TaxoClass: Human-like Classifier

Dayea Song
School of Computing
KAIST
Daejeon, Korea
sdy992000@kaist.ac.kr

Jaeseong Kim
School of Computing
KAIST
Daejeon, Korea
wotjd1102@kaist.ac.kr

Jinkyung Jo
Graduate School of Artificial Intelligence
KAIST
Daejeon, Korea
jinkyungjo@kaist.ac.kr

Yeongrok Kim
Department of Chemical and Biomolecular Engineering
KAIST
Daejeon, Korea
ff4270@kaist.ac.kr

***Abstract*—Hierarchical multi-label text classification is categorizing documents into multiple classes in the class hierarchy. Most HMTC methods need many human-labeled documents and it is too costly to obtain. In the paper we replicated, author proposed a novel HMTC framework, TaxoClass, as a human-like classifier to use only the class surface names. TaxoClass constructed with these four steps. First, calculates the similarity between document and class using a pretrained textual entailment model. Second, identify core classes of each documents. Third, do core class guided classifier training. Finally, generalize the text classifier through multi-label self-training. We follow these steps to implement our own HMTC framework TaxoReplica. We use relatively recent version of Amazon and DBPedia datasets to test our TaxoReplica. It achieved around 0.25 Example-F1 for Amazon dataset and 0.16 Example-F1 for DBPedia dataset. It is relatively low score compared with the paper. The limitations of the project with low accuracy were summarized in the conclusion. For further information, see https://github.com/team-corefinder/TaxoReplica.git**

I. Introduction

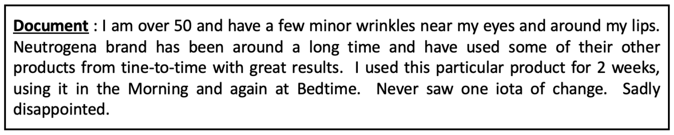
Our project is replicating paper, 'TaxoClass : Hierarchical Multi-Label Text Classification Using Only Class Names', which is accepted by NAACL 2021 [8]. Hierarchical multi-label text classification(HMTC) is categorizing documents with multiple classes in the class hierarchy. HMTC is useful in fields such as product categorization [9], semantic indexing [10], and fine-grained entity typing [11].

Most of methods for HMTC use supervised fashion. It first needs many human labeled documents and then train classifier for prediction. Despite high performance of this method, human labeling process takes a lot of time and cost.

Recent studies have been tried to solve this problem with small amount of labeled data. First, in semi-supervised method, abundant unlabeled data can assist model training on labeled data [12], [13]. However, it needs minimum labeled dataset including all classes in HMTC required and it could be expensive when number of classes is too large. Second, some weakly-supervised methods derive pseudo-labeled data by using class indicative keywords [7], [14], [15] or class surface names [16], [17]. These methods have some limitation such as document has only one class or all class surface names must appear in the document.

This is a paper being submitted to KAIST CS470 class as a final project.

In the paper, the author tried to solve this problem by following how human resolve the HMTC problem. When we are asked to assign multiple classes to a document, we will first find out 'core classes' that represent the important points of the document then check the ancestor classes in the taxonomy. Fig. 1 is an example of human labeling process. When human read this example document, they can quickly identify some core classes such as 'Lips', 'Face', 'Eyes', and 'Lip Care'. Following the ancestors of the most essential core class (in this example 'Face') will reveal the entire label.


__Document__ : I am over 50 and have a few minor wrinkles near my eyes and around my lips. Neutrogena brand has been around a long time and have used some of their other products from tine-to-time with great results. I used this particular product for 2 weeks, using it in the Morning and again at Bedtime. Never saw one iota of change. Sadly disappointed.

