NLP Service APIs and Models for Efficient Registration of New Clients

Anonymous EMNLP submission

Abstract

State-of-the-art NLP inference uses enormous neural architectures and models trained for GPU-months, well beyond the reach of most consumers of NLP. This has led to one-size-fits-all public API-based NLP service models by major AI companies, serving large numbers of clients. Neither (hardware deficient) clients nor (heavily subscribed) servers can afford traditional fine tuning. Many clients own little or no labeled data. We initiate a study of adaptation of centralized NLP services to clients, and present one practical and lightweight approach. Each client uses an unsupervised, corpus-based sketch to register to the service. The server uses an auxiliary network to map the sketch to an abstract vector representation, which then informs the main labeling network. When a new client registers with its sketch, it gets immediate accuracy benefits. We demonstrate the success of the proposed architecture using sentiment labeling, NER, and predictive language modeling.

1 Introduction

State-of-the-art NLP uses large neural networks with billions of parameters, enormous training data, and intensive optimization over weeks of GPU-time, causing more carbon emission than a car over its lifetime (Strubell et al., 2019). Such training prowess is (mercifully) out of reach for most users of NLP methods. Recognizing this, large AI companies have launched NLP cloud services[1] and also provided trained models for download and fine tuning. But many clients have too little data or hardware for fine tuning massive networks. Neither can the service be expected to fine-tune for each client.

Distributional mismatch between the giant general-purpose corpus used to train the central service and the corpus from which a client's instances arise leads to lower accuracy. A common source of trouble is mismatch of word salience (Paik, 2013) between client and server corpora (Ruder, 2019). In this respect, our setting also presents a new opportunity. Clients are numerous and form natural clusters, e.g., healthcare, sports, politics. We want the service to exploit commonalities in existing client clusters, without explicitly supervising this space, and provide some level of generalization to new clients without re-training or fine-tuning.

In response to the above challenges and constraints, we initiate an investigation of practical protocols for lightweight client adaptation of NLP services. We propose a system, KYC ("Know Your Client"), in which each client registers with the service using a simple sketch derived from its (unlabeled) corpus. The service network takes the sketch as additional input with each instance later submitted by the client. The service provides accuracy benefits to new clients immediately.

What form can a client sketch take? How should the service network incorporate it? While this will depend on the task, we initiate a study of these twin problems focused on predictive language modeling, sentiment labeling, and named entity recognition (NER). We show that a simple late-stage intervention in the server network gives visible accuracy benefits, and provide diagnostic analyses and insights. Our code and data will be made public.

Contributions In summary, we

- introduce the on-the-fly client adaptation problem motivated by networked NLP API services;
- present KYC, that *learns to compute* client-specific biases from unlabeled client sketches;
- show improved accuracy for predictive language modeling, NER and sentiment labeling;
- diagnose why KYC's simple client-specific label biases succeed, in terms of relations between word salience, instance length and label distribu-

[1] Google NLP, Microsoft Azure, IBM Watson

tions at diverse clients.

Related work Our method addresses the mismatch between a client's data distribution and the server model. The extensive domain adaptation literature (DauméIII, 2007; Blitzer et al., 2006; Ben-David et al., 2006) is driven by the same goal but most of these update model parameters using labeled or unlabeled data from the target domain (client). Some recent approaches attempt to make the adaptation light-weight (Lin and Lu, 2018; Li et al., 2020; Jia et al., 2019; Cai and Wan, 2019; Liu et al., 2020) while others propose to use entity description (Bapna et al., 2017; Shah et al., 2019) for zero-shot adaptation. Domain generalization is another relevant technique (Chen and Cardie, 2018; Guo et al., 2018; Li et al., 2018a; Wang et al., 2019; Shankar et al., 2018; Carlucci et al., 2019; Dou et al., 2019; Piratla et al., 2020) where multiple domains during training are used to train a model that can generalize to new domains. Of these, the method that seems most relevant to our setting is the mixture of experts network of (Guo et al., 2018), with which we present empirical comparison. Another option is to transform the client data style so as to match the data distribution used to train the server model. Existing style transfer techniques (Yang et al., 2018; Shen et al., 2017; Prabhumoye et al., 2018; Fu et al., 2018; Lample et al., 2019; Li et al., 2018b; Gong et al., 2019) require access to server data distribution.

2 Proposed service protocol

We formalize the constraints on the server and client in the API setting. (1) The server is expected to scale to a large number of clients making it impractical to adapt to individual clients. (2) After registration, the server is expected to provide labeling immediately and response latency per instance must be kept low implying that the server's inference network cannot be too compute-intensive. (3) Finally, the client cannot perform complex pre-processing of every instance before sending to the server, and does not have any labelled data.

