EasyChair Preprint

№ 7288

Ontology-Driven Scientific Literature Classification using Clustering and Self-Supervised Learning

Zhengtong Pan, Patrick Soong and Setareh Rafatirad

January 5, 2022

Ontology-Driven Scientific Literature Classification using Clustering and Self-Supervised Learning

Zhengtong Pan[1], Patrck Soong[2], Setareh Rafatirad[3]

[1,2,3] *Department of Computer Science, University of California, Davis*

[1] pztcookie@ucdavis.edu

[2] ppsoong@ucdavis.edu

[3] srafatirad@ucdavis.edu

Abstract

The rapid growth of scientific literature in the fields of computer engineering (CE) and computer science (CS) presents difficulties to researchers who are interested in exploring research publication records based on standard scientific categories. This urges the need for a context-aware, automatic classification of text documents into standard scientific categories. Document classification is a significant application of supervised learning which requires a labeled dataset for training the classifier. However, research publication records available on Google Scholar and dblp services are not labeled. First, manual annotation of a large body of scientific research work based on standard scientific terminology requires domain expertise and is extremely time-consuming. Second, hierarchical labeling of records facilitates a more effective and context-aware retrieval of documents. In this paper, we propose an ontology-driven classification technique based on zero-shot learning in conjunction with agglomerative clustering to automatically label a scientific literature dataset related to CE and CS. We further study and compare the effectiveness of multiple text classifiers such as logistic regression (LR), support vector machines (SVM), gradient boosting with Word2vec and bag of words (BOW) embedding, recurrent neural networks (RNN) with GloVe embedding, and feed-forward neural networks with BOW embedding. Our study showed that RNN with GloVe embedding outperforms other models with an above 0.85 F1 score on all granularity levels. Our proposed technique will help junior and experienced researchers identify new emerging technologies and domains for their research purposes.

Keywords

Hierarchical Document Classification, Natural Language Processing, Self-supervised Learning, Unsupervised Learning, Zero-Shot Learning

I. Introduction

Identifying potential emerging technologies and topics is an integral part of scientific research that has become increasingly difficult given the rapidly growing volume of research papers, journals, and conferences. Global research output grew about 4% annually from 2008 to 2018 and does not show signs of slowing down. For peer-reviewed publications, there is an increase from 1.8M to 2.6M articles per year [18]. Motivated by this challenge and in an effort to help junior researchers enter the field, we explored potential applications of machine learning through natural language processing (NLP), self-supervised learning, ontology construction, and agglomerative clustering for hierarchical document classification in order to automatically classify research papers in different sub-fields of CE and CS. While previous research papers have explored document classification, they did not use hierarchical document classification to

classify research publications to support research topics in CE and CS. In [3], [5], the authors proposed a hierarchical method for the classification of income tax and management-related documents. However, their datasets were manually labeled by domain experts. In [16], the authors proposed a document classification framework for the field of CS, using an unsupervised learning approach by applying clustering methods to unlabeled documents. Another research on scholarly paper classification only developed a flat model for classification with low F1 scores caused by data imbalance problem [22]. Furthermore, a promising document classification approach was proposed by [1] where the dataset was manually labeled by domain experts. However, human-induced tags can introduce noise and affect the performance of the model. Being inspired by all these previous research works, we propose a context-aware document classification framework based on the hierarchical structure of topics related to our dataset to achieve high classification performance using three granularity levels, from the coarsest to the finest level,i.e., domains, categories, and raw tags. The proposed framework supports the research fields of CE and CS. Our tri-level classifier obtained F1 scores ranging from 0.80 to 0.96. Moreover, this paper introduces an automatic labeling approach based on a self-supervised learning model along with an ontology to organize labels, and agglomerative clustering to verify label assignments. This novel approach in hierarchical document classification in the field of CS and CE is extremely valuable to researchers and would be a boon to the field. Compared to [16] which apply unsupervised learning methods to classify research papers, our proposed method provides more specialized and context-aware document labels to facilitate meaningful search and information retrieval. The outline of our proposed methodology for classifying scientific research documents into trending technological fields is illustrated in figure. 1.

The input dataset contains documents with titles, abstracts, published year, and conference names from 50K publication records collected from Google Scholar and dblp database [4] [15]. The metadata enrichment task used the papers' conference name and published year information to scrape each conferences' Call-For-Papers page to further obtain three-level of metadata corresponding to the topics of papers including domains (e.g. Machine Learning), categories (e.g. Applications), and raw tags (e.g. Crowdsourcing, Healthcare). Such contextual multi-level information was used in addition to the information mined from the title and abstract of paper publications to enrich the set of labels and to classify scientific papers. In order to create a standard, hierarchical and context-aware vocabulary for labeling documents in multiple granularity levels, we constructed an ontology to determine the labels. The metadata enrichment and ontology construction tasks were mainly conducted to use standardized and context-aware vocabulary rather than using unstructured, ad-hoc labels due to the noise associated with them. The standardized vocabulary was used in this work for the training of our text classifiers.

As part of feature extraction, we extracted n-grams and performed word embedding to generate the input for dataset labeling. Next, to avoid manual annotation which requires significant time and human effort, we leveraged self-supervised learning to annotate the unlabeled dataset [12]. Using self-supervised learning as Zero-shot learning (ZSL) in conjunction with the standard vocabulary obtained through data collection and metadata enrichment; the assignment of labels was automated to a massive collection of papers according to relevant scientific topics with a reasonable degree of confidence.

To ensure the accuracy of the assigned labels, labeling verification was performed. This verification task ensured that the document classifiers would not be trained on erroneous labels. In order to validate the labels, it is important to understand the content of the documents. This task is necessary to check if the counts of different assigned labels correspond to the distributions of the

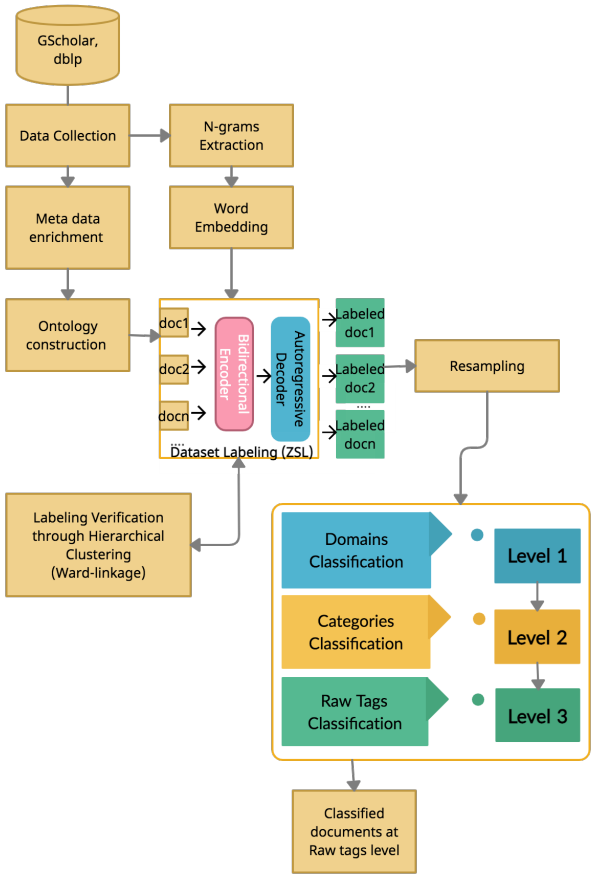

Figure 1. A Tri-Layer Document Classification Framework.

unlabeled documents categorized by their contents. To this end, we applied unsupervised learning as agglomerative clustering on word embedding. Motivated by the imbalanced data labeling results from the ZSL label assignment and the imbalances document clusters by agglomerative clustering, we resampled the dataset to have balanced label counts in order to achieve a higher classification performance.

We then used grid search to obtain the best performing combination of hyper-parameters for each classifier at each granularity level. Furthermore, we performed a comparative study of various word embedding techniques including BOW, Word2vec, and GloVe [9] embedding in conjunction with several text classifiers to report the winning classifier that yielded the highest F1 score at each topic-granularity level.