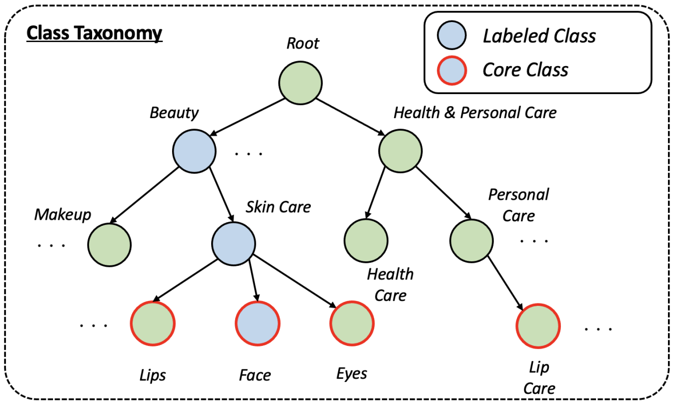


Fig. 1. An example of human labeling process.

Inspired by the human labeling process, the author proposed TaxoClass with three major steps and we follow these steps for implementing our HMTC framework, TaxoReplica. First, calculate the document-class similarity using pre-trained textual entailment model [1] and identify core classes. Second, train a text classifier that includes document encoder (pre-trained BERT based [2]) and class encoder. Finally, for generalization do the multi-label self-training.

II. Model

Our model consists of four parts: (1) document-class similarity calculation, (2) document core class mining, (3) core class guided classifier training, and (4) multi-label self-training. The details of each part will be explained below and Fig. 2 shows the architecture overview of our model.

A. Document-Class Similarity Calculation

We used the textual entailment approach proposed by [1] to calculate the semantic similarity between document and class pair. In particular, we used Roberta-Large-MNLI as our textual entailment model, which is a pretrained Roberta-Large model fine-tuned on the MNLI dataset. A document D_i and a class surface name s_j are used as inputs, and $P(D_i \rightarrow c_j)$, the probability that a document D_i belongs to a class c_j, is displayed as a output. We define this probability as document-class similarity $sim(D_i, c_j)$.

B. Document Core Class Mining

Core classes refer to a few classes that are most closely related to the document in the class taxonomy. In this chapter, we will describe how to find the core classes of each document.

- Core Class Candidate Selection: In order to find the core classes, we have to go through the entire class taxonomy and find a few classes, which is very inefficient. Therefore, we try to reduce the search space through the top-down approach as in Fig. 3. Let the starting level $l = 0$ for a given document D be the "Root" class. At level $l = 1$, the two children classes with the highest similarity to document D are selected, and in this way, at level l, select $l + 2$ classes and go down. After all these steps, among the collected children classes, choose the $(l+1)^2$ classes (at level $l+1$) with the highest path score(ps) defined as follows:

$$ps(c_j) = \begin{cases} ps(Root) = 1 \\ \max_{c_k \in Par(c_j)}\{ps(c_k) \cdot sim(c_j, D)\} \end{cases} \quad (1)$$

  where $Par(c_j)$ is class c_j's parent class set. In the chosen children classes, all classes except the "Root" class are called core class candidate set, and the candidate set for document D_i is denoted as $\mathbb{C}_i^{cand}$.
- Confident Core Class Identification: We will distinguish core classes from the selected candidate set above. For this, first, we define the confidence score as follows:

$$conf(D, c) = sim(D, c) - \max_{c' \in Par(c) \cup Sib(c)}\{sim(D, c')\} \quad (2)$$

  where $Sib(c)$ is the sibling class set of c. This confidence score is based on the assumption that core class c generally has a higher similarity than its parent and sibling classes.
  Second, if the core class of document D is c, the confidence score of c will be higher than the median confidence score of other documents with the core class c (denoted as $D(c)$). Therefore, we have:

$$conf(D, c) \geq median\{conf(D', c) | D' \in D(c)\} \quad (3)$$

  We define core class candidates that satisfy the above inequality as final core classes, and denote such a core class set as $\mathbb{C}_i$. Note that core class $\mathbb{C}_i$ may be empty if document D_i does not have any confident core class.

C. Core Class Guided Classifier Training

This chapter describes the process of training the classifier for hierarchical multi-label text classification with the document core class determined in the previous chapter. First, we explain about the architecture of classifier and then we will discuss about the training process.