Server network and model These constraints lead us to design a server model that *learns to compute* client-specific model parameters from the client sketch, and requires no client-specific fine-tuning or parameter learning. The original server network is written as $\hat{\boldsymbol{y}} = Y_\theta(E_\theta(\boldsymbol{x}))$ where $\boldsymbol{x}$ is the input instance, and Y_θ is a softmax layer to get the predicted label $\hat{\boldsymbol{y}}$. E_θ is a representation learning layer that may take diverse forms depending on the task; of late, BERT (Devlin et al., 2018) is used to design E_θ for many tasks.

We augment the server network to accept, with each input $\boldsymbol{x}$, a client-specific sketch S_c as shown in Figure 1. We discuss possible forms of S_c in the next subsection. (The dotted arrow represents a generative influence of S_c on $\boldsymbol{x}$.) The server implements an auxiliary network $\boldsymbol{g} = G_\phi(S_c)$. Here $\boldsymbol{g}$ can be regarded as a neural digest of the client sketch. Module $\bigoplus$ combines $E_\theta(\boldsymbol{x})$ and $\boldsymbol{g}$; concatenation was found adequate on the tasks we evaluated but we also discuss other options in Section 3. When the $\bigoplus$ module is concatenation we are computing a client-specific per-label bias, and even that provides significant gains, as we show in Section 3.

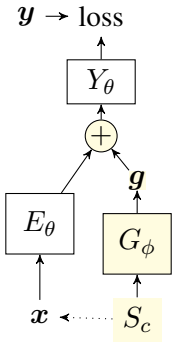

Figure 1: KYC overview.

Client sketch The design space of client sketch S_c is infinite. We initiate a study of designing S_c from the perspective of term weighting and salience in Information Retrieval (Paik, 2013). S_c needs to be computed once by each client, and thereafter reused with every input instance $\boldsymbol{x}$. Ideally, S_c and G_ϕ should be locality preserving, in the sense that clients with similar corpora and tasks should lead to similar $\boldsymbol{g}$s. Suppose the set of clients already registered is C.

A simple client sketch is just a vector of counts of all words in the client corpus. Suppose word w occurs $n_{c,w}$ times in a client c, with $\sum_w n_{c,w} = N_c$. Before input to G_ϕ, the server normalizes these counts using counts of other clients as follows: From all of C, the server will estimate a background unigram rate of word. Let the estimated rate for word w be p_w, which is calculated as:

$$p_w = \left(\sum_{c \in C} n_{c,w}\right) \Big/ \left(\sum_w \sum_{c \in C} n_{c,w}\right) . \quad (1)$$

The input into G_ϕ will encode, for each word w, how far the occurrence rate of w for client c deviates from the global estimate. Assuming the multinomial word event distribution, the marginal probability of having w occur $n_{c,w}$ times at client c is proportional to $p_w^{n_{c,w}}(1-p_w)^{(N_c - n_{c,w})}$. We finally pass a vector containing the normalized negative

	OOD			ID		
OOD Clients	Base	MoE	KYC	Base	MoE	KYC
BC/CCTV+Phoenix	63.8	66.9	71.8	86.0	83.8	86.7
BN/PRI+BN/VOA	88.7	87.9	90.7	84.5	83.0	86.0
NW/WSJ+Xinhua	73.9	78.9	80.9	80.8	77.2	82.5
BC/CNN+TC/CH	78.3	75.2	78.7	85.6	82.7	87.4
WB/Eng+WB/a2e	76.2	69.9	78.4	86.4	82.6	87.3
Average	76.2	75.8	**80.1**	84.7	81.9	**86.0**

Table 1: Test F1 on Ontonotes NER. OOD numbers are on the two listed domains whereas ID numbers are on test data of clients seen during training.

log probabilities as input to the model:

$$S_c \propto \Big(-n_{c,w} \log p_w - (N_c - n_{c,w}) \log(1 - p_w) : \forall w\Big). \quad (2)$$

We call this the **term-saliency** sketch. We discuss other sketches like TF-IDF and corpus-level statistics like average instance length in Sec. 3.

