TopWORDS_MEPA

Here is the source code for the TopWORDS-MEPA method proposed by Jiaze Xu, Changzai Pan, and Ke Deng in article "A Dual-Dictionary Model for Mining Domain-Specific Chinese Texts." This method is an extension of TopWORDS introduced in the paper "On the Unsupervised Analysis of Domain-Specific Chinese Texts" published in PNAS.

Background

TopWORDS_MEPA is a statistical algorithm that can achieve high-quality meta-pattern discovery, named entity recognition, text segmentation, and relation extraction simultaneously from unannotated target texts with little training information. It is proposed based on Dual-Dictionary Model, a extended version of Word Dictionary Model in TopWORDS Algorithm.

Compared to the original TopWORDS algorithm, TopWORDS-MEPA not only have improved performance on word discovery, but also extracts meta patterns and named entities of interest category from the text. For more details, please refer to our paper "A Dual-Dictionary Model for Mining Domain-Specific Chinese Texts".

Installation

To get started with this project, you need to have Python 3.8 or a higher version installed on your system. You can download the latest version of Python from the official Python website https://www.python.org/downloads/windows/.
Once you have Python installed, navigate to the project directory and run the following command to install all the required packages: For install relative packages, you can use `pip install` to install each package or run

```
pip install -r requirements.txt
```

This command will automatically install all the dependencies listed in the requirements.txt file. Make sure you have an active internet connection during the installation process.

Get Started

To get started quickly, we provide a short text (selected ten biographical chapters from the *History of the Song Dynasty*) as an example. You can quickly run TopWORDS-MEPA with the following command to get a quick result in 5 minutes:

```
python run_topwords_mepa.py --input_text_dir
'.\examples\data\HSD_tiny\HSD_tiny.txt' --technical_terms_dir
'.\examples\data\HSD_tiny\technical_terms.json' --punctuation_dir
'.\examples\data\HSD_tiny\punctuations.txt' --background_words_dir
'.\examples\data\HSD_tiny\background_words.txt'  --output_dir
'.\examples\output\HSD_tiny' --word_max_len_for_screening_tt 3 --
word_min_freq_for_screening_tt 50 --collocation_max_len_for_screening 3 --
collocation_min_freq_for_screening 20 --prune_by_score_iteration_num 0 --
num_of_processes 1
```

The output files are saved in `examples\output\HSD_tiny\output_dir`.

Parameters Explanation:

- **Input files directory:**
  - `--input_text_dir`: The path to the target text file (in `.txt` format).

    ```
    宋史卷二百四十九
    列傳第八
    范質子旻,兄子杲,王溥父祚,魏仁浦子咸信,孫昭亮
    范質字文素,大名宗城人.父守遇,鄭州防禦判官.質生之夕,母夢神人授以五色筆.九歲能屬文,十三
    治尚書,教授生徒
    ...
    ```

  - `--technical_terms_dir`: The path to the dictionary file (in `.json` format), which contains a small collection of known named entities of each category from independent data sources, used for building a weak word classifier based on Naive Bayes classification.

    ```
    {
      "name": ["范質", "王溥",...],
      "address": ["鄭州", "宗城", ...],
        ...
    }
    ```

  - `punctuation_dir`: The path to the punctuation file, containing punctuation strings for splitting paragraph into sentence pieces.
  - `background_words_dir`: Optional. The path to the background words file (in `.txt` format). You can provide background words for a better performance. In paper, we use the output of TopWORDS algorithm as an over-complete initial background word list.
- **Output files directory:**
  - `--output_dir`: The output directory for TopWORDS-MEPA results.
- **Key hyper-parameters**:
  - `--word_max_len_for_screening_tt`: The maximum word length $\tau_L^{(w)}$ for screening technical terms when constructing the initial word dictionary. Set 8 by default.
  - `--word_min_freq_for_screening_tt`: The minimum word frequency $\tau_F^{(w)}$ for screening technical terms when constructing the initial word dictionary. Set 5 by default.
  - `--collocation_max_len_for_screening`: The maximum meta-pattern length $\tau_L^{(p)}$ for screening when constructing the initial meta-pattern dictionary. Set 5 by default.
  - `--collocation_min_freq_for_screening`: The minimum meta-pattern frequency $\tau_F^{(p)}$ for screening when constructing the initial meta-pattern dictionary. Set 50 by default.
  - `--prune_by_score_iteration_num`: The number of additional EM iteration loops. Generally, after the EM algorithm converges, we prune words and meta-patterns based on their significance scores. This pruning disrupts the balance of the parameters, necessitating further EM iterations to achieve final convergence. The additional "pruning-EM algorithm" is typically performed 1 to 3 times. Here we set 0 for a quick result. The default value is 3.
  - `--num_of_processes`: The number of processes to use for parallel execution, enhancing performance on multi-core systems.

