Queer-E

By Guy Kachur

Erotica is just a Queer-E away

Table of Contents

Introduction

Over the years queer erotica has been persecuted from many websites. Media containing topics deemed inappropriate has been taken down from many sites. A few hosting websites have sprung up over the years, most notably Nifty.org which has proven to be a safe haven for LGBT stories of all kinds. The website is largely run by volunteers and almost entirely is supported by individual donations. As such they have not had a lot of resources to spend on organization or functionality of their search, which appears to be limited in robustness towards more complex queries. Many sites are similar to this, sprung up from necessity by the people that are using them. This project aims to build a more cohesive search engine for this erotica online. Allowing users to query and explore erotica in a user focused way, while providing an attempt to deliver the most relevant results for the query.

Problem summary

Queer erotica is scatterd online across many archives and hosting websites. The corpus has little structure and unknown formats between the different archives. The corpus is also user created, meaning there is a lot of data that has spelling errors or formatting problems which has made analysis more difficult. Aside from the searching giants there is no search engine for this topic.

Dataset Generation & Characteristics

To start, I created a set of seed urls. Using these seed urls I started to crawl the internet concerning queer erotica. This proved difficult as my own homemade crawler was not as fault tolerant or as respectful as needed. My initial dataset was around 10gb of crawled pages from many different websites. Transitioning to nutch I was able to generate a lot more data. Nutch helped me crawl the many websites much faster, and allowed for data saving between crawls. Overall improving my ability to mine the web of queer erotica.

To give myself more time to focus on the search task of this project I condensed my scope and decided to focus on one archive, Nifty.org. This allowed me to assure some structure, meaning I could gather and use more relevant metadata. Nifty uses user generated categories that are different between sexual orientations. The Gay orientation has many categories that are different than those found in the Transgender section and the Lesbian section. Some of these categories also include non-english stories or stories that do not fit anywhere else. To make matters more difficult after exploring some of these categories more, nifty accepts spelling problems in their urls. So the categories college, and collage both exist. Stories are also linked and hosted from many different places, i.e. a story could be in college and celebrity even though it's the same story. This meant that using the URL to detect duplicates did not always work.

To deal with some of these problems I spent some time creating a parser for nifty content. This parser tried to extract the authors name and email, as well as the orientation and category for each story. The data I ended up having is as follows:

Story	Example Data
Title	dylan's-summer-vacation-20
Date	2013-07-04T12:07:08Z
Email	thinat20@yahoo.com
Author	donny mumford
URL	www.nifty.org\\nifty\\gay\\college\\dylans-summer-vacation\\dylans-summer-vacation-20
Category	college

Orientation	gay
Body	<The text of the story>

Overall I ended up having two sets of data, that which I crawled and had not parsed/dealt with. The crawled data exceeded 30gb of data, excluding random video files from ad links.

Limiting that data, to only the data on the Nifty.org archive, I ended up having.

4.56 GB of data,

164372 Documents

6000 Duplicates

Problems with Crawling

I ran into many problems. First with my home grown crawler, It took many iterations to get a more fault tolerant crawler. That could deal with incorrect sources, bad parsing, etc. Adding in functionality to limit domains was also difficult as unlike most of the crawling we learned about in class I was crawling a few domains very deeply. This provided its own challenges. Limiting by domain worked well enough but Nifty.org is actually a conglomerate sight, there are multiple mirrors that handle it, each with their own naming scheme, with many links pointing to the different mirrors. So limiting to one is taxing on the domain/server, which are run by volunteers' contributions, and spreading the load to the others servers was hard to do depending on the urls grabbed the content tended to stay on the specific domain. Additionally crawling multiple domains for the same content generated many duplicates. Nutch, while powerful, when applied to a domain structured like a file system i.e. one inlink, no outlink per leaf, did not help as much as it could have. I ended up not incorporating the boost feature of Nutch as most links had almost no boosting factor, and most links just had similar scores. There is also a website, http://www.best-of-nifty.org/, which maintains a list of the best stories on nifty as voted on by their user base. Originally I wanted to use nutch to boost stories that were found on that website as these stories can be assured higher quality and if they share terms with the search query would almost assuredly be a more relevant result.

