Extended Plagcomps Tool & PMC Scrapper
User Manual

SeenuArun, AndiRajendran
University of Konstanz

I. Extended Plagcomps Tool

1. Introduction

This intrinsic plagiarism analysis tool was developed by a team called 'Copy Cats', a group of six computer science majors at Carleton College[1]. The tool was developed keeping in mind the different types of plagiarism detection schemes such Extrinsic and Intrinsic. The focus will be on intrinsic mechanism. This tool Plagcomps was developed to produce a one stop solution for different types of plagiarism detection mechanisms. It has two modules, extrinsic and intrinsic. The authors' intention was to bring extrinsic and intrinsic plagiarism detection approaches under one umbrella.

2. What are the main uses of the extended tool

The tool is used to assign suspicion scores to passages after atomizing, feature extraction and detecting outliers and presents visualizations to the end-user.

3. Source code links

Old Plagcomps Source code	https://github.com/NoahCarnahan/plagcomps https://github.com/nrjones8/plagcomps_site
Extended Plagcomps Source code	https://github.com/seenu-andi-rajendran/Intrinsic_Plagiarism_Analysis

4. Important Modules

The following important modules have been created/updated during my development.

[1] *http://www.cs.carleton.edu/cs_comps/1314/dlibenno/final-results/index.html*

<table>
<tr>
<td>Plagcomps

<table>
<tr><td>Tokenization.py</td><td>Divides the document into series of atoms. The atoms may be sentences, paragraphs, words and n-chars depending on the user's choice.</td></tr>
<tr><td>feature_extraction.py</td><td>Contains all the functions which extract stylistic features.</td></tr>
<tr><td>Cluster.py</td><td>Contains functions that perform clustering to detect outliers on the scores extracted. E.g kmeans, agglom, cosine, etc.</td></tr>
<tr><td>outlier_detection.py</td><td>Detects outliers among scores using density_based function and combines the scores</td></tr>
</table></td>
<td>The module that performs all the plagiarism detection operations</td>
</tr>
<tr>
<td>Plagapp

<table>
<tr><td>Routes.py</td><td>Python file that contains all possible urls. Each url contains a submodule of the plagcomps tool.</td></tr>
<tr><td>vroni_eval.py</td><td>This programme reads all the vroni files and tests all the features or configuration files and present the results in graphs.</td></tr>
</table></td>
<td>Contains all the programmes that build the GUI that operates the plagcomps tool as a whole</td>
</tr>
</table>

5. Setting Up the tool

This project is extended version of *plagcomps/plagapp* tool developed by Noah Carnahan & co. This project is developed in Python flask framework. Those who unfamiliar with this

framework, the link[2] would serve as good start. As part of my contribution, I have extended intrinsic plagiarism analysis module of this tool and updated with some new stylistic features and one additional outlier detection method. In the front end part, I have added some visualizations to make it more interactive to see the suspicious passages.

This project is developed in python programming language. To run the module the following packages are required to be installed in virtual environment. (Versions should be strictly followed as shown below)

```
Flask==0.10.1

Flask-WTF==0.9.4

Jinja2==2.7.1

MarkupSafe==0.18

PyBrain==0.3

PyYAML==3.10

SQLAlchemy==0.9.1

WTForms==1.0.5

Werkzeug==0.9.4

backports.ssl-match-hostname==3.4.0.2

gunicorn==18.0

itsdangerous==0.23

matplotlib==1.3.1

nltk==2.0.4

nose==1.3.0

numpy==1.8.0

psycopg2==2.5.2

pyparsing==2.0.1

python-dateutil==2.2

scikit-learn==0.14.1

scipy==0.13.2

six==1.5.2
```

[2] *https://code.tutsplus.com/tutorials/creating-a-web-app-from-scratch-using-python-flask-and-mysql--cms-22972*

```
tornado==3.2

wsgiref==0.1.2
```

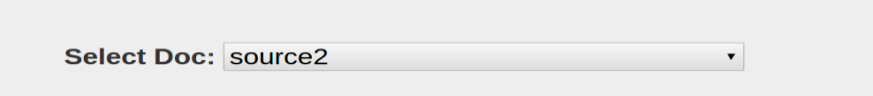
Additionaly, database connectivity is required. The database used is postgresql. The 'dbconstants.py' file in the 'plagcomps' folder must contain the database information as follows:

```
username='postgres'

password='12345678'

dbname='localhost:5432/postgres'
```

The `username, password` and the `port number` may differ from system to system. For those unfamiliar with PostgreSQL, the link[3] can serve as an introduction.

