	0.01055	**0.01883**	DynamoDB	**0.00283**	0.00250

attention classifier head on each encoder used in evaluation. [19] We conduct two experiments: in the first, we fine-tune the encoder weights, as has been done in previous work such as Devlin et al. 2019; Sanh et al. 2019; in the second, we freeze the encoder weights to demonstrate that this approach can be used in contexts where the same underlying encoder is to be used by multiple classifiers, removing the need to encode a text every time it is classified by a different classifier.

We compare our domain-adapted, distilled models using the tasks described in Section 3 with two baselines: the teacher model used for distillation (mBERT) and Sanh et al.'s domain-general distilled model (DistilmBERT). In each case, we train and evaluate separate classifiers for each language in the dataset. We evaluate model performance two ways, first fine-tuning the encoder weights during task training and second freezing the encoder weights to test the generalizability of a single encoder to multiple classifiers.

8 Results

The results for each language and encoder are broken down for each experiment in Table 8, where encoder weights were fine-tuned or frozen during task training. On average for these two tasks, the domain-adapted model achieves an F1 score that is 1.2% greater respective to the teacher mBERT model (an absolute difference of 0.9 F1) and 2.3% better respective to the domain-general DistilmBERT model (an absolute difference of 1.7 F1).

Performance is better for all models when the encoder weights are fine-tuned during task training. Still, domain-adapted models perform better relative to the baselines in both cases. The average absolute improvement of the domain-adapted models relative to the teacher model is 1.1% and the relative improvement over the domain-general distilled model is 2.1% when the encoder weights are tuned, while these improvements are 1.3% and 2.5% when the weights are frozen. This difference between the two training scenarios may be accounted for by the encoder undergoing some degree of domain adaptation during fine-tuning. Intuitively, the domain-general models would benefit from this more than the domain-adapted models, which are already tuned to the domain. Even so, the performance of domain-adapted models relative to the domain-general models when the encoder weights are fine-tuned demonstrates that domain adaptation is still beneficial in this scenario and contributes larger improvements in model performance than fine-tuning during task training alone.

Each result in Table 8 represents an average over 10 iterations of training and evaluation. We calculate the statistical significance of the improvement between the domain adapted models and baselines using the Kolmogorov-Smirnov (KS) test, as our data is non-normal (Brownlee, 2019). We found that the improvement in F1 score due to domain adaptation for the CC-DistilmBERT model was statistically significant ($p < .05$) compared to both baselines on all languages except Japanese, despite the small size of our datasets. The TD-DistilmBERT model's improvement in F1 over the DistilmBERT baseline was also statistically significant on all languages except Japanese, although the support for each class was much smaller for the TD task.

For Japanese, in the case of the TD model, very little data was available for domain adaptation (0.06 GB; shown in Table 1). We can speculate that this

[19] Code and data for reproduction are included in supplementary materials and will be made publicly available upon publication.

Table 8: Results for freezing and fine-tuning encoder weights during classifier training. F1, Precision and Recall are for the 'intentful' class in the CC binary classification task and are macro averages for the TD multi label task. Each result is an average across 10 iterations of training and evaluation. For the CC-DistilmBERT and TD-DistilmBERT models, * indicates a statistically significant difference ($p < 0.05$) between that model and DistilmBERT and † indicates a statistically significant difference ($p < 0.05$) between that model and mBERT.

Conversational Commerce

	Encoder Weights	Model	eng	esp	jap	por
F1-Score	Frozen	CC-DistilmBERT	**86.6***†	**82.0***†	**84.0**	**86.2***†
		DistilmBERT	84.3	79.5	**84.0**	85.2
		mBERT	85.4	78.7	83.6	85.2
	Fine-tuned	CC-DistilmBERT	**88.7***†	**82.3***†	**84.5**	**87.4***†
		DistilmBERT	85.8	79.9	83.9	85.6
		mBERT	86.7	80.7	84.1	86.3
Precision	Frozen	CC-DistilmBERT	**89.0**	**80.2**	**81.3**†	**86.9***
		DistilmBERT	87.0	78.4	79.0	83.6
		mBERT	87.0	78.6	77.0	85.7
	Fine-tuned	CC-DistilmBERT	**90.2***†	**81.4**	**79.9**	**87.3**
		DistilmBERT	87.6	79.6	**79.9**	85.8
		mBERT	88.3	78.8	79.2	86.1
Recall	Frozen	CC-DistilmBERT	**84.5**	**84.0**	87.0*†	85.5
		DistilmBERT	81.9	80.9	89.7	**87.0**
		mBERT	83.9	79.0	**91.6**	84.8
	Fine-tuned	CC-DistilmBERT	**87.3***†	**83.2**	89.7	**87.5**
		DistilmBERT	84.1	80.2	88.5	85.4
		mBERT	85.3	82.7	**89.8**	86.6

