

Measuring Fine-Grained Domain Relevance of Terms: A Hierarchical Core-Fringe Approach

Jie Huang^{1,3} Kevin Chen-Chuan Chang^{1,3} Jinjun Xiong^{2,3} Wen-mei Hwu^{1,3}

¹University of Illinois at Urbana-Champaign, USA

²IBM Thomas J. Watson Research Center, USA

³IBM-Illinois Center for Cognitive Computing Systems Research (C3SR), USA

{jeffhj, kcchang, w-hwu}@illinois.edu
jinjun@us.ibm.com

Abstract

We propose to measure *fine-grained domain relevance*—the degree that a term is relevant to a broad (e.g., computer science) or narrow (e.g., deep learning) domain. Such measurement is crucial for many downstream tasks in natural language processing. To handle *long-tail terms*, we build a *core-anchored semantic graph*, which uses *core terms* with rich description information to bridge the vast remaining *fringe terms* semantically. To support a *fine-grained domain* without relying on a matching corpus for supervision, we develop *hierarchical core-fringe learning*, which learns core and fringe terms jointly in a semi-supervised manner contextualized in the hierarchy of the domain. To reduce *expensive human efforts*, we employ *automatic annotation* and *hierarchical positive-unlabeled learning*. Our approach applies to big or small domains, covers head or tail terms, and requires little human effort. Extensive experiments demonstrate that our methods outperform strong baselines and even surpass professional human performance.¹

1 Introduction

With countless terms in human languages, no one can know all terms, especially those belonging to a technical domain. Even for domain experts, it is quite challenging to identify all terms in the domains they are specialized in. However, recognizing and understanding domain-relevant terms is the basis to master domain knowledge. And having a sense of domains that terms are relevant to is an initial and crucial step for term understanding.

In this paper, as our **problem**, we propose to measure *fine-grained domain relevance*, which is defined as the degree that a term is relevant to a

given domain, and the given domain can be broad or narrow— an important property of terms that has not been carefully studied before. E.g., *deep learning* is a term relevant to the domains of computer science and, more specifically, machine learning, but not so much to others like database or compiler. Thus, it has a high domain relevance for the former domains but a low one for the latter. From another perspective, we propose to decouple extraction and evaluation in automatic term extraction that aims to extract domain-specific terms from texts (Amjadian et al., 2018; Hätyy et al., 2020). This decoupling setting is novel and useful because it is not limited to broad domains where a domain-specific corpus is available, and also does not require terms must appear in the corpus.

A good command of domain relevance of terms will facilitate many downstream applications. E.g., to build a domain taxonomy or ontology, a crucial step is to acquire relevant terms (Al-Aswadi et al., 2019; Shang et al., 2020). Also, it can provide or filter necessary candidate terms for domain-focused natural language tasks (Huang et al., 2020). In addition, for text classification and recommendation, the domain relevance of a document can be measured by that of its terms.

We aim to measure fine-grained domain relevance as a semantic property of any term in human languages. Therefore, to be practical, the proposed model for domain relevance measuring must meet the following **requirements**: 1) covering almost all terms in human languages; 2) applying to a wide range of broad and narrow domains; and 3) relying on little or no human annotation.

However, among countless terms, only some of them are popular ones organized and associated with rich information on the Web, e.g., Wikipedia pages, which we can leverage to characterize the domain relevance of such “head terms.” In contrast, there are numerous “long-tail terms”— those not as

¹The code and data, along with several term lists with domain relevance scores produced by our methods are available at <https://github.com/jeffhj/domain-relevance>.

frequently used— which lack descriptive information. As **Challenge 1**, how to measure the domain relevance for such *long-tail terms*?

On the other hand, among possible domains of interest, only those broad ones (e.g., physics, computer science) naturally have domain-specific corpora. Many existing works (Velardi et al., 2001; Amjadian et al., 2018; Hätyy et al., 2020) have relied on such domain-specific corpora to identify domain-specific terms by contrasting their distributions to general ones. In contrast, those fine-grained domains (e.g., quantum mechanics, deep learning)— which can be any topics of interest— do not usually have a matching corpus. As **Challenge 2**, how to achieve good performance for a *fine-grained domain* without assuming a domain-specific corpus?

Finally, automatic learning usually requires large amounts of training data. Since there are countless terms and plentiful domains, human annotation is very time-consuming and laborious. As **Challenge 3**, how to reduce *expensive human efforts* when applying machine learning methods to our problem?

As our solutions, we propose a hierarchical core-fringe domain relevance learning approach that addresses these challenges. **First**, to deal with long-tail terms, we design the *core-anchored semantic graph*, which includes *core terms* which have rich description and *fringe terms* without that information. Based on this graph, we can bridge the domain relevance through term relevance and include any term in evaluation. **Second**, to leverage the graph and support fine-grained domains without relying on domain-specific corpora, we propose *hierarchical core-fringe learning*, which learns the domain relevance of core and fringe terms jointly in a semi-supervised manner contextualized in the hierarchy of the domain. **Third**, to reduce human effort, we employ *automatic annotation* and *hierarchical positive-unlabeled learning*, which allow to train our model with little even no human effort.

Overall, our framework consists of two processes: 1) the *offline construction process*, where a domain relevance measuring model is trained by taking a large set of seed terms and their features as input; 2) the *online query process*, where the trained model can return the domain relevance of query terms by including them in the core-anchored semantic graph. Our approach applies to a wide range of domains and can handle any query, while nearly no human effort is required. To validate the effectiveness of our proposed methods, we conduct

extensive experiments on various domains with different settings. Results show our methods significantly outperform well-designed baselines and even surpass human performance by professionals.

