Contribution

Literature Review

William Marsey
Imperial College London

June 2014

1 Introduction

1.1 The Research Question

Given a collaboratively edited document, and information about the collection of edits that constitute that document, may we measure the quality of each contribution? And may we use that to give all the collaborators a algorithmically-defined 'stake' in that final document?

Collaborative work is becoming a big deal. It is both interesting and an important trend in modern computer use. And the data is abundant.

Amongst many other things, this topic is a playground for sociology, machine learning, network studies, as well as more general studies of conflict, and personality. My work intends to focus on the algorithmic side of things - approximate string matching in particular. I look at how we may use Levenshtein distance, and the various favours, varieties and optimizations thereof, to measure contribution to a collaborative text, and how we may implement a version of this algorithm specifically tailored to our needs.

The main questions we ask are:

- What does Levenshtein distance define of a contribution in the context of massive online collaboration?
- What are the limitations and implications of defining contribution in this way?
- What else may we learn from analysing contribution?

We base our studies around data from Wikipedia. This study is defined by – and in some ways determined by – the specific context of Wikipedia, but, as we will see, is ultimately enriched by it. Due to its open-source nature, and its size, studies that touch upon Wikipedia cover a very broad range of topics. Many of them are directly related to the topic we concern ourselves with here, and many more may enrich our study tangentially.

2 Previous work

There are three sections here for the three different topics that come to bear on this subject:

- Wikipedia, studies of wikipedia, and the nature of Wikipedia
- The 'edit distance problem', Levenshtein distance
- The various pre-existing studies that apply the latter to the former

2.1 Wikipedia

Wikipedia's pre-eminence as an online resource is self-evident to anyone who has searched the internet for a generic topic. The website is ranked 6th globally in terms of website traffic,[1] and is the highest-ranked reference website by far - most of the sites it shares the top spots with are portals, search engines, shopping mega-sites, and social media websites.[2]

As a nexus of collaboration, a platform for negotiation, the site lends itself many different areas of study. That Wikipedia is a completely open platform, free of copyright restrictions, open database access and a comprehensive API has facilitated much study.

As a focal point of many people, the site, as a network, has been seen as a microcosm of the world wide web on the whole.[WHERE THE BOLLOCKS HAS THAT PAPER GONE I HAD IT] Previous work has seen the site reflecting society too, and has been used in social studies such as Lih's 2004 study of articles immediately after they have been cited in the press[9], and Metyan's 2012 use of the site to predict box-office success [12]. These studies give us a good idea of the community of Wikipedia colaborators.

emblem of the Web2.0 era [11]

used to predict box office success [12] But does it reveal westernness?

Denning: Wikipedia risks: Accuracy, Motives, Uncertain Expertise, Volatility, Coverage, Sources (not many offline sources) [5]

Denning says it cannot attain the status of a true encyclopedia without more formal content-inclusion and expert review procedures[5] this corroborates by findings in [6]?

2.1.1 On Wikipedia

'robust and remarkable growth' [7][14]

Wikipedia, at the last dump, consisted 800G of compressed data [3]

2.1.2 Evaluating Wikipedia articles

identify, analyse

after article mentioned in press [9]

compared by 'experts' to 'equivalent' Encyclopedia Britannica articles [6]

found metrics of article quality through factor analysis [13]

Analysis by conflict - revisions?[7]

WikiTrust. The most 'complete' of the many of the. Exists as firefox plugin (though it doesn't work any more) Culmination of various studies that try to QUOTE [4] and QUOTE CITE. It was assessed as recently as 2011 [10]

2.2 Edit difference

Standard: Levenshtein distance [8]

[1] According to Alexa, an Amazon-owned company. The rankings are based on a combined measure of Unique Visitors and Pageviews, and the data mined from around 25,000 different browser extensions, as well as sites that have installed Alexa's scripts.[1]

2.3 My work in context of these sources

different views emerging topics gaps and inconsistencies

3 Conclusions

PREDIFINED / NOT-PREDEFINED ideas of quality. look for when the the article levels off? And do this by DATE rather than REVISION. We may assume that pageviews are more well-distributed than revisions

summarize major contributions (which do we care about?) evaluate your current position point out any flaw in methodology/research/contradictions are there any gaps in the area which you will cover in your research? How will you integrate sources you have mentioned into your dissertation? Point out any areas for further study