3 Experiments

We evaluate KYC on three NLP tasks as services: NER, sentiment classification, and auto-completion based on predictive language modeling. We compare KYC against the baseline model (without the G_ϕ network in Figure 1) and the mixture of experts (MoE) model (Guo et al., 2018) (see Appendix B). For all three models, the E_θ network is identical in structure. In KYC, G_ϕ has two linear layers with ReLU giving a 128-dim vector $\boldsymbol{g}$, with slight exceptions (see Appendix A). We choose datasets that are partitioned naturally across domains, used to simulate clients. We evaluate in two settings: in-distribution (ID) on test instances from clients seen during training, and out-of-distribution (OOD) on instances from unseen clients. For this, we perform a leave-k-client-out evaluation where given a set D of clients, we remove k clients as OOD test and use remaining $D - k$ as the training client set C.

Named Entity Recognition (NER) We use Ontonotes (Pradhan et al., 2007) which has 18 entity classes from 31 sources which forms our set D of clients. We perform leave-2-out test five times with 29 training clients as C. We train a cased BERT-based NER model (Devlin et al., 2018) and report F-scores. Table 1 shows that KYC provides substantial gains for OOD clients. For the first two OOD clients (BC/CCTV,Phoenix), the baseline F1 score jumps from 63.8 to 71.8. MoE performs worse than baseline. We conjecture this is because separate softmax parameters over the large NER label space is not efficiently learnable.

	OOD			ID		
OOD Clients	Base	MoE	KYC	Base	MoE	KYC
Electronics+Games	86.6	87.4	88.1	88.5	88.7	89.0
Industrial+Tools	87.4	88.3	87.6	88.2	88.8	88.9
Books+Kindle Store	83.5	84.6	84.1	88.0	88.8	88.9
CDs+Digital Music	82.5	83.0	83.2	89.0	88.9	89.0
Arts+Automotive	89.9	90.6	90.8	88.2	88.6	88.6
Average	86.0	86.8	86.8	88.4	88.8	**88.9**

Table 2: Test Accuracy on Amazon Sentiment Data.

Sentiment Classification We use the popular Amazon dataset (Ni et al., 2019) with each product genre simulating a client. We retain genres with more than 1000 positive and negative reviews each and randomly sample 1000 positive and negative reviews from these 22 genres. We perform leave-2-out evaluation five times and Table 2 shows the five OOD genre pairs. We use an uncased BERT model for classifcation (Sun et al., 2019). Table 2 shows that average OOD client accuracy increases from 86.0 to 86.8 with KYC.

Auto-complete Task We model this task as a forward language model and measure perplexity. We used the 20 NewsGroup dataset and treat each of the twenty topics as a client. Thus D is of size 20. We use the state-of-art Mogrifier LSTM (Melis et al., 2020). We perform leave-1-topic-out evaluation six times and OOD topics are shown in Table 3. For MoE, the client-specific parameter is only the bias and not the full softmax parameters which would blow up the number of trainable parameters. Also it did not perform well. Table 3 shows that

OOD	OOD			ID		
Clients	Base	MoE	KYC	Base	MoE	KYC
sci.space	29.3	30.9	28.7	28.6	30.7	28.1
comp.hw	27.9	28.6	27.2	28.4	28.7	27.9
sci.crypt	29.6	29.8	29.3	27.6	28.1	27.5
atheism	28.5	28.1	28.1	28.6	28.2	27.9
autos	28.2	28.4	28.0	27.7	28.0	27.9
mideast	26.8	26.7	26.2	28.4	28.9	27.3
Average	28.4	28.7	**27.9**	28.2	28.8	**27.8**

Table 3: Perplexity comparison between the baseline and KYC on 20-NewsGroup dataset.

KYC performs consistently better than the baseline with average perplexity drop from 28.4 to 27.9. This drop is particularly significant because the Mogrifier LSTM is a strong baseline to start with. MoE is worse than baseline.

Diagnostics We provide insights on why KYC's simple method of learning per-client label biases

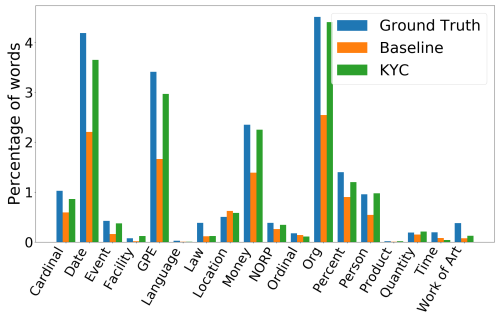

Figure 2: Proportion of true and predicted entity labels on OOD client NW/Xinhua. Similar trends observed on other OOD domains (Figure 4 of Appendix).