2 Related Work

The problem of domain relevance of terms is related to automatic term extraction, which aims to extract domain-specific terms from texts automatically. Compared to our task, automatic term extraction, where extraction and evaluation are combined, possesses a limited application and has a relatively large dependence on corpora and human annotation, so it is limited to several broad domains and may only cover a small number of terms. Existing approaches for automatic term extraction can be roughly divided into three categories: linguistic, statistical, and machine learning methods. Linguistic methods apply human-designed rules to identify technical/legal terms in a target corpus (Handler et al., 2016; Ha and Hyland, 2017). Statistical methods use statistical information, e.g., frequency of terms, to identify terms from a corpus (Frantzi et al., 2000; Nakagawa and Mori, 2002; Velardi et al., 2001; Drouin, 2003; Meijer et al., 2014). Machine learning methods learn a classifier, e.g., logistic regression classifier, with manually labeled data (Conrado et al., 2013; Fedorenko et al., 2014; Hätyy et al., 2017). There also exists some work on automatic term extraction with Wikipedia (Vivaldi et al., 2012; Wu et al., 2012). However, terms studied there are restricted to terms associated with a Wikipedia page.

Recently, inspired by distributed representations of words (Mikolov et al., 2013a), methods based on deep learning are proposed and achieve state-of-the-art performance. Amjadian et al. (2016, 2018) design supervised learning methods by taking the concatenation of domain-specific and general word embeddings as input. Hätyy et al. (2020) propose a multi-channel neural network model that leverages domain-specific and general word embeddings.

The techniques behind our hierarchical core-fringe learning methods are related to research on graph neural networks (GNNs) (Kipf and Welling, 2017; Hamilton et al., 2017); hierarchical text classification (Vens et al., 2008; Wehrmann et al., 2018; Zhou et al., 2020); and positive-unlabeled learning (Liu et al., 2003; Elkan and Noto, 2008; Bekker and Davis, 2020).

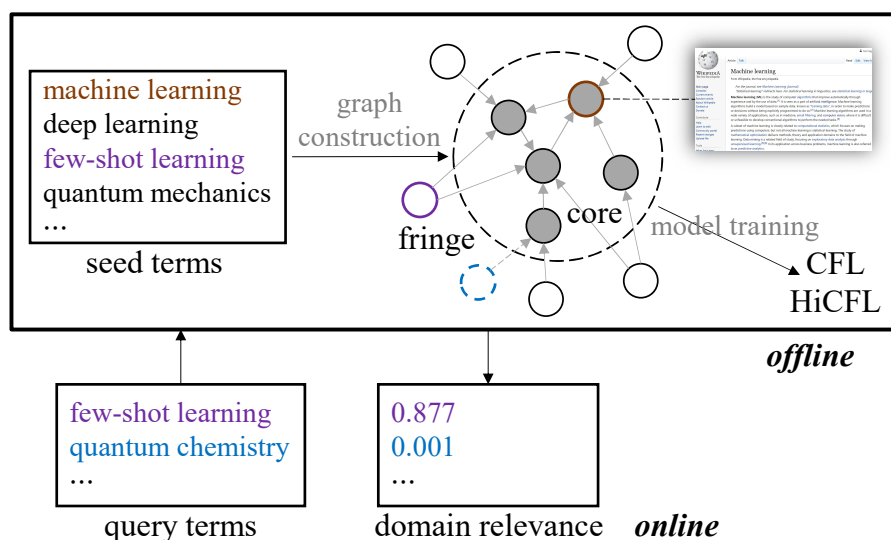


Figure 1: The overview of the framework. In this figure, *machine learning* is a core term associated with a Wikipedia page, *few-shot learning* is a fringe term included in the offline core-anchored semantic graph, and *quantum chemistry* is a fringe term included in the online process. Best viewed in color.

3 Methodology

We study the *Fine-Grained Domain Relevance* of terms, which is defined as follows:

Definition 1 (*Fine-Grained Domain Relevance*)
The fine-grained domain relevance of a term is the degree that the term is relevant to a given domain, and the given domain can be broad or narrow.

The domain relevance of terms depends on many factors. In general, a term with higher semantic relevance, broader meaning scope, and better usage possesses a higher domain relevance regarding the target domain. To measure the fine-grained domain relevance of terms, we propose a hierarchical core-fringe approach, which includes an offline training process and can handle any query term in evaluation. The overview of the framework is illustrated in Figure 1.

3.1 Core-Anchored Semantic Graph

There exist countless terms in human languages; thus it is impractical to include all terms in a system initially. To build the offline system, we need to provide seed terms, which can come from knowledge bases or be extracted from broad, large corpora by existing term/phrase extraction methods (Handler et al., 2016; Shang et al., 2018).

In addition to providing seed terms, we should also give some knowledge to machines so that they can differentiate whether a term is domain-relevant or not. To this end, we can leverage the description information of terms. For instance, Wikipedia

contains a large number of terms (the surface form of page titles), where each term is associated with a Wikipedia article page. With this page information, humans can easily judge whether a term is domain-relevant or not. In Section 3.3, we will show the labeling can even be done completely automatically.

However, considering the countless terms, the number of terms that are well-organized and associated with rich description is small. How to measure the fine-grained domain relevance of terms without rich information is quite challenging for both machines and humans.

Fortunately, terms are not isolated, while complex relations exist between them. If a term is relevant to a domain, it must also be relevant to some domain-relevant terms and vice versa. This is to say, we can bridge the domain relevance of terms through term relevance. Summarizing the observations, we divide terms into two categories: *core terms*, which are terms associated with rich description information, e.g., Wikipedia article pages, and *fringe terms*, which are terms without that information. We assume, for each term, there exist some relevant core terms that share similar domains. If we can find the most relevant core terms for a given term, its domain relevance can be evaluated with the help of those terms. To this end, we can utilize the rich information of core terms for ranking.