This paper proposes a tri-layer text classification framework (see figure. 1) which used structured metadata as a standard vocabulary to classify scientific published papers. The contribution of this paper is twofold:

- Performing a robust, ontology-based, automated labeling using a combination of ZSL and clustering which enabled us to achieve a 76% confidence score at the domain-level, 80 % confidence score at the category-level, and 70% confidence score at the raw-tags-level along with the proof of labels' relatedness to its corresponding document content through ward-linkage hierarchical clustering.

- Performing a comparative study that led to developing a robust classifier using Recurrent Neural Net (RNN) algorithm and GloVe embedding technique. Our comparative study showed that RNN with GloVe embedding outperformed other models studied in this work (including logistic regression, support vector machines, and gradient boosting with bag of words or word2vec) with 0.95 F1 scores after resampling at the domain level, while SVM with BOW embedding outperformed other models with 0.90 F1 scores at the category-level and 0.95 F1 scores at the raw-tags-level.

The structure of this paper is organized as follows: In section 2 a body of related work is studied. Section 3 presents methods in our multi-stage proposed architecture. In Section 4, we discussed experimental results and model comparative studies such as ZSL labeling assignments and hierarchical text classification. Finally, section 5 concludes the paper.

II. Related Works

A. Self-Supervised Learning

Unlike supervised and semi-supervised learning methods that require labeled documents, zero-shot learning is a form of self-supervised learning that can generalize a model trained on a labeled set for a specific task to a different task without requiring further training on any of the new labels. This is based on the transfer of knowledge from seen classes to unseen classes based on semantic attributes, descriptions of all classes, correlations among classes [17]. BART is known for fine-tuning for topic labeling tasks for multi-genre datasets. It matches the performance of RoBERTa with comparable training resources on GLUE and SQuAD, achieves new state-of-the-art results on a range of topic labels generation, abstract dialogue, question answering, and summarization tasks, with gains of up to 6 ROUGE [24]. BART's flexibility and performance on a variety of tasks allow researchers to bypass the need of having a large amount of properly annotated training data.

B. Hierarchically Classifying Documents Using Very Limited Information

Instead of building a single huge classifier for the entire task, the structure of topics can help to break down the classification problem into a manageable sizes. Using one classifier in the coarse level labels, and another classifier in the finer levels of labels so that classifiers are facing much simpler sub-tasks. Experiments show that the performance of Naive Bayes appears to yield the same performance in both flat classification and hierarchical classification, but more complex models such as KDB1 provide an 80 percent reduction in error with significance $P < 0.01$ in hierarchical classification [1].

Instead of approaching this task with a single large classifier, they used two separate classifiers to form a hierarchical classification structure. They leverage the structure of the ontology in order to reduce the classifier's scope. the first classier used the coarse-level labels while the second classifier is for the finer-level of labels. Previous experiments showed that the performance of Naive Bayes yielded the same performance in both flat classification and hierarchical classification use cases, but more complex models such as KDB1 provided an 80 percent reduction in error with significance $P < 0.01$ in hierarchical classification [1].

III. Proposed Methodology

In this section, we present our problem statement and an overview of our methodology.

Context plays a key role in text classification using a standard vocabulary. For instance, a categorical label such as "Healthcare Application" provides enriched semantics when it is

modeled within a context such as "Machine Learning". The goal of this work is to classify scientific documents with context-aware standard vocabulary.

Given a dataset $D = \{d_i\}$, such that $1 \leq i \leq n$, d_i indicates a text document represented with a feature vector in form of n-grams, extracted from the title and abstract of a document, and n represents the number of documents. For instance, we want to classify scientific literature with extracted tri-grams such as "exploits machine learning", "link browser fingerprints", bi-grams such as "tracking users", "long duration", "browser fingerprinting" and uni-gram such as "fingerprints", we would know the scientific literature belonged to either the domain "Machine learning", with the category "Machine learning applications", or the domain "Privacy and Security", with the raw tag "Machine learning and AI security". It's extremely difficult to classify the scientific literature into an accurate domain, category, or raw tag without extracting meaningful phrases or collecting context-aware labels.

In order to collect context-aware labels, given all the scientific literature as vocabulary context, we collected labels $L_h = l_i$ such that $l_i =< l_i, ...d_m >$ where l_i was each label at different granularity level (eg. domain as Machine learning, category as deep learning, raw tag as deep reinforcement learning) and m represented the number of labels. We used an ontology to organize L_h in hierarchical structure.

Next, we generated numerical vectors as readable input for text classifiers, given $J = f_1(D)$ where f_1 is the word embedding applied to n-grams, we got numerical vectors J that closely represented document contents.

To assign all labels L_h to the dataset D. With $Z_{h1} = f2(D, J, L_h)$ where $f2$ was the development of Zero-shot Learning (ZSL) model to automatically label documents at different granularity levels, and Z_{h1} was the labeled dataset. Z_{h1} was chosen either from $f2(D, L_h)$ or $f2(J, L_h)$ based on whichever method yielded a more accurate labelled dataset.

To verify the automatic label assignment Z_{h1} matches how manual labeling of documents was done, $f3$ was applied to Z_{h1} so that we get $Z_{h2} = f3(J, Z_{h1})$ where Z_{h2} represents the finalized accurate labelled training data at each granularity level. Finally, after the split of training and testing the labelled data Z_{h2}, given $R_h = f4(Z_{h2})$ where $R_h = r_i$ s.t. $r_i =< r_1, ...r_n >$ and r_i was each classified document at the finest granularity level (eg. raw tags), $f4$ was the development of several text classifiers at different hierarchical levels (eg. domains, categories, raw tags), and n represented the number of documents.

Overall, to generate the classified scientific literature R_h, we summarized the solutions to our problem statement as the mathematical formula below:

$$R_h = f4(f3(J, f2(f1(D), f1(D), L_h))) \tag{1}$$

The following subsections will elaborate details about methodologies including metadata enrichment and Ontology, feature extraction, dataset labeling, and labeling verification, and the comparative study on the tri-level document classification.

A. Meta Data Enrichment and Ontology Construction

What motivated us to enrich our labels was the low text classification performance by a flat model on only 30 labels collected from trending technological terms based on Gartner Hype Cycle [18]. The best model had only a 0.50 F1 score in classifying thousands of documents in these 30 labels. Besides, 30 labels were too general to classify a document into a fine granularity level.

In order to come up with a number that best describes how many topics these documents can be classified, we tried a Self Organizing Map (SOM) on BOW embedding based on extracted

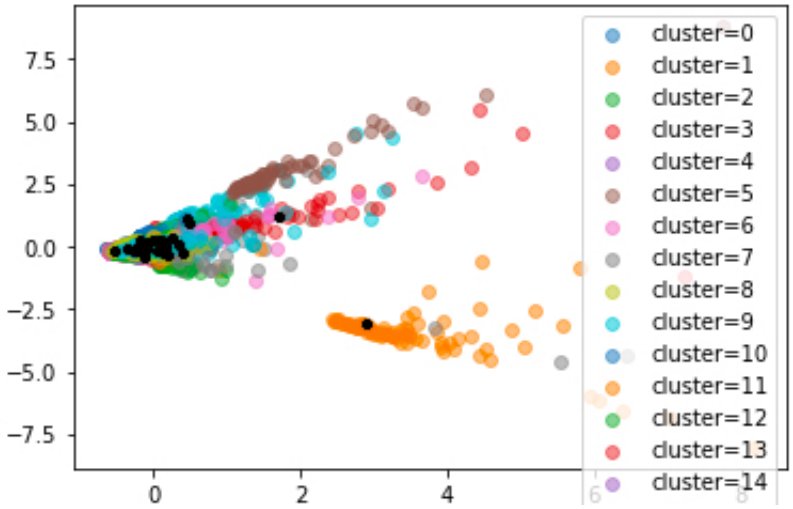

Figure 2. The SOM shows the fuzzy clustering on BOW embedding based on extracted n-grams from documents. This clustering is done on 5% of random sampling data for visualization purposes.

n-grams to study document distributions. SOM showed fuzzy clustering presented in figure. 2. This made it hard to know the actual number of clusters. The fuzzy clustering indicated that the document might contain overlapping content, and they could be better classified with hierarchical granularities, from the coarse level labels to the finer level labels. This motivated us to collect more labels, and to organize them in hierarchical structures. To enrich our standard vocabularies to label Google Scholar and dblp [4] [15] scientific literature with high degrees of confidence related to thousands of different domains, we scraped the *Call-For-Papers* section in conference web pages to obtain domains, categories, and raw tags based on conference names and years from our dataset.