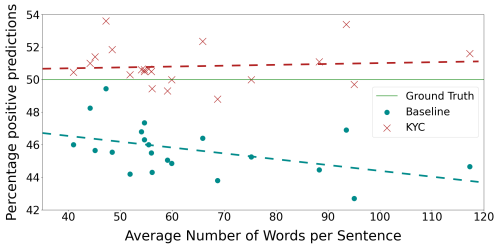

Figure 3: Fraction Positive Predicted versus average review length by baseline and KYC. Each dot/cross is a domain and the dotted lines indicate the best fit lines.

from client sketches is so effective. One explanation is that the baseline had large discrepancy between the true and predicted class proportions for several OOD clients. KYC corrects this discrepancy via *computed* per-client biases. Figure 2 shows true, baseline, and KYC predicted class proportions for one OOD client on NER. Observe how labels like `date`, `GPE`, `money` and `org` are underpredicted by baseline and corrected by KYC. Since KYC only corrects label biases, instances most impacted are those close to the shared decision boundary, and exhibiting properties correlated with labels but diverging across clients. We uncovered two such properties:

Ambiguous Tokens In NER the label of several tokens changes across clients, E.g. tokens like `million, billion` in finance clients like NW/Xinhua are `money` 92% of the times whereas in general only 50% of the times. Based on client sketches, it is easy to spot finance-related topics and increase the bias of `money` label. This helps KYC correct labels of borderline tokens.

Instance Length For sentiment labeling, review length is another such property. Figure 3 is a scatter plot of the average review length of a client versus the fraction predicted as positive by the baseline. For most clients, review length is clustered around the mean of 61, but four clients have length > 90. Length of review is correlated with label: on average, negative reviews contain 20 words more than positive ones. This causes baseline to underpredict positives on the few clients with longer reviews. The topics of the four outlying clients (video games, CDs, Toys&Games) are related so that the client sketch is able to shift the decision boundary to correct for this bias. Using only normalized average sentence length as the client sketch bridges part of the improvement of KYC over the baseline (details in Appendix C) implying that average instance length should be part of client sketch for classification tasks.

	Salience	TF	Binary	Sum-	Architecture		
	Concat	IDF	BOW	mary	Deep	Decomp	MoE-$\boldsymbol{g}$
OD	80.1	80.0	81.0	75.4	80.9	76.0	74.9
ID	86.0	85.9	77.8	81.8	85.9	85.0	79.8

Table 4: Comparing variant client sketches (S_c) and network architectures ($\bigoplus$ and Y_θ) of KYC in Fig 1.

Ablation Studies We explored a number of alternative client sketches and models for harnessing them. We present a summary here; details are in the Appendix C and D. Table 4 shows average F1 on NER for three other sketches: TF-IDF, Binary bag of words, and a 768-dim pooled BERT embedding of ten summary sentences extracted from client corpus (Barrios et al., 2016). KYC's default term saliency features provides best accuracy with TF-IDF a close second, and embedding-based sketches the worst. Next, we compare three other architectures for harnessing $\boldsymbol{g}$ in Table 4: **Deep**, where module $\bigoplus$ after concatenating $\boldsymbol{g}$ and E adds an additional non-linear layer so that now the whole decision boundary, and not just bias, is client-specific. KYC's OOD performance increases a bit over plain concat. **Decompose**, which mixes two softmax matrices with a client-specific weight α learned from $\boldsymbol{g}$. **MoE-$\boldsymbol{g}$**, which is like MoE but uses the client sketch for expert gating. We observe that the last two options are worse than KYC.

4 Conclusion

We introduced the problem of lightweight client adaption in NLP service settings. This is a promising area, ripe for further research on more complex tasks like translation. We proposed client sketches and KYC: an early prototype server network for on-the-fly adaptation. Three NLP tasks showed considerable benefits from simple, per-label bias correction. Alternative architectures and ablations provide additional insights.

References

Ankur Bapna, Gokhan Tür, Dilek Hakkani-Tür, and Larry Heck. 2017. Towards zero-shot frame semantic parsing for domain scaling.

Federico Barrios, Federico Lpez, Luis Argerich, and Rosa Wachenchauzer. 2016. Variations of the similarity function of textrank for automated summarization. *Proc. Argentine Symposium on Artificial Intelligence, ASAI*.

S. Ben-David, J. Blitzer, K. Crammer, and F. Pereira. 2006. Analysis of representations for domain adaptation. In *Proceedings of the 19th International Conference on Neural Information Processing Systems*, NIPS'06.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 conference on empirical methods in natural language processing*, pages 120–128. Association for Computational Linguistics.

Yitao Cai and Xiaojun Wan. 2019. Multi-domain sentiment classification based on domain-aware embedding and attention. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 4904–4910. AAAI Press.