Taking Wikipedia as an example, each core term is associated with an article page, so they can

be returned as the ranking results (result term) for a given term (query term). Considering the data resources, we use the built-in Elasticsearch based Wikipedia search engine² (Gormley and Tong, 2015). More specifically, we set the maximum number of links as k (5 as default). For a query term v , i.e., any seed term, we first achieve the top $2k$ Wikipedia pages with exact match. For each result term u in the core, we create a link from u to v . If the number of links is smaller than k , we do this process again without exact match and build additional links. Finally, we construct a term graph, named **Core-Anchored Semantic Graph**, where nodes are terms and edges are links between terms.

In addition, for terms that are not provided initially, we can also handle them as fringe terms and connect them to core terms in evaluation. In this way, we can include any term in the graph.

3.2 Hierarchical Core-Fringe Learning

In this section, we aim to design learning methods to learn the fine-grained domain relevance of core and fringe terms jointly. In addition to using the term graph, we can achieve features of both core and fringe terms based on their linguistic and statistical properties (Terry et al., 2019; Conrado et al., 2013) or distributed representations (Mikolov et al., 2013b; Yu and Dredze, 2015). We assume the labels, i.e., domain-relevant or not, of core terms are available, which can be achieved by an automatic annotation mechanism introduced in Section 3.3.

As stated above, if a term is highly relevant to a given domain, it must also be highly relevant to some other terms with a high domain relevance and vice versa. Therefore, to measure the domain relevance of a term, in addition to using its own features, we aggregate its neighbors' features. Specifically, we propagate the features of terms via the term graph and use the label information of core terms for supervision. In this way, core and fringe terms help each other, and the domain relevance is learned jointly. The propagation process can be achieved by graph convolutions (Hammond et al., 2011). We first apply the vanilla graph convolutional networks (GCNs) (Kipf and Welling, 2017) in our framework. The graph convolution operation (GCNConv) at the l -th layer is formulated as the

following aggregation and update process:

$$\mathbf{h}_i^{(l+1)} = \phi\left(\sum_{j \in \mathcal{N}_i \cup \{i\}} \frac{1}{c_{ij}} \mathbf{W}_c^{(l)} \mathbf{h}_j^{(l)} + \mathbf{b}_c^{(l)}\right), \quad (1)$$

where \mathcal{N}_i is the neighbor set of node i . c_{ij} is the normalization constant. $\mathbf{h}_j^{(l)} \in \mathbb{R}^{d^{(l)} \times 1}$ is the hidden state of node j at the l -th layer, with $d^{(l)}$ being the number of units; $\mathbf{h}_j^{(0)} = \mathbf{x}_j$, which is the feature vector of node j . $\mathbf{W}_c^{(l)} \in \mathbb{R}^{d^{(l+1)} \times d^{(l)}}$ is the trainable weight matrix at the l -th layer, and $\mathbf{b}_c^{(l)}$ is the bias vector. $\phi(\cdot)$ is the nonlinearity activation function, e.g., $\text{ReLU}(\cdot) = \max(0, \cdot)$.

Since core terms are labeled as domain-relevant or not, we can use the labels to calculate the loss:

$$\mathcal{L} = - \sum_{i \in \mathcal{V}_{core}} (y_i \log z_i + (1 - y_i) \log(1 - z_i)), \quad (2)$$

where y_i is the label of node i regarding the target domain, and $z_i = \sigma(h_i^o)$, with h_i^o being the output of the last GCNConv layer for node i and $\sigma(\cdot)$ being the sigmoid function. The weights of the model are trained by minimizing the loss. The relative domain relevance is obtained as $\mathbf{s} = \mathbf{z}$.

Combining with the overall framework, we get the first domain relevance measuring model, **CFL**, i.e., **Core-Fringe Domain Relevance Learning**.

CFL is useful to measure the domain relevance for broad domains such as computer science. For domains with relatively narrow scopes, e.g., machine learning, we can also leverage the label information of domains at the higher level of the hierarchy, e.g., $\text{CS} \rightarrow \text{AI} \rightarrow \text{ML}$, which is based on the idea that a domain-relevant term regarding the target domain should also be relevant to the parent domain. Inspired by related work on hierarchical multi-label classification (Vens et al., 2008; Wehrmann et al., 2018), we introduce a hierarchical learning method considering both global and local information.

We first apply l_c GCNConv layers according to Eq. (1) and get the output of the last GCNConv layer, which is $\mathbf{h}_i^{(l_c)}$. In order not to confuse, we omit the subscript that identifies the node number. For each domain in the hierarchy, we introduce a hierarchical global activation \mathbf{a}_p . The activation at the $(l + 1)$ -th level of the hierarchy is given as

$$\mathbf{a}_p^{(l+1)} = \phi(\mathbf{W}_p^{(l)}[\mathbf{a}_p^{(l)}; \mathbf{h}^{(l_c)}] + \mathbf{b}_p^{(l)}), \quad (3)$$

where $[\cdot; \cdot]$ indicates the concatenation of two vectors; $\mathbf{a}_p^{(1)} = \phi(\mathbf{W}_p^{(0)} \mathbf{h}^{(l_c)} + \mathbf{b}_p^{(0)})$. The global in-

²<https://en.wikipedia.org/w/index.php?search>

formation is produced after a fully connected layer:

$$\mathbf{z}_p = \sigma(\mathbf{W}_p^{(l_p)} \mathbf{a}_p^{(l_p)} + \mathbf{b}_p^{(l_p)}), \quad (4)$$

where l_p is the total number of hierarchical levels.

