General-to-Specific Transfer Labeling for Domain Adaptable Keyphrase Generation

Rui Meng[1], **Tong Wang**[2], **Xingdi Yuan**[2], **Yingbo Zhou**[1], **Daqing He**[3]
[1]Salesforce Research, [2]Microsoft Research, Montréal, [3]University of Pittsburgh
ruimeng@salesforce.com

Abstract

Training keyphrase generation (KPG) models require a large amount of annotated data, which can be prohibitively expensive and often limited to specific domains. In this study, we first demonstrate that large distribution shifts among different domains severely hinder the transferability of KPG models. We then propose a three-stage pipeline, which gradually guides KPG models' learning focus from general syntactical features to domain-related semantics, in a data-efficient manner. With domain-general phrase pre-training, we pre-train Sequence-to-Sequence models with generic phrase annotations that are widely available on the web, which enables the models to generate phrases in a wide range of domains. The resulting model is then applied in the Transfer Labeling stage to produce domain-specific pseudo keyphrases, which help adapt models to a new domain. Finally, we fine-tune the model with limited data with true labels to fully adapt it to the target domain. Our experiment results show that the proposed process can produce good quality keyphrases in new domains and achieve consistent improvements after adaptation with limited in-domain annotated data[1][2].

1 Introduction

The last decade has seen major advances in deep neural networks and their applications in natural language processing. Particularly, the subarea of neural keyphrase generation (KPG) has made great progress with the aid of large language models (Lewis et al., 2020) and large-scale datasets (Meng et al., 2017a; Yuan et al., 2020a). Due to the high cost of data annotation, most, if not all, of the large-scale KPG datasets are constructed by scraping domain-specific data from the internet. For example, Meng et al. collected more than 500k scientific papers of which keyphrases are provided by paper authors. Gallina et al. crawled about 280k news articles from New York Times with editor-assigned keyphrases. Following Gururangan et al. (2020), we use "domain" to denote a distribution over language characterizing a given topic or genre. Specifically in KPG tasks, domains can be "computer science papers", "online forum articles", "news" etc.

Despite recent neural models can to some extent learn KPG skills from existing datasets (Meng et al., 2021a; Gallina et al., 2019; Yuan et al., 2020a), because most of these datasets are limited to a single domain, it remains unclear how the trained models can be transferred to new domains, especially in a real-world setting. Some existing studies claim their models demonstrate a certain degree of transferability across domains. For instance, Meng et al. show that models trained with scientific paper datasets can generate decent quality keyphrases from news articles, in a zero-shot manner. Xiong et al. present that training with open-domain web documents can improve the model's generalizability. However, there is a lack of systematic studies on domain transferring KPG, and thus the observations reported in prior works do not support a comprehensive understanding of this topic.

To investigate this question, we conduct an empirical study on how well KPG models can transfer across domains. We utilize commonly used KPG datasets covering four different domains (Science, News, Web, Q&A). We first show experiment results (§2.2) that suggest models trained with data in a specific domain do not generalize well to other domains, even in cases where they are initialized with pre-trained language models such as BART (Lewis et al., 2020). We also visualize the domain gaps among datasets by inspecting their phrase overlaps. Keyphrases often represent the specific knowledge of a domain and this may result in the failure of transferring models across domains.

[1]All code and datasets are available at https://github.com/memray/OpenNMT-kpg-release.

[2]The research was mostly accomplished when the first author was at the University of Pittsburgh.

Findings of the Association for Computational Linguistics: ACL 2023, pages 1602–1618
July 9-14, 2023

The empirical study motivates us to explore novel methods that can help models possess the ability of generating high quality keyphrases and more importantly, can quickly adapt to a new domain with limited amount of annotation. We propose a three-stage training pipeline, in which we gradually guide a KPG model's learning focus from general syntactical features to domain-specific information. First, we pre-train the model using community labeled phrases in Wikipedia (§3.1). Then, we use a novel self-training-based domain adaptation method, namely Transfer Labeling, to adapt the model to the new domain. Note this domain adaptation method does not require ground-truth labels, we leverage the model pre-trained in the previous stage to generate pseudo-labels for training itself. Finally, we use a limited amount of in-domain data with true annotations to fully adapt the model to the new domain. We report extensive experiment results and thorough analyses to demonstrate the effectiveness of the proposed methods.

2 Background and Motivation

2.1 Background

Keyphrase Generation (KPG) Typically, the task is to generate a set of keyphrases $P = \{p_1, \ldots, p_n\}$ given a source text t. Semantically, these phrases summarize and highlight important information contained in t, while syntactically, each keyphrase may consist of multiple words and serve a component of a sentence. Depending on a particular domain the source text belongs to (e.g., scientific paper, news) and downstream applications (e.g., article classification, information retrieval), the extent to which a phrase is important can vary, i.e. the criteria of keyphrase can be different in various datasets. Following Meng et al., we denote a keyphrase as *present* if it is a sub-string of the source text, or as *absent* otherwise. We adopt the `One2Seq` training paradigm (Yuan et al., 2020a). Given a source text t and a set of ground-truth keyphrases P, we concatenate all ground-truth keyphrases into a single string: <bos>p_1<sep>$\cdots$<sep>p_n<eos>, where <bos>, <sep>, and <eos> are special tokens. This string is paired with t to train a sequence-to-sequence model. We refer readers to (Meng et al., 2021a) for more details in common KPG practice.

2.2 Domain Gap in KPG Tasks

Previous studies have touched on how much KPG models can transfer their skills when applied across domains (Meng et al., 2017a; Xiong et al., 2019a), but not in a systematic way. In this subsection, we revisit this topic and try to ground our discussion with thorough empirical results. Specifically, we consider four broadly used datasets in the KPG community: `KP20k` (Meng et al., 2017a) contains scientific papers in computer science; `OpenKP` (Xiong et al., 2019a) is a collection of web documents; `KPTimes` (Gallina et al., 2019) contains a set of news articles; `StackEx` (Yuan et al., 2020a) are community-based Q&A posts collected from StackExchange. All the four datasets are large enough to train KPG models from scratch. At the same time, the documents in these datasets cover a wide spectrum of domains. We report statistics of these four datasets in appendix Table 7.

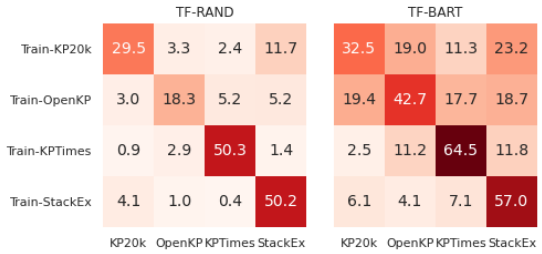

Figure 1: Cross-domain transfer performance of `TF-Rand` and `TF-Bart` (F@O, the higher the better). Y-axis: training dataset; X-axis: test dataset.

On the model dimension, we consider two model architectures: `TF-Rand`, a 6-layer encoder-decoder Transformer with random initialization (Vaswani et al., 2017); and `TF-Bart`, a 12-layer Transformer initialized with BART-large (Lewis et al., 2020). We train the two models on the four datasets individually and subsequently evaluate all the resulting eight checkpoints on the test split of each dataset. As shown in Figure 1, in-domain scores (i.e., trained and tested on the same datasets) are placed along the diagonal, the other elements represent cross-domain testing scores. We observe that both models exhibit a large gap between in-domain and out-of-domain performance. Even though the initialization with BART can alleviate the gap to a certain degree, the difference remains significant.