- Text Classifier Architecture: Our text classifier has a dual-encoder structure. One is a document encoder and one is a class encoder. And the representations generated through encoders use a text matching network to calculate the probability that document D_i has core class c_j.
  The document encoder uses the pretrained BERT-base-uncased model introduced by [2]. In order to implement the class Encoder, we referred to the implementation of [6], and graph natural network (GNN) [3] was used. The class encoder we used captures information on class taxonomy and information that can be obtained through surface name. For Class c_j an ego network which includes the parent and child class of c_j is obtained. The GNN propagates the feature of the node to the node which belongs to the ego network. Node feature was initialized using pre-trained word embedding [7]. If the label surface name is consists of multiple words, concatenate words with under bar(_). For example, label name 'sports season' becomes 'sports_season'. For multi-gram class names which contains ',' or '&' , we use average of the word embeddings. For instance, feature of 'Home & Kitchen' is average embedding of the 'Home' and 'Kitchen' . When the word is not exists in the pre-trained embedding, embedding vector was randomly initialized. We defined the GNNs of L-layers as follows:

$$h_u^{(l)} = \text{ReLU}\left(\sum_{v \in N(u)} \alpha_{uv}^{(l-1)} \mathbf{W}^{(l-1)} h_v^{(l-1)}\right) \quad (4)$$

  where $l \in 1, \cdots, L$, $N(u)$ includes node u's neighbors and itself, $\alpha_{uv}^{(l-1)} = \frac{1}{\sqrt{|N(u)||N(v)|}}$ is a normalization constant (same for all layers), and $W^{(l-1)}$ are learnable parameters. Finally, the node feature is obtained as an