To achieve the local information for each level of the hierarchy, the model first generates the local hidden state $\mathbf{a}_q^{(l)}$ by a fully connected layer:

$$\mathbf{a}_q^{(l)} = \phi(\mathbf{W}_t^{(l)} \mathbf{a}_p^{(l)} + \mathbf{b}_t^{(l)}). \quad (5)$$

The local information at the l -th level of the hierarchy is then produced as

$$\mathbf{z}_q^{(l)} = \sigma(\mathbf{W}_q^{(l)} \mathbf{a}_q^{(l)} + \mathbf{b}_q^{(l)}). \quad (6)$$

In our core-fringe framework, all the core terms are labeled at each level of the hierarchy. Therefore, the loss of hierarchical learning is computed as

$$\mathcal{L}_h = \epsilon(\mathbf{z}_p, \mathbf{y}^{(l_p)}) + \sum_{l=1}^{l_p} \epsilon(\mathbf{z}_q^{(l)}, \mathbf{y}^{(l)}), \quad (7)$$

where $\mathbf{y}^{(l)}$ denotes the labels regarding the domain at the l -th level of the hierarchy and $\epsilon(\mathbf{z}, \mathbf{y})$ is the binary cross-entropy loss described in Eq. (2). In testing, The relative domain relevance \mathbf{s} is calculated as

$$\mathbf{s} = \alpha \cdot \mathbf{z}_p + (1 - \alpha) \cdot (\mathbf{z}_q^{(1)} \circ \mathbf{z}_q^{(2)}, \dots, \mathbf{z}_q^{(l_p)}), \quad (8)$$

where \circ denotes element-wise multiplication. α is a hyperparameter to balance the global and local information (0.5 as default). Combining with our general framework, we refer to this model as **HiCFL**, i.e., **Hierarchical CFL**.

Online Query Process. If seed terms are provided by extracting from broad, large corpora relevant to the target domain, most terms of interest will be already included in the offline process. In evaluation, for terms that are not provided initially, our model treats them as fringe terms. Specifically, when receiving such a term, the model connects it to core terms by the method described in Section 3.1. With its features (e.g., compositional term embeddings) or only its neighbors' features (when features cannot be generated directly), the trained model can return the domain relevance of any query.

3.3 Automatic Annotation and Hierarchical Positive-Unlabeled Learning

Automatic Annotation. For the fine-grained domain relevance problem, human annotation is very

time-consuming and laborious because the number of core terms is very large regarding a wide range of domains. Fortunately, in addition to building the term graph, we can also leverage the rich information of core terms for automatic annotation.

In the core-anchored semantic graph constructed with Wikipedia, each core term is associated with a Wikipedia page, and each page is assigned one or more categories. All the categories form a hierarchy, furthermore providing a category tree. For a given domain, we can first traverse from a root category and collect some gold subcategories. For instance, for computer science, we treat *category: subfields of computer science*³ as the root category and take categories at the first three levels of it as gold subcategories. Then we collect categories for each core term and examine whether the term itself or one of the categories is a gold subcategory. If so, we label the term as positive. Otherwise, we label it as negative. We can also combine gold subcategories from some existing domain taxonomies and extract the categories of core terms from the text description, which usually contains useful text patterns like “*x is a subfield of y*”.

Hierarchical Positive-Unlabeled Learning. According to the above methods, we can learn the fine-grained domain relevance of terms for any domain as long as we can collect enough gold subcategories for that domain. However, for domains at the low level of the hierarchy, e.g., deep learning, a category tree might not be available in Wikipedia. To deal with this issue, we apply our learning methods in a positive-unlabeled (PU) setting (Bekker and Davis, 2020), where only a small number of terms, e.g., 10, are labeled as positive, and all the other terms are unlabeled. We use this setting based on the following consideration: if a user is interested in a specific domain, it is quite easy for her to give some important terms relevant to that domain.

Benefiting from our hierarchical core-fringe learning approach, we can still obtain labels for domains at the high level of the hierarchy with the automatic annotation mechanism. Therefore, all the negative examples of the last labeled hierarchy can be used as reliable negatives for the target domain. For instance, if the target domain is deep learning, which is in the CS \rightarrow AI \rightarrow ML \rightarrow DL hierarchy, we consider all the non-ML terms as the reliable negatives for DL. Taking the positively

³https://en.wikipedia.org/wiki/Category:Subfields_of_computer_science

labeled examples and the reliable negatives for supervision, we can learn the domain relevance of terms by our proposed HiCFL model contextualized in the hierarchy of the domain.

4 Experiments

In this section, we evaluate our model from different perspectives. 1) We **compare with baselines** by treating some labeled terms as queries. 2) We **compare with human professionals** by letting humans and machines judge which term in a query pair is more relevant to a target domain. 3) We conduct intuitive **case studies** by ranking terms according to their domain relevance.

4.1 Experimental Setup

Datasets and Preprocessing. To build the system, for offline processing, we extract seed terms from the arXiv dataset (version 6)⁴. As an example, for computer science or its sub-domains, we collect the abstracts in computer science according to the arXiv Category Taxonomy⁵, and apply *phrasemachine* to extract terms (Handler et al., 2016) with lemmatization and several filtering rules: frequency > 10; length ≤ 6; only contain letters, numbers, and hyphen; not a stopword or a single letter.

We select three broad domains, including *computer science* (CS), *physics* (Phy), and *mathematics* (Math); and three narrow sub-domains of them, including *machine learning* (ML), *quantum mechanics* (QM), and *abstract algebra* (AA), with the hierarchies CS → AI → ML, Phy → mechanics → QM, and Math → algebra → AA. Each broad domain and its sub-domains share seed terms because they share a corpus. To achieve gold subcategories for automatic annotation (Section 3.3), we collect subcategories at the first three levels of a root category (e.g., *category: subfields of physics*) for broad domains (e.g., physics); or the first two levels for narrow domains, e.g., *category: machine learning* for machine learning. Table 1 reports the total sizes and the ratios that are core terms.

