

Topics in Social Computing

Progress Report

Knowledge Building Analysis Toolkit



WIKIPEDIA
The Free Encyclopedia

Mentor: Dr. Sudarshan Iyengar

Submitted by: Paras Kumar

Introduction

With the success of crowdsourced portals like Wikipedia, Stack Overflow, Quora, GitHub, etc., a class of researchers is driven towards understanding the dynamics of knowledge building on these portals. This project focuses on the analysis of the Wikipedia dataset.

Wikipedia is a platform to share knowledge where people across the globe collaborate to write an article. Like any collaboration process, articles on Wikipedia contain a lot of information apart from just what is present in the article itself and we can compare different revisions of a Wikipedia article to find out various interesting patterns in the collaboration or knowledge building process by looking at the evolution of a Wikipedia article.

Aim

Our aim is to build a programming library in Python which provides high-level operations for the analysis of knowledge data on Wikipedia articles. This wiki analysis toolkit will be available as a PyPi package and then anyone can easily use this toolkit in their project.

Github Repository

The code for the analysis which are completed can be found in my [Github repository](#).

Analysis 1

Controversy Analysis using wiki-links

What is the question?

Can we measure the relative controversy level of various wiki-links present in a wikipedia article.

Why is the question important?

Finding relative controversy level can help us to divide the set of all wiki-links present in a wikipedia article into two classes:- controversial and non-controversial. Researchers can set the threshold for controversy level based on their requirements. Then further studies can be conducted on these two classes of wiki-links.

What does the world know about it?

The research paper [Societal Controversies in Wikipedia Articles](#) tells us how we can measure the relative controversy level of wiki-links. The method presented in this paper is very similar to finding fake friendship on a social network:- two normal friends usually also have more mutual friends, but, in the case of fake friendship there would be very few or no mutual friends.

My contribution

I followed the research paper [Societal Controversies in Wikipedia Articles](#) to write a Python program to rank all the wiki-links present in a wikipedia article based on their controversy level.

Steps followed in this analysis:-

1. Finding all the sentences found in two subsequent revisions of a wikipedia article in which some words were deleted or both added and deleted, but not only added. The research paper didn't clearly mentioned the algorithm to find such sentences, so I used diff algorithm (which is used in **Git** to find changes in a **Git Commit**) to find these sentences. This was the most time consuming step.
2. Now, we remove all other text except the wiki-links from all these sentences. So, now each sentence effectively contains a set of wiki-links.
3. Now, we assume each sentence carries controversy score of 1 which is equally divided between all the wiki-links present in a sentence. Same wiki-links can occur in multiple sentences because wiki-links can occur multiple times in a wikipedia article.
4. Following step 3 we keep adding controversy scores to wiki-links until all sentences are processed. This way the wiki-links which occur alone will get higher controversy scores added each time they occur in the subsequent revisions of the wikipedia article.
5. Finally, we sort all the wiki-links based on their final controversy scores.

Conclusion

We finally have a program in Python which can find the relative controversy level of all the wiki-links present in a wikipedia article. This program reads all the revisions of a wikipedia article to arrive at these final controversy levels, so this analysis is based on the evolution of a wikipedia article over time.