Technical Discussion

	Encoder Weights	Model	eng	esp	jap	por	rus
F1-Score	Frozen	TD-DistilmBERT	**71.2***	70.1*	**67.4**	**71.2***†	**66.2***
		DistilmBERT	68.9	68.8	65.8	68.2	64.3
		mBERT	70.3	**70.9**	66.5	69.8	66.0
	Fine-tuned	TD-DistilmBERT	**72.5***	**72.9***†	66.7	**71.9***	**67.2***
		DistilmBERT	70.6	71.5	65.9	70.4	66.0
		mBERT	72.3	71.3	**67.2**	71.4	66.5
Precision	Frozen	TD-DistilmBERT	**81.4***	**80.8**	**78.5***†	**79.8**	**78.3***
		DistilmBERT	79.7	78.5	75.7	78.6	75.5
		mBERT	80.8	79.2	76.6	78.2	77.6
	Fine-tuned	TD-DistilmBERT	**80.5***†	**81.2***†	74.7	**78.1**†	**76.5***†
		DistilmBERT	77.6	77.5	73.8	76.8	72.6
		mBERT	77.4	76.4	**74.8**	76.3	73.7
Recall	Frozen	TD-DistilmBERT	**64.3***	63.5	**60.5**	**64.9***	**58.3**
		DistilmBERT	61.6	62.5	59.3	60.7	56.8
		mBERT	63.2	**65.4**	59.8	63.7	**58.3**
	Fine-tuned	TD-DistilmBERT	66.9†	67.6	61.5	67.3	60.7
		DistilmBERT	65.6	67.7	60.9	65.4	61.2
		mBERT	**68.7**	**68.4**	**62.6**	**67.9**	**61.7**

lack of data made adaptation via knowledge distillation less effective. In this case, adaptation via fine-tuning was relatively more effective than adaptation via knowledge distillation. For the CC model, while somewhat more Japanese data was available (0.44 GB), it is still less relative to English and Spanish, so we again attribute the less significant results for Japanese to data scarcity. We note that for this domain, even less data was available for Portuguese (0.35 GB) than Japanese, although for Portuguese the domain adapted model did show a significant improvement over mBERT. In this case, we speculate that the domain adapted model's performance on Portuguese benefited from lexical overlap with the Spanish training data.

9 Conclusion

We addressed the problem of encoder scalability in the context of conversational commerce by showing that knowledge distillation with domain-specific data reduces model size, while simultaneously improving model performance. This approach allows for the training of an encoder that can be used across a variety of languages, is smaller than a state-of-the-art model like mBERT, and performs better on domain-specific tasks. Because our approach uses only training data for the domain and languages of interest, less data is necessary for training, reducing the time, cost and environmental impact of training, while accommodating limited

data availability. A key advantage of domain adaptation during encoder rather than classifier training is that it allows for the deployment of a single encoder, which can serve multiple classifiers at runtime. This reduces storage and maintenance cost, and due to the much larger size of the encoders compared with the classifiers, provides dramatically better scalability in real-world, e-commerce applications that must support multiple languages and tasks.

References

Jason Brownlee. 2019. How to use statistical significance tests to interpret machine learning results. Accessed: October 11, 2021.

Cristian Bucilă, Rich Caruana, and Alexandru Niculescu-Mizil. 2006. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541.

Debajyoti Chatterjee. 2019. Making neural machine reading comprehension faster. *arXiv preprint arXiv:1904.00796*.