A Appendix A: Python class for scraping Wikipedia article version

A.1 Code

```
import requests
import time
import json
import csv
import wikipedia
from bs4 import BeautifulSoup
from datetime import datetime, timedelta
from decimal import Decimal

WIKI_API_URL = 'http://en.wikipedia.org/w/api.php'
WIKI_USER_AGENT = 'wikipedia (https://github.com/goldsmith/Wikipedia/)'

class WikiRevisionScrape:
    par = {
        'format': 'json',
        'action': 'query',
        'prop': 'revisions'
        }
    head = {
        'User-Agent': WIKI_USER_AGENT
        }
    rand = True
    pagelimit = 1
    historylimit = -1
    rl = False
    rl_minwait = None
    rl_lastcall = None
    pageid = 0
    parentid = 0
    childid = 0

    #atm naively assuming headers, params, titles to be in correct format
    def __init__(self, pagelimit=1, historylimit=-1, _headers=None, _params=None,
        _titles=None):
        if(_params):
            params = _params

        if(_headers):
            self.head = _headers

        if(_titles):
            self.params['titles'] = _titles
            self.rand = False

        self.pagelimit = pagelimit
        self.historylimit = historylimit


    def scrape(self, indexfilename, contentsfilename):
        self._pace()
        index_f = open(indexfilename + ".csv", "ab") #HACK = needs to migrate to postrgres
        contents_f = open(contentsfilename + ".csv", "ab") #HACK = needs to migrate to
            postrgres
        index = csv.writer(index_f)
```

```
        contents = csv.writer(contents_f)
        index.writerow(["PAGEID","REVISION","USER","USERID","TIMSTAMP","SIZE","COMMENT"])
        contents.writerow(["PAGEID","REVISION","CONTENT"])

        for i in range(self.pagelimit):
            if 'rvprop' in self.par:
                del self.par['rvprop']
            if 'revids' in self.par:
                del self.par['revids']
            print "fetching page"
            if(self.rand):
                self.par['titles'] = wikipedia.random() #get random title
            self.childid = self._getlatest()
            r = requests.get(WIKI_API_URL, params=self.par, headers=self.head)
            self._rate()
            del self.par['titles']
            self._tracehist(index, contents)

    def _getlatest(self):
        r = requests.get(WIKI_API_URL, params=self.par, headers=self.head)
        r = r.json()

        #HACK = should grab multiple pages
        for key, value in r['query']['pages'].iteritems():
            self.pageid = key
        #HACK = chould grab multiple revisions (for each pageid)
        self.parentid = self.childid =
            r['query']['pages'][self.pageid]['revisions'][0]['revid']
        return self.childid

    def _tracehist(self, index, contents):
        ##We store revisions we've visited
        ##loops can occur in revision histories
        visited = []
        i = self.historylimit
        j = 0

        self.par['rvprop'] =
            'userid|user|ids|flags|tags|size|comment|contentmodel|timestamp|content'

        while (self.parentid not in visited) and i is not 0 and self.parentid is not 0:
            self.par['revids'] = self.parentid

            self._pace()

            r = requests.get(WIKI_API_URL, params=self.par, headers=self.head)
            r = r.json()

            self._rate()

            visited.append(self.childid)

            #print r

            self.childid = r['query']['pages'][self.pageid]['revisions'][0]['revid']
            self.parentid = r['query']['pages'][self.pageid]['revisions'][0]['parentid']
            user = r['query']['pages'][self.pageid]['revisions'][0]['user']
            userid = r['query']['pages'][self.pageid]['revisions'][0]['userid']
            size = r['query']['pages'][self.pageid]['revisions'][0]['size']
            timestamp = r['query']['pages'][self.pageid]['revisions'][0]['timestamp']
            comment = "" #comments sometimes don't return from old revisions...
```

```
            try:
                comment = r['query']['pages'][self.pageid]['revisions'][0]['comment']
            except:
                comment = ""
            content = r['query']['pages'][self.pageid]['revisions'][0]['*']

            index.writerow([self.pageid, self.childid, user.encode("UTF-8"), userid,
                timestamp, size, comment.encode("UTF-8")])
            contents.writerow([self.pageid, self.childid, content.encode("UTF-8")])

            if(self.historylimit > 0):
                print self.pageid, "fetch", j+1, "of", self.historylimit, ", revid",
                    self.childid, "timestamp", str(timestamp)
                i = i - 1
            else:
                print self.pageid, "fetch", j+1, ", revid", self.childid, "timestamp",
                    str(timestamp)
            j = j + 1
        print "limit reached"

    def _pace(self):
        if self.rl and self.rl_last_call and self.rl_lastcall + self.rl_minwait >
            datetime.now():
            wait_time = (self.rl_lastcall + self.rl_minwait) - datetime.now()
            time.sleep(int(wait_time.total_seconds()))

    def _rate(self):
        if self.rl:
            self.rl_lastcall = datetime.now()
```