Due to the hierarchical nature of the collected domains, categories, and raw tags, we manually created an ontology using Protege software [10] to organize all these labels from coarse to fine levels of information. figure. 3 contains the ontology that presents an example of the organization of hierarchical labels. In this Ontology, the *domain* superclass represented the highest level of topics across conferences. Since subclasses represented concepts that were more specific than the domain superclass, we set categories as subclasses. Raw tags were the finest level, so we set raw tags as instances of the subclass categories to constitute a knowledge base for research topics [2]. The ontology can be reused so that future topics can be directly added at a certain hierarchical level [6], [8], [23]. The ontology helped to determine the labels for text classification at each granularity level. We first performed text classification at the domains level (such as Machine Learning, Field programming devices, Manufacturing, design, and EDA, etc.). Next, we collected all the documents classified to the specific domain such as Machine Learning, and we used categories under the domain Machine Learning as labels (e.g., Machine Learning Applications, Deep Learning, Learning Theory, etc.), and performed text classification again. For documents classified to Machine Learning Applications, they were assigned with labels at the raw tags level under Machine Learning Applications (e.g., Crowdsourcing, Healthcare, Social Good, etc.) and the final text classification would classify documents into the raw tags level.

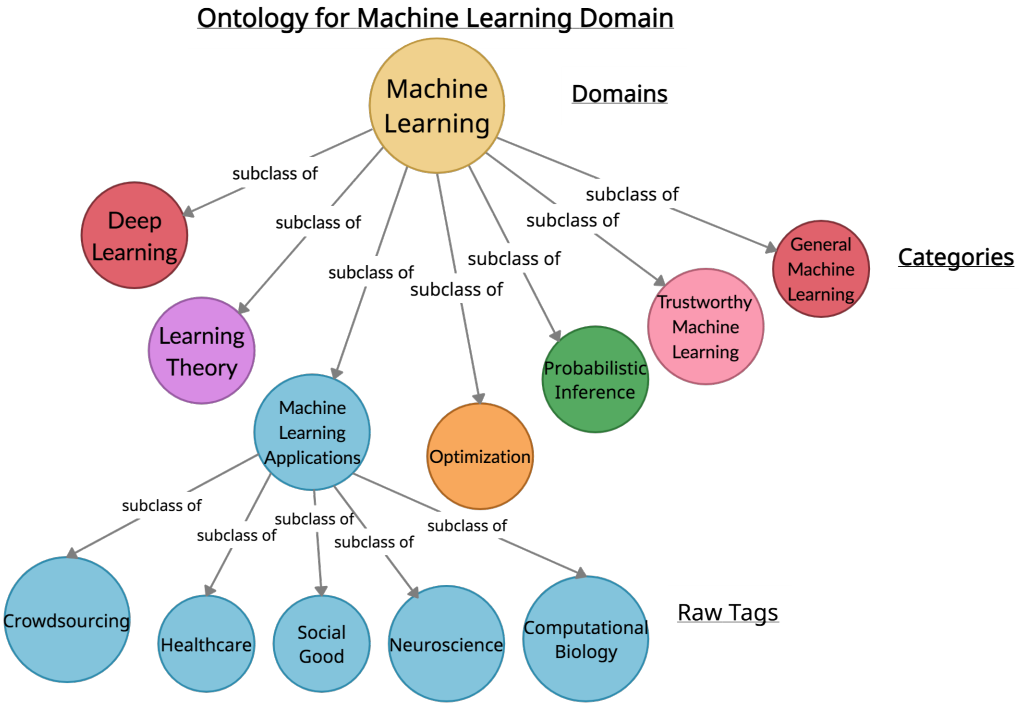

Figure 3. The ontology shows the hierarchical structure for labels for hierarchical classification in the Machine Learning domain. This ontology only showed the branch of the Machine Learning domain from coarse to fine levels. The details of other domains, categories, and raw tags were omitted.

B. Feature Extraction

After organizing the labels hierarchically, we collected the title and abstract for each document as a single input to the text classification pipeline. We extracted features from each input in the form of n-grams by stop words removal, tokenization, and additional splitting and filtering to obtain n-grams in the format of uni-grams, bi-grams, and tri-grams. Since all the text classifiers require the input to be represented as numerical vectors, word embedding is essential to transform human understanding of text into machine-readable vectors. We applied word embedding on n-grams so that titles and abstracts were presented as n-dimensional vector space. We studied the performance of word embedding techniques including Bag of Words (BOW), Word2vec, and GloVe [9], in conjunction with several text classifiers before and after re-sampling the result of word embedding. The outcome of these experiments is discussed in *Experimental Results* section.

To extract meaningful n-grams, the document titles and abstracts had special characters filtered out and then tokenized. With stop words acting as boundary words, the words between two boundary words were considered candidate words to join a phrase. All the candidate words were joined together in the form of uni-grams, bi-grams, and tri-grams. For long phrases with more than 3 words, we split each long phrase into two bi-grams, or a combination of bi-gram and tri-gram, or a combination of uni-gram and tri-gram. So we ensured the phrases were short and meaningful. As for word embedding, we applied Word2vec, Bag of Words (BOW), and GloVe embedding on n-grams extracted from titles and abstracts. We studied different combinations of

word embedding and classifiers to get the word embedding that yield the highest F1 score.

1) Word2Vec: Word2Vec is a feed-forward neural network-based model that predicts all of the neighboring words for every occurrence of every word in an entire corpus. Word2vec is good at capturing the local statistics of a corpus. It assigns similar numerical values to similar phrases because those are the values that minimize the loss function. The loss function for Word2Vec is the difference between the weights assigned to the word vectors. We used Word2Vec because we wanted to capture the phrase context within documents. The algorithm we used for Word2vec was Skip-gram (SG) because we wanted to guess the context words for each word, and SG is known for good performance for rare words or phrases which we wanted to capture in rare documents [21].

2) BOW: BOW counts how many times a word appears in a document. Each document was represented as a bag of word vectors. Each phrase in the document was regarded as a feature, and the occurrence of each feature was recorded in the numerical vector that represented each document. We used BOW because different documents shared the same phrases. BOW embedding gave similar vectors to documents with similar occurrences of the same phrases.

3) GloVe: Compared to Word2vec, GloVe can capture both global and local statistics of a corpus. GloVe embedding is based on matrix factorization techniques on the word-context matrix with co-occurrence information. GloVe allows us to take a text corpus and transform each word into a position in a high dimensional space so that similar words are placed together [9]. Below is the cost function about how Glove creates the word vectors:

$$J = \sum_{i,j=1}^{V} f(X_{ij})(w_i^T \widetilde{w_j} + b_i + \widetilde{b_j} - logX_{ij})^2 \tag{2}$$

V is the size of the vocabulary, and X is the co-occurrence matrix,X_{ij} is occurrences of i_{th} word in j_{th} word context, for example, the i_{th} word is "high-performance computing", and the j_{th} word is "high-performance memory". X_{ij} is the occurrence of "high-performance computing" with "high-performance memory". f is a weighting function to reduce the effect of long-tail, w_i is word vector for i_{th} word, and w_j is context word vector for j_{th} word, and b_i and b_j are bias values for i and j. Glove intended to minimize this cost function, so it assigned similar numerical vectors to documents with similar phrase pairs [9]. In this example, those documents with co-occurring phrases such as "high-performance computing" and "high-performance memory" were assigned with similar numerical vectors by Glove.