Indexing

As my data was (largely) in english so I decided to take advantage of solar ability to stem and tokenize with the text_en field. I also copied both the email and url fields into two fields, one un-tokenized and one tokenized so I could search the URL and email but I also wanted to keep them un-tokenized as they lose a lot of their meaning when they are tokenized. Also people do not really want to search based on the domain of a url.

Indexing was interesting as the main problem I had was my dataset had many duplicates. To tackle that I took advantage of Solr's duplicate detection which needed to be done when the documents were indexed. I created my own signature field. Using the default of all fields seemed to result in too many duplicates. I used a few different combinations of fields. Title alone produced way too many duplicates as the title for one chapter was often only 1 character different than the other chapters. Eventually I settled on title, body, and author. As that seemed to do the best at catching duplicates but leaving series chapters alone.

Ranking

To rank my results I mostly boosted specific fields. Title being the most relevant. Content was next, as that would have the most results. Since I took the category and orientation out, I didn't boost by those. Instead I filtered results by them. Especially since most of the time the term would not have appeared in either. Boosting by the author rarely seemed to have any effect, except when searching for the author's exact name. I ended up removing that boost as the author field I had been searching on was tokenized and when boosted sometimes returned results that I deemed non-relevant. When I made the field a string my Solr would randomly include spaces which messed up even exact word matching. I.r. test123@yahoo.com is different than test123@yahoo.com (the second one has spaces around it). This made even querying by author more difficult, but if you know the exact name of the author, it is found and returned well, as that term really only appears in the stories the author created.

MoreLikeThis

More like this was hard to narrow down. I tried using many different fields to decide them. As I wanted it to favor the similar entries to the stories, but also provide a way to get different stories. I weighed author heavily, as that should be the best indication that a story will be similar to the one selected, than the title which I boosted secondly because I wanted some of the series to show up, but I didn't want it to dominate the results, and lastly content of the story. URL provided to be useless as most of the stories have a very similar URL. I was hoping to be able to separate based on series here, but I did not have a lot of luck in that regard. I do like how I boosted my more like this seems to give me what I wanted, i.e. you usually see other sections of the series, but it is not dominated by only that, and the same author shows up a lot.

Frontend

My frontend was very important to me. I wanted to create a clean and easy experience that enabled filtering for different categories and orientations. The UI needed to be fun and inviting, a little bit queer, and easy to use. By using a tree to display the orientations and categories can easily be toggled, and will filter the results based on the orientation and category. I decided to represent the results in a card format, with the title being a link to the story. The author being a link to the authors email. I also used solr to generate a snippet, fiddling with the snippet size until it seemed to give me a good snippet, that had some context, but also wasn't too long that it was more confusing than helpful. Using only one snippet seemed like a better idea than using all of them, but perhaps I could have used a collapsing feature, and only displayed more on a users prompt.

Since the rainbow is representative of the queer community at large, it seemed in theme to match it to the logo and to use it as a stylistic element in parts of the page. I used it to highlight the term that people were searching for across the page. I also used it to highlight the suggested more reading. All of which are links to the stories on the archives themselves.

I also played around making the logo. My original design was too childish, and upon user feedback I settled on something that was a bit more book-ish to stay in theme for the erotica.

Future Improvements

UI: In the future I want to use color/icons to denote all the categories and orientations. I also would like to grab the favicons of the websites and use those to display where the story was indexed from, adding a functionality to sort by domain. Giving the users more freedom to choose what domain they want to get content from.

Ranking: I would love to be able to allow users to give feedback on the resultlist, upvoting or downvoting stories. As I talked about earlier the ability to give results a higher score if they are more linked on websites.

Figures & Tables

Top most frequent adult words

1. Cock: 117846
2. Fuck : 109690
3. Ass : 107024
4. Sex : 106115
5. Cum : 105619

Top most frequent Authors *

*These are just the times the term appears, not the amount of stories they have created

