**NLPCC2022 Shared Task5**

# Multi-label Classification, NER, Content Extraction for Scientific Literature

# Guideline

CNPIEC KEXIN LTD

April 6th, 2022

# Table of Contents

# I. TASK OVERVIEW

Managing and exploring scientific literature efficiently becomes more and more important for people such as researchers due to the growing large number of publications. Owning to the development of artificial intelligence (AI) and natural language processing (NLP) technology, literature management and exploration can be more efficiently by using AI-based literature classification, literature search and recommendation as well as PDF content extraction functions. Aiming at solving the related challenging fundamental problems, we set three tracks in this task (with three datasets provided):

**Track 1:** Multi-label Classification Model for English Scientific Literature: develop a multi-label classification model for scientific research literature based on the given metadata (title and abstract) of scientific research literature and corresponding hierarchical labels from a domain-specific topic taxonomy.

**Track 2:** Supervised Named Entity Recognition Model for English Domain-specific Texts: develop a named entity recognition model for domain-specific texts based on state-of-the-art NLP and deep learning technique with the labeled domain-specific sentences corresponding to seven entity types.

**Track 3:** Detection and Extraction of Inline and Isolated Mathematical Expressions Model for English PDF Articles: design and develop a model that can automatically recognize and extract the inline and isolated mathematical expressions from the provided English PDF articles without using any commercial third-party APIs.

**Organizer:** CNPIEC KEXIN LTD and Data Intelligence

**Contact:** He ZHANG (rd_kexin@cnpiec.com.cn)

## II. TRACK 1 - Multi-label Classification Model for English Scientific Literature

### A. Background

In recent years, the indexing for scientific research literature is becoming more and more important with the increasing number of such literature in various fields. According to incomplete statistics, about 2.5 million research papers published from about 28100 journals every year. Search engines, digital libraries and citation indexes are widely used to search these documents and publications. The intelligent classification of scientific research literature is helpful to improve the retrieval efficiency and quality of users. Hierarchical multi-label classification is a common method for indexing scientific research literature. For example, both arXiv and Microsoft Academic adopt this method. Moreover, hierarchical multi-label classification is also a challenging task in text classification.

### B. Objectives

The aim of this task is to develop a multi-label classification model for scientific research literature, based on the given metadata (title and abstract) of scientific research literature and corresponding hierarchical labels from a domain-specific topic taxonomy.

### C. Expected Outputs

Python-based model, development report (including instruction for model usage) and prediction results of the testing datasets.

It should be noted that the format of model input should be the same as that of the testing datasets.

For submission, please write the prediction results into a JSON file (in format of **[{"title":"","abstract":"","pred_labels":[""]}, ...]**) with the same sample order in testing datasets.

### D. Training datasets

We provide around 95,000 English scientific research literature, of which 90,000 can be used as training set and 5,000 can be used as verification set. Each data sample contains the title and summary of the article and the corresponding hierarchical multi

labels (level1、level2、level3) and the union of hierarchical multi labels (levels). (Note: data other than the training set cannot be used in the process of model development). The number of label categories for level1, level2, level3 and levels is 21, 260, 1272 and 1530, respectively. In addition, these labels follow a long-tailed distribution.

The model input is provided as JSON files which are displayed in table as follows:

| | title | abstract | level1 | level2 | level3 | levels |
|---|---|---|---|---|---|---|
| 0 | Evidence for the Superatom–Superatom Bonding f... | Metal clusters with specific number of valence... | [Energy, Physical chemistry, Inorganic chemistry] | [Cluster chemistry, Kinetics, Binding energy, ... | [Metal clusters, Kinetic parameters] | [Energy, Physical chemistry, Inorganic chemist... |
| 1 | Global Structure of a Three-Way Junction in a ... | The condensation of bacteriophage phi29 genomi... | [Materials science, Physical chemistry, Polyme... | [Chemical structure, Polymers, Spectroscopy, P... | [Genetics, Crystal structure, Oligomers, Elect... | [Materials science, Physical chemistry, Polyme... |
| 2 | Influence of Dithiolate Bridges on the Structu... | As a further exploration of the asymmetrically... | [Physical chemistry, Cross-disciplinary concep... | [Chemical structure, Inorganic compounds, Chem... | [Ligands, Diirons, Molecular structure, Redox ... | [Physical chemistry, Cross-disciplinary concep... |
| 3 | Plasticity of the Binding Site of Renin: Optim... | Protein flexibility poses a major challenge to... | [Analytical chemistry, Physical chemistry, Ino... | [Chemical structure, Cluster chemistry, Imagin... | [Ligands, Crystal structure, Protein structure... | [Analytical chemistry, Physical chemistry, Ino... |
| 4 | Regioselective Synthesis of Multifunctional Al... | We describe, for the first time, a highly regi... | [Organic chemistry, Catalysis] | [Organic compounds, Functional groups, Hydrosi... | [Amides, Aziridines, Amines] | [Organic chemistry, Catalysis, Organic compoun... |

Fig 1. The format for model input

## E. Testing datasets

5,000 English scientific articles are provided as testing datasets. The testing datasets will be provided as a JSON file in format of **[{"title":"","abstract":"", ...]**.

## F. Evaluation Metrics

The model evaluation is based on the three indicators of precision, recall and F1 value, and the three indicators will be calculated by the provided evaluation code i.e. "**eval.py**". It is required that the performance of the submitted prediction results shall not be lower than that of the mainstream methods.

# III. TRACK 2 - Supervised Named Entity Recognition Model for English Domain-specific Texts

## A. Background

Domain knowledge graph has been widely adopted for various domains, e.g., medicine, agriculture and service industry, because it can provide promising functions including intelligent search and personalized recommendation. Knowledge graph is normally composited of a huge number of entities and relations (to connect entities), and the utility of knowledge graph largely depends on the richness of these entities/relations. To construct high-value knowledge graph (e.g., informative domain knowledge graph), researchers aim at automatically extracting entities from massive heterogeneous sources, which is often impossible to achieve with pure manual labor.

With the blooming of natural language processing (NLP), researchers have proposed Named Entity Recognition (NER) technique to automatically extract entities from raw texts. NER is mostly regarded as a supervised sequence labeling/tagging task; that is, recognizing entities from unseen texts according to the patterns learned from labeled texts. As an essential step for knowledge graph construction, as well as some other NLP tasks, the development of NER is one of the main focuses in both the academia and the industry in recent years. Under this background, this competition targets at exploring novel and insightful NER methods to better capture the entities, especially for the construction of domain knowledge graphs.

## B. Objectives

NER is normally defined as a supervised sequence labeling task. Therefore, this competition will provide a certain number of labeled sentences, which are collected from domain-specific English publications, for the model training. The trained model will be used to capture entities in unlabeled sentences.

The training datasets provided to the participants includes 5000+ sentences, which are labeled following BIO format, including 7 different entity types. The sentences are collected from open-source domain-specific (material science) English publications, and thus they may contain entities only belong to this domain. The expectation is that

participants can explore state-of-the-art NLP and deep learning techniques to improve the accuracy as well as the generalization of the NER model.

### C. Expected Outputs

Python-based NER model trained with the provided training datasets, development report (including instruction for model usage) and prediction results of the testing datasets.

The output of predicted model needs to be a text file containing sentences with predicted BIO labels, for example:

```
transport O
resistance O
for O
the O
cp-NiOx B-MA
device O
```

Fig 2. The example of predicted BIO labels

Please name the output file as "**predict.txt**". Please separate the word and the tag by **space**.

### D. Training Datasets

5000+ domain-specific English sentences that are labeled following BIO format. There are eight entity types, including seven domain-specific types ("MA", "PR", "AP", "CH", "EQ", "SY", "ST", "DE") and the other type ("O").