Baselines. Since our task on fine-grained domain relevance is new, there is no existing baseline for model comparison. We adapt the following models on relevant tasks in our setting with additional inputs (e.g., domain-specific corpora):

⁴<https://www.kaggle.com/Cornell-University/arxiv>

⁵https://arxiv.org/category_taxonomy

domain		#terms	core ratio
CS	ML	113,038	27.7%
Phy	QM	416,431	12.1%
Math	AA	103,984	26.4%

Table 1: The statistics of the data.

- **Relative Domain Frequency (RDF):** Since domain-relevant terms usually occur more in a domain-specific corpus, we apply a statistical method using $\text{freq}_s(w)/\text{freq}_g(w)$ to measure the domain relevance of term w , where $\text{freq}_s(\cdot)$ and $\text{freq}_g(\cdot)$ denote the frequency of occurrence in the domain-specific/general corpora respectively.
- **Logistic Regression (LR):** Logistic regression is a standard supervised learning method. We use core terms with labels (domain-relevant or not) as training data, where features are term embeddings trained by a general corpus.
- **Multilayer Perceptron (MLP):** MLP is a standard neural neural-based model. We train MLP using embeddings trained with a domain-specific corpus or a general corpus as term features, respectively. We also concatenate the two embeddings as features (Amjadian et al., 2016, 2018).
- **Multi-Channel (MC):** Multi-Channel (Hätty et al., 2020) is the state-of-the-art model for automatic term extraction, which is based on a multi-channel neural network that takes domain-specific and general corpora as input.

Training. For all supervised learning methods, we apply automatic annotation in Section 3.3, i.e., we automatically label all the core terms for model training. In the PU setting, we remove labels on target domains. Only 20 (10 in the case studies) domain-relevant core terms are randomly selected as the positives, with the remaining terms unlabeled. In training, all the negative examples at the previous level of the hierarchy are used as reliable negatives.

Implementation Details. Though our proposed methods are independent of corpora, some baselines (e.g., MC) require term embeddings trained from general/domain-specific corpora. For easy and fair comparison, we adopt the following approach to generate term features. We consider each term as a single token, and apply word2vec CBOW (Mikolov et al., 2013a) with negative sampling, where dimensionality is 100, window size is 5, and number of negative samples is 5. The training cor-

		Computer Science		Physics		Mathematics	
		ROC-AUC	PR-AUC	ROC-AUC	PR-AUC	ROC-AUC	PR-AUC
RDF	SG	0.714	0.417	0.736	0.496	0.694	0.579
LR	G	0.802 \pm 0.000	0.535 \pm 0.000	0.822 \pm 0.000	0.670 \pm 0.000	0.854 \pm 0.000	0.769 \pm 0.000
MLP	S	0.819 \pm 0.003	0.594 \pm 0.003	0.853 \pm 0.001	0.739 \pm 0.004	0.868 \pm 0.000	0.803 \pm 0.001
MLP	G	0.863 \pm 0.001	0.674 \pm 0.002	0.874 \pm 0.001	0.761 \pm 0.003	0.904 \pm 0.001	0.846 \pm 0.002
MLP	SG	0.867 \pm 0.001	0.667 \pm 0.002	0.875 \pm 0.001	0.765 \pm 0.002	0.904 \pm 0.001	0.843 \pm 0.003
MC	SG	0.868 \pm 0.002	0.664 \pm 0.006	0.877 \pm 0.003	0.768 \pm 0.004	0.903 \pm 0.001	0.843 \pm 0.002
CFL	G	0.885 \pm 0.001	0.712 \pm 0.002	0.905 \pm 0.000	0.812 \pm 0.002	0.918 \pm 0.001	0.870 \pm 0.002
CFL	C	0.883 \pm 0.001	0.708 \pm 0.002	0.901 \pm 0.000	0.800 \pm 0.001	0.919 \pm 0.001	0.879 \pm 0.002

S and G indicate the corpus used. S: domain-specific corpus, G: general corpus, SG: both. C means the pre-trained compositional GloVe embeddings are used.

Table 2: Results for broad domains.

pus can be a general one (the entire arXiv corpus, denoted as G), or a domain-specific one (the sub-corpus in the branch of the corresponding domain, denoted as S). We also apply compositional GloVe embeddings (Pennington et al., 2014) (element-wise addition of the pre-trained 100d word embeddings, denoted as C) as non-corpus-specific features of terms for reference.

For all the neural network-based models, we use Adam (Kingma and Ba, 2015) with learning rate of 0.01 for optimization, and adopt a fixed hidden dimensionality of 256 and a fixed dropout ratio of 0.5. For the learning part of CFL and HiCFL, we apply two GCNConv layers and use the symmetric graph for training. To avoid overfitting, we adopt batch normalization (Ioffe and Szegedy, 2015) right after each layer (except for the output layer) and before activation and apply dropout (Hinton et al., 2012) after the activation. We also try to add regularizations for MLP and MC with full-batch or mini-batch training, and select the best architecture. To construct the core-anchored semantic graph, we set k as 5. All experiments are run on an NVIDIA Quadro RTX 5000 with 16GB of memory under the PyTorch framework. The training of CFL for the CS domain can finish in 1 minute.

We report the mean and standard deviation of the test results corresponding to the best validation results with 5 different random seeds.