We recommend to set $\tau_L^{(w)} \geq 8, \tau_F^{(w)} \geq 2, \tau_L^{(p)} \geq 5, \tau_F^{(p)} \geq 50$ based on various characteristics of different applications. In this toy example, we set these parameters just for a quick start.

Note: In this simple example, we set the hyper-parameters to focus on high-frequency, short-length words and collocations to achieve a quick but rough result. If you want a more comprehensive experience, please refer to the three additional examples provided in our paper (***History of the Song Dynasty***, ***History of the Ming Dynasty*** and ***CHIP2020***). The input files for these two examples are also provided in the directory `examples`. Due to file size limitations, the input files for ***Jin Yong's Novels*** (26M) are not provided here.

The command for running *History of the Song Dynasty*:

```
python run_topwords_mepa.py --input_text_dir '.\examples\data\HSD\HSD.txt' --
technical_terms_dir '.\examples\data\HSD\technical_terms.json' --punctuation_dir
'.\examples\data\HSD\punctuations.txt' --background_words_dir
'.\examples\data\HSD\background_words.txt'  --output_dir '.\examples\output\HSD'
--word_max_len_for_screening_tt 8 --word_min_freq_for_screening_tt 5 --
collocation_max_len_for_screening 5 --collocation_min_freq_for_screening 100 --
num_of_processes 8
```

The command for running *History of the Ming Dynasty*:

```
python run_topwords_mepa.py --input_text_dir '.\examples\data\HMD\HMD.txt' --
technical_terms_dir '.\examples\data\HMD\technical_terms.json' --punctuation_dir
'.\examples\data\HMD\punctuations.txt' --background_words_dir
'.\examples\data\HMD\background_words.txt'  --output_dir '.\examples\output\HMD'
--word_max_len_for_screening_tt 8 --word_min_freq_for_screening_tt 5 --
collocation_max_len_for_screening 5 --collocation_min_freq_for_screening 100 --
num_of_processes 8
```

The command for running *CHIP2020*:

```
python run_topwords_mepa.py --input_text_dir
'.\examples\data\CHIP2020\short_sents.txt' --technical_terms_dir
'.\examples\data\CHIP2020\technical_terms.json' --punctuation_dir
'.\examples\data\CHIP2020\punctuations.txt' --background_words_dir
'.\examples\data\CHIP2020\background_words.txt'  --output_dir
'.\examples\output\CHIP2020' --word_max_len_for_screening_tt 8 --
word_min_freq_for_screening_tt 5 --collocation_max_len_for_screening 5 --
collocation_min_freq_for_screening 50 --num_of_processes 8
```

Simulation Study

To validate the effectiveness of TopWORDS-MEPA, we designed a simulation study for evaluating tasks of dictionary discovery and parameter estimation. We constructed a Dual Dictionary Model with 100 meta-patterns and 1,000 non-trivial words belonging to 4 categories (background, address, name, and office title). These elements are selected from the texts of

History of the Song Dynasty, with the corresponding meta-pattern sampling distribution γ and word sampling distributions $\theta_{w|c}$ properly specified to mimic their practical usage frequencies in the book. For more details, please refer to the Section 4 's detailed simulation studies.

We have made available the source code along with the input files (including 100 meta-patterns and 1,000 non-trivial words) under the directory `examples/simulation`. To reproduce the results of our simulation study, execute the following command:

```
python generate_simulation_text.py --seed 0 --sentence_num 50000 --input_dir
./simulation/input --output_dir ./simulation/output
```

Explanation:

- `--seed`: An integer value that serves as the seed for the random number generator. The default value is `0`.
- `--sentence_num`: The number of sentences for generated texts. In our paper, there are three settings: 50,000, 100,000, and 200,000. The default value is `200000`.
- `--input_dir`: The path to the directory containing the input files. We provide the input files (a collocation dictionary file

  file and a word dictionary file) in `./simulation/input` .
- `--output_dir`: The path to the directory where the generated text data will be saved. By default, the output is directed to `./simulation/output`.

Advanced Usage

If you want to apply TopWORDS-MEPA to your own data, following the three steps below:

1. **Prepare the Input Data**
   Ensure that you have the input text file, the technical terms file and punctuation file ready. Here's an example:

   - `input_text_dir`: The target text file for analysis (in `.txt` format).

     ```
     宋史卷二百四十九
     列傳第八
     范質子旻,兄子杲,王溥父祚,魏仁浦子咸信,孫昭亮
     范質字文素,大名宗城人.父守遇,鄭州防禦判官.質生之夕,母夢神人授以五色筆.九歲能屬文,十三
     治尚書,教授生徒
     ...
     ```

   - `technical_terms_dir`: A dictionary file (in `.json` format), which contains a small collection of known named entities of each category from independent data sources, used for building a weak word classifier based on Naive Bayes classification.