C. Dataset Labeling and Labeling Verification

Labeling Verification visualized the resulting clusters, the count of documents in each cluster, and the frequent n-grams per cluster to verify the labeled assignments. With all n-grams and all hierarchical labels, we performed label assignments percolating down from coarse level i.e., domains to finer levels as categories and raw tags. We used the BART method for Zero-Shot Learning (ZSL) at each granularity level. Next, for labeling verification, ward-linkage agglomerative clustering was applied on the BOW embedding based on n-grams extracted from titles and abstracts. The most frequent n-grams in each cluster were manually checked to ensure that their content was compatible with the ZSL assigned labels.

1) Dataset Labeling with ZSL: As for ZSL, the transformer we used was Facebook's BART-large (BART), a large pre-trained model with 12 encoder and decoder layers and 400M parameters [19]. The model we used was one that has been pre-trained on the MNLI (Multi-genre Natural Language Inference) dataset [11]. BART is an effective model for sequence or text

classification as it includes encoders and decoders that are connected by cross-attention. BART is trained by corrupting documents and then optimizing a reconstruction loss, which is the cross-entropy between the decoder's output and the original document. Inputs to the encoder do not need to be aligned with decoder outputs, allowing random noise transformations. The input is a corrupted document (left), and it's encoded with a bidirectional model. The likelihood of the original document (right) is calculated with an autoregressive decoder [19]. The details of the schematic of BART are illustrated in figure 4.

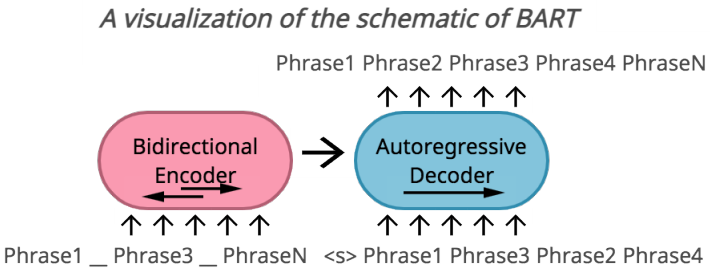

Figure 4. The left block shows that the corrupted document is input into the bidirectional encoder. The right block represents the decoder where the autoregressive decoder calculates the likelihood of the original document.

Each decoder layer performs attention over the final hidden state of the encoder output [19]. This design allowed BART to assign labels that were closely related to the original input as extracted n-grams from titles and abstracts. We used the BART-large transformer to label the dataset at each granularity level. We also studied the performance of ZSL with BART on our extracted n-grams and word embeddings (Glove, Word2vec, BOW) corpus. The text corpus contained all the extracted n-grams and the labels with all the technical phrases from the domains level. We only chose the domains level to study ZSL because the domains level contained the most diverse topics. If ZSL performs well at the domains level, we assume that it also performs well at the categories and raw tags level, because the label assignments range in the finer granularity levels (categories, raw tags) are narrowed down to each domain with less noise from other topics. For example, considering assigning categories labels such as "General Machine Learning", "Deep Learning" and "Optimization", the label assignment was performed on the text corpus that was labeled with the domain "Machine Learning". Topics became narrower in the finer levels, so the noises from other domains at the finer levels (categories, raw tags) were reduced. The experimental results about the kernel density of confidence scores are displayed in 8. Details about ZSL confidence scores on the text corpus and word embeddings are discussed in *Experimental Results* section. BART with the highest confidence score trained at domains level was developed to label documents at categories and raw tags level.

2) Labeling Verification with Ward-linkage Agglomerative Clustering: In order to get the most frequent n-grams from each distinct cluster based on document content, we chose the ward linkage for agglomerative clustering to categorize documents, because ward linkage can merge the pair of clusters that minimize the variance of the clusters being merged. This helped categorize documents into pure and unique clusters. We chose the metric euclidean to compute the linkage of agglomerative clustering because the linkage ward only accepts euclidean metrics. The ward-linkage agglomerative clustering applied on BOW embedding of documents' abstracts and titles

showed around 9 clusters at the threshold level of 10000 documents. As a result, cluster sizes were imbalanced, corresponding to the label distributions based on the ZSL label assignments at all granularity levels. Our manual verification started from the most general clusters and per located down to their subclusters. According to the cluster size and the frequency of the assigned labels, we visualized the most frequent n-grams within hierarchical clusters and manually verified that the content of n-grams matched the ZSL assigned labels. figure. 5 shows the cut threshold and a number of estimated clusters by ward-linkage agglomerative clustering.

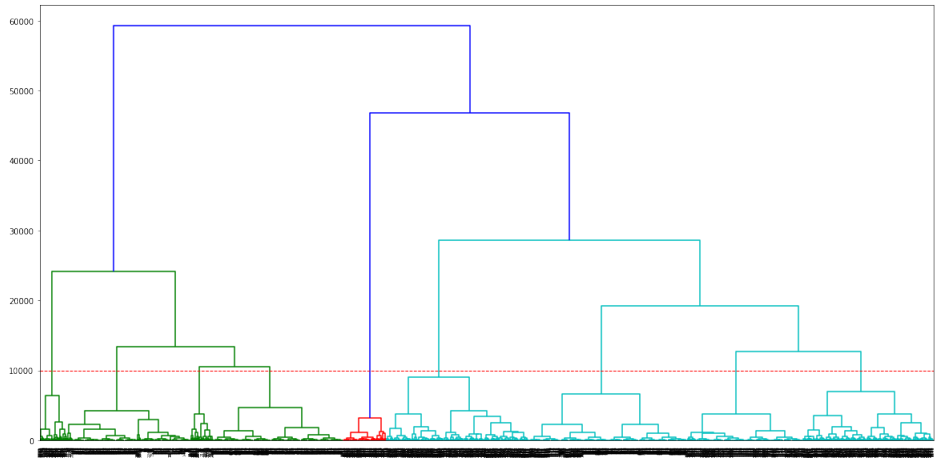

Figure 5. The dendrogram shows the hierarchical clusters by ward-linkage hierarchical clustering. We selected the threshold as 1000 documents, and the threshold seperated documents into approximately 9 clusters.

In order to verify the matching between clusters' content and labels, we gathered the top 6 most frequent phrases in unigrams, bigrams, trigrams and visualized them in the form of pie charts. Most unigrams contained common phrases such as "behavior", "framework" and "process". Most bigrams provided useful information that distinguished topics across clusters. For example, while doing the manual label verification at the domains level, the largest cluster contained bigrams such as "power dissipation", "energy efficiency", "embedding system" and "case study". So we got an intuition that most documents were related to the functionality of computers. As for the smallest cluster, the most frequent bigrams contained "virtual machine", "high sensitivity", "performance degradation" etc. We expected that documents related to performance and testing took up the smallest percentage across the entire corpus.

With this base knowledge, at the domains level, we manually checked the most popular label assigned by ZSL related to the functionality of computers, and the least popular label assigned by ZSL related to performance and testing. In order to verify label assignment at the next granularity level, we directly accessed the subclusters and visualized the frequent n-grams within clusters. Based on the frequency of ZSL assigned labels, the size of each cluster and the most frequent n-grams in clusters, we conducted manual checking for the rest of the ZSL assigned labels at each granularity level.

figure. 6 shows the pie charts with the n-gram frequencies in the largest cluster based on ward-linkage agglomerative clustering. The top 6 phrases with frequency larger than one from each type of n-gram were selected to represent the cluster content. The actual content of the phrases was manually analyzed to ensure the most frequent label at the domains level correlated with the selected phrases.