4.2 Comparison to Baselines

To compare with baselines, we separate a portion of core terms as queries for evaluation. Specifically, for each domain, we use 80% labeled terms for training, 10% for validation, and 10% for testing

(with automatic annotation). Terms in the validation and testing sets are treated as fringe terms. By doing this, the evaluation can represent the general performance for all fringe terms to some extent. And the model comparison is fair since the rich information of terms for evaluation is not used in training. We also create a test set with careful human annotation on *machine learning* to support our overall evaluation, which contains 2000 terms, with half for evaluation and half for testing.

As evaluation metrics, we calculate both ROC-AUC and PR-AUC with automatic or manually created labels. ROC-AUC is the area under the receiver operating characteristic curve, and PR-AUC is the area under the precision-recall curve. If a model achieves higher values, most of the domain-relevant terms are ranked higher, which means the model has a better measurement on the domain relevance of terms.

Table 2 and Table 3 show the results for three broad/narrow domains respectively. We observe our proposed CFL and HiCFL outperform all the baselines, and the standard deviations are low. Compared to MLP, CFL achieves much better performance benefiting from the core-anchored semantic graph and feature aggregation, which demonstrates the domain relevance can be bridged via term relevance. Compared to CFL, HiCFL works better owing to hierarchical learning.

In the PU setting— the situation when automatic annotation is not applied to the target domain, although only 20 positives are given, HiCFL still achieves satisfactory performance and significantly outperforms all the baselines (Table 4).

The PR-AUC scores on the manually created test

		Machine Learning		Quantum Mechanics		Abstract Algebra	
		ROC-AUC	PR-AUC	ROC-AUC	PR-AUC	ROC-AUC	PR-AUC
LR	G	0.917 \pm 0.000	0.346 \pm 0.000	0.879 \pm 0.000	0.421 \pm 0.000	0.872 \pm 0.000	0.525 \pm 0.000
MLP	S	0.902 \pm 0.001	0.453 \pm 0.009	0.903 \pm 0.001	0.545 \pm 0.004	0.910 \pm 0.000	0.641 \pm 0.007
MLP	G	0.932 \pm 0.001	0.562 \pm 0.010	0.922 \pm 0.001	0.587 \pm 0.014	0.923 \pm 0.000	0.658 \pm 0.006
MLP	SG	0.928 \pm 0.001	0.574 \pm 0.011	0.923 \pm 0.000	0.574 \pm 0.007	0.925 \pm 0.001	0.673 \pm 0.004
MC	SG	0.928 \pm 0.002	0.554 \pm 0.007	0.924 \pm 0.001	0.590 \pm 0.003	0.924 \pm 0.001	0.685 \pm 0.005
CFL	G	0.950 \pm 0.002	0.627 \pm 0.013	0.950 \pm 0.000	0.678 \pm 0.003	0.938 \pm 0.001	0.751 \pm 0.009
HiCFL	G	0.965 \pm 0.003	0.645 \pm 0.014	0.957 \pm 0.001	0.691 \pm 0.003	0.942 \pm 0.002	0.769 \pm 0.006

S and G indicate the corpus used. S: domain-specific corpus, G: general corpus, SG: both.

Table 3: Results for narrow domains.

		Machine Learning		Quantum Mechanics		Abstract Algebra	
		ROC-AUC	PR-AUC	ROC-AUC	PR-AUC	ROC-AUC	PR-AUC
LR	G	0.860 \pm 0.000	0.206 \pm 0.000	0.788 \pm 0.000	0.280 \pm 0.000	0.833 \pm 0.000	0.429 \pm 0.000
MLP	S	0.804 \pm 0.003	0.144 \pm 0.003	0.767 \pm 0.009	0.260 \pm 0.005	0.804 \pm 0.006	0.421 \pm 0.010
MLP	G	0.836 \pm 0.005	0.234 \pm 0.016	0.813 \pm 0.006	0.295 \pm 0.011	0.842 \pm 0.003	0.467 \pm 0.011
MLP	SG	0.844 \pm 0.003	0.230 \pm 0.015	0.796 \pm 0.008	0.291 \pm 0.011	0.839 \pm 0.006	0.463 \pm 0.013
MC	SG	0.852 \pm 0.006	0.251 \pm 0.019	0.795 \pm 0.014	0.303 \pm 0.017	0.861 \pm 0.004	0.547 \pm 0.006
CFL	G	0.918 \pm 0.001	0.441 \pm 0.009	0.897 \pm 0.002	0.408 \pm 0.004	0.887 \pm 0.002	0.563 \pm 0.018
HiCFL	G	0.940 \pm 0.008	0.508 \pm 0.026	0.897 \pm 0.004	0.421 \pm 0.014	0.915 \pm 0.002	0.648 \pm 0.009

Table 4: Results for narrow domains (PU learning).

		PR-AUC	PR-AUC (PU)
LR	G	0.509 \pm 0.000	0.449 \pm 0.000
MLP	S	0.550 \pm 0.017	0.113 \pm 0.010
MLP	G	0.586 \pm 0.016	0.299 \pm 0.027
MLP	SG	0.590 \pm 0.005	0.217 \pm 0.013
MC	SG	0.603 \pm 0.016	0.281 \pm 0.012
CFL	G	0.703 \pm 0.017	0.525 \pm 0.013
HiCFL	G	0.755 \pm 0.011	0.581 \pm 0.036

Table 5: Results (PR-AUC) for machine learning with manual labeling.

set without and with the PU setting are reported in Table 5. We observe that the results are generally consistent with results reported in Table 3 and Table 4, which indicates the evaluation with core terms can work just as well.