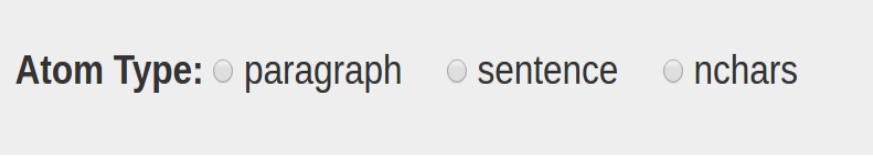
6. Plagcomps Front End

a. *Selecting single file*



Click on drop down list to select file. The list derives files from the *`plagcomps/sample_corpus`*. Therefore, the files need to be pasted there is they are to appear in the drop down list along with the ground truth in *xml* if they exist.

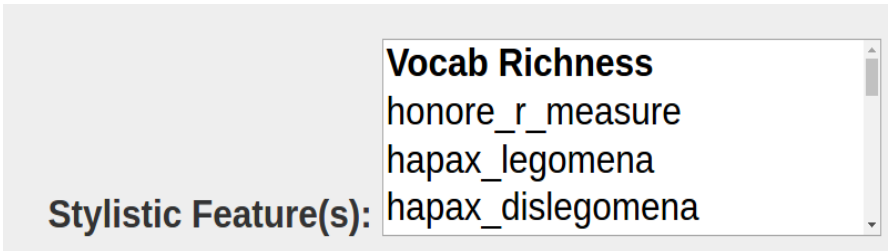
b. *Selecting atom type*



c. *Selecting Stylometry*

Clicking on the atom type will make the stylometric features to appear.

[3] *https://www.digitalocean.com/community/tutorials/how-to-install-and-use-postgresql-on-ubuntu-14-04*



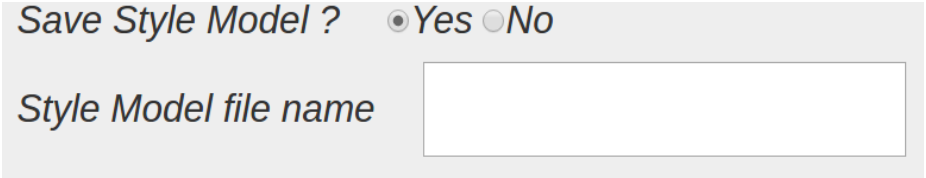
d. Selecting Outlier Detection method, mention about cluster values

Select outlier detection algorithm. Please note that the cluster value is only valid for k-means and will have no effect on any other outlier detection

e. Style model

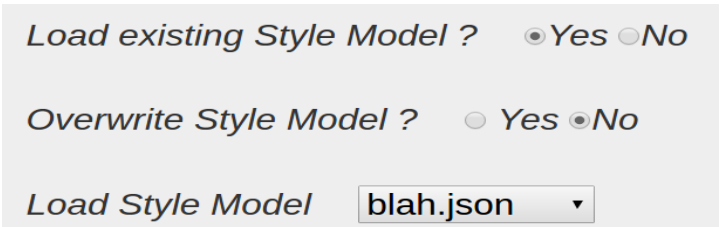
i. Step by step guide of creating style model

By selecting multiple features and clustering algorithm. How the configurations are saved in the behind the scenes.



Once selecting the stylometric features and clustering method, the style model can be saved. Selecting the 'Yes' radio button under save style model heading will make the text box appear where the style model file name can be given.

ii. How to reload style model and overwrite the style model?