Below is a sample run with 2006_Westchester_County_torna.kno1ml as input, the first image shows wiki-links with lower controversy levels and the second image shows the wiki-links with higher controversy levels.

```
Activities Terminal Sun 16:33
paras@paras-Vostro-15-3568: ~/Desktop/Knolml-Analysis

File Edit View Search Terminal Help
Checking Revision 225 ...
Checking Revision 226 ...
Checking Revision 227 ...
Checking Revision 228 ...
Checking Revision 229 ...
Checking Revision 230 ...
Checking Revision 231 ...

=====Ranking based on controversy=====

[[miles - per - hour|mph]]
--> 0.058823529411764705
[[kilometres - per - hour|km/h]]
--> 0.058823529411764705
[[Warren - County, - New - Jersey|Warren - County]]
--> 0.058823529411764705
[[Category:2006 in New - York|Westchester + York + (state)|Westchester County tornado]]
--> 0.058823529411764705
[[Category:Natural disasters in New - York]]
--> 0.058823529411764705
[[Category:Tornadoes in New - York]]
--> 0.058823529411764705
[[Category:Tornadoes in New York (state)]]
--> 0.058823529411764705
[[Storm + Prediction + Center]]
--> 0.0625
[[Category:2006 + natural + disasters + in + the + United + States|Westchester + County]]
--> 0.0625
[[Category:Four Award articles]]
--> 0.0625
[[Category:Tornadoes - in - New - York]]
--> 0.0625
[[Category:Tornadoes - in - Connecticut]]
--> 0.0625
[[Category:2006 - natural - disasters - in - the - United - States|Westchester - County]]
--> 0.0625
```

```
Activities Terminal Sun 16:34
paras@paras-Vostro-15-3568: ~/Desktop/Knolml-Analysis

File Edit View Search Terminal Help
--> 4.012394957983194
[[New York State Route 9A]]
--> 4.05959595959596
[[New - York - State - Route - 9A]]
--> 4.25
[[Hudson - River]]
--> 4.2857142857142865
[[Westchester County, New York|Westchester County]]
--> 4.333333333333333
[[Fuj|ta_scale#Parameters|F2]]
--> 4.7
*[[List of North American tornadoes and tornado outbreaks]]
--> 4.853259485612425
[[United States Dollar|USD]]
--> 5.0
[[Fairfield County, Connecticut]]
--> 5.166666666666667
[[Sleepy - Hollow, - New - York|Sleepy - Hollow]]
--> 5.166666666666667
[[Sleepy Hollow, New York|Sleepy Hollow]]
--> 6.999999999999999
[[Greenwich, - Connecticut|Greenwich]]
--> 7.249999999999999
[[United - States - Dollar|USD]]
--> 8.0
[[Consolidated Edison]]
--> 8.5
[[waterspout]]
--> 9.15
[[Westchester County, New York|Westchester]]
--> 9.68858543417367
[[Greenwich, Connecticut|Greenwich]]
--> 11.226262626262624
[[tornado]]
--> 12.18144257703081
[[Connecticut]]
--> 18.759595959595966
paras@paras-Vostro-15-3568:~/Desktop/Knolml-Analysis$
```

Analysis 2

Contributions of an author over a given period of time in a wikipedia article

What is the question?

Can we develop a function in Python to find contributions of an author in terms of words, sentences, bytes etc over a given period of time (given starting and ending dates)?

Why is the question important?

Finding a specific author's contribution over various revisions of a wikipedia article can help us to find a lot of further information about the quantity of knowledge contributed by the author. Also, this can be a common step for many different kinds of analysis on wikipedia, so this function should be present in our toolkit.

What does the world know about it?