Fabio M Carlucci, Antonio D'Innocente, Silvia Bucci, Barbara Caputo, and Tatiana Tommasi. 2019. Domain generalization by solving jigsaw puzzles. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2229–2238.

Xilun Chen and Claire Cardie. 2018. Multinomial adversarial networks for multi-domain text classification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1226–1240.

H. DauméIII. 2007. Frustratingly easy domain adaptation. pages 256–263.

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

Qi Dou, Daniel Coelho de Castro, Konstantinos Kamnitsas, and Ben Glocker. 2019. Domain generalization via model-agnostic learning of semantic features. In *Advances in Neural Information Processing Systems*, pages 6447–6458.

Zhenxin Fu, Xiaoye Tan, Nanyun Peng, Dongyan Zhao, and Rui Yan. 2018. Style transfer in text: Exploration and evaluation. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Hongyu Gong, Suma Bhat, Lingfei Wu, Jinjun Xiong, and Wen-mei Hwu. 2019. Reinforcement learning based text style transfer without parallel training corpus. *arXiv preprint arXiv:1903.10671*.

Jiang Guo, Darsh J. Shah, and Regina Barzilay. 2018. Multi-source domain adaptation with mixture of experts. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 4694–4703.

Chen Jia, Xiaobo Liang, and Yue Zhang. 2019. Cross-domain ner using cross-domain language modeling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2464–2474.

Guillaume Lample, Sandeep Subramanian, Eric Smith, Ludovic Denoyer, Marc'Aurelio Ranzato, and Y-Lan Boureau. 2019. Multiple-attribute text rewriting. In *International Conference on Learning Representations*.

Haoliang Li, Sinno Jialin Pan, Shiqi Wang, and Alex Chichung Kot. 2018a. Domain generalization with adversarial feature learning. *CVPR*.

Jing Li, Shuo Shang, and Ling Shao. 2020. Metaner: Named entity recognition with meta-learning. In *Proceedings of The Web Conference 2020*, pages 429–440.

Juncen Li, Robin Jia, He He, and Percy Liang. 2018b. Delete, retrieve, generate: A simple approach to sentiment and style transfer. *arXiv preprint arXiv:1804.06437*.

Bill Yuchen Lin and Wei Lu. 2018. Neural adaptation layers for cross-domain named entity recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2012–2022.

Zihan Liu, Genta Indra Winata, and Pascale Fung. 2020. Zero-resource cross-domain named entity recognition. *arXiv preprint arXiv:2002.05923*.

Gbor Melis, Tom Koisk, and Phil Blunsom. 2020. Mogrifier lstm. In *International Conference on Learning Representations*.

Jianmo Ni, Jiacheng Li, and Julian McAuley. 2019. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 188–197, Hong Kong, China. Association for Computational Linguistics.

Jiaul H Paik. 2013. A novel tf-idf weighting scheme for effective ranking. In *SIGIR Conference*, pages 343–352.

Vihari Piratla, Praneeth Netrapalli, and Sunita Sarawagi. 2020. Efficient domain generalization via common-specific low-rank decomposition. *arXiv preprint arXiv:2003.12815*.

Shrimai Prabhumoye, Yulia Tsvetkov, Ruslan Salakhutdinov, and Alan W Black. 2018. Style transfer through back-translation. *arXiv preprint arXiv:1804.09000*.

Sameer S. Pradhan, Eduard Hovy, Mitch Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2007. Ontonotes: A unified relational semantic representation. In *Proceedings of the International Conference on Semantic Computing*, ICSC 07, page 517526, USA. IEEE Computer Society.

Sebastian Ruder. 2019. *Neural Transfer Learning for Natural Language Processing*. Ph.D. thesis, National University of Ireland, Galway.

Darsh J Shah, Raghav Gupta, Amir A Fayazi, and Dilek Hakkani-Tur. 2019. Robust zero-shot cross-domain slot filling with example values. *arXiv preprint arXiv:1906.06870*.

Shiv Shankar, Vihari Piratla, Soumen Chakrabarti, Siddhartha Chaudhuri, Preethi Jyothi, and Sunita Sarawagi. 2018. Generalizing across domains via cross-gradient training. In *International Conference on Learning Representations*.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. In *Advances in neural information processing systems*, pages 6830–6841.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in nlp. *arXiv preprint arXiv:1906.02243*.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019. How to fine-tune bert for text classification? In *Chinese Computational Linguistics*, pages 194–206, Cham. Springer International Publishing.

Haohan Wang, Zexue He, Zachary C Lipton, and Eric P Xing. 2019. Learning robust representations by projecting superficial statistics out. *arXiv preprint arXiv:1903.06256*.