4.3 Comparison to Human Performance

In this section, we aim to compare our model with human professionals in measuring the fine-grained domain relevance of terms. Because it is difficult for humans to assign a score representing do-

	ML-AI	ML-CS	AI-CS
Human	0.698 \pm 0.087	0.846 \pm 0.074	0.716 \pm 0.115
HiCFL	0.854 \pm 0.017	0.932 \pm 0.007	0.768 \pm 0.023

Table 6: Accuracies of domain relevance comparison.

main relevance directly, we generate term pairs as queries and let humans judge which one in a pair is more relevant to *machine learning*. Specifically, we create 100 ML-AI, ML-CS, and AI-CS pairs respectively. Taking ML-AI as an example, each query pair consists of an ML term and an AI term, and the judgment is considered right if the ML term is selected.

The human annotation is conducted by five senior students majoring in computer science and doing research related to terminology. Because there is no clear boundary between ML, AI, and CS, it is possible that a CS term is more relevant to machine learning than an AI term. However, the overall trend is that the higher the accuracy, the better the performance. From Table 6, we observe that HiCFL far outperforms human performance.

The depth of the background color indicates the domain relevance. The darker the color, the higher the domain relevance (annotated by the authors); * indicates the term is a core term, otherwise it is a fringe term.

1-10	101-110	1001-1010	10001-10010	100001-100010
supervised learning*	adversarial machine learning*	regularization strategy	method for detection	tumor region
convolutional neural network*	temporal-difference learning*	weakly-supervised approach	gait parameter	mutual trust
machine learning*	restricted boltzmann machine	learned embedding	stochastic method	inherent problem
deep learning*	backpropagation through time*	node classification problem	recommendation diversity	healthcare system*
semi-supervised learning*	svms	non-convex learning	numerical experiment	two-phase*
q-learning*	word2vec*	sample-efficient learning	second-order method	posetrack
reinforcement learning*	rbms	cnm-rnn model	landmark dataset	half*
unsupervised learning*	hierarchical clustering*	deep bayesian	general object detection	mfc5
recurrent neural network*	stochastic gradient descent*	classification score	cold-start recommendation	borda count*
generative adversarial network*	svm*	classification algorithm*	similarity of image	diverse way

Table 7: Ranking results for machine learning with HiCFL.

Given positives (10): deep learning, neural network, deep neural network, deep reinforcement learning, multilayer perceptron, convolutional neural network, recurrent neural network, long short-term memory, backpropagation, activation function.

1-10	101-110	1001-1010	10001-10010	100001-100010
convolutional neural network*	discriminative loss	multi-task deep learning	low light image	law enforcement agency*
recurrent neural network*	dropout regularization	self-supervision	face dataset	case of channel
artificial neural network*	semantic segmentation*	state-of-the-art deep learning algorithm	estimation network	release*
feedforward neural network*	mask-rcnn	generative probabilistic model	method on benchmark datasets	ahonen*
deep learning*	probabilistic neural network*	translation model	distributed constraint	electoral control
neural network*	pretrained network	probabilistic segmentation	gradient information	runge*
generative adversarial network*	discriminator model	handwritten digit classification	model on a variety	many study
multilayer perceptron*	sequence-to-sequence learning	deep learning classification	model constraint	mean value*
long short-term memory*	autoencoders	multi-task reinforcement learning	automatic detection	efficient beam
neural architecture search*	conditional variational autoencoder	skip-gram*	feature redundancy	pvt*

Table 8: Ranking results for deep learning with HiCFL (PU learning).

Although we have reduced the difficulty, the task is still very challenging for human professionals.

4.4 Case Studies

We interpret our results by ranking terms according to their domain relevance regarding *machine learning* or *deep learning*, with hierarchy CS \rightarrow AI \rightarrow ML \rightarrow DL. For CS-ML, we label terms with automatic annotation. For DL, we create 10 DL terms manually as the positives for PU learning.

Table 7 and Table 8 show the ranking results (1-10 represents terms ranked 1st to 10th). We observe the performance is satisfactory. For ML, important concepts such as supervised learning, unsupervised learning, and deep learning are ranked very high. Also, terms ranked before 1010th are all good domain-relevant terms. For DL, although only 10 positives are provided, the ranking results are quite impressive. E.g., unlabeled positive terms like artificial neural network, generative adversarial network, and neural architecture search are ranked very high. Besides, terms ranked 101st to 110th are all highly relevant to DL, and terms ranked 1001st to 1010th are related to ML.

5 Conclusion

We introduce and study the fine-grained domain relevance of terms— an important property of terms that has not been carefully studied before. We

propose a hierarchical core-fringe domain relevance learning approach, which can cover almost all terms in human languages and various domains, while requires little or even no human annotation.

We believe this work will inspire an automated solution for knowledge management and help a wide range of downstream applications in natural language processing. It is also interesting to integrate our methods to more challenging tasks, for example, to characterize more complex properties of terms even understand terms.

Acknowledgments

We thank the anonymous reviewers for their valuable comments and suggestions. This material is based upon work supported by the National Science Foundation IIS 16-19302 and IIS 16-33755, Zhejiang University ZJU Research 083650, IBM-Illinois Center for Cognitive Computing Systems Research (C3SR) - a research collaboration as part of the IBM Cognitive Horizon Network, grants from eBay and Microsoft Azure, UIUC OVCR CCIL Planning Grant 434S34, UIUC CSBS Small Grant 434C8U, and UIUC New Frontiers Initiative. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of the funding agencies.