### E. Testing Datasets

600+ domain-specific English sentences without entity label, including the same entity types as those of the training datasets.

The input of the trained model for the final evaluation is the file "**test.txt**", which contains unlabeled sentences, for example:

```
transport
resistance
for
the
cp-NiOx
device
```

Fig 3. The example of testing datasets

### F. Evaluation Metrics

Accuracy measured with F1-score on the testing datasets. The accuracy is expected to

be higher than existing main-stream NER methods. F1-score is defined as follows:

$$F_1 = \frac{2}{\frac{1}{recall} \times \frac{1}{precision}} = 2 \times \frac{precision \times recall}{precision + recall}$$

where *precision* is calculated by dividing the number of correctly recognized entities with the number of all recognized entities, *recall* is calculated by dividing the number of correctly recognized entities with the number all entities in the ground truth. The predicted labels will be compared with the ground truth to measure the final prediction accuracy, with the provided code i.e. "**ner_evaluation.py**".

# IV. TRACK 3 - Detection and Extraction of Inline and Isolated Mathematical Expressions Model for English PDF Articles

## A. Background

Detecting and extracting inline and isolated mathematical expressions (including mathematical characters, symbols, and equations) are among the most important and challenging tasks in text mining. Although OCR techniques are effective on digitizing ordinary text, their performance on mathematical expression are quite limited. Mathematical expressions are essential information containers, especially for the fields of academic research, regulation and education. The accurate recognition of inline and isolated mathematical expressions has a wide variety of applications and is favored by readers and practitioners in related fields.

## B. Objectives

The objective of this task is to design and develop algorithmic models that can automatically recognize and extract the inline and isolated mathematical expressions from the provided English PDF articles without using any commercial third-party APIs, based on Python platform, machine learning models and the provided datasets (optional). The designed models should meet the following requirements:

1) The designed model should detect and locate the inline mathematical expressions. The coordinates of an inline mathematical expression should be displayed in the form of [x0, x1, y0, y1], where x0 is the distance calculated in pixels between the left boundary of the mathematical expression and the left edge of the current page, x1 is the distance between the right boundary of the expression and the left edge of the page, y0 is the distance between the top boundary of the expression and the top edge of the page, and y1 is the distance between the bottom boundary of the expression and the top edge of the page, respectively.
2) The designed model should extract the inline mathematical expressions and output the expressions in image formats and LaTex expressions.
3) The designed model should detect and locate the isolated mathematical

expressions. The coordinates of an isolated mathematical expression should be displayed in the form of [x0, x1, y0, y1], where x0 is the distance calculated in pixels between the left boundary of the mathematical expression and the left edge of the current page, x1 is the distance between the right boundary of the expression and the left edge of the page, y0 is the distance between the top boundary of the expression and the top edge of the page, and y1 is the distance between the bottom boundary of the expression and the top edge of the page, respectively.

4) The designed model should extract the isolated mathematical expressions and output the expressions in image formats and LaTex expressions.

## C. Expected Outputs

Development report (including instruction for model usage), prediction results and Python-based model that can detect, locate and extract the inline and isolated mathematical expressions for English PDF articles. The outputs of the designed models should be both the image files and LaTex expressions of the extracted expressions.

For evaluation convenience, please call the model as follows:

**char_data, line_data, bbox_data = model(pdf_page, input_path)**

where **pdf_page** represents the full directory of the image file of the PDF page, **input_path** represents the full directory to save the image file of the characters whose type is other and the image file of the marked PDF page where the boundary of each character is marked using rectangle, **char_data** represents a dict with the following structure:

```
{
 str(char_idx): { # char_idx: The index of the current character. For each page, char_idx starts from int(0)
  "block_id": int(), # The block id of the current character. For each page, block_id starts from int(0)
  "line_id": int(), # The line id of the current character. For each block, line_id starts from int(0)
  "char_id": int(), # The character id of the current character. For each line, char_id starts from int(0)
  "type": int(), # The type of the current character. 0: normal text; 1: mathematical symbol; 2: other
                 # If type is 2, the current block should contain no other characters and should be
                 # exported to the appointed directory in png format
  "x0": int(), # The minimum horizontal coordinate of the current character
  "x1": int(), # The maximum horizontal coordinate of the current character
  "y0": int(), # The minimum vertical coordinate of the current character
  "y1": int(), # The maximum vertical coordinate of the current character
  "char": "" # The current character
  }
 }
```

**line_data** represents a dict with the following structure:

```
{
 str(line_idx): { # line_idx: The index of the current line. For each page, line_idx starts from int(0)
  "block_id": int(), # The block id of the current line. For each page, block_id starts from int(0)
  "line_id": int(), # The line id of the current line. For each block, line_id starts from int(0)
  str(seg_id): { # seg_idx: The segment id of the current line. For each line, seg_id starts from int(0)
                 # The segment of different types in the current line should be split into different segments
   "seg_id": int(), # The segment id of the current line. For each line, seg_id starts from int(0)
   "type": int(), # The type of the current segment. 0: normal text; 1: mathematical expression; 2: other
   "text": "", # The text of the current segment
   "path": "" # The directory of the image file of the current segment
   }
  }
 }
```

**bbox_data** represents an image file of the marked PDF page where the boundary of each character is marked using rectangle and is saved to input_path.

### D. Training datasets

Participants can choose their training data freely. Some optional open-source datasets related to this task are provided. The first two are **GTDB-1** and **GTDB-2** (data source：W. Ohyama, M. Suzuki and S. Uchida, "Detecting Mathematical Expressions in Scientific Document Images Using a U-Net Trained on a Diverse Dataset," in IEEE Access, vol. 7, pp. 144030-144042, 2019, doi: 10.1109/ACCESS.2019.2945825) that can be used to locate mathematical expressions. The detailed information of these datasets is listed in the following table:

| Category | Datasets 1: GTDB-1 | Datasets 2: GTDB-2 |
|---|---|---|
| Number of articles | 31 | 16 |
| Number of pages | 544 | 343 |
| Number of math symbols | 162406 | 115433 |
| Number of ordinary text characters | 646714 | 507412 |

Both datasets include the PDF articles and their corresponding CSV files. In each CSV file, the attributes of a character, such as file name, page number, type (text or image), line number, coordinates, text mode (ordinary text or math symbol), positional relationship to the preceding character (horizontal, right-superscript, right-subscript, left-superscript, left-subscript, upper, lower, or has no preceding character), and OCR code, are recorded in details. For copyright reasons, we selected 27/31 articles in GTDB-1 and 13/16 articles in GTDB-2 for this task.

The third datasets is **im2latex-100k** that can be used to transform mathematical

expression into LaTex expressions[1]. This datasets includes around 100k formulas and images that are split into train, validation and test sets.

### E. Testing datasets

The image files of 50 English PDF article pages will be used for testing. Each page may contain inline mathematical expressions, isolated mathematical expressions, figures, tables, article titles, author information, institution information, or ordinary text only.

### F. Evaluation metrics

1) $R_s \geq 85\%,\ R_s = \frac{n_{TP}}{n_{TP}+n_{FN}}$

2) $P_s \geq 85\%,\ P_s = \frac{n_{TP}}{n_{TP}+n_{FP}}$

3) $F_s \geq 85\%,\ F_s = \frac{2P_sR_s}{P_s+R_s}$

where $n_{TP}$, $n_{FP}$, and $n_{FN}$ are the numbers of correctly extracted mathematical symbols, falsely extracted symbols or ordinary text, and undetected mathematical symbols, respectively.

[1] https://zenodo.org/record/56198#.YjgDQOpByUl