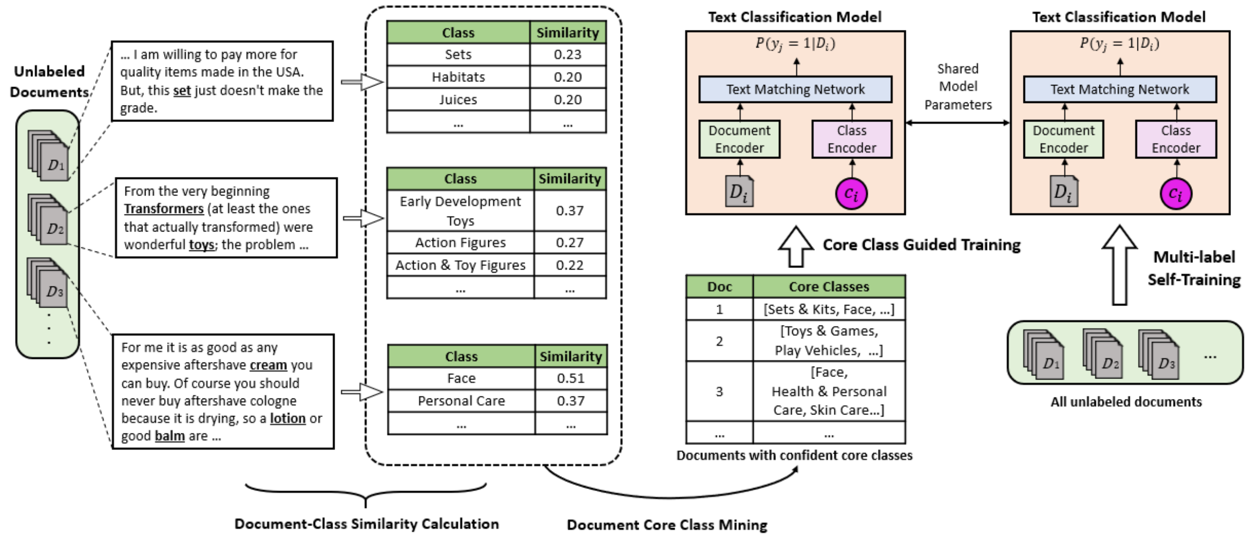

Unlabeled Documents
D_1
D_2
D_3
... I am willing to pay more for quality items made in the USA. But, this set just doesn't make the grade.
From the very beginning Transformers (at least the ones that actually transformed) were wonderful toys; the problem ...
For me it is as good as any expensive aftershave cream you can buy. Of course you should never buy aftershave cologne because it is drying, so a lotion or good balm are ...
Class | Similarity
Sets 0.23
Habitats 0.20
Juices 0.20
Early Development Toys 0.37
Action Figures 0.27
Action & Toy Figures 0.22
Face 0.51
Personal Care 0.37
Document-Class Similarity Calculation
Document Core Class Mining
Doc | Core Classes
1 [Sets & Kits, Face, ...]
2 [Toys & Games, Play Vehicles, ...]
3 [Face, Health & Personal Care, Skin Care...]
Documents with confident core classes
Core Class Guided Training
Text Classification Model
$P(y_j = 1|D_i)$
Text Matching Network
Document Encoder
Class Encoder
D_i
c_i
Shared Model Parameters
Multi-label Self-Training
All unlabeled documents

Fig. 2. Overview of our TaxoClass architecture. First, we calculate document-class similarity using a textual entailment model. Second, we find document core classes. Third, we train a text classifier using document core classes. Finally, we generalize the text classifier through multi-label self-training using the entire unlabeled documents. During self-training, we share the parameters of the text classifier.

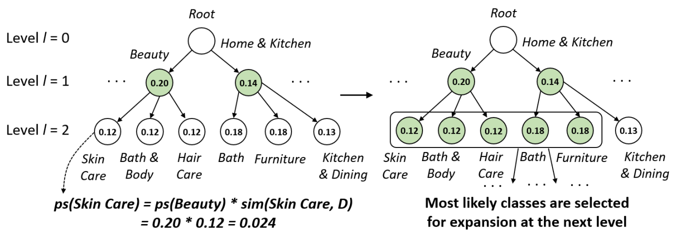


Fig. 3. Top-down core class candidate selection.

average of all $h_u^{(L)}$ which belonging to the ego network, G:

$$h_G = \frac{1}{|G|}\sum_{u\in G} h_u^{(L)} \tag{5}$$

We use a log-bilinear text matching model to calculate the probability that document D_i has core class c_j as below:

$$p_{ij} = \mathbf{P}(y_j = 1|D_i) = \sigma(c_j^T B D_i) \tag{6}$$

where $\sigma(\cdot)$ is the sigmoid function and B is a learnable interaction matrix. In the original paper, the exponential function is applied before the sigmoid function. However, when exponential function is applied, it is mapped to the [1/2, 1] range due to the sigmoid function. In this case, training converges to negative label $\rightarrow$ 1/2, positive label $\rightarrow$ 1. Instead of doing it, we subtracted the exponential function because it was trained so that all the labels were 1/2.

- Text Classifier Training: We use document confident core classes to train a text classifier. We define positive and negative classes for training. In general, if a class is the core class of a given document, it is highly likely that its parent classes and children classes are also tagged with the document. Therefore, we define positive classes as core classes and their parent classes, and define negative classes as a set excluding positive classes and children classes of core class from all classes. This can be expressed as a formula as below:

$$\begin{aligned}\mathbb{C}_i^{pos} &= \left(\bigcup_{c_j\in\mathbb{C}_i} Par(c_j)\right)\cup\mathbb{C}_i,\\ \mathbb{C}_i^{neg} &= \mathbb{C} - \mathbb{C}_i^{pos} - \bigcup_{c_j\in\mathbb{C}_i} Chd(c_j)\end{aligned} \tag{7}$$

where $\mathbb{C}$ is all classes, $Par(c_j)$ is class c_j's parent class set, and $Chd(c_j)$ is class c_j's children class set. And we use binary cross entropy (BCE) loss for training, and the equation is as follows:

$$\mathcal{L} = -\sum_{\substack{i=1\\ \mathbb{C}_i\neq\emptyset}}^{|\mathcal{D}|}\Big(\sum_{c_j\in\mathbb{C}_i^{pos}}\log p_{ij} + \sum_{c_j\in\mathbb{C}_i^{neg}}\log(1-p_{ij})\Big) \tag{8}$$

where $\emptyset$ means an empty set and we exclude the documents which have no confident core class.