A.2 Example output

B Appendix B: Python class for basic, space-naive Levenshtein implementation

B.1 Code

```
class LevDistBasic:
    e = [] #edit operation array
    t = [] #grid array
    x = "" #string1
    y = "" #string2
    m = 0 #length string1
    n = 0 #length string2
    dist = 0 #Levenshtein distance
    ed = [] #the edit operation, calculated in _calculate()

    def __init__(self, _x, _y):
        self.x = _x
        self.y = _y
        self.m = len(_x)
        self.n = len(_y)
        self.t = [[0]*(self.n+1) for _ in xrange(self.m+1)]
        self.e = [[" "]*(self.n+1) for _ in xrange(self.m+1)]
        self.dist = self._calculate()

    def __str__(self):
        return str(self.distance())

    def distance(self):
        return self.dist

    def strings(self):
        return self.x, self.y

    def table(self):
        return self.t

    def operation(self):
        return self.ed

    ##ADD WARNING for long strings / deal with them
    def showtable(self):
        result = ""
        for ch in self.y:
            result = result + ch + " "
        print "      ", result
        for r in range(len(self.t)):
            s = ' '
            if r:
                s = self.x[r-1]
            print s, ' ', self.t[r]

    def showop(self):
        for i, op in enumerate(self.ed):
            l = str(i) + ": "
            if op[0] == 'I':
                l += "insert " + op[-1]
            elif op[0] == 'K':
                l += "keep " + op[-1]
            elif op[0] == 'D':
```

```
            l += "delete " + op[-1]
        elif op[0] == 'S':
            l += "swap " + op[-1][0] + " for " + op[-1][-1]
        else:
            return "FAIL: incorrect operation"
        print l

    def _ed(self):
        i, j = len(self.e)-1, len(self.e[0])-1
        self._ed_recursive(i,j)

    def _ed_recursive(self,i,j):
        if self.e[i][j] == ' ':
            if i == 0 and j > 0:
                self.ed.append(('D', self.y[0]))
            if j == 0 and i > 0:
                self.ed.append(('D', self.x[0]))
            return
        if self.e[i][j] == 'K':
            self._ed_recursive(i-1, j-1)
            self.ed.append((self.e[i][j], self.x[i-1]))
        elif self.e[i][j] == 'S':
            self._ed_recursive(i-1, j-1)
            self.ed.append((self.e[i][j], (self.x[i-1] + ',' + self.y[j-1])))
        elif self.e[i][j] == 'D':
            self._ed_recursive(i-1,j)
            self.ed.append((self.e[i][j], self.x[i-1]))
        else:
            self._ed_recursive(i,j-1)
            self.ed.append((self.e[i][j], self.y[j-1]))

    def _calculate(self):
        for i in xrange(self.m+1):
            self.t[i][0] = i
        for j in xrange(self.n+1):
            self.t[0][j] = j
        j = 1
        while j < self.n+1:
            i = 1
            while i < self.m+1:
                c = (self.x[i-1] != self.y[j-1])
                dl = self.t[i-1][j] + 1
                ins = self.t[i][j-1] + 1
                sbs = self.t[i-1][j-1] + c
                self.t[i][j] = min(ins, dl, sbs)
                if ins < dl and ins < sbs:
                    self.e[i][j] = 'I'
                elif dl <= sbs:
                    self.e[i][j] = 'D'
                else:
                    if(self.x[i-1] != self.y[j-1]):
                        self.e[i][j] = 'S'
                    else:
                        self.e[i][j] = 'K'
                i += 1
            j += 1
        self._ed()
        return self.t[self.m][self.n]
```

B.2 Example output

References

[1] Alexa about us. `http://www.alexa.com/about`. Accessed: 2014-04-31.

[2] Alexa the top 500 sites on the web. `http://www.alexa.com/topsites`. Accessed: 2014-04-31 (updated daily).

[3] Wikimedia enwiki dump progress on 20140502. `http://dumps.wikimedia.org/enwiki/20140502/`. Accessed: 2014-04-29.

[4] B. Thomas Adler and Luca de Alfaro. A Content-driven Reputation System for the Wikipedia. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages 261–270, New York, NY, USA, 2007. ACM.

[5] Peter Denning, Jim Horning, David Parnas, and Lauren Weinstein. Wikipedia risks. *Commun. ACM*, 48(12):152–152, December 2005.

[6] Jim Giles. Internet encyclopaedias go head to head, 2005.

[7] Aniket Kittur, Bongwon Suh, Bryan A. Pendleton, and Ed H. Chi. He says, she says: Conflict and coordination in wikipedia. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '07, pages 453–462, New York, NY, USA, 2007. ACM.

[8] VI Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10:707, 1966.

[9] Andrew Lih. Wikipedia as participatory journalism: reliable sources? metrics for evaluating collaborative media as a news resource. In *In Proceedings of the 5th International Symposium on Online Journalism*, pages 16–17, 2004.

[10] Teun Lucassen and Jan Schraagen. Evaluating wikitrust: A trust support tool for wikipedia. *First Monday*, 16(5), 2011.

[11] Mostafa Mesgari, Chitu Okoli, Mohamad Mehdi, Finn Årup Nielsen, and Arto Lanamäki. "the sum of all human knowledge": A systematic review of scholarly research on the content of wikipedia. *Journal of the American Society for Information Science and Technology*, April 2014. This is a postprint of an article accepted for publication in Journal of the American Society for Information Science and Technology copyright © 2014 (American Society for Information Science and Technology).

[12] Mrton Mestyn, Taha Yasseri, and Jnos Kertsz. Early prediction of movie box office success based on wikipedia activity big data. *CoRR*, abs/1211.0970, 2012.

[13] B. Stvilia, M.B. Twidale, L.C. Smith, and L. Gasser. Assessing information quality of a community-based encyclopedia. In *Proceedings of the International Conference on Information Quality*, pages 442–454, 2005.

[14] J Voss. Measuring wikipedia. *Proceedings 10th International Conference of the ...*, January 2005.