Keyphrases are typically concepts or entities that represent important information of a document. The collection of keyphrases in a domain can also be deemed as a representation of domain knowl-

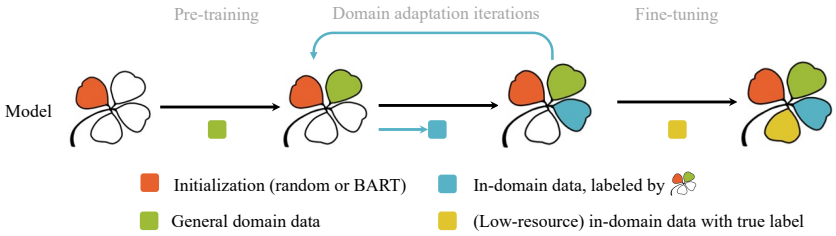

Figure 2: The proposed three-stage pipeline. A model is first pre-trained with general domain data and learns to generate syntactically correct phrases. In the domain adaptation stage, the model adapts to the target domain by training on domain-specific data, where the pseudo labels are generated by the model itself. Finally, we fine-tune the model with limited amount of target domain data with true label, to fully accomplish domain adaptation.

	KP20k	OpenKP	KPTimes	StackEx
KP20k	100	10.6	3.5	55.7
OpenKP	3.2	100	8.5	33.1
KPTimes	0.5	4.3	100	6.9
StackEx	0.7	1.3	0.5	100

Table 1: Overlap (%) of unique keyphrases between domains (train split). Numbers are normalized by dividing the diagonal element in its column. For example, the overlap keyphrases between KP20k and StackEx make a proportion of 0.7% in KP20k and a 55.7% in StackEx.

edge. Therefore, to better investigate the domain gaps, we further look into the keyphrase overlap between datasets. As shown in Table 1, only a small proportion of phrases are in common between the four domains. We provide a T-SNE visualization of a set of phrases sampled from these dataset in appendix Figure 8, the phrase clusters present clear domain gaps in their semantic space.

We hypothesize that the domain specific traits in annotated data make models difficult to learn keyphrase patterns in a domain-general sense. Furthermore, humans may label keyphrases under an application-oriented consideration and thus a one-size-fits-all standard for keyphrase annotation may not exist. For example, on StackExchange, users tend to assign common tags to better expose their questions to community experts, resulting in a small keyphrase vocabulary size. On the contrary, the topics are more specialized in scientific papers and authors would emphasize novel concepts in their studies. This may explain the large number of unique keyphrases found in KP20k.

2.3 Disentanglement of "Key" and "Phrase"

In §2.2, we empirically show that KPG models do not adequately transfer to out-of-domain data, even initialized with pre-trained language models. However, data annotation for every single domain or application does not seem practical either, due to the high cost and the potential need of domain-specific annotators. Inspired by some prior works, we attempt to disentangle the important properties of a keyphrase as *keyness* (Bondi and Scott, 2010; Gabrielatos, 2018) and *phraseness* (Tomokiyo and Hurst, 2003). We believe a proficient KPG model should generate outputs that satisfy both properties.

Keyness refers to the attribute that how well a phrase represents important information of a piece of text. The degree of keyness can be document dependent and domain dependent. For example, "cloud" is a common keyphrase in Computer Science papers, it is, in most cases, less likely to be important in Meteorology studies. Due to its high dependence on domain-specific information, we believe that the knowledge/notion of keyness is more likely to be acquired from in-domain data.

Phraseness, on the other hand, focuses more on the syntactical aspect. It denotes that given a short piece of text, without even taking into account its context, to what extent it can be grammatically functional as a meaningful unit. Although the majority of keyphrases in existing datasets are noun phrases (Chuang et al., 2012), they can present in variant grammatical forms in the real world (Sun et al., 2021). We believe that phraseness can be independent from domains and thus can be obtained from domain-general data.

3 Methodology

In the spirit of the motivation discussed above, we propose a three-stage training procedure in which a model gradually moves its focus from learning domain-general phraseness towards domain-specific keyness, and eventually adapts to a new domain with only limited amount of annotated data. An overview of the proposed pipeline is illustrated in Figure 2. First, with a Pre-Training stage (**PT**), the model is trained with domain-general data to learn phraseness (§3.1). Subsequently, in the Do-

main Adaption stage (**DA**), the model is exposed with *unlabeled* in-domain data. Within a few iterations, the model labels the data itself and use them to gradually adapt to the new domain (§3.2). Lastly, in the Fine-Tuning stage (**FT**), the model fully adapts itself to the new domain by leveraging a limited amount of in-domain data with true annotations (§3.3). In this section, we describe each of the three stages in detail.

3.1 Domain-General Phrase Pre-training

The first training stage aims to capture the phraseness in general, we leverage the Wikipedia data and community labeled phrases from the text. Wikipedia is an open-domain knowledge base that contains rich entity-centric annotations, its articles cover a wide spectrum of topics and domains and thus it has been extensively used as a resource of distant supervision for NLP tasks related to entities and knowledge (Ghaddar and Langlais, 2017; Yamada et al., 2020; Xiong et al., 2019b). In this work, we consider four types of markup patterns in Wikipedia text to form distant keyphrase labels:

- in-text phrases with special formatting (italic, boldface, and quotation marks);
- in-text phrases with wikilinks (denoting an entity in Wikipedia);
- "see also" phrases (denoting related entities);
- "categories" phrases (denoting superordinate entities).

Although the constructed targets using the above heuristics can be noisy if considering the keyness aspect, we show that they work sufficiently for training general phrase generation models.

Given a piece of Wikipedia text t and a set of community labeled phrases, we convert this data point to the format of `One2Seq` as described in §2.1. In practice, the number of phrases within t can be large and thus we sample a subset from them to form the target. We group all the phrases appear in t as present candidates, the rest (e.g., "see-also" and categories) are grouped as absent candidates. Additionally, we take several random spans from t as infilling candidates (similar as (Raffel et al., 2020)) for robustness. Finally, we sample a few candidates from each group and concatenate them as the final target sequence.

On the source side, we prepend a string suggesting the cardinality of phrases in each target group to the beginning of t. We also corrupt the source

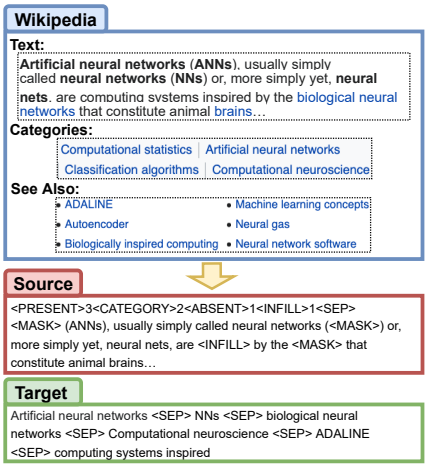

Figure 3: Illustration of processing Wikipedia to source-target pairs in domain general phrase pre-training.

sequence by replacing a small proportion of present and infilling phrases with a special token `[MASK]`, expecting to improve models' robustness (Raffel et al., 2020). We show an example of a processed Wikipedia data instance in Figure 3.

