UrbanDictionary PageRank Report

Group: Brian Xie, Emily Miller, & Quinn Tran
EE126
Prof. Ramchandran

Introduction

Our objective is to find the hit rate of synonyms on UrbanDictionary and create a separation factor from the collection of words with the most hits.

Pseudocode

```
#Returns outgoing edges of word
procedure arcs(word):
    return {hyperlink | hyperlink in definition of word or "related words" section of word}
#Updates maps and counters for visited word
procedure visit(word):
    if word not in cache:
        outgoingLinks = arcs(word)
        associatedWords[word] = unique elements of outgoingLinks:  counts
        cache[words] = outgoingLinks
    counter[word] += 1
procedure pagerank(n, d):
    n := number of root words
    d := max depth to explore each root word
    counter := {word:#appearances = 0}
    associatedWords := {rootWord:associates and counts on page}
    cache := {word:list of links on page}
    do n times:
        word = randomly seeded root word
        do d times:
                visit(word)
                with small probability:
                        word = randomly seeded word
                else:
                        outgoingLinks = arcs(word)
                        if outgoingLinks not empty:
                            word = random(outgoingLinks)
                        else:
                            break inner loop
    return counter
```

Theory

We start by selecting multiple root words and a set depth rather than a single root; this is more advantageous because selecting only a single root risks putting us in a relatively inaccessible part of the graph, so we could spend a large number of iterations traversing a large, yet unimportant subset of nodes. We traverse using *Beautiful Soup* to one of the words referenced in root's definition or the related words section. For words with no outgoing links supplied by by URBANDICTIONARY users, we terminate the search and pick a new random root to continue the process.

Optimal:

The solution that seemed the most optimal was to cache web crawling and to traverse a depth greater than 10 for every root word. We also believed that a cache would speed up the traverse and the data structure Dictionaries would keep re-accessing simple.

Alternative:

We considered having one thread web crawling for the same amount of depth for every root word.

```
bushwacked mcdonalds 7.04740507498e-07
bushwacked subway 7.04740507498e-07
bushwacked stinson 7.04740507498e-07
bushwacked stripper 1.409481015e-06
bushwacked tv 1.409481015e-06
bushwacked how+i+met+your+mother 1.409481015e-06
bushwacked metro 1.409481015e-06
bushwacked prejudice 1.409481015e-06
bushwacked nyc 2.11422152249e-06
bushwacked barney 2.81896202999e-06
bushwacked blacks 2.81896202999e-06
bushwacked racists 2.81896202999e-06
bushwacked slit 4.93318355248e-06
bushwacked metrosexual 5.63792405998e-06
bushwacked lie 6.34266456748e-06
bushwacked hag 7.75214558248e-06
bushwacked muff 1.47995506575e-05
bushwacked gash 1.62090316724e-05
bushwacked cooter 1.62090316724e-05
bushwacked asian 1.69137721799e-05
bushwacked white 2.04374747174e-05
bushwacked racist 2.25516962399e-05
bushwacked heterosexual 2.32564367474e-05
bushwacked neat 2.32564367474e-05
bushwacked gay+old+time 2.88943608074e-05
bushwacked black 2.88943608074e-05
```

(a) Least associated with root

```
bushwacked lovely 3.17133228374e-05
bushwacked fucked 3.52370253749e-05
bushwacked kind 3.73512468974e-05
bushwacked caring 4.29891709574e-05
bushwacked vag 5.49697595848e-05
bushwacked snatch 5.63792405998e-05
bushwacked gorgeous 6.27219051673e-05
bushwacked fucking 6.62456077048e-05
bushwacked perfect 6.62456077048e-05
bushwacked loving 6.97693102423e-05
bushwacked sweet 0.000136014917947
bushwacked nice 0.000143062323022
bushwacked funny 0.000169137721799
bushwacked amazing 0.000192394158547
bushwacked hot 0.000214945854787
bushwacked cool 0.000218469557324
bushwacked pussy 0.000342503886644
bushwacked gay 0.000388312019631
bushwacked sex 0.000654703931465
```