D. Multi-label Self-Training

After training the text classifier, we refine our model through self-training on the entire unlabeled documents. This helps to generalize the model. Self training ([4]) is to iteratively compute the target distribution Q using the model's current predicted distribution P. In general, we train self training using KL divergence loss, and the expression is:

$$\mathcal{L}_{ST} = \text{KL}(Q||P) = \sum_{i=1}^{|\mathcal{D}|} \sum_{j=1}^{|\mathcal{C}|} q_{ij} \log \frac{q_{ij}}{p_{ij}} \tag{9}$$

The target distribution Q is designed to increase the weight of high confidence predictions and decrease the weight of low confidence predictions:

$$q_{ij} = \frac{p_{ij}^2/(\sum_i p_{ij})}{p_{ij}^2/(\sum_i p_{ij}) + (1-p_{ij})^2/(\sum_i (1-p_{ij}))} \tag{10}$$

During practical training, the target distribution Q is not updated every time, but is updated every 25 batches using Eq. (9). This makes the self-training process more effective and more robust. This training method is described in detail in Algorithm 1.

```
Algorithm 1: TaxoClass Framework
Input: An unlabeled corpus $\mathcal{D}$, a class taxonomy $\mathcal{T}$ with class names $\mathcal{S}$, an entailment model $\mathcal{M}$, total number of batches $\mathcal{B}$.
Output: A trained classifier $f(\cdot)$.
1 Use model $\mathcal{M}$ to compute document-class similarity;
2 Obtain document core classes $\{(D_i, \mathbb{C}_i)|D_i \in \mathcal{D}\}$;
3 Train classifier $f(\cdot)$ with Eq. (8);
4 for $i$ from $1$ to $\mathcal{B}$ do
5     if $i$ mod $25 = 0$ then
6         Update $Q$ with Eq. (10);
7     Train classifier $f(\cdot)$ with Eq. (9);
8 Return $f(\cdot)$;
```

III. Experiments

A. Datasets

We use two datasets for training and evaluation: (1) Amazon-297 and (2) DBPedia-298. Amazon-297 contains 45,000 product reviews and a three-level class taxonomy consisting of 297 classes. DBPedia-298 contains 45,000 Wikipedia articles and a three-level class taxonomy consisting of 298 classes. Detailed data statistics can be found in Table I.

TABLE I
Dataset statistics. There are two types of datasets. One is Amazon product reviews dataset and the other is DBPedia Wikipedia articles dataset. # Train means the number of train data, # Test means the number of test data, and # Classes means the number of classes in class taxonomy.

Dataset	# Train	# Test	# Classes
Amazon-297	30,000	15,000	297
DBPedia-298	30,000	15,000	298

Some notable preprocessing and modification we did includes following: First, removing categories in Amazon dataset which have too unconmmon special characters, they are also removed from our taxonomy json to reduce redundant traverse of classes. Second, categories of DBPedia were CamelCased instead of space separation of words, so we changed the cases to better encoded and interpreted by pretrained NLP models.

B. Evaluation Metrics

We followed the evaluation metric presented in the paper almost the same. The first metric is example-F1. The average example-F1 of all documents can be obtained by the following formula:

$$Example - F1 = \frac{1}{N} \sum_{i=1}^{N} \frac{2|\mathbb{C}_i^{true} \cap \mathbb{C}_i^{pred}|}{|\mathbb{C}_i^{true}| + |\mathbb{C}_i^{pred}|} \tag{11}$$

where $\mathbb{C}_i^{true}$ ($\mathbb{C}_i^{pred}$) is the true (model predicted) class set of document D_i. Since the paper did not present a method of determining $\mathbb{C}_i^{pred}$, we included only labels exceeding threshold (0.95) in $\mathbb{C}_i^{pred}$.
We also use Precision at k ($P@k$), which is often used for HMTC evaluation, and the expression is:

$$P@k = \frac{1}{N} \sum_{i=1}^{N} \frac{|\mathbb{C}_i^{true} \cap \mathbb{R}_{i,1:k}^{pred}|}{min(k, |\mathbb{C}_i^{true}|)} \tag{12}$$