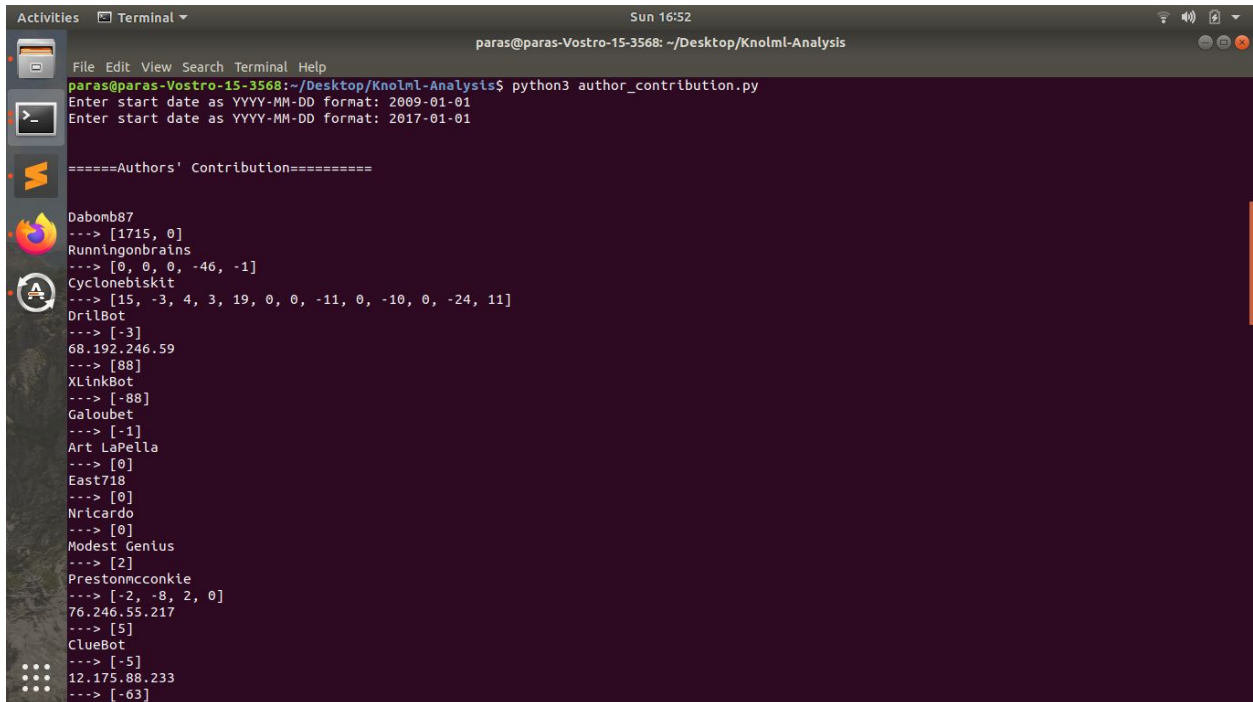
The users who see only the final article don't know anything about an author's contribution over a given period of time, only those who have all the revisions of an article can find contributions made by an author, but they have to either do this manually or implement their own function for doing this common analysis.

My contribution

I implemented the program to find contribution of an author over a given period of time. You have to give the start date, end date and author name and contribution measure(bytes/words/sentences/wiki-links) and the output will be an array of contribution in terms of given contribution measure.

Conclusion

I have generated following plots to show contributions (in terms of words) vs revisions starting between 2019-01-01 and 2017-01-01 found in the file.

A terminal window titled 'Terminal' with a menu bar (File, Edit, View, Search, Terminal, Help) and a status bar (Sun 16:52, paras@paras-Vostro-15-3568: ~/Desktop/Knolml-Analysis). The terminal shows the execution of a Python script 'author_contribution.py'. It prompts for start and end dates (2009-01-01 and 2017-01-01) and displays the output '====Authors' Contribution===='. The output lists authors and their contributions as Python lists: Dabomb87 [1715, 0], Runningonbrains [0, 0, 0, -46, -1], Cyclonebiskitt [15, -3, 4, 3, 19, 0, 0, -11, 0, -10, 0, -24, 11], DrillBot [-3], 68.192.246.59 [88], XLinkBot [-88], Galloubet [-1], Art LaPella [0], East718 [0], Nricardo [0], Modest Genius [2], Prestonmconkie [-2, -8, 2, 0], 76.246.55.217 [5], ClueBot [-5], and 12.175.88.233 [-63].

```
File Edit View Search Terminal Help
paras@paras-Vostro-15-3568:~/Desktop/Knolml-Analysis$ python3 author_contribution.py
Enter start date as YYYY-MM-DD format: 2009-01-01
Enter start date as YYYY-MM-DD format: 2017-01-01

====Authors' Contribution====

Dabomb87
--> [1715, 0]
Runningonbrains
--> [0, 0, 0, -46, -1]
Cyclonebiskitt
--> [15, -3, 4, 3, 19, 0, 0, -11, 0, -10, 0, -24, 11]
DrillBot
--> [-3]
68.192.246.59
--> [88]
XLinkBot
--> [-88]
Galloubet
--> [-1]
Art LaPella
--> [0]
East718
--> [0]
Nricardo
--> [0]
Modest Genius
--> [2]
Prestonmconkie
--> [-2, -8, 2, 0]
76.246.55.217
--> [5]
ClueBot
--> [-5]
12.175.88.233
--> [-63]
```