Trained with this data, we expect a model to become a general phrase generator — given a source text, the model can generate a sequence of phrases, regardless the specific domain a text belongs to.

3.2 Domain Adaption with Transfer Labeling

In the second stage, we aim to expose the model with data from a domain of interest, so it can learn the notion of domain-specific keyness. We propose a method, namely General-to-Specific Transfer Labeling , which does not require any in-domain annotated data. Transfer labeling can be considered as a special self-training method (Yarowsky, 1995; Culp and Michailidis, 2008; Mukherjee and Awadallah, 2020), where the key notion is to train a model with its own predictions iteratively.

Distinct from common practice of self-training where initial models are bootstrapped with annotated data, transfer labeling regards the domain-general model from the pre-training stage 3.1 as a qualified phrase predictor. We directly transfer the model to documents in a new domain to predict pseudo labels. The resulting phrases, paired with these documents, are used to tune the model so as to adapt it to the target domain distribution. Note that this process can be run iteratively, to gradually adapt models to target domains.

3.3 Low-resource Fine-Tuning

In the third stage, we expose the model to a small amount of in-domain data with annotated

keyphrases. This aims to help the model fully adapt to the new domain and reduce the bias caused by noisy labels from previous stages.

4 Experiments

We reuse the model architecture described in §2.2 throughout this paper. And most models apply a single iteration of transfer labeling. We discuss the effect of multi-iteration transfer labeling in §4.2.5. See Appendix A.1 for implementation details.

4.1 Datasets and Evaluation Metric

We consider the same four large-scale KPG datasets as described in §2.2, but instead of training models with all annotated document-keyphrases pairs, we take a large set of unannotated documents from each dataset for domain adaptation, and a small set of annotated examples for few-shot fine-tuning. Specifically, in the pre-training stage (**PT**), we use the 2021-05-21 release of English Wikipedia dump and process it with wikiextractor package, which results in 3,247,850 passages. In the domain adaptation stage (**DA**), for each domain, we take the first 100k examples from the training split (without keyphrases), and apply different strategies to produce pseudo labels and subsequently train the models. In the fine-tuning stage (**FT**), we take the first 100/1k/10k annotated examples (document-keyphrases pairs) from the training split to train the models. We report the statistics of used datasets in appendix Table 7.

We follow previous studies to split training/validation/test sets, and report model performance on test splits of each dataset. A common practice in KPG studies is to evaluate the model performance on present/absent keyphrases separately. However, the ratios of present/absent keyphrases differ drastically among the four datasets (e.g. `OpenKP` is strongly extraction-oriented). Since we aim to improve the model's out-of-domain performance in general regardless of the keyphrases being present or absent, we follow Bahuleyan and El Asri (2020) and simply evaluate present and absent keyphrases altogether. We report the F@O scores (Yuan et al., 2020a) between the generated keyphrases and the ground-truth. This metric requires systems to model the cardinality of predicted keyphrases themselves.

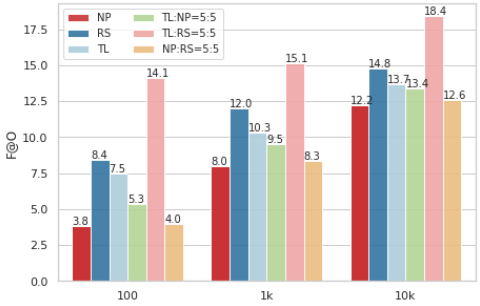

Figure 4: Comparison of different strategies for domain adaptation with `TF-Rand`. **TL**: Transfer Labeling. **NP**: Noun Phrases. **RS**: Random Span.

4.2 Results and Analyses

4.2.1 Zero-shot Performance

We first investigate how well models can perform after the pre-training stage, without utilizing any in-domain annotated data. Since Wikipedia articles contain a rather wide range of phrase types, we expect models trained on this data are capable of predicting relevant and well-formed phrases from documents in general. We show our models' testing scores in the first row of Table 2 and 3, where only **PT** is checked. We observe that pre-training with Wikipedia data can provide decent zero-shot performance in both settings, i.e., model is initialized randomly (Table 2) and with pre-trained language models (3). Both settings achieve the same average F@O score of 12.2, which evinces the feasibility of using **PT** model to generate pseudo labels for further domain adaptation. The scores also suggest that at the pre-training stage, the BART model (with pre-trained initialization and more parameters) does not present an advantage in comparison to a smaller model trained from scratch.

4.2.2 Domain Adaptation Strategies

We compare transfer labeling (**TL**, proposed in §3.2) with two unsupervised strategies: (1) Noun Phrase (**NP**) and (2) Random Span (**RS**). For NP, we employ SpaCy (Honnibal et al., 2020) to POS-tag source texts and extract noun phrases based on regular expressions. For RS, we follow Raffel et al. (2020), extracting random spans as targets and masking them in the source text. For TL, all pseudo phrases are generated by a **PT** model in a zero-shot manner (with greedy decoding).

As shown in Figure 4, in the single strategy setting, RS performs the best among the three strategies and TL follows. We speculate that RS models are trained to predict randomly masked spans based on their context, and this results in the best gener-

alization among the three. As for the NP strategy, since targets are only noun phrases appear in the source text, the models may have the risk of overfitting to recognize a subset of possible phrases. TL lies in between the two discussed strategies, the generated pseudo labels contain both present and absent phrases, and thanks to the **PT** model trained with Wikipedia data, the generated targets can contain many phrase types beyond noun phrases.

We further investigate the performance gap between RS and TL. On KP20k, the **PT** model can generate 5.1 present and 2.6 absent keyphrases on average. The generated pseudo labels, albeit of good quality, are always fixed during the training. This is due to the deterministic nature of the **PT** model, which may cause overfitting and limit the model's generalizability. In contrast, random spans in RS are dynamically generated, therefore a model can learn to generate different target phrases even the same documents appear multiple times during training. This motivates us to investigate if these strategies can be synergistic by combining them. As shown in Figure 4, we observe that combining TL and RS can lead to a significant improvement over all other strategies, indicating that these two strategies are somewhat complementary and thus can be used together in domain adaption. In the rest of the paper, we by default combine TL and RS in the domain adaptation stage, by taking equal amount of data from both sides, we discuss other mixing strategies in Appendix A.3.

It is worth noting that, if we apply domain adaptation with the TL+RS mixing strategy and evaluate models without any fine-tuning (2nd row in Table 2/3), we can observe a clear drop in the performance of randomly initialized model (Table 2). We believe it is because using random spans for targets worsens the phraseness of the predictions. BART initialized models, on the other hand, show robust performance against these noisy targets.

4.2.3 Performance in Low-Data Setting

As described in §4.1, we use 100/1k/10k in-domain examples with gold standard keyphrases to fine-tune the model. To investigate the necessity of the **PT** and **DA** stages given the **FT** stage, we conduct a set of ablation experiments, skipping some of the training stages in the full pipeline.