(b) Most associated with root

Figure 1: Sample root and weight of hits per word

Experiments/Implementation

Using words from the favorites list as root resulted in low variance which gave us no substantial information about how modern day users shape and contribute to the English Language. When we changed the root word to a random word on UrbanDictionary the results varied enough that we could calculate a trend from the number of hits after ranking the synonyms. We found that the cache was sufficient to speed up traversal because the terms in urban dictionary converge to the same, small list of inappropriate words in the steady state. Thus, multi-threading isn't necessary: almost every word can be linked back to the same limited set of words relatively quickly through traversal. We realized that multi-threading would be difficult to implement in conjunction with caching as well. When each thread tries to access the same cache there would be conflicts where one thread tries to check the cache while another thread is writing to the cache. If not handled properly you will have many idle threads, so the benefits of multi-threading are less significant. The only idea we kept from our original optimal theory was to use dictionaries.

```
Selected words in the top 100 most popular:
1 sex 0.0301163808474
10 sexy 0.0118001750575
13 awesome 0.010146853827
14 cool 0.0100495996369
15 hot 0.00988750932019
16 amazing 0.00885013129316
17 love 0.00803967970953
19 funny 0.00778033520277
20 beautiful 0.00739131844264
22 cute 0.00690504749246
25 nice 0.00658086685901
27 sweet 0.00625668622556
28 girl 0.00609459590884
31 stupid 0.00596492365546
33 pretty 0.00525172626187
34 balls 0.00466820112166
35 smart 0.00460336499498
37 fun 0.00440885661491
40 ugly 0.00359840503128
42 weed 0.00343631471456
43 loser 0.00337147858787
47 loving 0.00320938827114
49 butt 0.00311213408111
50 perfect 0.00304729795442
53 great 0.00301487989108
54 idiot 0.00301487989108
55 ho 0.00291762570104
56 gorgeous 0.0028852076377
58 hoe 0.00262586313094
60 snatch 0.00259344506759
61 douche 0.00256102700425
65 sick 0.00246377281421
66 attractive 0.00243135475087
67 fat 0.00230168249749
68 marijuana 0.00220442830745
69 crazy 0.00217201024411
71 drugs 0.00210717411742
73 caring 0.00197750186404
74 pot 0.00194508380069
75 drunk 0.00191266573735
76 joint 0.00181541154731
77 lol 0.00171815735728
82 douchebag 0.00171815735728
83 toilet 0.00171815735728
84 kind 0.00171815735728
85 high 0.00168573929393
88 music 0.00162090316724
89 smoke 0.00162090316724
91 tramp 0.00162090316724
92 good 0.00162090316724
94 straight 0.00149123091387
95 moron 0.00149123091387
96 prostitute 0.00149123091387
97 dump 0.00149123091387
98 lovely 0.00145881285052
100 annoying 0.00142639478718

Average length of 100 most popular words: 5.19
Average length of 200 most popular words: 5.49
Average length of 500 most popular words: 5.736
Average length of 1000 most popular words: 6.153
Average length of 5000 most popular words: 7.9118

Average length of 100 least popular words: 9.39
Average length of 200 least popular words: 9.845
Average length of 500 least popular words: 9.278
Average length of 1000 least popular words: 8.861
Average length of 5000 least popular words: 8.6772
```

Figure 2: Top 100 Most Popular Words

Results & Analysis

Our results linked words that occur with the highest frequency in the lexicons of UrbanDictionary users.UrbanDictionary does contain a list of popular words, but these may be sorted by some other metric, such as upvotes or staff picks; this may list words that are considered interesting, but not words which are actually used. In addition, because our Pagerank links to related words as well, we can gather a higher-level overview of the topics which users are concerned with. We notice that the UrbanDictionary website is mostly static, with changes only occurring when users vote, add words, or add definitions. Since these events are very infrequent, they can be ignored and the website can be modeled as a graph where each word is a node whose characteristics do not change. Then each unique word only needs to be visited once, with its outgoing links stored in a data-structure; the next time the same link is queried, we avoid the expensive operation of loading the page over a network and parsing through the html. Then we can simply access its outgoing links through the data-structure because we assume they will be the same as they were when the page was initially accessed. Using this method, we are able to gather much more data in the same amount of time. Our weighted graphs depict how more than half of the first couple of thousand iterations were previously visited words.

We conclude that Pageranking urbandictionary reveals the unsurprising and common theme of sex, humor, and the feeling of being devastated. Ironically, although more and more words are being crowdsourced, the terms converged to cuss words or sexual descriptions. The top words were all closely related to each other, which was why finding separate subgraphs was difficult. As seen in Figure 1, given a large depth, the cache would grow logarithmically and the crawler would essentially visit the same set of words as the number of links visited goes to infinity. The popular words were essentially in one cluster. In a high level, urban dictionary represents current events and pop culture. In a metaphorical level, urban dictionary perpetuates common themes in humanity. It is channel for the masses to voice their sexual frustrations.

Discussion/Limits

The experiment was carried out on a small scale with the algorithm returning back to root after reaching a certain finite depth while crawling through UrbanDictionary but if the algorithm was not restricted and we had the necessary computing power, the results would be applicable to the field of natural language processing. We could have tried parsing through the actual definitions to construct full sentences (though that would be a more general Markov chain operation and not strictly a Pagerank one). We were not able to create a new word because our PageRank algorithm would have given us the most popular words and concatenating them together does not necessarily create a "new" word. Also by time constraint we were not able to create a new word because peer review on UrbanDictionary takes a few days to approve of a new word.

Citing

Report: ee126pagerank.pdf
Our Code: crawler.py , formatted_graph.py
Outside Code: *BeautifulSoup, Pickle, Urlparse/Urllib, Networkx*

Figure 3: Set of 4 roots with 4 click depth per root

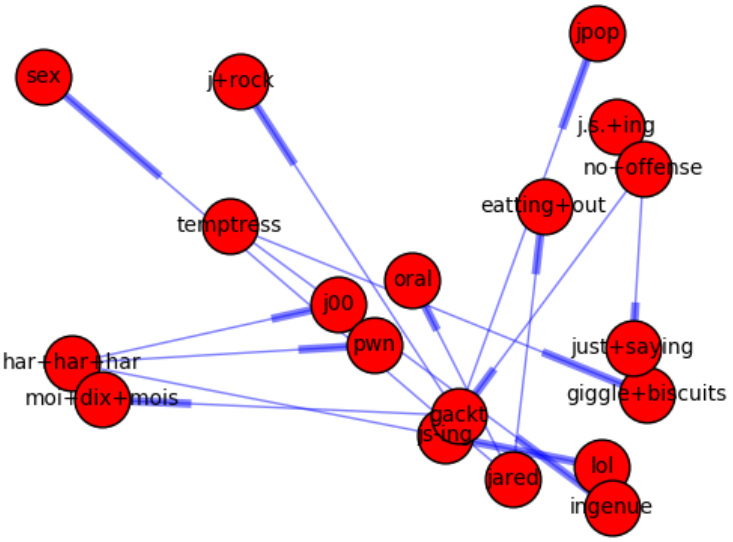

Figure 4: Set of 10 roots with 5 click depth per root