where $\mathbb{R}_i^{pred}$ is a rank list of predicted class set $\mathbb{C}_i^{pred}$ based on the predicted probability and $\mathbb{R}_{i,1:k}^{pred}$ is predicted top k most likely classes for document D_i.
Finally, Mean Reciprocal Rank (MRR) introduced by [5] is used and it means an average measure of reciprocal ranks over a set of listed results. The definition of MRR is as follows:

$$\text{MRR} = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{|\mathbb{C}_i^{true}|} \sum_{c_j \in \mathbb{C}_i^{true}} \frac{1}{R_{ij}} \tag{13}$$

where R_{ij} is the rank of document D_j's true class c_j in model predicted rank list (over all classes).

C. Experiment Settings

We used most of the hyperparameters in the original paper as-is. We use AdamW optimizer to train our model with batch size 4, learning rate 5e-5 for all parameters in BERT document encoder and learning rate 4e-3 for all remaining parameters. We run all experiments on a single cluster with 40 CPU cores and a TITAN RTX-24G GPU. With batch size 4, the TaxoClass framework consumes about 7GB GPU memory.

D. Results

We experiment our TaxoClass model on two datasets. In addition, we evaluate two ablations: TaxoClass-noST and TaxoClass-noGNN. The TaxoClass-noST is a model without self-training and shows the effect of self-training. The TaxoClass-noGNN replaces the GNN-based class encoder with a simple embedding layer. Table II shows the result of evaluation. We use four types of evaluation metrics. As we can see, TaxoClass with self-training and GNN-based class encoder has the best performance among three models. In the case of not doing self-training, it showed a performance drop of about one percentage. When the GNN-based class encoder was not used, the performance was much lower. In the case of our model using both self training and GNN, the F1 score was about 20 percentage in both the Amazon and DBPedia datasets.

TABLE II
EVALUATION OF OUR METHODS ON TWO DATASETS. TAXOCLASS IS OUR MODEL. TAXOCLASS-NOST IS A MODEL WITHOUT SELF-TRAINING AND TAXOCLASS-NOGNN IS A MODEL WITHOUT GNN-BASED CLASS ENCODER.

Method	**Amazon-297**				**DBPedia-298**			
	Example-F1	P@1	P@3	MRR	Example-F1	P@1	P@3	MRR
TaxoClass-NoST	24.66	43.70	30.74	25.75	16.88	31.87	19.59	16.93
TaxoClass-NoGNN	17.82	28.81	26.53	7.31	8.78	32.00	16.63	15.28
TaxoClass	25.47	47.56	32.12	27.37	18.01	31.55	19.81	17.42

IV. RELATED WORK

Related work consists of three parts: (1) Weakly-supervised text classification, (2) zero-shot text classification, and (3) hierarchical text classification. The details of each part will be explained below and Table III shows the shortcomings of related work.

TABLE III
SHORTCOMINGS OF RELATED WORK

Related Work	**Description**	**Shortcoming**	**Improvement**
Weakly-supervised	Few labeled documents and many unlabeled documents	1. Document has only one class 2. All class names must appear in the corpus	1. Document has multiple class 2. No need to appear in the corpus
Zero-shot	Train with seen classes and predict with unseen classes	Need labeled document with seen classes	Do not require labeled document

A. Weakly-supervised Text Classification

In weakly-supervised text classification, few labeled documents and many unlabeled documents are needed to train model. Shortcomings of some previous research of weakly-supervised method is documents has only one class or all class names must appear in the corpus. In our improvement, document can have multiple class and class names are no need to appear in the corpus.

B. Zero-shot Text Classification

In zero-shot text classification, classifier is trained with documents that belongs to seen classes and it predict the documents that belongs to unseen classes. Documents are jointly embedded so knowledge of seen classes can be transferred to unseen classes. Shortcoming of zero-shot text classification is that it also need labeled documents with seen classes. In out improvement, we do not require labeled documents.