On the other hand, the existing style model can be loaded after selecting the file to be analyzed. The correct style model file needs to be selected and the option for loading needs to be selected as a 'Yes'.

In case, the style model is to be overwritten, the option for loading has to be selected as 'Yes'. The atom type and clustering has to be selected and the correct style model needs to be mentioned in the drop down list.

7. Results page

a. Text view:

Showing all the atoms tool tip. Differentiate for paragraph and sentence as atom type. The textual view shows all the text. The text is divided into atoms. Each atom is colored orange, red and black, if the plagiarism confidence crosses the 90th percentile and 0.85 respectively and the rest respectively.

b. Text view with ground truth

The texts that are marked as plagiarized in the ground truth file appear as underlined in the textual view in the results page.

c. Tabular column view

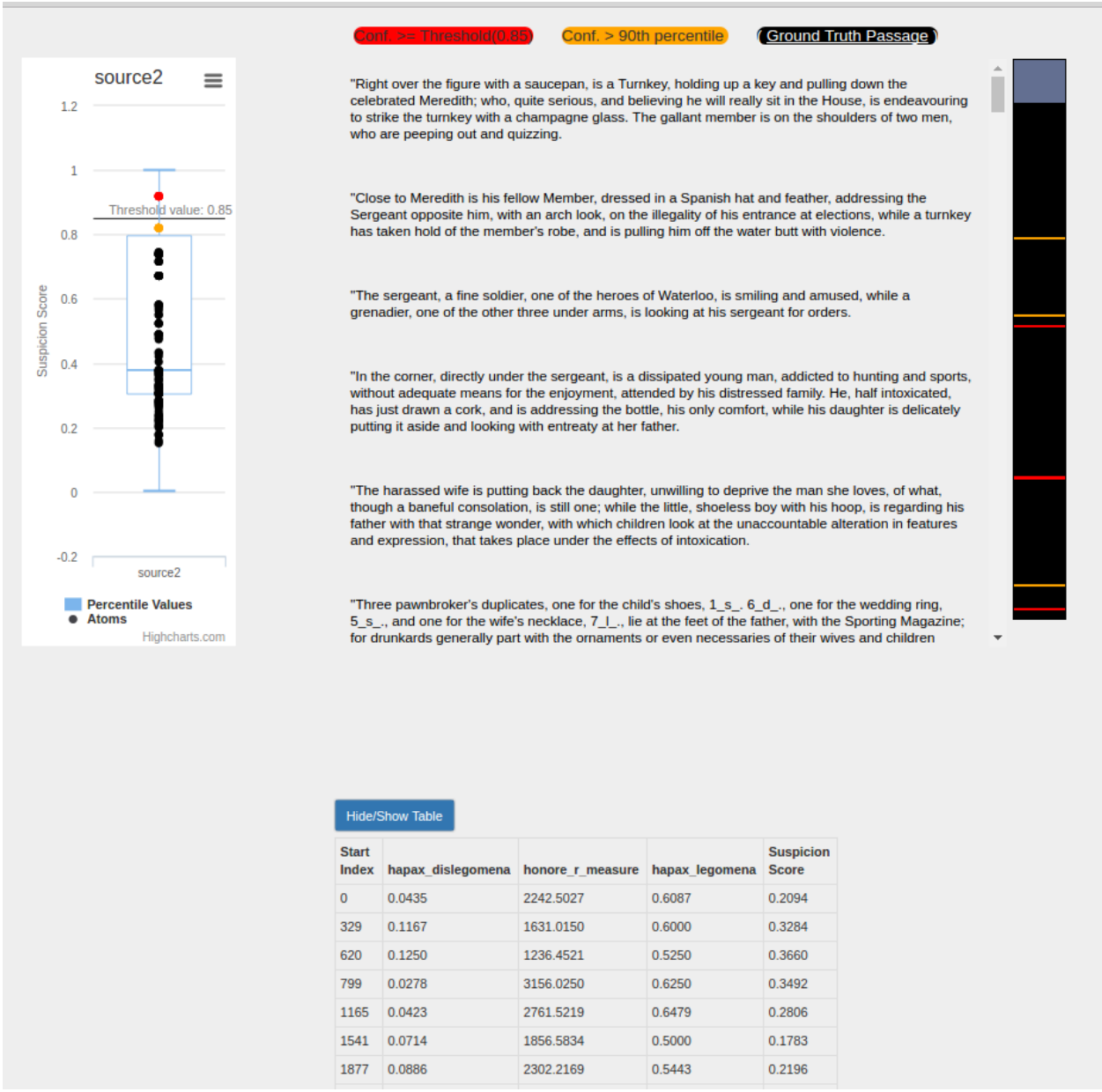
The tabular column contains three columns. The start index, where the starting position of each atom is displayed. The second column is actually divided into multiple columns where each column represents the stylometric feature selected for analysis and the scores for each atom is displayed. The final column is the plagiarism confidence which is decided by the clustering algorithm. The color changes to orange and red, if the plagiarism confidence crosses the 90th percentile and 0.85 respectively.