The above contributions are in terms of number of words added. Therefore in case of replacing words net contribution is zero, in case of net addition of words the value is positive and in case net deletion of words contribution is negative. The values of all contributions for a particular author is returned in the form of a Python list.

Analysis 3

Ranking all the authors based on their contribution to a given paragraph

What is the question?

Can we rank all the authors of a wikipedia article based on their contribution to a particular paragraph present in the article? The paragraph will be given as input to the program.

Why is the question important?

Suppose a phrase or a paragraph from a wikipedia article gets viral over social media and we want to know which author(s) contributed to this viral phrase then we can use this program to track them.

What does the world know about it?

The idea of finding original contributor(s) for an important phrase or paragraph is not new, but we are doing this analysis for a specific purpose:- for wikipedia data in .knolml format. Also, in our analysis we are not using simple string matching for finding the amount of knowledge contributed by a particular author, rather we are considering those words which can change the meaning conveyed by a particular paragraph.

My contribution

This analysis may look quite similar to the previous analysis, but here we have to find a contributions of all authors to a given paragraph (unlike the complete article in case of previous analysis) and also here we are ranking the authors according to

their contribution to this given paragraph.

But, these two modifications pose following challenges:-

- The paragraph given as input will change over time, so simple string matching will not work. One more reason why simple string matching will not work in this analysis is because let's say the paragraph given as input has the word "don't" while originally the paragraph had the word "do not". So, the author who made this change didn't change the meaning of the paragraph or added any new knowledge to it and we shouldn't measure it as a valid contribution.
- The author who first wrote the paragraph should be ranked higher because he was the original author of the knowledge presented by that particular paragraph. Later, on other other authors would have only refined the original idea and they should be ranked lower.