Anna Currey, Prashant Mathur, and Georgiana Dinu. 2020. Distilling multiple domains for neural machine translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4500–4511.

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

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. *arXiv preprint arXiv:2004.10964*.

Song Han, Xingyu Liu, Huizi Mao, Jing Pu, Ardavan Pedram, Mark A Horowitz, and William J Dally. 2016. Eie: efficient inference engine on compressed deep neural network. *ACM SIGARCH Computer Architecture News*, 44(3):243–254.

Song Han, Jeff Pool, John Tran, and William Dally. 2015. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28:1135–1143.

Xiaochuang Han and Jacob Eisenstein. 2019. Unsupervised domain adaptation of contextualized embeddings for sequence labeling. 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 4238–4248.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

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.

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*.

Vimal Manohar, Pegah Ghahremani, Daniel Povey, and Sanjeev Khudanpur. 2018. A teacher-student learning approach for unsupervised domain adaptation of sequence-trained asr models. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 250–257. IEEE.

Idriss Mghabbar and Pirashanth Ratnamogan. 2020. Building a multi-domain neural machine translation model using knowledge distillation. *arXiv preprint arXiv:2004.07324*.

Matthew E Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. *arXiv preprint arXiv:1705.00108*.

Alexander Rietzler, Sebastian Stabinger, Paul Opitz, and Stefan Engl. 2020. Adapt or get left behind: Domain adaptation through bert language model finetuning for aspect-target sentiment classification. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 4933–4941.

Sebastian Ruder, Parsa Ghaffari, and John G Breslin. 2017. Knowledge adaptation: Teaching to adapt. *arXiv preprint arXiv:1702.02052*.

Minho Ryu and Kichun Lee. 2020. Knowledge distillation for BERT unsupervised domain adaptation. *arXiv preprint arXiv:2010.11478*.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. In *NeurIPS EMC^2 Workshop*.

Sheng Shen, Zhen Dong, Jiayu Ye, Linjian Ma, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. 2020. Q-BERT: Hessian Based Ultra Low Precision Quantization of BERT. In *AAAI*, pages 8815–8821.

Raphael Tang, Yao Lu, Linqing Liu, Lili Mou, Olga Vechtomova, and Jimmy Lin. 2019. Distilling task-specific knowledge from BERT into simple neural networks. *arXiv preprint arXiv:1903.12136*.

Taesun Whang, Dongyub Lee, Chanhee Lee, Kisu Yang, Dongsuk Oh, and Heuiseok Lim. 2020. An effective domain adaptive post-training method for bert in response selection. *Interspeech 2020*, page 1585–1589.

Edward WD Whittaker and Bhiksha Raj. 2001. Quantization-based language model compression. In *Seventh European Conference on Speech Communication and Technology*.

Dong Yu, Kaisheng Yao, Hang Su, Gang Li, and Frank Seide. 2013. Kl-divergence regularized deep neural network adaptation for improved large vocabulary speech recognition. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 7893–7897. IEEE.

A Appendix: Sentence Pairs

The sentences used to produce the embeddings compared in Table 8 are listed below.

A.1 Non-technical Sentences

1. Jerry has had a lot of late nights recently and can't start his day without a cup of {Java / coffee}.
2. I was scared to death when I saw that {Python / snake}.
3. The ship is outfitted with lots of safety features, including {floats / rafts} in case of an emergency.
4. The {terminal / ending} scene of the movie was a big surprise!
5. Emily had to drop out of school after she missed too many {classes / lectures}.
6. In times of uncertainty, people would take offerings to the temple and ask the {Oracle / Prophet} for help.

A.2 Technical Sentences

1. I can't find any code or post on how to get traffic data in {Java / C#} for Windows Phone 8.
2. I have {Python / PHP} code for shortening a URL.
3. Suppose I need to parse space delimited lists of numbers where some lists contain integers and some lists contain {floats / doubles}.
4. I am a spark newbie and I want to run a Python script from the {terminal / command line}.
5. fill up a div with this data with relevent markup - divs id's {classes / objects} surrounding this data.
6. So what I'm doing is reading a lot of data from remote Nettezza database and inserting them into another remote {Oracle / DynamoDB} database.