We start with discussing the results of randomly initialized models (Table 2). **FT-only**: in the case where models are only fine-tuned with a small subset of annotated examples, models perform rather poorly on all datasets, especially on KP20k and OpenKP, where more unique target phrases are involved. **DA+FT**: different from the previous setting, here all models are first trained with 100k pseudo labeled in-domain data points. We expect these pseudo labeled data to improve models on both phraseness and keyness dimensions. Indeed, we observe DA+FT leads to a large performance boost in almost all settings. This suggests the feasibility of leveraging unlabeled in-domain data using the proposed adaptation method (TL+RS). **PT+FT**: the pre-training stage provides a rather significant improvement in all settings, averaging over datasets and k-shot settings, PT+FT (23.8) nearly doubles the performance of DA+FT (12.6). This observation indicates that the large-scale pre-training with domain-general phrase data can be beneficial in various down-stream domains, which is consistent with prior studies for text generation pre-training. **PT+DA+FT**: we observe a further performance boost when both PT and DA stages are applied before FT. This to some extent verifies our design that PT and DA can guide the models to focus on different perspectives of KPG and thus work in an complementary manner.

We also investigate when the model is initialized with a pre-trained large language model, i.e., BART (Lewis et al., 2020). Due to space limit, we only report models' average scores (over the four datasets, and over the k-shot settings) in Table 3, we refer readers to appendix Table 9 for the full results. We observe that in the pipeline, the fine-tuning stage provides TF-Bart the most significant performance boost — the average score is tripled, compared to the 0-shot settings, even performing solely the fine-tuning stage. This may be because the BART model was trained on a much wider range of domains of data (compared to Wikipedia, which is already domain-general), so it may have already contained knowledge in our four testing domains. However, the auto-regressive pre-training of BART does not train particularly on the KPG task. This explains why it requires the BART model to fine-tune on KPG data to achieve higher performance. The above assumption can also be support by further observations in Table 3. Results suggest that the DA stage is not notably helpful to TF-Bart's scores, and the PT stage, on the other hand, seems to contribute to a better score. We believe this is because the quality difference between labels used in these two stages: PT uses

	PT	DA	FT	KP20k	OpenKP	KPTimes	StackEx	Avg
0-shot	x			15.0	10.0	9.1	14.8	12.2
	x	x		11.2	4.6	7.7	4.3	6.9
100-shot			x	0.5	0.2	2.4	5.1	2.1
		x	x	14.1	5.6	5.3	11.7	9.2
	x		x	14.5	20.1	22.6	13.0	17.6
	x	x	x	16.7	24.4	22.0	18.4	20.4
1k-shot			x	0.5	0.6	5.4	7.0	3.4
		x	x	15.0	8.6	8.9	15.4	12.0
	x		x	17.6	25.5	30.5	21.1	23.7
	x	x	x	19.7	28.0	30.7	26.3	26.2
10k-shot			x	3.4	1.5	19.2	20.8	11.3
		x	x	16.5	13.1	13.4	23.4	16.6
	x		x	20.6	30.6	38.6	31.4	30.3
	x	x	x	22.1	31.6	36.7	34.7	31.3
Avg			x	1.5	0.8	9.0	11.0	5.6
		x	x	15.2	9.1	9.2	16.8	12.6
	x		x	17.6	25.4	**30.6**	21.8	23.8
	x	x	x	**19.5**	**28.0**	29.8	**26.5**	**25.9**

Table 2: Zero-shot and low-data results obtained by TF-Rand. The best average score in each column is **boldfaced**.

	PT	DA	FT	Avg
0-shot	x			12.2
	x	x		12.0
Average of few-shot (100/1k/10k)			x	36.2
		x	x	36.3
	x		x	**36.6**
	x	x	x	36.1

Table 3: Zero-shot and low-data results of TF-Bart model. Full results are reported in appendix Table 9.

Model	DA Data	100-shot	1k-shot	10k-shot
TF-Rand	KP20k 100k	16.7	19.7	22.1
	MAG-CS 1m	16.8	19.4	21.8
	MAG-CS 12m	**17.6**	**20.4**	**22.8**
TF-Bart	KP20k 100k	22.2	25.3	28.4
	MAG-CS 1m	22.3	25.4	28.4
	MAG-CS 12m	**22.5**	**25.4**	**28.6**

Table 4: Average scores (over 4 datasets) with different amount of transfer labeled data for domain adaptation. All models are trained through three stages. The best score in each block is **boldfaced**.

community-labeled phrases (high phrase quality but domain-general) and DA uses labels generated by the model itself (no guarantee on phrase quality but closer to target domains). Since TF-Bart only needs specific knowledge about the KPG task, the PT stage can therefore be more helpful.

We run Wilcoxon signed-rank tests on the results of Table 2, and we find all differences between neighboring experiments (e.g. PT+FT vs. PT+DA+FT, both trained with KP20k and 10k-shot) are significant ($p < 0.05$). For Table 3, the improvement of PT+FT over the other three settings is also significant.

4.2.4 Scaling the Domain Adaptation

One advantage of self-labeling is the potential to leverage large scale unlabeled data in target domains. We also investigate this idea and build a large domain adaptation dataset by pairing an unlabeled dataset with pseudo labels produced by a **PT** model. To this end, we resort to the MAG (Microsoft Academic Graph) dataset (Sinha et al., 2015) and collect paper titles and abstracts from 12 million scientific papers in the domain of Computer Science, filtered by 'field of study'. The resulting subset MAG-CS is supposed to be in a domain close to KP20k, yet it may contain noisy data points due to errors in the MAG's data construction process. We follow the same experiment setting as reported in the above subsections, except that in the DA stage we either use 1 million or 12 million pseudo-labeled MAG data points for domain adaptation. We train the models with the PT+DA+FT pipeline and report models' scores on KP20k test split.

As shown in Table 4, compared to our default setting which uses 100k unlabeled KP20k data points for domain adaptation, larger scale domain adaptation data can indeed benefit model performance — models adapted with MAG-CS 12m documents show consistent improvements. However, the MAG-CS 1m data (still 10 times the size of KP20k) does not show clear evidence being helpful. We suspect the distribution gap between the domain adaptation data (i.e., MAG-CS) and the testing data (i.e., KP20k) may have caused the extra need of generalization. Therefore, the MAG-CS 12m data may represent a data distribution that has more overlap with KP20k and thus being more helpful. We also observe that models initialized with BART are more robust against such a distribution gap, on account of BART's pre-training with large scale of text in general domain.

4.2.5 Multi-iteration Domain Adaptation

Prior self-training studies have demonstrated the benefit of multi-iterations of label propagation (Triguero et al., 2015; Li et al., 2019). We conduct experiments to investigate its effects on KPG. Specifically, we first pre-train a TF-Rand model using Wikipedia data as in previous subsections. Then, we repeatedly perform the domain adaptation stage multiple times. In each iteration, the model produces pseudo labels from the in-domain documents and then train itself with this data. Finally, we fine-tune the model with 10k annotated data points, and report its test scores on KP20k. We consider two datasets, KP20k and MAG-CS 1m, as the in-domain data for domain adaptation. As illustrated in Figure 5, the TF-Rand model can gradually gain better test performance by iteratively performing domain adaptation using both datasets. Due to limited computing resources, we set the maximum number of iterations to 10. But the trend suggests that models may benefit from more DA iterations.