Overcoming these challenges

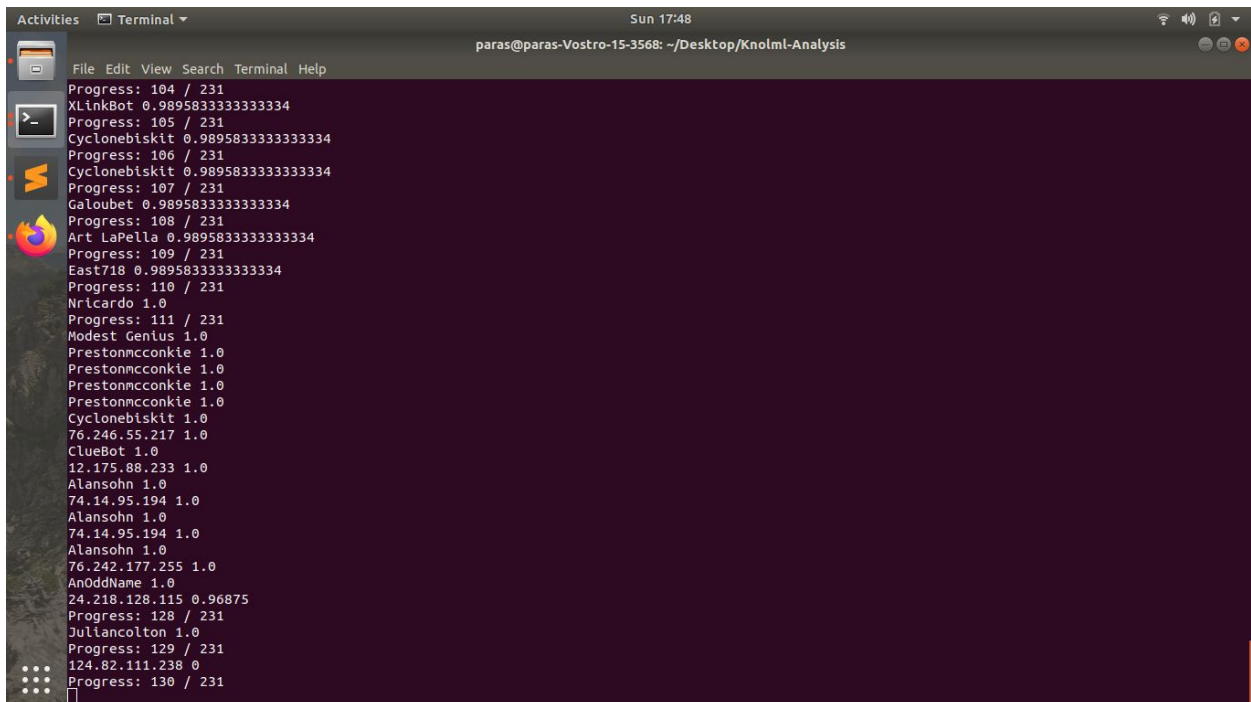
- Instead of using simple string matching we are using a similarity measure between two paragraphs or sentences. Currently, I'm using cosine similarity which I may change in the future if I can get better results.
- Before calculating the contributions for all authors we first have to preprocess all the revisions of the wikipedia article. The words which don't carry any useful information like "don't", "this", "you" etc are removed because they don't correspond to any significant contribution to the wikipedia article.
- Similarity measure of 1 corresponds to perfect match after preprocessing the articles which gives a contribution score of 1 to the first author, if the similarity remains 1 we don't assign any score to next authors since no change was made by them in other cases the similarity value is assigned as score.

Conclusion

We finally have a program in Python which can rank all the authors of a wikipedia article based on their contribution to a particular paragraph present in the article. The paragraph will be given as input to the program.

Below is a sample run of the code for finding contributions for the paragraph:-

"Thousands of trees were either uprooted or snapped along the tornado's path through the state. Minor damage was inflicted upon several structures. The tornado left 1,700 residences in Greenwich without power and blocked six roads. Most of the damage was concentrated to the northwestern corner of the town. Damages in the state totaled to \$2 million (2006 USD). ==Aftermath== In the wake of the tornado, the mayor of Sleepy Hollow declared a village-wide state of emergency. Two hundred emergency personnel responded to the storm. (conEdison) crews were sent out to repair downed power lines and clear roads. By the next night, power was restored to all but 600 of the previous 10,000 residences without power in . Westchester County opened its Emergency Operations Center after the storm to quickly respond to the event. Two days after the storm, many of the roads had been cleared and power was fully restored. A recreational path in , was not expected to be open for another two weeks due to numerous fallen trees." which was selected from 116th revision of the article in 2006_Westchester_County_torna.knolml having total 231 revisions:-



```
Activities  Terminal  Sun 17:48
paras@paras-Vostro-15-3568: ~/Desktop/Knolml-Analysis

Progress: 104 / 231
XLinkBot 0.9895833333333334
Progress: 105 / 231
Cyclonebiskit 0.9895833333333334
Progress: 106 / 231
Cyclonebiskit 0.9895833333333334
Progress: 107 / 231
Galoubet 0.9895833333333334
Progress: 108 / 231
Art LaPella 0.9895833333333334
Progress: 109 / 231
East718 0.9895833333333334
Progress: 110 / 231
Nrlcardo 1.0
Progress: 111 / 231
Modest Genius 1.0
Prestonmconkie 1.0
Prestonmconkie 1.0
Prestonmconkie 1.0
Prestonmconkie 1.0
Cyclonebiskit 1.0
76.246.55.217 1.0
ClueBot 1.0
12.175.88.233 1.0
Alansohn 1.0
74.14.95.194 1.0
Alansohn 1.0
74.14.95.194 1.0
Alansohn 1.0
76.242.177.255 1.0
AnOddName 1.0
24.218.128.115 0.96875
Progress: 128 / 231
Juliancolton 1.0
Progress: 129 / 231
124.82.111.238 0
Progress: 130 / 231
```

