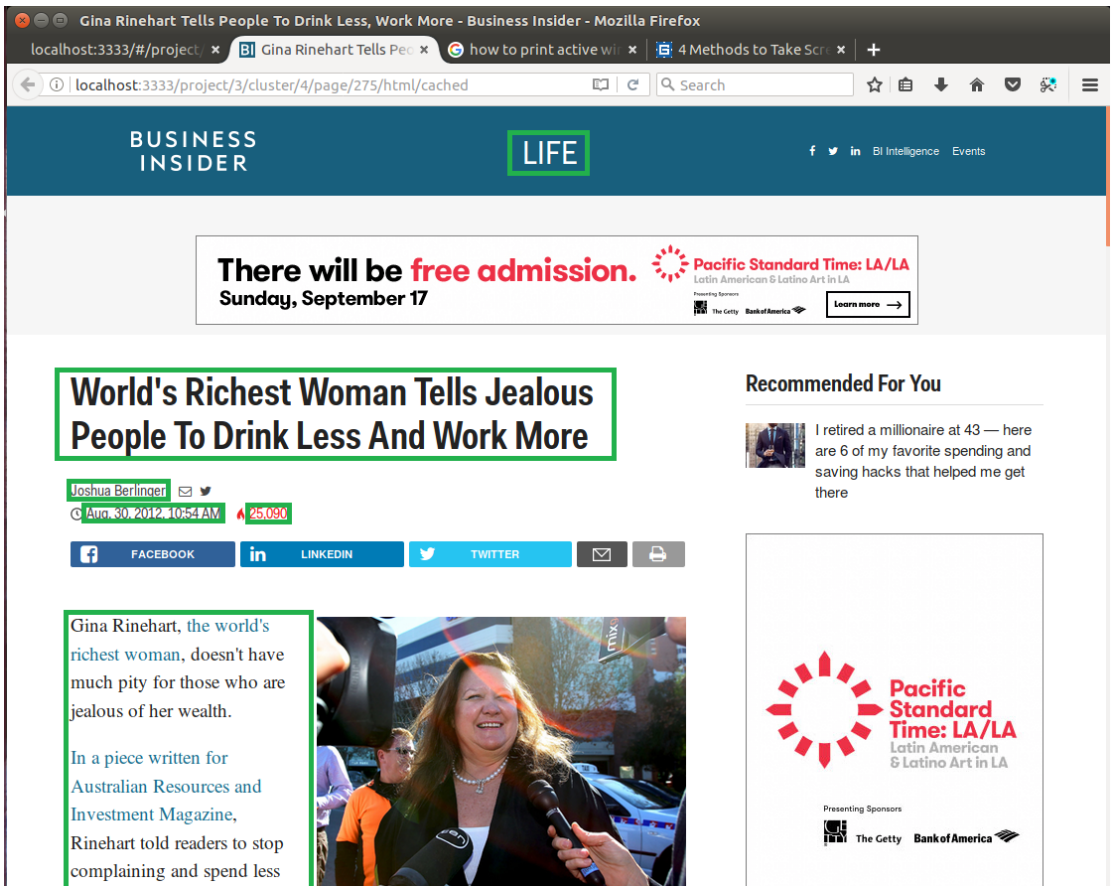
INF 558 Information Graph – Assignment 2

Student: Cheng-Lin Li USC ID: 9799716705

Task 1:

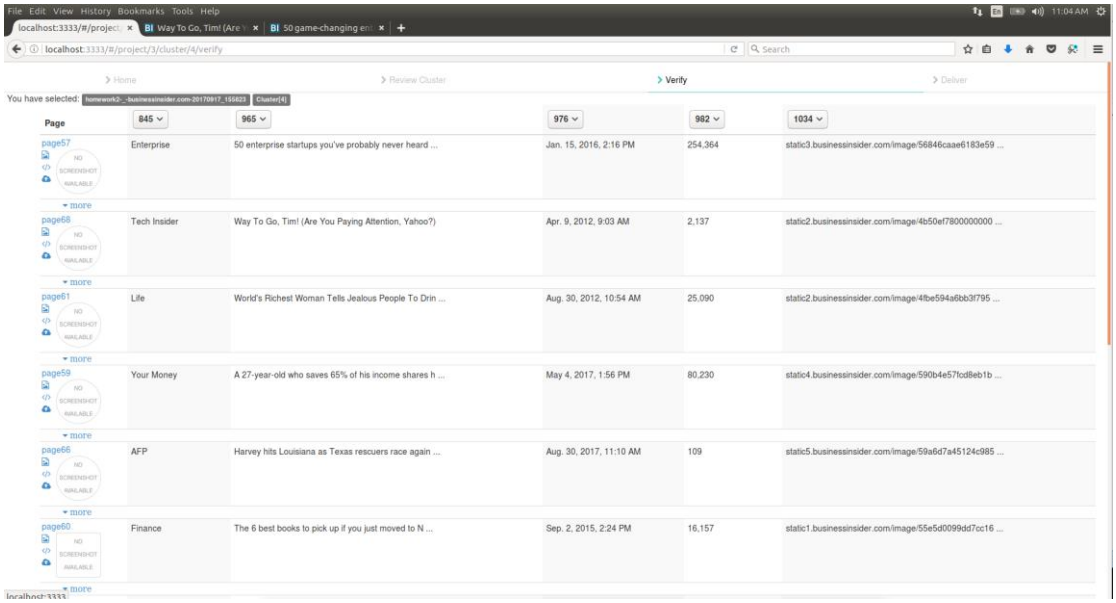
1. Interesting fields from business insider webpage:

- Answer:
    - Content:
        - Category
        - Subject
        - Content
        - Publish Date
        - Views
    - Author/Contributor:
        - Name



INF 558 Information Graph – Assignment 2

Task 2:



1. Can Inferlink tool extract your highlighted fields?

- Answer: It can extract most of fields except “author/contributor”.

2. If not, list up to 3 fields that cannot be extracted and explain why the tool cannot extract these fields. Hint: using lectures about wrapper learning and Inferlink extraction tool.

- Answer: During the template learning phase, Inferlink finds those n-grams that occur on all pages but exactly once. The “author/contributor” attributes are NOT exactly once in all training pages, i.e. Some of pages contain “ks-author-byline” in class of <li> tag but some other pages contain “single-author” in class of <li> tag for author(s) in training pages.

3. Choose one extracted rule and explain how the rules can be used to extract field from webpages.

- Answer:
    - Select rule 2:

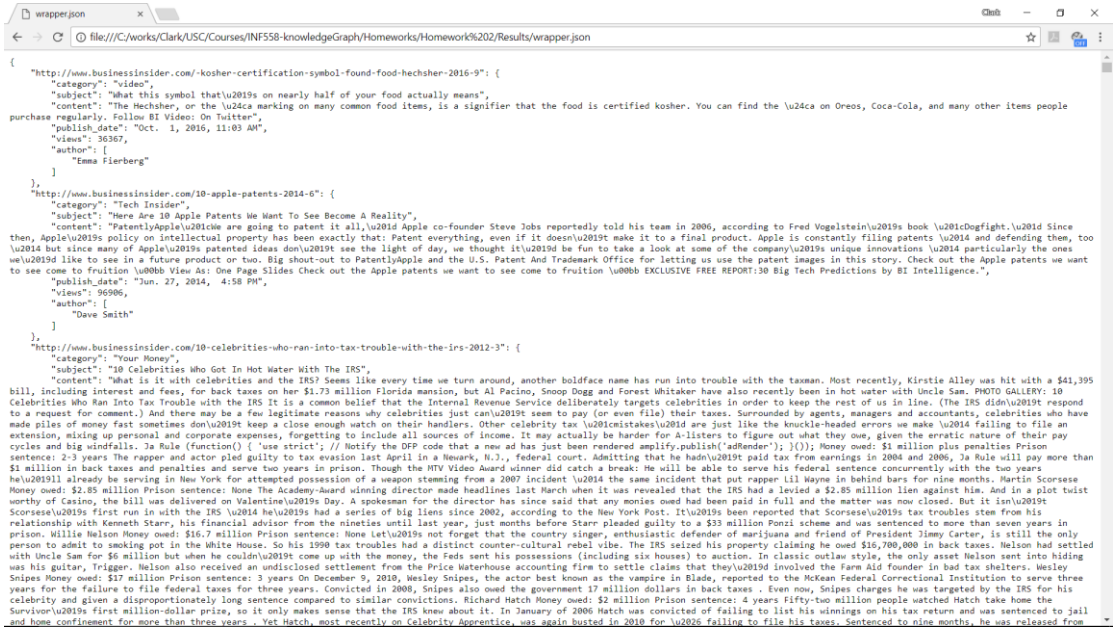

```
{
    "begin_regex":
"\\>\\s+\\</div\\>\\s+\\</div\\>\\s+\\</div\\>\\s+\\<div\\s+class=\"content\\s+post.*?\"\\>\\s
+\\<div\\s+class=\"sl\\-layout.*?\\-post\"\\>\\s+\\<.*?h1\\>",
    "end_regex":
"\\</h1\\>\\s+\\<div\\s+id=\"content\"\\s+class=\"content\"\\>\\s+\\<div\\s+class",
    "id": "af701a31-8f37-4b4b-8aae-eff7f5a24fc4",
    "include_end_regex": true,
    "name": "965",
```

```
      "removehtml": false,
      "rule_type": "ItemRule",
      "strip_end_regex":
"\\</h1\\>\\s+\\<div\\s+id=\"content\"\\s+class=\"content\"\\>\\s+\\<div\\s+class"
    },
```


- Explanation:
  - String Begin Pattern:
    “>” + whitespace (include carriage return) + 3 continuous (” ”+ whitespace) + “ ” + whitespace + ” ” + “<” + zero to multiple character + “h1>”
  - String End Pattern:
    “</h1>” + whitespace (include carriage return) + “ ” + whitespace + “<div class=”
  - Strip End Pattern: The same as String End Pattern
    “</h1>” + whitespace (include carriage return) + “ ” + whitespace + “<div class=”
  - All characters between pattern begin and end will be treated as article subject.

INF 558 Information Graph – Assignment 2

Task 3: Manual developed wrapper in Python 3.6 with Beautiful Soup 4.6




```
Usage: python wrapper.py <JSON Lines input file name> </output/path>
       The program requires Python 3.6 to execute.
       - JSON Lines input file path = input file path of JSON Lines file
       - output_path = Output file path of JSON file
        Output JSON format:
        {
            "url": {
                        "category": " Category Text ",
                        "subject": "Subject Text",
                        "content": "Article content",
                        "publish_date": "Article publish date",
                        "views": How many views on this article,
                        "author": ["author1", "author 2"...]
            },
        ...
        }
```

How to wrap:

1. Category:
   a. Locate <h2> and its class contains string ”vert-name” then get the text field.
2. Subject: Locate <h1> to get the text field for the subject of the article.
3. Content:

   a. locate and its class contains string ”KonaBody post-content” or
   b. locate and its class contains string “intro-content”

   then get the text field and remove \n, \t characters.
4. Publish Date: Locate and its class contains string ” svg sprites date-icon”, then get the text of the field from its next, next sibling tag.
5. Views: Locate and its class contains string “Engagement” then get its text and convert to integer number.
6. Author:
   a. Locate <li> and its class contains string ”ks-author-byline” then get its content, replace ‘and’ to ‘,’ if any.

      Or
   b. Locate <li> and its class contains string “single-author” then get its content.

   If the string includes ‘,’ ; then split the string by ‘,’ else put author into list.
7. During above process, if the field cannot be located, put null into the data field.
8. All above information was packaged into a dictionary object as a value which will be assigned into an external dictionary data structure with url as key.
9. Dump the externa dictionary as JSON object to file.
10. Check the JSON file and revise rules to reduce the null fields until all those null fields are real no contains cases.