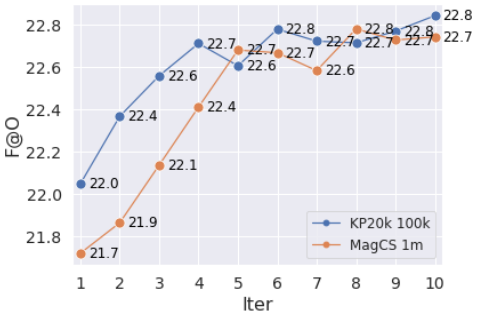

Figure 5: Trend of 10k-shot performance on KP20k with iterative self-labeling for 10 iterations.

5 Related Work

Keyphrase Generation. Meng et al. (2017b) first propose KPG, which enables models to generate keyphrases according to importance and semantics, rather than extracting sub-strings from the text (Witten et al., 1999; Liu et al., 2011; Wang et al., 2016). Following this idea, Chen et al. (2019); Wang et al. (2019); Liang et al. (2021) propose to leverage extra structure information (e.g., title, topic) to guide the generation. Chan et al. (2019); Luo et al. (2021) propose a model using reinforcement learning, and Swaminathan et al. (2020) propose using GAN for KPG. Ye et al. (2021) propose to dynamically align target phrases to eliminate the influence of target phrase order, a problem highlighted by Meng et al. (2021a). Mu et al. (2020); Liu et al. (2020); Park and Caragea (2020) use pre-trained language models for better representations of documents. In a similar vein, Ye and Wang utilize self-learning to generate synthetic phrases for data augmentation, whereas we use self-labeling for domain adapation. Gao et al. use a dense retriever to augment keyphrase generation in the cross-lingual scenario.

Pre-training for Phrase/Entity Understanding. Meng et al. (2021a) show that pre-train models with noisy annotation can deliver great improvements on KPG. Kulkarni et al. (2021) pre-train an understanding and a generation model with a large-scale annotated dataset OAGKX (Çano and Bojar, 2020) and the resulting models achieve decent performance on various NLP tasks. Both studies use a large amount of annotated data for pre-training, which is only available for certain domains. Wang et al. (2021); Li et al. (2022) use contrastive learning to train phrase encoders. Wang et al. (2021); Li et al. (2022) use contrastive learning to train phrase encoders. Lee et al. (2021) find open-domain QA datasets can be used to learn strong dense phrase representations. Wikipedia is also frequently used in training models for entity-centric and knowledge-rich tasks. (Yamada et al., 2020; Liu et al., 2021; Xiong et al., 2019b; Meng et al., 2021b; Huang et al., 2021) use Wikipedia and its related resources as distant supervision to enhance BERT's abilities on modeling entities.

Self-labeling. Self-labeling or self-training is a typical means for utilizing unannotated data and it has been applied in various machine learning tasks (He et al., 2019; Mukherjee and Awadallah, 2020). Yu et al. (2021) define rules as weak supervision for text classification and use self training to propagate labels to new documents. In our case, the pseudo labels are induced by models pre-trained with weak phrase annotation in Wikipedia. Liang et al. (2020) use self-training to supplement distantly supervised NER and Huang et al. (2021) use self-training to leverage unlabeled in-domain data.

6 Conclusion

In this study, we investigate domain gaps in the KPG task that hinder models from generalization. We attempt to alleviate this issue by proposing a three-stage pipeline to strategically enhance models' abilities on keyness and phraseness. Essentially, we consider phraseness as a domain-general property and can be acquired from Wikipedia data

as distant supervision. Then we use self-labeling to distill the phraseness into data in a new domain, and the resulting pseudo labels are used for domain adaptation, as the labels can reflect the keyness and phraseness of the new domain. Finally, we fine-tune the model with limited amount of target domain data with true labels. By taking the advantage of open-domain knowledge on the web, we believe this general-to-specific paradigm is generic and can be applied to a wide variety of machine learning tasks. As a next step, we plan to employ the proposed method for text classification and information retrieval, to see whether the domain-general phrase model can produce reliable class labels and queries for domain adaptation.

Limitations

In this study, we provide empirical evidence of the impact of domain gap in keyphrase tasks, and we propose effective methods to alleviate it. However, we acknowledge that this study is limited in the following aspects: (1) As the first study discussing domain adapation and few-shot results, there is few studies to refer to as fair baselines. Nevertheless, we attempt to show the improvements of the proposed methods over base models by extensive experiments. (2) The pretrained keyphrase generation model can be used off-the-shelf, but the multi-stage adaptation pipeline might increase the engineering complexity in practice. (3) We have only explored three strategies for domain adaptation, and they all require generating hard pseudo labels in different ways. Soft-labeling (Liang et al., 2020) and knowledge distillation (Zhou et al., 2021) methods are worth investigating. (4) We train a model with Wikipedia annotation to predict pseudo keyphrases, and it would be interesting to see if we can use large language models (e.g. GPT-3 (Brown et al., 2020)) to zero-shot predict phrases.

Ethics Statement

Dataset Biases The domain-general pseudo phrases were produced based on public web-scale data (Wikipedia), and it mainly represents the culture of the Englishspeaking populace. Political or gender biases may also exist in the dataset, and models trained on these datasets may propagate these biases. Additionally, the pretrained BART models can carry biases from the data it was pretrained on.

Environmental Cost The experiments described in the paper primarily make use of V100 GPUs. We typically used four GPUs per experiment, and the first-stage pretraining may take up to four days. The backbone model BART-LARGE 400 million parameters. While our work required extensive experiments, future work and applications can draw upon ourinsights and need not repeat these comparisons.

References

Hareesh Bahuleyan and Layla El Asri. 2020. Diverse keyphrase generation with neural unlikelihood training. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5271–5287.

Marina Bondi and Mike Scott. 2010. *Keyness in texts*, volume 41. John Benjamins Publishing.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Erion Çano and Ondřej Bojar. 2020. Two huge title and keyword generation corpora of research articles. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 6663–6671.

Hou Pong Chan, Wang Chen, Lu Wang, and Irwin King. 2019. Neural keyphrase generation via reinforcement learning with adaptive rewards. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2163–2174, Florence, Italy. Association for Computational Linguistics.

Wang Chen, Yifan Gao, Jiani Zhang, Irwin King, and Michael R. Lyu. 2019. Title-guided encoding for keyphrase generation. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 6268–6275. AAAI Press.

Jason Chuang, Christopher D Manning, and Jeffrey Heer. 2012. "without the clutter of unimportant words" descriptive keyphrases for text visualization. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 19(3):1–29.

Mark Culp and George Michailidis. 2008. An iterative algorithm for extending learners to a semi-supervised setting. *Journal of Computational and Graphical Statistics*, 17(3):545–571.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep

bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Costas Gabrielatos. 2018. Keyness analysis: Nature, metrics and techniques. In *Corpus approaches to discourse*, pages 225–258. Routledge.

Ygor Gallina, Florian Boudin, and Béatrice Daille. 2019. Kptimes: A large-scale dataset for keyphrase generation on news documents. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 130–135.

Yifan Gao, Qingyu Yin, Zheng Li, Rui Meng, Tong Zhao, Bing Yin, Irwin King, and Michael Lyu. 2022. Retrieval-augmented multilingual keyphrase generation with retriever-generator iterative training. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 1233–1246, Seattle, United States. Association for Computational Linguistics.

Abbas Ghaddar and Phillippe Langlais. 2017. WiNER: A Wikipedia annotated corpus for named entity recognition. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 413–422, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360.