Zichao Yang, Zhiting Hu, Chris Dyer, Eric P Xing, and Taylor Berg-Kirkpatrick. 2018. Unsupervised text style transfer using language models as discriminators. In *Advances in Neural Information Processing Systems*, pages 7287–7298.

NLP Service APIs and Models for Efficient Registration of New Clients (Appendix)

A Reproducibility/Implementation Details

In this section we provide details about the dataset, architecture and training procedures used for each of the three tasks. We provide the datasets used, code, hyperparameters for all the tasks in the code submitted along with the submission.

A.1 NER

We use the standard splits provided in the Ontonotes dataset (Pradhan et al., 2007). Our codebase builds on the official PyTorch implementation released by (Devlin et al., 2018). We finetune a cased BERT base model with a maximum sequence length of 128 tokens for 3 epochs which takes 3 hours on a Titan X GPU.

A.2 Sentiment Classification

As described previously, we use the Amazon dataset (Ni et al., 2019). For each review, we use the standard protocol to convert the rating to a binary class label by marking reviews with 4 or 5 stars as positive, reviews with 1 or 2 stars as negative and leaving out reviews with 3 stars. We randomly sample data points from each domain to select 1000, 200 and 500 positive and negative reviews each for the train, validation and test splits, respectively. We leave out the domains that have insufficient examples, leaving us with 22 domains. We use the finetuning protocol provided by the authors of (Sun et al., 2019) and use the uncased BERT base model with a maximum sequence length of 256 for this task. We train for 5 epochs (which takes 4 hours on a Titan X GPU) and use the validation set accuracy after every epoch to select the best model.

A.3 Auto Complete Task

We use 20NewsGraoup dataset while regarding each content class label as a client. We remove header, footer from the content of the documents and truncate the size of each client to around 1MB. We use word based tokenizer with a vocabulary restricted to top 10,000 tokens and demarcate sentence after 50 tokens. The reported numbers in Table 3 are when using TF-IDF vector for domain sketch. We did not evaluate other kinds of domain sketch on this task. We train all the methods for 40 epochs with per epoch train time of 4 minutes on a Titan X GPU.

We adopt the tuned hyperparameters corresponding to PTB dataset to configure the baseline Melis et al. (2020). Since the salience information from the client sketch can be trivially exploited in perplexity reduction and thereby impede learning desired hypothesis beyond trivially copying the salience information, we project the sketch vector to a very small dimension of 32 before fanning it out to the size of vocabulary. We did not use any non-linearity in G_ϕ and also employ dropout on the sketches.

B Details of MoE method (Guo et al., 2018)

MoE employs a shared encoder and a client specific classifier. We implemented their proposal to work with our latest encoder networks. Our implementation of their method is to the best of our efforts faithful to their scheme. The only digression we made is in the design of discriminator: we use a learnable discriminator module that the encoder fools while they adopt MMD based metric to quantify and minimize divergence between clients. This should, in our opinion, only work towards their advantage since MMD is not sample efficient especially given the small size of our clients.

	OOD		ID	
OOD Clients	Base	KYC	Base	KYC
BC/CCTV + BC/Phoenix	63.8	70.1	86.00	86.7
BN/PRI + BN/VOA	88.7	91.6	84.5	86.2
NW/WSJ + NW/Xinhua	73.9	79.2	80.8	82.2
BC/CNN + TC/CH	78.3	80.4	85.6	87.1
WB/Eng + WB/a2e	76.2	78.9	86.4	87.5
Average	76.2	80.0	84.7	85.9

Table 5: Performance on the NER task on the Ontonotes dataset when using TF-IDF as the client sketch.

C Results with Different Client Sketches

In this section we provide results on every OOD split for the different client sketches described in Section 3 along with more details.

- **TF-IDF**: This is a standard vectorizer used in Information Retrieval community for document similarity. We regard all the data of the client as a

