Progress Report

170025298

16/03/2022

1 Introduction

Trafilatura is actually an excellent content extractor for English Websites; however, the content extractor does not support well for those Chinese websites when using the Chinese web-pages dataset as is shown in section 1. As a result, the research direction of this project is to enhance the performance of the Trafilatura supporting Chinese Websites. In this progression report, the author will introduce what the author have done in two months, and will introduce how those designs are implemented to make the content extraction heuristic more reliable.

2 Code comprehension

The first thing the author have done is to investigate how the Trafilatura work. Comprehends the work flow of the Trafilatura leads to a easier understanding what might be the breakthrough point of the project. By reading through a majority of codes in Trafilatura, the work flow can be thought as follows:

Steps	Achievement	Methods
1	Loads the HTML as a tree	Use the lxml package to load it as a DOM tree parser
2	Cleaning the DOM tree with filters and extract metadata	Iterate through the head node to retrieve the metadata of the webpage; manually clean the DOM tree with a series of xpath expressions
3	Convert tags in the DOM tree into specified tags (and also remove formats)	Convert all tags of links into another tag: ref; convert h into head; etc.
4	Extract and remove comments	Extract comments using the heuristic as follows: Trafilatura have a series of lxml expressions relates to the feature of comments. It will iterate through those expressions to find nodes that is most likely to be comments.
5	Extract Contents with several heuristics	Extract contents using lxml expressions first and filter those with rich links with link density heuristics.
6	If checking in step 5 failed, using comparison extract	Using another heuristics for content extractions. Those extraction heuristics may imported from a third party extraction package.
7	If the extraction fails, use the baseline extraction	Using the baseline extraction heuristics
8	Export the extraction result	————————

Table 1: How Trafilatura works

3 Evaluate Trafilatura performance of the Chinese websites.

For the evaluation of the content extraction model, the author uses four dimensions to judge the performance of the Trafilatura content extraction model. The experiment uses four kinds of data-sets to evaluate the performance of the Trafilatura: Content extraction benchmark[**?**], the Chinese version and English version of the CleanEval data-set[**?**] and my self-made data-set manually extract contents from four types of Chinese websites: newspapers, blogs, video websites and Q & A websites. The result of the evaluation is as shown in table 2.

Data Set (number of data)	Precision	Recall	F1-Score	Accuracy
CleanEval EN (60)	0.740 ± 0.047	0.507 ± 0.055	0.602 ± 0.049	0.000 ± 0.000
CleanEval CH (50)	0.249 ± 0.039	0.180 ± 0.033	0.209 ± 0.034	0.000 ± 0.000
Content Extraction Benchmark (181)	0.913 ± 0.013	0.958 ± 0.011	0.935 ± 0.011	0.276 ± 0.032
Self-made CH Dataset (20)	0.770 ± 0.068	0.744 ± 0.074	0.757 ± 0.066	0.048 ± 0.054

Table 2: Content Extraction Evaluation using original Trafilatura CE tool.

4 Learning and improving xpaths.py

While learning from Trafilatura, the author have found that the heuristic of filtering unnecessary tags are written in xpaths.py, which includes a number of tags that are manually filtered. xpath.py is a list of expressions that allows for algorithms to identify unnecessary elements in the HTML DOM tree. A vital part of doing such work is to identify what kind of classes or ids that matches such "unnecessary". By searching for pages (Figure 1) where content extraction was not quite accurate, I found more classes and ids that match the kind of elements in Trafilatura that needed to be cleaned up.

Figure 1: The unfiltered tag / id

By comparing the difference between the output of the Trafilatura and the ground truth of the web content, the author have found a number of pages that contain more tags that can manually filtered. Th author added those tags into xpaths.py and the result of filtering is shown as figure 3.

Comparison	Precision	Recall	F-Score	Accuracy
New	**0.779**	**0.831**	**0.804**	0
Old	0.770	0.744	0.757	**0.048**

Table 3: Caption

However, there are a number of elements that have quite confused class names or id names such as a letter with a number, or directly named with uuids. To make such modifications more generalized, we may use some more flexible algorithms to clean those unnecessary tags.

5 Data-set improvement & creation

At the very beginning of the project research, two datasets were used for this project: The CleanEval dataset with 50 webpages and corresponding results, and a hand made dataset with 20 web pages with its corresponding results. When evaluating the accuracy of the model, the use of two datasets for two tests made the calculation of the accuracy of this model gets quite difficult. At the same time, the extraction of the author and the date of a web page was considered to be one of the research directions to create a better web content extraction tool, whereas in most current datasets the author and the corresponding date of the extracted content text are rarely included as part of the extraction. In case of evaluating that, I merged two datasets that were previously used and created the author and date dataset for web pages

However, merging two datasets are not as easy as what the author was expected. In the process of collecting data to create a dataset, I sometimes find that the dataset, "CleanEval", is not quite accurate. Using such a dataset to evaluate the performance of the content extracting model can sometimes be quite scrappy. Therefore, the datasets downloaded from third-party websites have to be cleaned by removing and modifying a lot of data that had garbled content. The cleaning process was done manually and a python script was written for merging those data together and have produced a ground truth JSON file for the comparison between the ground truth and the result of the extraction. In addition to that, based on the webpage collected, I made the corresponding author and the date dataset manually. Those datasets are used for the further investigations.

6 A Chinese-Specified Implementation of the code using the link density

6.1 Design

When using Trafilatura to extract the content of web pages, we sometimes find out that there are several navigation bars that are not completely cleaned (as is shown in figure 2).

Figure 2: Findings

Those links are not filtered by Trafilatura, which influences the accuracy rate of the content extractors. To filter those unnecessary navigation bars, an algorithm is adopted, which can be called the link density algorithm:

> **Algorithm 1**: When there are more than a certain number of links in a <list> tag in the DOM tree, there is a high probability that this tag is the container of the navigation bar.

The algorithm is shown above actually increased the model's accuracy, but sometimes those navigation bars were contained in a tag. In addition to that, the algorithm sometimes leads to the issue that it may filter a normal list. For example, the blog's author sometimes uses hyperlinks to redirect to the detail of the concept. As a result, the following algorithm may solve the issue:

> **Algorithm 2**: A div tag is considered to be a container for a navigation bar if the total amount of text in the tag divided by the number of <a> tags inside is higher than a certain value. We call such a number the "Ratio "of the component.

$$Ratio = \frac{Total\ number\ of\ characters}{Number\ of\ links} \tag{1}$$

6.2 Implementation

A function named *remove_nav* is created for adopting algorithm 2. During the implementation of the function, the lxml package was used to iterate all s to calculate the ratio of the component. We test using the set collected in the dataset and decide the ratio leading to the best accuracy in testing.

6.3 Evaluation

With adopting algorithm 2, the accuracy of the precision increases from 0.847 to 0.885, as the recall rate decreased from 0.878 to 0.872. The F1-Score increases from 0.862 to 0.878, and the accuracy increases from 0.159 to 0.174. Overall, the algorithm enhanced the performance of the model.