Junxian He, Jiatao Gu, Jiajun Shen, and Marc'Aurelio Ranzato. 2019. Revisiting self-training for neural sequence generation. In *International Conference on Learning Representations*.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength Natural Language Processing in Python.

Jiaxin Huang, Chunyuan Li, Krishan Subudhi, Damien Jose, Shobana Balakrishnan, Weizhu Chen, Baolin Peng, Jianfeng Gao, and Jiawei Han. 2021. Few-shot named entity recognition: An empirical baseline study. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10408–10423.

Mayank Kulkarni, Debanjan Mahata, Ravneet Arora, and Rajarshi Bhowmik. 2021. Learning rich representation of keyphrases from text. *arXiv preprint arXiv:2112.08547*.

Jinhyuk Lee, Mujeen Sung, Jaewoo Kang, and Danqi Chen. 2021. Learning dense representations of phrases at scale. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6634–6647.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880.

Jiacheng Li, Jingbo Shang, and Julian McAuley. 2022. Uctopic: Unsupervised contrastive learning for phrase representations and topic mining. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6159–6169.

Xinzhe Li, Qianru Sun, Yaoyao Liu, Qin Zhou, Shibao Zheng, Tat-Seng Chua, and Bernt Schiele. 2019. Learning to self-train for semi-supervised few-shot classification. *Advances in Neural Information Processing Systems*, 32.

Chen Liang, Yue Yu, Haoming Jiang, Siawpeng Er, Ruijia Wang, Tuo Zhao, and Chao Zhang. 2020. Bond: Bert-assisted open-domain named entity recognition with distant supervision. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1054–1064.

Xinnian Liang, Shuangzhi Wu, Mu Li, and Zhoujun Li. 2021. Unsupervised keyphrase extraction by jointly modeling local and global context. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 155–164, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Rui Liu, Zheng Lin, Peng Fu, and Weiping Wang. 2020. Reinforced keyphrase generation with bert-based sentence scorer. In *2020 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)*, pages 1–8. IEEE.

Zhiyuan Liu, Xinxiong Chen, Yabin Zheng, and Maosong Sun. 2011. Automatic keyphrase extraction by bridging vocabulary gap. *the Fifteenth Conference on Computational Natural Language Learning*.

Zihan Liu, Feijun Jiang, Yuxiang Hu, Chen Shi, and Pascale Fung. 2021. Ner-bert: a pre-trained model for low-resource entity tagging. *arXiv preprint arXiv:2112.00405*.

Yichao Luo, Yige Xu, Jiacheng Ye, Xipeng Qiu, and Qi Zhang. 2021. Keyphrase generation with fine-grained evaluation-guided reinforcement learning. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 497–507, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Rui Meng, Xingdi Yuan, Tong Wang, Sanqiang Zhao, Adam Trischler, and Daqing He. 2021a. An empirical study on neural keyphrase generation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4985–5007, Online. Association for Computational Linguistics.

Rui Meng, Sanqiang Zhao, Shuguang Han, Daqing He, Peter Brusilovsky, and Yu Chi. 2017a. Deep keyphrase generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 582–592. Association for Computational Linguistics.

Rui Meng, Sanqiang Zhao, Shuguang Han, Daqing He, Peter Brusilovsky, and Yu Chi. 2017b. Deep keyphrase generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 582–592, Vancouver, Canada. Association for Computational Linguistics.

Yu Meng, Yunyi Zhang, Jiaxin Huang, Xuan Wang, Yu Zhang, Heng Ji, and Jiawei Han. 2021b. Distantly-supervised named entity recognition with noise-robust learning and language model augmented self-training. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10367–10378.

Funan Mu, Zhenting Yu, LiFeng Wang, Yequan Wang, Qingyu Yin, Yibo Sun, Liqun Liu, Teng Ma, Jing Tang, and Xing Zhou. 2020. Keyphrase extraction with span-based feature representations. *arXiv preprint arXiv:2002.05407*.

Subhabrata Mukherjee and Ahmed Awadallah. 2020. Uncertainty-aware self-training for few-shot text classification. *Advances in Neural Information Processing Systems*, 33:21199–21212.

Seoyeon Park and Cornelia Caragea. 2020. Scientific keyphrase identification and classification by pre-trained language models intermediate task transfer learning. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5409–5419, Barcelona, Spain (Online). International Committee on Computational Linguistics.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67.

Arnab Sinha, Zhihong Shen, Yang Song, Hao Ma, Darrin Eide, Bo-June Hsu, and Kuansan Wang. 2015. An overview of microsoft academic service (mas) and applications. In *Proceedings of the 24th international conference on world wide web*, pages 243–246.

Si Sun, Zhenghao Liu, Chenyan Xiong, Zhiyuan Liu, and Jie Bao. 2021. Capturing global informativeness in open domain keyphrase extraction. In *CCF International Conference on Natural Language Processing and Chinese Computing*, pages 275–287. Springer.

Avinash Swaminathan, Haimin Zhang, Debanjan Mahata, Rakesh Gosangi, Rajiv Shah, and Amanda Stent. 2020. A preliminary exploration of gans for keyphrase generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8021–8030.

Takashi Tomokiyo and Matthew Hurst. 2003. A language model approach to keyphrase extraction. In *Proceedings of the ACL 2003 workshop on Multiword expressions: analysis, acquisition and treatment*, pages 33–40.

Isaac Triguero, Salvador García, and Francisco Herrera. 2015. Self-labeled techniques for semi-supervised learning: taxonomy, software and empirical study. *Knowledge and Information systems*, 42(2):245–284.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Minmei Wang, Bo Zhao, and Yihua Huang. 2016. Ptr: Phrase-based topical ranking for automatic keyphrase extraction in scientific publications. *23rd International Conference, ICONIP 2016*.

Shufan Wang, Laure Thompson, and Mohit Iyyer. 2021. Phrase-bert: Improved phrase embeddings from bert with an application to corpus exploration. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10837–10851.

Yue Wang, Jing Li, Hou Pong Chan, Irwin King, Michael R. Lyu, and Shuming Shi. 2019. Topic-aware neural keyphrase generation for social media language. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2516–2526, Florence, Italy. Association for Computational Linguistics.

Ian H. Witten, Gordon W. Paynter, Eibe Frank, Carl Gutwin, and Craig G. Nevill-Manning. 1999. Kea:

Practical automatic keyphrase extraction. In *Proceedings of the Fourth ACM Conference on Digital Libraries*, DL '99, pages 254–255, New York, NY, USA. ACM.

Lee Xiong, Chuan Hu, Chenyan Xiong, Daniel Campos, and Arnold Overwijk. 2019a. Open domain web keyphrase extraction beyond language modeling. 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 5175–5184.

Wenhan Xiong, Jingfei Du, William Yang Wang, and Veselin Stoyanov. 2019b. Pretrained encyclopedia: Weakly supervised knowledge-pretrained language model. In *International Conference on Learning Representations*.

Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. 2020. Luke: Deep contextualized entity representations with entity-aware self-attention. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6442–6454.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *33rd annual meeting of the association for computational linguistics*, pages 189–196.

Hai Ye and Lu Wang. 2018. Semi-supervised learning for neural keyphrase generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4142–4153, Brussels, Belgium. Association for Computational Linguistics.