References

- Fatima N Al-Aswadi, Huah Yong Chan, and Keng Hoon Gan. 2019. Automatic ontology construction from text: a review from shallow to deep learning trend. *Artificial Intelligence Review*, pages 1–28.
- Ehsan Amjadian, Diana Inkpen, T Sima Paribakht, and Farahnaz Faez. 2018. Distributed specificity for automatic terminology extraction. *Terminology. International Journal of Theoretical and Applied Issues in Specialized Communication*, 24(1):23–40.
- Ehsan Amjadian, Diana Inkpen, Tahereh Paribakht, and Farahnaz Faez. 2016. Local-global vectors to improve unigram terminology extraction. In *Proceedings of the 5th International Workshop on Computational Terminology*, pages 2–11.
- Jessa Bekker and Jesse Davis. 2020. Learning from positive and unlabeled data: a survey. *Mach. Learn.*, 109(4):719–760.
- Merley Conrado, Thiago Pardo, and Solange Oliveira Rezende. 2013. A machine learning approach to automatic term extraction using a rich feature set. In *Proceedings of the 2013 NAACL HLT Student Research Workshop*, pages 16–23.
- Patrick Drouin. 2003. Term extraction using non-technical corpora as a point of leverage. *Terminology*, 9(1):99–115.
- Charles Elkan and Keith Noto. 2008. Learning classifiers from only positive and unlabeled data. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 213–220.
- Denis Fedorenko, N Astrakhantsev, and D Turdakov. 2014. Automatic recognition of domain-specific terms: an experimental evaluation. *Proceedings of the Institute for System Programming*, 26(4):55–72.
- Katerina Frantzi, Sophia Ananiadou, and Hideki Mima. 2000. Automatic recognition of multi-word terms: the c-value/nc-value method. *International journal on digital libraries*, 3(2):115–130.
- Clinton Gormley and Zachary Tong. 2015. *Elasticsearch: the definitive guide: a distributed real-time search and analytics engine*. ” O’Reilly Media, Inc.”.
- Althea Ying Ho Ha and Ken Hyland. 2017. What is technicality? a technicality analysis model for eap vocabulary. *Journal of English for Academic Purposes*, 28:35–49.
- William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 1025–1035.
- David K Hammond, Pierre Vandergheynst, and Rémi Gribonval. 2011. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150.
- Abram Handler, Matthew Denny, Hanna Wallach, and Brendan O’Connor. 2016. Bag of what? simple noun phrase extraction for text analysis. In *Proceedings of the First Workshop on NLP and Computational Social Science*, pages 114–124.
- Anna HäTTY, Michael Dorna, and Sabine Schulte im Walde. 2017. Evaluating the reliability and interaction of recursively used feature classes for terminology extraction. In *Proceedings of the student research workshop at the 15th conference of the European chapter of the association for computational linguistics*, pages 113–121.
- Anna HäTTY, Dominik Schlechtweg, Michael Dorna, and Sabine Schulte im Walde. 2020. Predicting degrees of technicality in automatic terminology extraction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2883–2889.
- Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.
- Jie Huang, Zilong Wang, Kevin Chen-Chuan Chang, Wen-mei Hwu, and Jinjun Xiong. 2020. Exploring semantic capacity of terms. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*.
- Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456.
- Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *Proceedings of the 3rd International Conference on Learning Representations*.
- Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *Proceedings of International Conference on Learning Representations*.
- Bing Liu, Yang Dai, Xiaoli Li, Wee Sun Lee, and Philip S Yu. 2003. Building text classifiers using positive and unlabeled examples. In *Third IEEE International Conference on Data Mining*, pages 179–186. IEEE.
- Kevin Meijer, Flavius Frasinca, and Frederik Hogenboom. 2014. A semantic approach for extracting domain taxonomies from text. *Decision Support Systems*, 62:78–93.

- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.
- Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.
- Hiroshi Nakagawa and Tatsunori Mori. 2002. A simple but powerful automatic term extraction method. In *COLING-02: COMPUTERM 2002: Second International Workshop on Computational Terminology*.
- Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.
- Chao Shang, Sarthak Dash, Md Faisal Mahub Chowdhury, Nandana Mihindukulasooriya, and Alfio Gliozzo. 2020. Taxonomy construction of unseen domains via graph-based cross-domain knowledge transfer. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2198–2208.
- Jingbo Shang, Jialu Liu, Meng Jiang, Xiang Ren, Clare R Voss, and Jiawei Han. 2018. Automated phrase mining from massive text corpora. *IEEE Transactions on Knowledge and Data Engineering*, 30(10):1825–1837.
- Ayla Rigouts Terryn, Véronique Hoste, and Els Lefever. 2019. In no uncertain terms: a dataset for monolingual and multilingual automatic term extraction from comparable corpora. *Language Resources and Evaluation*, pages 1–34.
- Paola Velardi, Michele Missikoff, and Roberto Basili. 2001. Identification of relevant terms to support the construction of domain ontologies. In *Proceedings of the ACL 2001 Workshop on Human Language Technology and Knowledge Management*.
- Celine Vens, Jan Struyf, Leander Schietgat, Sašo Džeroski, and Hendrik Blockeel. 2008. Decision trees for hierarchical multi-label classification. *Machine learning*, 73(2):185.
- Jorge Vivaldi, Luis Adrián Cabrera-Diego, Gerardo Sierra, and María Pozzi. 2012. Using wikipedia to validate the terminology found in a corpus of basic textbooks. In *LREC*, pages 3820–3827.
- Jonatas Wehrmann, Ricardo Cerri, and Rodrigo Barros. 2018. Hierarchical multi-label classification networks. In *International Conference on Machine Learning*, pages 5075–5084.
- Wenjuan Wu, Tao Liu, He Hu, and Xiaoyong Du. 2012. Extracting domain-relevant term using wikipedia based on random walk model. In *2012 Seventh ChinaGrid Annual Conference*, pages 68–75. IEEE.
- Mo Yu and Mark Dredze. 2015. Learning composition models for phrase embeddings. *Transactions of the Association for Computational Linguistics*, 3:227–242.
- Jie Zhou, Chunping Ma, Dingkun Long, Guangwei Xu, Ning Ding, Haoyu Zhang, Pengjun Xie, and Gongshen Liu. 2020. Hierarchy-aware global model for hierarchical text classification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1106–1117.