In the above image we see that the similarity value was 1 for the first time in 112th revision when the author was 'Modest Genius', so he is the original author of this paragraph, although we copied this paragraph from 116th revision who's the author was 'Prestonmcconkie', this means either he didn't change this paragraph or he changed some words which didn't change the meaning of the searched paragraph.

Another pattern that we observe here is that similarity value first increases and reaches 1 and then again decreases and it can then keep repeating this pattern. This happens because some part of knowledge present in the searched paragraph might be derived from some previous revisions and there the similarity values are less than 1. When there is a perfect match of knowledge the similarity value is 1 after that some authors may change the searched paragraph which lowers the value of similarity. But, one thing to note here is that the similarity value can again become 1 in later revisions because the changes which resulted in decrease of similarity value of 1 might be undone by some authors in later revisions, but again they can't be considered the original author of that paragraph in terms of the knowledge represented by the paragraph.

So, in summary the original author is the author of the revision where the similarity value reaches 1 for the first time and other authors have lesser contribution to the given paragraph who's contributions can be measured by the similarity values.

Analysis 4 (Currently working on it)

Finding knowledge gaps in a wikipedia article

What is the question?

A wikipedia article represents knowledge about some related topics, like a wikipedia article on IIT Ropar may be talking about placements of IIT Ropar in a particular section. But, in this section there was no information regarding a new branch say Biotechnology which was newly introduced. So, can we write a Python program that can tell that the information regarding placements of Biotechnology is missing from the IIT Ropar wikipedia page? Or in general can we tell that there is a knowledge gap in a wikipedia article?

Why is the question important?

Finding knowledge gaps can have various applications and one such application can be in developing an app which suggests the author of an article about possible knowledge gaps and thus it can help the author to improve the quality of articles. This analysis can further be extended to enrich the text, but, my current analysis is to identify the knowledge gaps.

What does the world know about it?

There exists a research paper by Qiao and Hu titled [Measuring Knowledge Gaps in Student Responses by Mining Networked Representations of Texts](#). This study is based on educational question answering (QA). In this QA, the primary goal was to assess whether students have mastered certain knowledge. They examined the gap between two sources, student response vs assessed knowledge based on concretized text representations of both the sources.

My contribution

My first task was to understand what is knowledge gap and then understanding the difference between **internal and external knowledge gap**. External gap arises because of a certain jump of information in an article. Here by jump of information I mean that the reader expected some information between two subsequent segments of the text in an article, but it was not found. Second type of knowledge gap is due to ambiguity or difficulty in understanding of a particular segment of text which is termed as internal knowledge gap.

After understanding the problem at a high level, my next task was to identify the steps which I may have to follow to find knowledge gaps. As discussed above both internal and external knowledge gaps are defined in terms of sections in an article. But, how do we first find these sections or segments of text which are talking about a particular topic. So, the **first step is to segment the text**. Once we have found these segments then our **second step is to identify internal knowledge gaps within a text segment and external knowledge gaps between consequent text segments**.

Currently I am reading a research paper by Alemi and Ginsparg titled [Text Segmentation based on Semantic Word Embeddings](#). This article presents a method to segment texts into naturally coherent sections using word2vec. Here they learnt a model for word-word co-occurrence statistics, such that the probability of two words appearing near one another is proportional to the exponential of their dot product in the vector form. I am still reading this research paper and trying to understand the word embedding they have used.

Conclusion

This analysis is not yet complete, so I can't reach any conclusions yet.