Jiacheng Ye, Tao Gui, Yichao Luo, Yige Xu, and Qi Zhang. 2021. One2Set: Generating diverse keyphrases as a set. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4598–4608, Online. Association for Computational Linguistics.

Yue Yu, Simiao Zuo, Haoming Jiang, Wendi Ren, Tuo Zhao, and Chao Zhang. 2021. Fine-tuning pretrained language model with weak supervision: A contrastive-regularized self-training approach. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1063–1077.

Xingdi Yuan, Tong Wang, Rui Meng, Khushboo Thaker, Peter Brusilovsky, Daqing He, and Adam Trischler. 2020a. One size does not fit all: Generating and evaluating variable number of keyphrases. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7961–7975.

Xingdi Yuan, Tong Wang, Rui Meng, Khushboo Thaker, Peter Brusilovsky, Daqing He, and Adam Trischler. 2020b. One size does not fit all: Generating and evaluating variable number of keyphrases. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7961–7975, Online. Association for Computational Linguistics.

Xuan Zhou, Xiao Zhang, Chenyang Tao, Junya Chen, Bing Xu, Wei Wang, and Jing Xiao. 2021. Multi-grained knowledge distillation for named entity recognition. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5704–5716.

A Appendix

A.1 Implementation Details

Most experiments make use of four V100 GPUs. We elaborate the training hyper-parameters for reproducing our results in Table 5 and 6. For inference, we follow previous studies (Yuan et al., 2020b; Meng et al., 2021a) that uses beam search to produce multiple keyphrase predictions (beam width of 50, max length of 40 tokens). We report testing scores with best checkpoints, which achieve best performance on valid set (2,000 data instances for all domains).

Phrase masking ratio denotes for $p\%$ of target phrases, replacing their appearances in the source text with a special token `[PRESENT]`.

Random span ratio denotes replacing $p\%$ of words in the source text with a special token `[MASK]`.

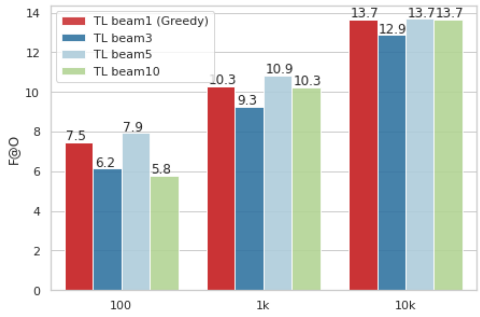

Figure 6: Comparison of domain adaptation with Transfer Labeling using different beam width to produce pseudo labels (`TF-Rand`).

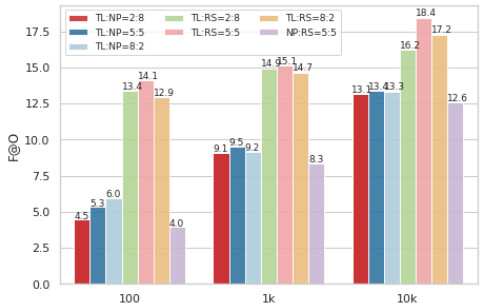

Figure 7: More results on mixing techniques for domain adaptation (`TF-Rand`, with different mixing ratios). **TL**: Transfer Labeling. **NP**: Noun Phrases. **RS**: Random Span.

A.2 Data Statistics

See Table 7.

A.3 Additional Results and Analyses

Figure 6 and 7 show additional results of domain adaptation. In Figure 6, we find that larger beam widths do not lead to significantly better scores after fine-tuning and thus we use simple greedy decoding for most of this study. In Figure 7, we compare various domain adapation strategies of mixing different pseudo labels. Overall, we find that mixing labels of transfer labeling (**TL**) and random spans (**RS**) by 50%:50% leads to best performance.

In Figure 8, we use T-SNE to visualize 1,000 most frequent keyphrases from each of four datasets (100k data examples from the training split) in the semantic space. We use BERT-base (Devlin et al., 2019) to generate phrase embeddings (we feed forward each phrase independently as a sequence and take the `[CLS]` embedding as output). We use the T-SNE of Scikit-Learn (Pedregosa et al., 2011) with default hyperparameters. The result shows that phrases from each domain tend to gather into clusters. Particularly, we can see that a big overlap between `KP20k` and `StackEx` since both domains are related to Computer Science. The distribution of `OpenKP` is more spread out, as its documents are collected from the web and cover a broader range of topics.

We present the full results of `TF-Rand` and `TF-Bart` in Table 8 and 9. Besides, we supplement the evaluation with another two popular datasets: `JPTimes` (for models trained in the `JPTimes` domain) and `DUC-2001` (for models trained in the `OpenKP` domain).

Hparam	PT	DA	PT+DA/PT+DA$_{\texttt{MAG-CS}}$	FT 100/1k/10k	*FT 100/1k/10k
Max source length	512				
Max target length	128				
Max phrase number	16	8			
Max phrase length	16	8			
Phrase masking rate	0.1				
Random span ratio	0.05				
Batch size	≈80	100	100	100	100
Learning rate	3e-4	3e-4	1e-5	3e-4	1e-5
Number of steps	200k	40k	20k/200k	2k/4k/8k	1k/2k/4k
Warmup steps	10%				
Learning rate decay	linear				
Optimizer	Adam				
Adam β_1	0.9				
Adam β_2	0.998				
Adam epsilon	1e-6				
Max gradient norm	2.0	2.0	1.0	1.0	1.0
Dropout	0.1				
BPE Dropout	0.1	0.0	0.0	0.0	0.0
Label smoothing	0.1				
Save checkpoint freq	ending step	ending step	ending step	100/200/400	50/100/200

Table 5: Training hyperparameters for TF-Rand. *FT denotes the fine-tuning stage in cases of PT+FT or PT+DA+FT. Empty cell means it is the same as the leftmost value.

Hparam	PT	DA	PT+DA/PT+DA$_{\texttt{MAG-CS}}$	FT 100/1k/10k
Max source length	512			
Max target length	256	256	256	128
Max phrase number	16			
Max phrase length	6	8	8	8
Phrase masking rate	0.1			
Random span ratio	0.05			
Batch size	256	256	256	16
Learning rate	1e-5			
Number of steps	40k	5k	5k/20k	2k/4k/8k
Warmup steps	2.4k	300	300/1.2k	200/400/800
Learning rate decay	linear			
Optimizer	Adam			
Adam β_1	0.9			
Adam β_2	0.98	0.98	0.98	0.999
Adam epsilon	1e-8			
Weight decay	0.01			
Max gradient norm	1.0	-	-	0.1
Dropout	0.1			
Label smoothing	0.1			
Save checkpoint freq	ending step	ending step	100/200/400	50/100/200

Table 6: Training hyperparameters for TF-Bart. Empty cell means it is the same as the leftmost value.