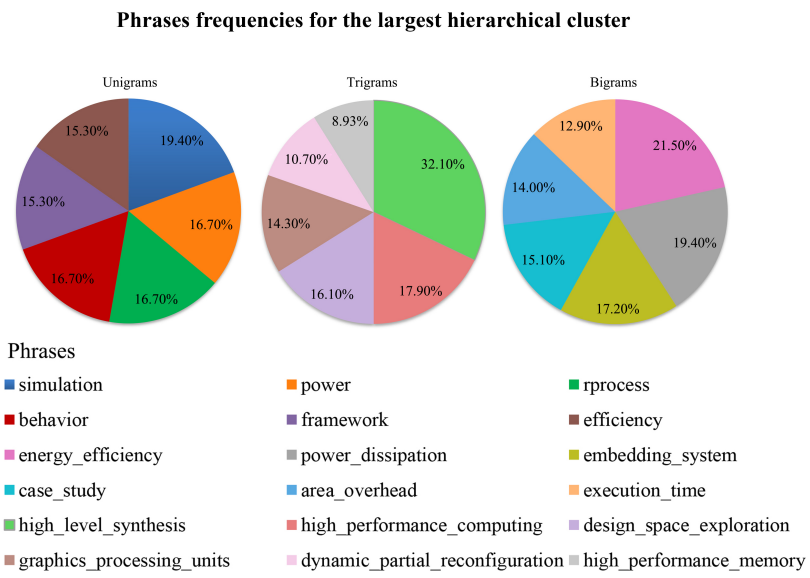

Figure 6. The pie charts show the n-gram frequencies in the largest cluster based on ward-linkage agglomerative clustering.

figure. 7 shows the pie charts with the n-gram frequencies in the smallest cluster based on ward-linkage agglomerative clustering. The top 6 phrases with frequency larger than one from each type of n-gram were selected to represent the cluster content. The actual content of the phrases was manually analyzed to ensure the most frequent label at the domains level correlated with the selected phrases.

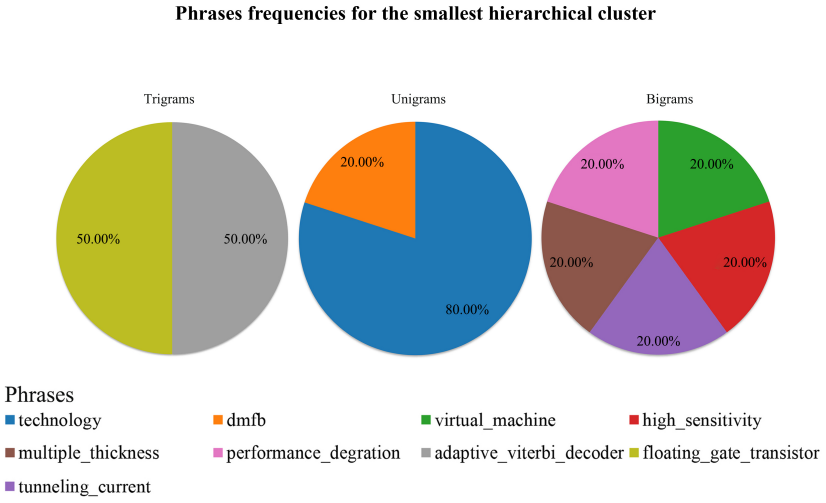

Figure 7. The pie charts show the n-gram frequencies in the smallest cluster based on ward-linkage agglomerative clustering.

D. Resampling

Due to the uneven label distributions from the ZSL label assignment results presented in figure. 10 and the uneven cluster size by the ward-linkage agglomerative clustering, we compared the performance of word embedding methods combined with classifiers before and after resampling. As for resampling methods, we tried Synthetic Minority Over-Sampling Technique (SMOTE), Random Oversampling, Random Undersampling, and the hybrid resampling method with Random Oversampling and Random Undersampling. Since the hybrid resampling method yielded the most balanced word embedding with all labels obtaining a similar amount of embedding vectors, we selected the hybrid resampling method to resample word embedding. All classifiers' performances were evaluated with an F1 score before and after resampling.

E. Comparative Study on the Tri-level Document Classification

First, the labeled abstracts were split into the proportion as 80/20 ratio for training and testing. Next, we studied several machine learning algorithms and reported the classifier with the highest F1 score. We chose the F1 score because the data was distributed unevenly based on label assignment results. Our analysis compared Logistic Regression (LR), Support Vector Machine (SVM), Gradient Boosting (GB), Recurrent Neural Network (RNN), and Feed Forward Neural Network with BOW embedding (BOW Model) based on F1 score metric to select the best combination of word embedding and a classifier at each granularity level.

1) LR: We used LR as a baseline model to compare the model performances with more complex models. Even Though Naive Bayes was famous for its effectiveness in text classification, Naive Bayes was not chosen as the baseline model because the word embedding Word2Vec generated negative values which were incompatible with Naive Bayes. LR was chosen because of its fast classification of unknown values, and it adjusted well to linear text distribution. Logistic regression was trained on both BOW and Word2vec embedding at each granularity level.

2) SVM: We chose SVM because it was effective in high-dimensional space, which was suitable for the high-dimensional word vectors created by BOW embeddings and Word2vec embeddings. With the assumption of Gaussian distributed data, the Support vector machine aims to find a hyperplane or a set of hyperplanes in an n-dimensional space that distinctly classifies the data points. The objective of the SVM was to find a hyperplane in n-dimensional space such that the margin is maximized. SVM was trained on both BOW and Word2vec embedding at each granularity level.

3) GB: We choose GB because of its good adjustment to imbalanced data distribution as we observed from the result of zero-shot learning (ZSL) label assignment, and GB provides parallelization in tree building. GB is an iterative functional gradient algorithm. It minimizes a loss function by iteratively choosing a function that points towards the negative gradient [7]. GB was trained on both BOW and Word2vec embedding at each granularity level.

4) RNN: In our model comparison study, we only trained RNN with Glove embedding, because we wanted to study classification performance with the order of phrase pairs preserved in each document. We used Glove embedding to create word representations that capture co-occurrence phrases and using RNN to capture the order of the n-grams in each document. Other models with the combination of either BOW or Word2vec embedding didn't achieve this. RNN makes use of sequential data when the current step has some kind of relationship with the previous steps. RNN is good at classification in a long-range semantic dependency [13]. Even though RNN is famous for time component application, our dataset might have word pairs that

often appear to be together in a document, which satisfied the sequential data requirement for RNN. Below is a formula about how RNN classifies documents:

$$a^{<t>} = g_1(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a) \tag{3}$$

$$y^{<t>} = g_2(W_{ya}a^{<t>} + b_y) \tag{4}$$

For each document, at each time step t (word phrase), each word phrase in the document goes through activation $a^{<t>}$, and generates the output $y^{<t>}$. $W_{ax}x$, W_{aa}, W_{ya}, b_a, by are coefficients that were shared temporarily and g1, g2 are activation functions. As for the architecture of RNN, we used the embedding vector length with a maximum number of words fewer than 200, sigmoid activation function, binary cross-entropy loss function, 32 batch size, and 20 epochs.

5) BOW Model: BOW Model is a simple feedforward neural network model with the input as BOW embedding. We used the BOW Model to compare the simple neural network performance with the complex neural network models such as Gradient Boosting with BOW embedding before and after resampling the dataset.