1. Unknown_Author 37040
2. Billy Billy 1302
3. DOGG 985
4. Bob Archman 907
5. Bill Jonners 770

Orientations and Frequencies

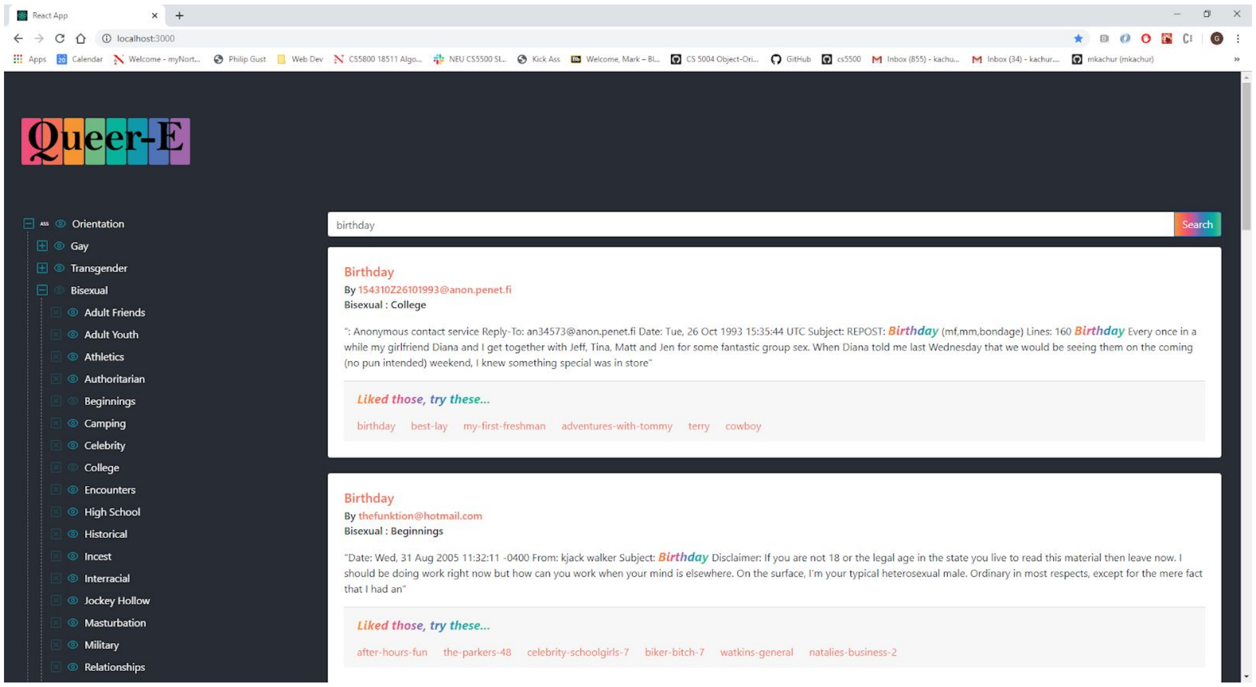
1. Gay 121038
2. Bisexual 17309
3. Lesbian 10983
4. Transgender 10737

Top 5 Categories

1. Adult-youth 14607
2. Incest 13492
3. Highschool 11881

4. Authoritarian 9789
5. College 7331

UI Sample



birthday

Birthday
By 154310Z26101993@anon.penet.fi
Bisexual : College

": Anonymous contact service Reply-To: an34573@anon.penet.fi Date: Tue, 26 Oct 1993 15:35:44 UTC Subject: REPOST: *Birthday* (mf,mm,bondage) Lines: 160 *Birthday* Every once in a while my girlfriend Diana and I get together with Jeff, Tina, Matt and Jen for some fantastic group sex. When Diana told me last Wednesday that we would be seeing them on the coming (no pun intended) weekend, I knew something special was in store"

Liked those, try these...

birthday best-lay my-first-freshman adventures-with-tommy terry cowboy

Birthday
By thefunktion@hotmail.com
Bisexual : Beginnings

"Date: Wed, 31 Aug 2005 11:32:11 -0400 From: kjack walker Subject: *Birthday* Disclaimer: If you are not 18 or the legal age in the state you live to read this material then leave now. I should be doing work right now but how can you work when your mind is elsewhere. On the surface, I'm your typical heterosexual male. Ordinary in most respects, except for the mere fact that I had an"

Liked those, try these...

after-hours-fun the-parkers-48 celebrity-schoolgirls-7 biker-bitch-7 watkins-general natalies-business-2

Logo

Queer-E

Stack

Solr was used to host the index, and allowing fast searching.

My own custom parser and uploader was built in java.

The frontend is built with React and Javascript.

Crawling was done with my own custom crawler in java, than with a third party website copying tool, than eventually with Nutch.