d. Box plot view

The box plot shows the position of the dots based on their confidence scores. If they cross the threshold, the color changes to that of the atom they represent. Clicking on the dots in boxplot scrolls the text to where the atom appears in the view.

e. Text scroll

The scrollbar on the right is used to scroll the text. It also gives an overview of all the colors of the fonts of all the atoms which change as per the plagiarism confidence into black, orange and red.



7. How tool can be further improved?

a. Front end part
   i. Finding way to display what the style model configuration consists of when it is selected.

ii. Innovative and easily understable way to connect the boxplot, text scroll, tabular column view with proper colors.

II. Technical Details of PMC Scrapper

The PubMed Central (PMC) articles to be extracted are divided into following three categories.

Category	Specifications of files in their category
X	Contains single-author publications where each author has three or more publications in his/her name.
Y	The author appearing in 'X' must have three or more collaborative publications.
Z	Single-author Publications authored by those unrelated to 'X' having a minimum of two publications each.

Table : Criteria for PMC Test Collection

Following steps were undertaken to create the test collections using python script:

1. Accessing PubMed database to search for mesh term ids under open access licensing ('CC Licenses') or user defined license type.
    a. Retrieving publication ids that belong to mesh terms given as input filtered by open access licensing ('cc license').

2. Noting down all authors (by their full names with initials) with non-collaborative publications (all_authors).
    a. Storing the authors, of non-collaborative publications, in a list. Non-collaborative publications only have one author in the list of authors.

3. Identifying those authors who appear three times or more and storing them separately (x_authors).
    a. Counting frequency of authors in the list and then storing those authors having a count of 3 or more in a separate list (x_authors).

4. Finding all the publications of these authors and downloading non-collaborative publications in the folder created for the author.
    a. Again going through the retrieved list of publication ids and downloading all non-collaborative publications of those authors in the list x_authors.

5. If the same authors have three or more collaborative publications, then store them in the folder and marking them as 'collaborative' in the filename. This falls under category 'y'.
    a. Search in the list of retrieved publications the collaborative publications and see whether any author has 3 or more collaborative publications by determining whether the author is included in the list of authors of each publication id.

6. Identify authors not belonging to list 'x_authors' but belonging to list 'all_authors' and seeing if they have two or more appearances, thereby storing them in a new list called 'z_authors', download the publications into folders created for each author again stored in folders belonging to respective MeSH Terms.

Developed scripts are as follows

Script	Input	Output
`getstats.py`	MeSH Term(s) in mesh.txt	`res.json` (containing pubmed ids) and graph to display statistics of MeSH Term(s).
`down.py`	`res.json`	Publications in Folders for all authors under their respective mesh terms.

Table : Source code details of PMC scrapper and statistics generation module.

Scripts can be found in following GitHub link[4]

[4] https://github.com/seenu-andi-rajendran/pubmed_scrape