     ```
     {
       "name": ["范質", "王溥",...],
       "address": ["鄭州", "宗城", ...],
         ...
     }
     ```

- `punctuation_dir`: The path to the punctuation file(in `.txt` format), containing punctuation strings for splitting paragraph into sentence pieces.

  ```
  。
  ，
  ！
  ？
  ```

- `background_words_dir`: Optional (in `.txt` format). You can provide background words for a better performance.

  ```
  如果
  我
  这样
  ...
  ```

2. **Set Up the Output Directory**
   Create the output directory where the results will be saved. For example, the output directory can be `examples\output\test`.
   - `output_dir`: The output directory for TopWORDS-MEPA results.
3. **Run the Script**
   Open your command prompt or terminal, navigate to the project directory, and run the following command by replacing your own parameters.

   ```
   python run_topwords_mepa.py --input_text_dir
   '.\examples\data\HSD_tiny\HSD_tiny.txt' --technical_terms_dir
   '.\examples\data\HSD_tiny\technical_terms.json' --punctuation_dir
   '.\examples\data\HSD_tiny\punctuations.txt' --background_words_dir
   '.\examples\data\HSD_tiny\background_words.txt'  --output_dir
   '.\examples\output\HSD_tiny' --word_max_len_for_screening_tt 3 --
   word_min_freq_for_screening_tt 50 --collocation_max_len_for_screening 3 --
   collocation_min_freq_for_screening 20 --prune_by_score_iteration_num 0 --
   num_of_processes 1
   ```

Parameter Explanation:

- **Screening Parameters for Initializing Dictionary**:
  - `--word_max_len_for_screening_tt`: The maximum word length $\tau_L^{(w)}$ for screening technical terms when constructing the initial word dictionary. Set 8 by default.
  - `--word_min_freq_for_screening_tt`: The minimum word frequency $\tau_F^{(w)}$ for screening technical terms when constructing the initial word dictionary. Set 5 by default.
  - `--word_max_len_for_screening`: The maximum word length $\tau_L^{(w)}$ for screening all words (including backgrounds) when constructing the initial word dictionary. If you have provided an over-complete background words file in `background_words_dir`, please set this parameter to 1 to avoid repeating enumeration.
  - `--word_min_freq_for_screening`: The minimum word frequency $\tau_F^{(w)}$ for screening all words (including backgrounds) when constructing the initial word dictionary. If you have provided an over-complete background words file in `background_words_dir`, please set this parameter to 1 to avoid repeating enumeration.

    - `--collocation_max_len_for_screening`: The maximum meta-pattern length $\tau_L^{(p)}$ for screening when constructing the initial meta-pattern dictionary. Set 5 by default.
    - `--collocation_min_freq_for_screening`: The minimum meta-pattern frequency $\tau_F^{(p)}$ for screening when constructing the initial meta-pattern dictionary. Set 50 by default.

As mentioned before, We recommend to set $\tau_L^{(w)} \geq 8, \tau_F^{(w)} \geq 2, \tau_L^{(p)} \geq 5, \tau_F^{(p)} \geq 50$ based on various characteristics of different applications. In this toy example, we set these parameters just for a quick start.

- **EM iteration Parameters**:
    - `--em_iteration_num`: Number of maximum steps in each EM algorithm iteration. The default value is 100.
    - `--prune_by_score_iteration_num`: The number of additional EM iteration loops. Generally, after the EM algorithm converges, we prune words and meta-patterns based on their significance scores. This pruning disrupts the balance of the parameters, necessitating further EM iterations to achieve final convergence. The additional "pruning-EM algorithm" is typically performed 1 to 3 times. Here we set 0 for a quick result. The default value is 3.
- **Pruning Parameters**:
    - `prune_by_count_threshold_collocation`: Threshold for pruning collocations based on count. The default value is 0.1.
    - `prune_by_count_threshold_word`: Threshold for pruning words based on count. The default value is 0.1.
    - `prune_by_para_threshold_collocation`: Threshold for pruning collocations based on parameters. The default value is 1e-6.
    - `prune_by_para_threshold_word`: Threshold for pruning words based on parameters. The default value is 1e-6.
- **Other Parameters**:
    - `num_of_open_categories_of_a_word`: Number of open categories for each word. The default value is None, which means open all word categories. The computational time of the algorithm increases exponentially with the growth in the number of word categories. Therefore, for computational efficiency, we typically keep the number of word categories below 6.
    - `min_prob_in_nb_smooth`: Minimum probability for smoothing in Naive Bayes when constructing word classifier.
    - `alpha`: Pseudo count α assigned to each word w in word dictionary.
    - `num_of_processes`: Number of processes for parallel processing.

Contact

If you are interested in collaborating or have any questions, please feel free to contact the author Ke Deng (kdeng@tsinghua.edu.cn). We look forward to hearing from you!