C. Hierarchical Text Classification

Hierarchical text classification use a class hierarchy (taxonomy) to improve the performance of text classification. It typically divided into two categories, local and global approaches. Local approach train a model per class, parent class or level. Global approach train a model with whole taxonomy with recursive regularization or graph neural network (GNN) based encoder. We follow global approach with GNN based encoder.

V. CONCLUSIONS & FUTURE WORK

This mini RD replicates TaxoClass, a solution to the hierarchical multi-label text classification problem when only class surface names, instead of massive labeled documents, are given. We used more recent amazon, DBPEDIA data than original paper, which is from 2014, and more recent version of pretrained BERT-large-mnli when generating core classes of documents, and checked if this model fits well for the newer data and slightly different settings.
The resulting model accuracy was a little unsatisfactory. We think this is due to limitations as follow:

- Among document categories in given taxonomy, there were many over-specific ones such as surface name includes uncommon acronyms, uncommon special characters, or too lengthy to regard it as category but small group of products, which is less well recognized by pretrained models we use.
- Orignal paper took the whole corpus into account when finding confidence of classes, but we considered documents only within same batch as target document due to performance reason.
- Batch size in core class guided / self training is restricted to be less than 8 because pretrained models cannot process bigger batch due to its inherent performance limitation.
- Some label names that do not exist in the pretrained word embedding model were randomly initialized.

As future works, we suggest these to overcome limitations mentioned above:

- Restrict taxonomy classes to have max length, not to have acronyms in it, or word which is very uncommon using existing word frequency statistics.
- Cache similarity calculation results persistently to use larger batch size (or even use whole corpus) when core class mining.
- Workaround with out of memory problem such as reducing document length, try versions of pretrained model which WAS latest at the time of original research.

REFERENCES

[1] W. Yin, J. Hay, and D. Roth, “Benchmarking zero-shot text classification: datasets, evaluation and entailment approach,” EMNLP/IJCNLP, 2019.
[2] J. Devlin, M. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” NAACL-HLT, 2019.
[3] T. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” ICLR, 2017.

[4] J. Xie, R. B. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," ICML, 2016.
[5] N. Craswell, "Mean reciprocal rank," Encyclopedia of Database Systems, 2009.
[6] J. Shen, Z. Shen, C. Xiong, C. Wang, K. Wang, and J. Han, "Taxoexpan: Self-supervised taxonomy expansion with position-enhanced graph neural network," WWW, 2020.
[7] Y. Meng, J. Shen, C. Zhang, and J. Han, "Weakly-supervised neural text classification," CIKM, 2018.
[8] J. Shen, W. Qiu, Y. Meng, J. Shang, X. Ren, and J. Han, "Taxoclass: Hierarchical multi-label text classification using only class names," NAACL, 2021.
[9] S. Goumy and M. A. Mejri, "Ecommerce product title classification," eCOM@ SIGIR, 2018.
[10] K. Li, S. Li, S. Yavuz, H. Zha, Y. Su, and X. Yan, "Hiercon: Hierarchical organization of technical documents based on concepts," ICDM, 2019.
[11] P. Xu and D. Barbosa, "Neural fine-grained entity type classification with hierarchy-aware loss," NAACL-HLT, 2018.
[12] S. Gururangan, T. Dang, D. Card, and N. A. Smith, "Variational pretraining for semi-supervised text classification," ACL, 2019.
[13] D. Berthelot, N. Carlini, I. Goodfellow, N. Papernot, A. Oliver, and C. Raffel, "Mixmatch: a holistic approach to semi-supervised learning," NeurIPS, 2019.
[14] Z. Zeng, W. Zhou, X. Liu, and Y. Song, "A variational approach to weakly supervised document-level multi-aspect sentiment classification," NAACL-HLT, 2019.
[15] D. Mekala and J. Shang, "Contextualized weak supervision for text classification," ACL, 2020.
[16] Y. Meng, Y. Zhang, J. Huang, C. Xiong, H. Ji, C. Zhang, and J. Han, "Text classification using label names only: a language model self-training approach," EMNLP, 2020.
[17] Z. Wang, D. Mekala, and J. Shang, "X-class: Text classification with extremely weak supervision," NAACL, 2021.