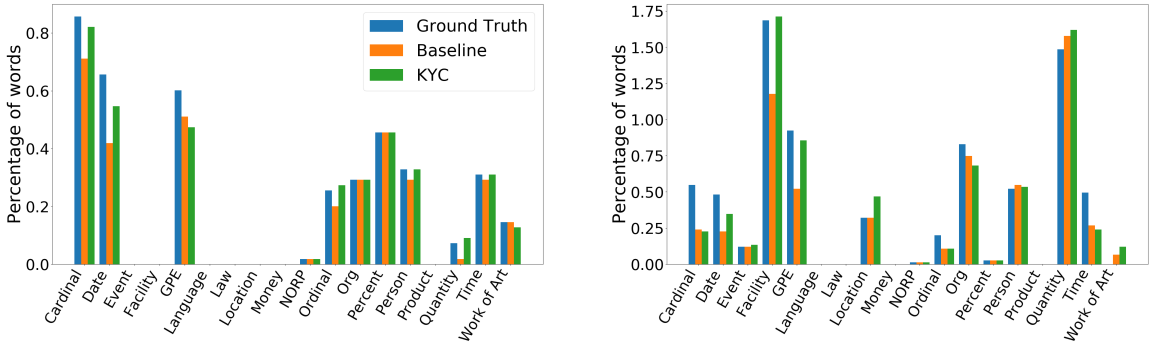

Figure 4: Proportion of true and predicted entity labels for different OOD clients (left) BC/Phoenix (right) BC/CCTV.

OOD Clients	OOD Base	OOD KYC	ID Base	ID KYC
BC/CCTV + BC/Phoenix	63.8	75.3	86.0	79.3
BN/PRI + BN/VOA	88.7	90.5	84.5	78.7
NW/WSJ + NW/Xinhua	73.9	82.7	80.8	71.4
BC/CNN + TC/CH	78.3	80.3	85.6	79.9
WB/Eng + WB/a2e	76.2	76.4	86.4	79.6
Average	76.2	81.0	84.7	77.8

Table 6: Performance on the NER task on the Ontonotes dataset when using Binary Bag of Words as the client sketch.

OOD Clients	OOD Base	OOD KYC	ID Base	ID KYC
BC/CCTV + BC/Phoenix	63.8	61.5	86.0	83.0
BN/PRI + BN/VOA	88.7	82.3	84.5	85.2
NW/WSJ + NW/Xinhua	73.9	82.3	80.8	75.0
BC/CNN + TC/CH	78.3	72.5	85.6	83.2
WB/Eng + WB/a2e	76.2	78.3	86.4	82.5
Average	76.2	75.4	84.7	81.8

Table 7: Performance on the NER task on the Ontonotes dataset when using sentence embddings averaged over an extracted summary.

OOD Clients	OOD Base	OOD Salience	OOD Avg Len	ID Base	ID Salience	ID Avg Len
Electronics+Games	86.4	88.1	86.9	88.5	89.0	88.6
Industrial+Tools	87.4	87.6	88.3	88.2	88.9	88.8
Books+Kindle Store	83.5	84.6	84.5	88.0	88.9	89.0
CDs+Digital Music	82.5	83.0	83.1	89.0	89.0	89.0
Arts+Automotive	89.9	90.6	90.2	88.2	88.6	88.5
Average	86.0	86.8	86.6	88.4	88.8	88.8

Table 8: Accuracy on the Sentiment Analysis task when using average review length as the client sketch. Columns "Saliency" and "Avg Len" refer to using KYC with the default saliency features and normalized review lengths as client sketches, respectively.

document when computing this vector. The corresponding numbers using this sketch are shown in Table 5 and are only slightly worse than the salience features.

- **Binary Bag of Words (BBoW)**: A binary vector of the same size as vocabulary is assigned to each client while setting the bit corresponding to a word on if the word has occurred in the client's data. We notice an improvement on the OOD set but a significant drop in ID numbers as seen in Table 6, 4. We attribute this to the strictly low representative power of BBoW sketches compared to the other sketches. The available train data for NER is laced with rogue clients which are not labeled and are instead assigned the default tag: "O". Proportion of KYC's improvement on this task comes from the ability to distinguish bad clients and keeping their parameters from not affecting other clients. This, however, is not possible when the representative capacity of the sketch is compromised. Thereby we do worse on ID using this sketch but not on OOD meaning the model does worse on the bad clients (which are only part of ID, and not OOD).
- **Contextualized Embedding of Summary**: We also experiment with using deep-learning based techniques to extract the topic and style of a client by using the "pooled" BERT embeddings averaged over sentences from the client. Since the large number of sentences from every client would lead to most useful signals being killed upon averaging, we first use a Summary Extractor (Barrios et al., 2016) to extract roughly 10 sentences per client and average the sentence embeddings over these sentences only. This method turns out to be ineffective in comparison to the other client sketches, indicating that sentence embeddings do not capture all the word-distribution

information needed to extract useful correction.