IV. Experiment Results and Discussions

A. ZSL Confidence Scores

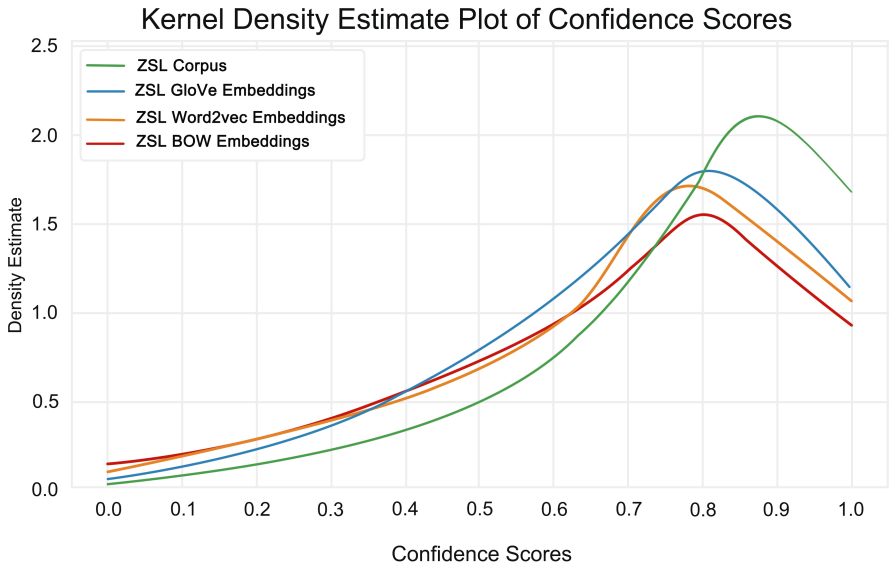

Figure 8. All ZSL models were trained with BART with all the domains level labels. The kernel density estimate plots showed ZSL trained on extracted n-grams from text corpus, Glove embeddings, Word2vec embeddings and BOW embeddings.

figure. 8 displays the kernel density estimate plot of confidence scores for BART trained on text corpus (extracted n-grams) and word embeddings (Glove, Word2vec, BOW) with all domains labels. The peaks of the kernel density plot told us the concentrated values of the confidence scores. ZSL trained with Corpus yielded the highest concentrated confidence score 0.85. The next highest was ZSL trained with Glove embedding with a concentrated confidence score of 0.80. Based on the high performance of ZSL with BART on text corpus (extracted n-grams), we decided to label documents with BART based on the corpus (extracted n-grams) for the domains,

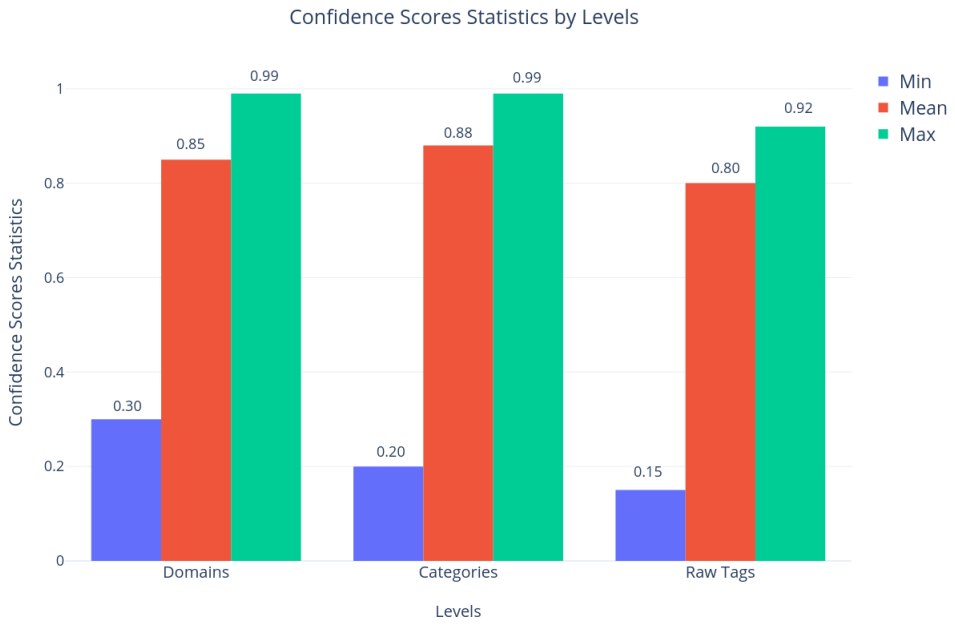

Figure 9. Bar plots show the min, mean, and max confidence scores for the ZSL label assignment results at domains, categories and raw tags levels.

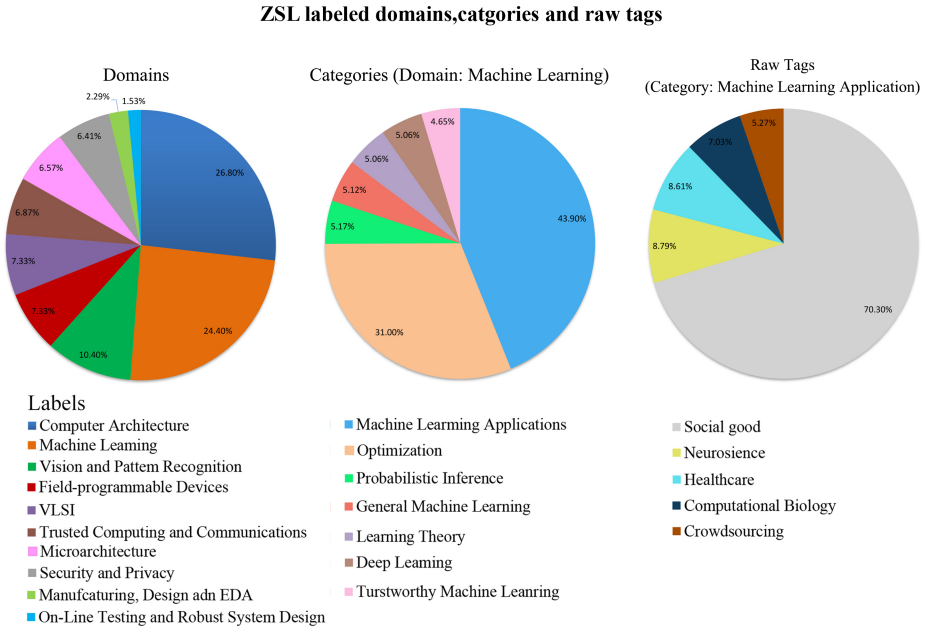

Figure 10. Pie charts show the distributions of labels at coarse level Domains to finer levels as categories and raw tags.

categories, and raw tags levels. figure. 9 contains the bar plots with the min, mean and max confidence score based on ZSL. Categories had the highest mean confidence score 0.88. Even though raw tags obtained a relatively lower confidence score of 0.80, we have manually checked

the labeling results and ensured most documents at raw tags levels were reasonable.

Based on figure. 10, the label distributions were not balanced at each granularity level. For the presented pie chart, the displayed categories were under the domain “Machine Learning”, and the displayed raw tags were under the category “Machine Learning Applications”.The major label at the domains level was “Computer Architecture” with 26.80% of all domains labels. At the categories level under the domain “Machine Learning”, the major category label was “Machine Learning Applications” with 43.90% of all categories labels. At the raw tags level, the major label was “Social good” under the category “Machine Learning Applications” with 70.30% of all raw tags labels. These imbalanced label distributions motivated the use of the F1 score as the evaluation metric and the resampling for further text classification.

B. Comparative Study of Word Embedding and Text Classifiers

figure. 11 shows the F1 score at each classification level before and after resampling for BOW embedding. figure. 11-A shows the results of text classifiers before resampling, and figure. 11-B. shows the results of text classifiers after resampling. Text classification F1 scores increased at finer granularity levels before and after resampling.

figure. 12 shows the F1 score at each classification level before and after resampling for Word2vec embedding. figure. 12-A shows the results of text classifiers before resampling, and figure. 12-B. shows the results of text classifiers after resampling. Text classification F1 scores increased at finer granularity levels before and after resampling.

According to figure. 11 and figure. 12, models yielded better performances in classifying documents at finer granularity levels. Most classifiers had improved F1 scores at finer granularity levels since the documents covered much narrower topics at categories and raw tags levels with less noise from other topics. For example, classified documents that belonged to the domain ”Machine Learning” required further classification at the categories level, but the categories under other domains such as ”Security and Privacy” were no longer considered as labels for these documents. We can take the performances of Gradient Boosting (GB) as proof, before resampling, as it is shown in the figure. 11-A, GB with BOW embedding had improved 17% F1 score from domains to categories and improved 47% F1 score from categories to raw tags level. Meanwhile, as it is shown in the figure. 12-A, before resampling, GB with Word2vec embedding had improved 29% F1 score from domains to categories level and improved 36% F1 score from categories to raw tags level. Based on figure. 11-A and figure. 12-A, even though LR, SVM with both BOW and Word2vec embedding obtained high F1 scores above 0.80 before resampling at raw tags level, they performed badly at domains and categories levels. The low F1 scores were caused by the imbalanced data problem. This motivated us to resample data to achieve consistent performance at all granularity levels.