	#doc	#words in doc	#kp	#unique kp	#kp per doc	#uni kp per doc	#present kp per doc	#absent kp per doc
Training Sets								
Wikipedia	3.2m	-	-	-	-	-	-	-
KP20k	514.2k	161	2.7m	680.1k	5.3	1.3	3.3	1.9
OpenKP	134.9k	1104	294.2k	206.8k	2.2	1.5	2.1	0.0
KPTimes	259.9k	803	1.3m	104.8k	5.0	0.4	2.4	2.6
StackEx	299.0k	207	803.9k	8.1k	2.7	0.0	1.6	1.1
MAG-CS 1M	1.0m	151	9.6m	1.7m	9.6	1.7	3.4	6.2
MAG-CS 12M†	12.1m	151	115.9m	14.3m	9.6	1.2	3.4	6.2
Test Sets								
KP20k	19,987	161	105.2k	55.9k	5.3	2.8	3.3	1.9
OpenKP	6,614	894	14.6k	13.6k	2.2	2.0	2.0	0.2
KPTimes	10,000	804	50.4k	13.9k	5.0	1.4	2.4	2.6
StackEx	16,000	205	43.1k	4.5k	2.7	0.3	1.6	1.1
JPTimes	10,000	517	50.3k	9.0k	5.0	0.9	4.0	1.0
DUC-2001	308	701	2.5k	1.8k	8.1	6.0	7.9	0.2

Table 7: Statistics of training/testing datasets used in this study. †Only 7.7m papers in MAG-CS 12M have keyphrases.

	PT	DA	FT	KP20k	OpenKP	KPTimes	StackEx	Average over 4	JPTimes	DUC-2001
0-shot	x			15.0	10.0	9.1	14.8	12.2	15.8	9.4
	x	x		11.2	4.6	7.7	4.3	6.9	12.7	6.6
100-shot			x	0.5	0.2	2.4	5.1	2.1	2.4	0.2
		x	x	14.1	5.6	5.3	11.7	9.2	7.6	4.0
	x		x	14.5	20.1	22.6	13.0	17.6	24.1	20.5
	x	x	x	16.7	24.4	22.0	18.4	20.4	24.2	20.3
1k-shot			x	0.5	0.6	5.4	7.0	3.4	2.0	0.6
		x	x	15.0	8.6	8.9	15.4	12.0	9.0	4.8
	x		x	17.6	25.5	30.5	21.1	23.7	25.8	20.6
	x	x	x	19.7	28.0	30.7	26.3	26.2	26.1	22.5
10k-shot			x	3.4	1.5	19.2	20.8	11.3	8.5	0.7
		x	x	16.5	13.1	13.4	23.4	16.6	9.6	6.4
	x		x	20.6	30.6	38.6	31.4	30.3	25.7	24.3
	x	x	x	22.1	31.6	36.7	34.7	31.3	27.1	23.6
Avg			x	1.5	0.8	9.0	11.0	5.6	4.3	0.5
		x	x	15.2	9.1	9.2	16.8	12.6	8.7	5.1
	x		x	17.6	25.4	**30.6**	21.8	23.8	25.2	21.8
	x	x	x	**19.5**	**28.0**	29.8	**26.5**	**25.9**	**25.8**	**22.1**

Table 8: Zero-shot and low-data results. Models are randomly initialized. The best average score is boldfaced.

	PT	DA	FT	KP20k	OpenKP	KPTimes	StackEx	Average over 4	JPTimes	DUC-2001
0-shot	x			14.7	9.7	10.5	13.9	12.2	16.3	9.8
	x	x		13.8	10.7	12.0	11.5	12.0	17.5	11.6
100-shot			x	22.3	32.8	31.6	29.6	29.1	27.9	16.6
		x	x	22.5	33.3	32.0	29.2	29.3	28.7	20.7
	x		x	22.4	33.7	31.6	31.1	29.7	28.4	21.5
	x	x	x	22.2	32.0	31.6	29.7	28.9	28.4	21.5
1k-shot			x	25.1	36.4	43.6	41.1	36.5	33.2	21.1
		x	x	25.3	36.2	43.2	40.9	36.4	31.8	21.0
	x		x	24.9	36.9	44.3	41.2	36.8	34.0	22.7
	x	x	x	25.3	36.5	42.9	40.1	36.2	31.9	22.1
10k-shot			x	28.2	40.8	53.3	49.3	42.9	34.4	23.2
		x	x	28.0	41.5	53.4	49.6	43.1	34.5	25.0
	x		x	28.2	41.3	53.4	49.7	43.1	34.2	25.0
	x	x	x	28.4	41.2	53.2	49.8	43.1	34.9	25.6
Avg			x	25.2	36.6	42.8	40.0	36.2	31.8	20.3
		x	x	**25.3**	37.0	42.9	39.9	36.3	31.7	22.2
	x		x	25.2	**37.3**	**43.1**	**40.7**	**36.6**	**32.2**	23.0
	x	x	x	**25.3**	36.6	42.6	39.9	36.1	31.8	**23.1**

Table 9: Zero-shot and Few-shot results. Models are initialized with BART-large (Lewis et al., 2020). The best average score is boldfaced.

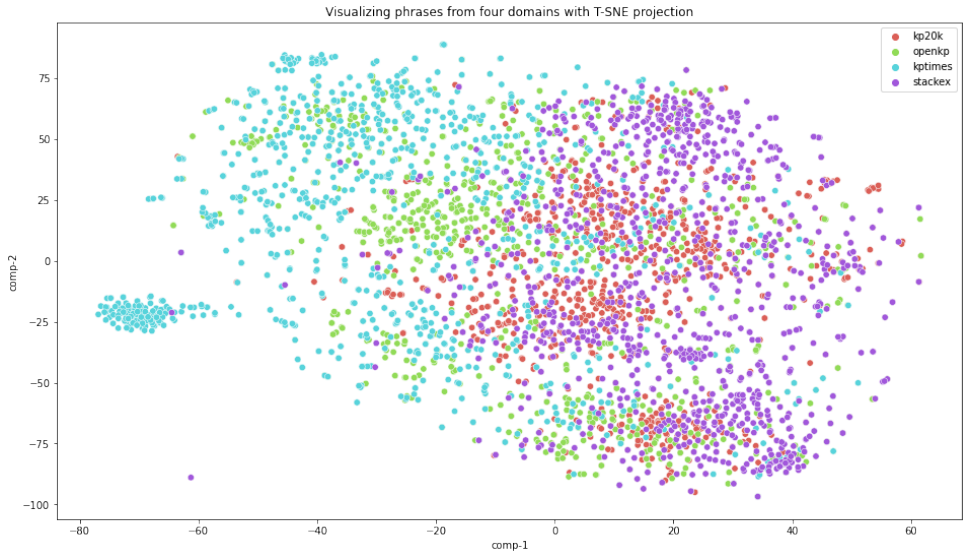

Figure 8: T-SNE visualization of keyphrase representations from four datasets.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
Sec 7

☐ A2. Did you discuss any potential risks of your work?
Not applicable. Left blank.

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Sec 1

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

Code will be open-sourced under MIT License

☑ B1. Did you cite the creators of artifacts you used?
Sec 4.1

☒ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Code will be open-sourced under MIT License

☒ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Code will be open-sourced under MIT License

☐ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
Not applicable. Left blank.

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Sec 2.2 and A.2

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Sec 2.2 and A.2

C ☑ Did you run computational experiments?

Sec 4

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
All models are based on public checkpoints. Computational budget was not recorded and multiple configs of infrastructure have been used. But all codes are public.

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Sec A.1

☒ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Just a single run due to computational constraints, but we experimented with various models/datasets.

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Sec 4.1 and A.1

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.