- **Average Instance Length:** For the task of Sentiment Analysis, we also experiment with passing a single scalar indicating average review length as the client sketch in order to better understand and quantify the importance of average review length on the performance of KYC. We linearly scale the average lengths so that all train clients have values in the range $[-1, 1]$. As can be seen in Table 8, this leads to a significant improvement over the baseline. In particular, the OOD splits CDs + Digital Music and Books + Kindle Store have reviews that are longer than the average and consequently result in improvements when augmented with average length information. The gains from review length alone are not higher than our default term-saliency sketch indicating that term frequency captures other meaningful properties as well.

D Results with Different Model Architectures

In this section we provide results for the different network architecture choices described in Section 3

- **Deep**: The architecture used is identical to that shown in Figure 1 barring $\bigoplus$, which now consists of an additional 128-dimensional non-linear layer before the final softmax transform Y_θ.
- **Decompose**: The final softmax layers is decomposed in to two. A scalar α is predicted from the client sketch using G_ϕ similar to KYC. The final softmax layer then is obtained through convex combination of the two softmax layers using α. Figure 5 shows the overview of the architecture.
- **MoE-$\boldsymbol{g}$**: We use the client sketch as the drop-in replacement for encoded instance representation employed in Guo et al. (2018). The architecture is sketched in Figure 6. As shown in Table 11, this method works better than the standard MoE model, but worse than KYC.

	OOD		ID	
OOD Clients	Base	KYC	Base	KYC
BC/CCTV + BC/Phoenix	64.8	74.5	85.6	86.8
BN/PRI + BN/VOA	89.5	90.0	84.1	85.6
NW/WSJ + NW/Xinhua	74.4	80.6	80.2	92.8
BC/CNN + TC/CH	78.0	79.6	86.1	87.5
WB/Eng + WB/a2e	75.6	79.9	85.8	87.1
Average	76.5	80.9	84.4	86.0

Table 9: Performance on the NER task on the Ontonotes dataset using KYC-Deep.

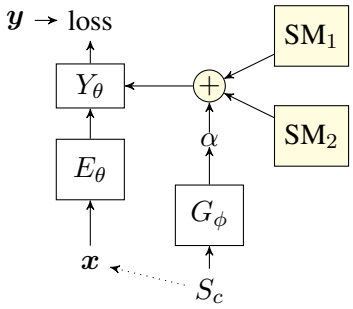

Figure 5: Decompose overview: $\bigoplus$ indicates a weighted linear combination. SM_i, $i \in \{1, 2\}$ represent the softmax matrices which are combined using weights α.

	OOD		ID	
OOD Clients	Base	KYC	Base	KYC
BC/CCTV + BC/Phoenix	64.1	56.0	85.6	86.3
BN/PRI + BN/VOA	89.6	89.9	84.6	85.5
NW/WSJ + NW/Xinhua	72.3	68.2	81.2	80.0
BC/CNN + TC/CH	78.5	77.5	85.9	86.6
WB/Eng + WB/a2e	75.5	71.0	86.1	86.7
Average	76.0	72.5	84.7	85.2

Table 10: Performance on the NER task on the Ontonotes dataset using Decompose.

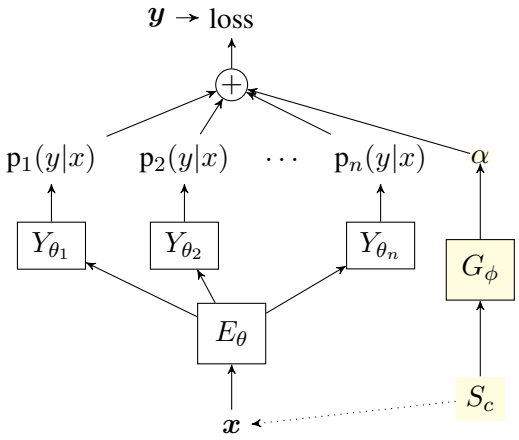

Figure 6: MoE-$\boldsymbol{g}$ overview: $\bigoplus$ indicates a weighted linear combination. $\text{p}_i(y|x)$ represents the i^{th} expert's predictions and α represents weights for expert gating.

	OOD		ID	
OOD Clients	Base	KYC	Base	KYC
BC/CCTV + BC/Phoenix	64.8	74.7	85.6	84.0
BN/PRI + BN/VOA	89.5	88.3	84.1	83.6
NW/WSJ + NW/Xinhua	74.4	61.6	80.2	64.8
BC/CNN + TC/CH	78.0	73.7	86.1	82.1
WB/Eng + WB/a2e	75.6	76.3	85.8	84.4
Average	76.5	74.9	84.4	79.8

Table 11: Performance on the NER task on the Ontonotes dataset using MoE-$\boldsymbol{g}$.