1) Model Performances after resampling: Resampling helped simple models such as LR, SVM with both BOW and Word2vec embedding to improve performances at all granularity levels, but most models with BOW embedding outperformed models with Word2vec embedding after resampling. Before resampling, for example, according to figure. 11-A, LR with BOW embedding reached 0.15 F1 score at domains level, 0.30 F1 score at categories level, and 0.82 F1 scores at raw tags level. According to figure. 12-A, LR with Word2vec embedding reached 0.20 F1 score at domains level, 0.36 F1 score at categories level, and 0.85 F1 scores at raw tags level. In contrast, after resampling, according to the figure. 11-B, most models with BOW embedding such as LR and SVM improved F1 scores above 0.80 at all granularity levels. According to figure. 12-B, LR, and SVM with Word2vec embedding obtained relatively lower F1 scores below 0.80 at domains and categories levels. Besides, taking GB to compare the performances

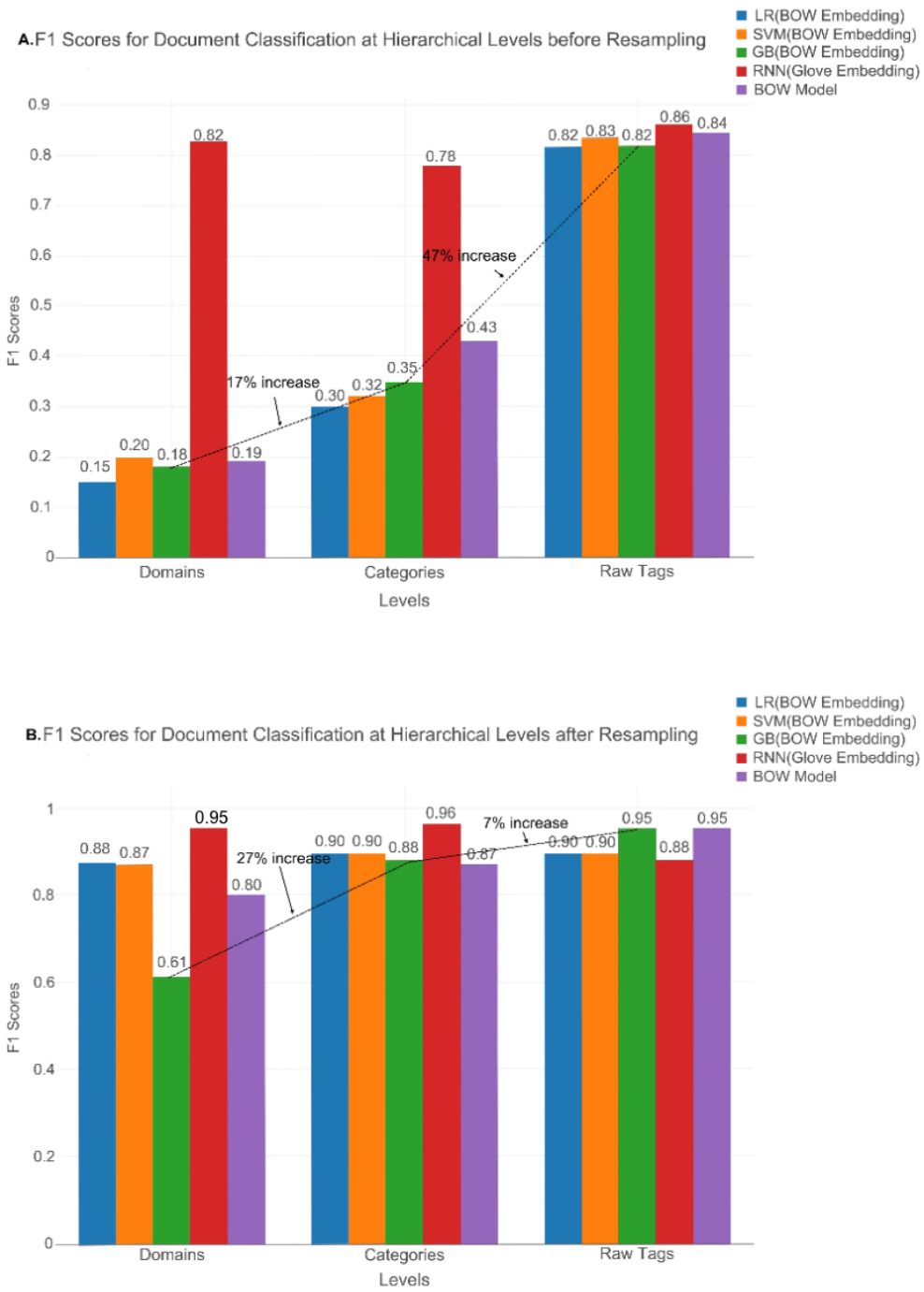

Figure 11. At each classification level, BOW Embedding was applied for LR, SVM, GB, and BOW Model. Glove embedding was applied for RNN.

on BOW and Word2vec embedding after resampling, based on the figure. 11-B, GB with BOW embedding reached F1 score 0.61 at domains level, F1 score 0.88 at categories level, and F1

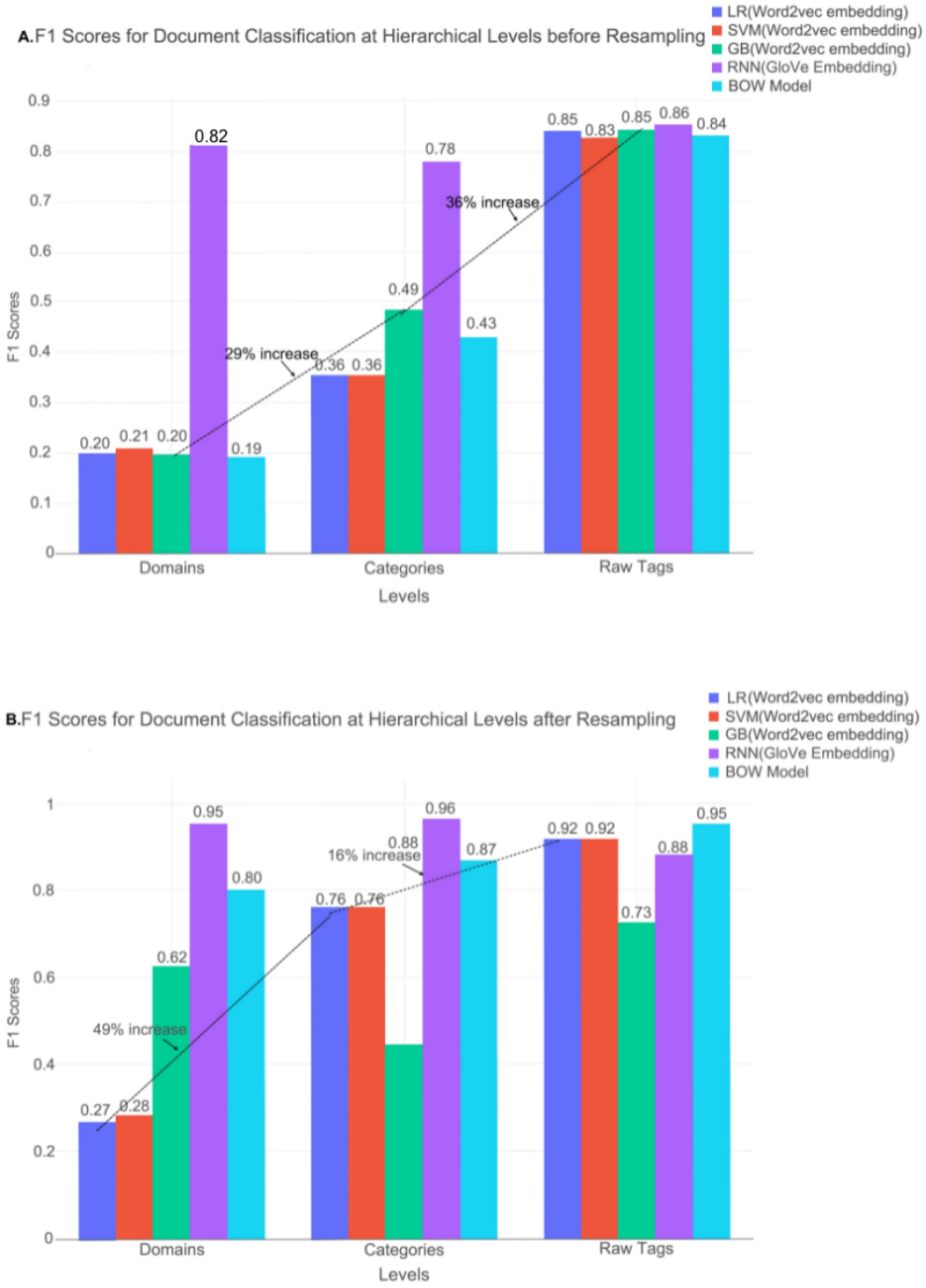

Figure 12. At each classification level, Word2vec Embedding was applied for LR, SVM, and GB. Glove embedding was applied for RNN, and BOW embedding was applied for BOW Model.

score 0.95 at raw tags level. In comparison, based on the figure. 12-B, GB with Word2vec embedding reached F1 score 0.62 at domains level, but obtained a lower F1 score of 0.45 at

categories level, and lower F1 score of 0.73 at raw tags level.

2) Discussions about Text Classifiers with High Performance: There was one model that remained consistent with high performances before and after resampling at all granularity levels. According to figure. 11 and figure. 12, RNN with Glove embedding reached high F1 scores no lower than 0.78 before resampling at all levels. After resampling, it also reached the highest F1 score 0.95 at the domains level, the highest F1 score 0.96 at the categories level. However, after resampling at raw tags level, BOW Model with 0.95 F1 score beat RNN with Glove embedding, regardless of the high F1 score 0.88 from RNN with Glove embedding at raw tags level. The high performances of RNN with Glove embedding at domains, categories, and raw tags levels might be because that co-occurrences of phrases and sequential information were more valuable in distinguishing documents across more diverse topics. RNN's ability in capturing ordered pairs of phrases to compare documents was similar to how we manually compared documents across a large number of topics by comparing documents' abstracts based on similar sentences. The BOW Model obtained the highest performance at raw tags level after resampling might be because that word representations of rare phrases were duplicated after resampling, and models required more word representations of rare phrases to classify documents in more uncommon topics. Besides BOW Model, other simple models such as LR and SVM with BOW embedding both reached high F1 scores above 0.90 after resampling at categories and raw tags levels. To conclude, simple models were able to perform well at raw tags level with smaller amounts of features before resampling and might be better at classifying rare documents after resampling, but a larger and more complex model such as RNN with Glove embedding performed well at coarser levels with larger amounts of features and more noises.

V. Conclusion

In this paper, we propose a novel multi-stage machine learning pipeline that utilizes self-supervised learning, Ontology, ward-linkage agglomerative clustering, and hierarchical document classification to classify research papers into specific trending fields. The construction of the ontology decomposed labels in 3 granularity levels, and the application of agglomerative clustering verified that the hierarchy of documents was consistent with the hierarchy of labels. This hierarchical decomposition on both labels and documents' content has simplified the large-scale automatic labeling and multi-class classification, which allowed even simple models to perform well at the finer granularity levels and more robustness to model performances across all granularity levels. Our pipeline classified unlabelled published papers in the field of CS and CE with high F1 scores across all granularity levels. For example, RNN with Glove Embedding yielded the highest F1 scores no less than 0.95 in the domain and category levels, and BOW Model yielded the highest F1 score of 0.95 in the raw tag level. Our proposed multi-layer classification framework automates the process of document labeling, while yielding a high F1 score (no less than 0.95) in all granularity levels after resampling. By modeling our multi-layer vocabulary (i.e., domain, category, and raw tag level) our context-aware classification framework showed to be way more effective that other works such as [22] which applied a deep attentive neural network trained with a manually labeled dataset. Our proposed approach is also capable of classifying continuous stream of raw text related to documents over time. Our novel context-aware hierarchical documents classification technique can be applied to any research domain with different technical backgrounds. The Initial results showed that our approach could yield an effective tool for researchers of all levels and fields.

REFERENCES

[1] D. Koller and M. Sahami. Hierarchically classifying documents using very few words. In Proceedings of the 14th International Conference on Machine Learning (ICML), (1997).

[2] N. F. Noy and D. L. Mcguinness, Ontology development 101: A guide to creating your first ontology, (2001).

[3] C. H. Caldas and L. Soibelman, Automating hierarchical document classification for construction management information systems, Automation in Construction, vol. 12(4), (2003), 395–406.

[4] Google Scholar, Google Scholar Top Publications, (2008).

[5] J. Xu, V. Singh, V. Govindaraju and D. Neogi, A Hierarchical Classification Model for Document Categorization, 2009 10th International Conference on Document Analysis and Recognition, (2009), 486-490.

[6] S. Rafatirad and R. Jain, Contextual augmentation of ontology for recognizing sub-events, 2011 IEEE Fifth International Conference on Semantic Computing, (2011), 546-553.

[7] A. Natekin and A. Knoll, Gradient boosting machines, a tutorial, Frontiers in Neurorobotics, vol. 7, (2013).

[8] S. Rafatirad, R. Jain, and K. Laskey, Context-based event ontology extension in Multimedia Applications, 2013 IEEE Seventh International Conference on Semantic Computing, (2013), 278-285.

[9] J. Pennington, R. Socher, and C. D. Manning, Glove: Global vectors for word representation, in Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), (2014), 1532–1543.

[10] M. A. Musen, The protégé project: a look back and a look forward, AI Matters, vol. 1(4), (2015), 4–12.

[11] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning, A large annotated corpus for learning natural language inference, In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP), (2015).

[12] Pushpankar Kumar Pushp and Muktabh Mayank Srivastava, Train once, test anywhere: Zero-shot learning for text classification, ArXiv, abs/1712.05972, (2017).

[13] W. Yin, K. Kann, M. Yu, and H. Schütze, Comparative study of CNN and RNN for natural language processing, ArXiv, abs/1702.01923, (2017).

[14] Wang, Alex et al, GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding, ArXiv, abs/1804.07461, (2018).

[15] Schloss Dagstuhl - Leibniz Center for Informatics, dblp computer science bibliography, (2019).

[16] S.-W. Kim and J.-M. Gil, Research paper classification systems based on TF-IDF and Lda schemes, Human-centric Computing and Information Sciences, vol. 9, (2019), 1-21.

[17] Yin, Wenpeng et al, Benchmarking Zero-shot Text Classification: Datasets, Evaluation and Entailment Approach, ArXiv, abs/1909.00161, (2019).

[18] Gartner Inc, 5 trends drive the gartner hype cycle for emerging technologies, (2020).

[19] Lewis, M. et al, BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension, ArXiv, abs/1910.13461, (2020).

[20] National Science Board, National Science Foundation, U.S. Trends and International Comparisons: Science and Engineering Indicators 2020, National Center for Science and Engineering Statistics (NCSES), (2020).

[21] Y. Wu, S. Zhao, and W. Li, Phrase2Vec: Phrase embedding based on parsing, Information Sciences, vol. 517, (2020), 100–127.

[22] B. Kandimalla, S. Rohatgi, J. Wu, and C. L. Giles, Large scale subject category classification of scholarly papers with deep attentive neural networks, Frontiers, (2021).

[23] C. Bandi, S. Salehi, R. Hassan, S. M. P D, H. Homayoun, and S. Rafatirad, Ontology-driven framework for trend analysis of vulnerabilities and impacts in IOT hardware, IEEE 15th International Conference on Semantic Computing (ICSC), (2021), 211-214.

[24] P. Cristian and R. Trainan, BART: Weakly-Supervised Topic Label Generation, Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics (EACL), (2021), 1418–1425.