Measuring Success of Open Source Projects Using Web Search Engines

Dawid Weiss
Poznań University of Technology
Poznań, Poland
dawid.weiss@cs.put.poznan.pl

Abstract— **What makes an open source project successful? In this paper we show that the traditional factors of success of open source projects, such as the number of downloads, deployments or community activity are inconvenient to collect or insufficient. We then correlate success of an open source project with its *popularity* on the Web. We show several ideas of how such popularity could be measured using Web search engines and provide experimental results from quantitative analysis of the measures we introduce on representative large samples of open source projects from SourceForge.**

I. Introduction and motivation for the work

Success of a commercial software project is usually measured by the profit it brings to its creators or copyright owners. This definition, does not translate directly to majority of the open source world, where software projects are developed by enthusiasts and volunteers for free. Even if, for certain larger projects, an indirect income gained from consultancy or packaged distributions of the software satisfies the needs of its maintainers (developers), it is nowhere near a general rule.

A question comes to mind: what makes an open source project successful? The following factors seem to contribute to the overall success:

- **Number of deployments/ uses.** Open source projects that are actively used provide satisfaction and sense of pride for their developers. "Use" of an open source project can be understood as direct use by end-users, or indirect use as a component within another program.
- **Number of active committers and e-mailing list traffic.** For projects that have not yet reached the level of stability to be used in real applications, the number of active committers can be an indication of interest of the OS community in the project. This, for example, is the case with projects like Nutch (`http://www.nutch.org`), which is not yet mature, but has raised a significant flurry of interest.

In practice, measuring the above factors may prove to be a difficult task. The number of deployments of a project is often unknown and impossible to determine, because open source projects expose their artifacts for uncontrolled download and rarely keep a list of real use cases. Besides, number of downloads of a project may not translate to its deployments. Number of committers and e-mailing list traffic is also not always a good indication of a project's popularity. Number of committers (or rather: commits) usually reflects the dynamics of project's growth, but certain projects are extremely popular and their development is rather stale than dynamic — take JUnit (`http://www.junit.org`) as an example. The latest version of JUnit is dated September 3, 2002, which is over two years ago.

Straightforward indicators of open source projects' success may be misleading, inaccurate and difficult to collect. The motivation for this work is to fill this gap by providing an alternative, easier way of establishing the success ratio of a project.

Let us define *popularity* of a project as being proportional to the number of Web pages that mention this project somehow. This approach is clearly inspired by how we perceive and measure the influence (significance, success) of research papers — the more citations a given paper has, the more influential it appears to be to the research community. Considering the ways people nowadays exchange information — through mailing lists, blogs, newsgroups and other Web-enabled environments — we believe this "Web popularity" may serve as a good estimation of a project's success.

The remaining part of this paper provides ways of narrowing the concept of popularity to concrete, measurable properties and a demonstration of how these properties can be calculated by mining knowledge already present in existing Web search engines.

II. Web Search Engines and popularity

Search engines nowadays keep the most up-to-date state of a large part of the Internet's resources.[1] Mining the information stored in a Web Search engine for extracting a project's popularity has the following advantages:

- **Objective (neutral) point of view**. Web crawlers are conceptually simple automata that traverse the Web pretty much at random.[2] The number of references to any open source project will reflect its actual importance on the Web.
- **Relevance**. Most open source licenses *require* that a credit is given in case of embedding, or reusing parts of the source code. This credit usually takes the form of a link pointing back to the Web page of the original project, or its name (or the name of its authors). Every such link thus constitutes a use case, or at least a strong expression of interest in the project.
- **Up-to-date**. Search engines keep more or less up to date with the state of pages in the internet, so by using the information they provide we can assure the accuracy of data on an almost daily basis. An additional bonus is that several search engines provide

[1] Google's index as of December 2004 is approximated at a little bit over 8 billion pages

[2] This is a major simplification, but with no negative effect for further considerations here.

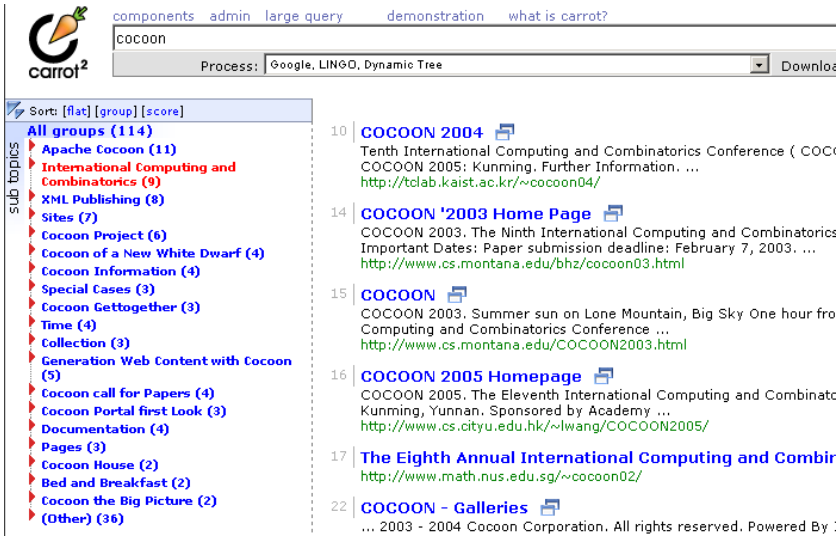

Fig. 1. Different topics for a simple query *Cocoon* clustered dynamically in a clustering engine Carrot^2 (`http://carrot2.sourceforge.net`).

the capability of searching within the desired time spans. This should allow us to perform retrospective analysis of a project's popularity over time.

- **A data source that is easy to collect**. Querying search engines requires substantially less effort compared to the same work performed by humans (in the form of surveys, for example). Most of the work can also be performed automatically using simple parsers or applications interfaces provided by a search engine.

III. Methods of measuring projects popularity

Let us define notational convention which we will use further in this text:

p — the project which we analyze,
$M_x(p)$ — popularity of project p counted with method x,
N_p — name of the project p,
$\mathrm{hp}(p)$ — home page of project p.

A. Method 1 (M_1): simple word counting

The simplest idea of measuring popularity is to **count the number of pages that contain all of the words in project's p name**.

$$M_1(p) = \sum (\text{pages containing words in } N_p). \quad (1)$$

1) Advantages:

- No direct links from the pages referencing the project are needed to count it (it is enough that the project's name is mentioned).
- Works with all search engines — there is a possibility of cross-verification of results.

2) Disadvantages: Simple as it is, the method has a major drawback: it counts all pages that contain *other meanings* of words building up the name of the project. For example, a word *Cocoon* is mentioned on over 3 million pages, but only a fraction of them is indeed related to the open source project Cocoon. Clustering the top portion of Web search result by topic reveals other meanings the query was related to (see Figure 1).

3) Discussion of estimation quality: This method is an *over-estimation* of the number of pages that truly reference the project. We believe that it sets the upper bound of the popularity of a project.

B. Method 2 (M_2): license-reference counting

Many open source licenses require that the name of the project, or some other specific phrase, is explicitly mentioned whenever the project or its subcomponents are reused. An example demonstrating this practice is shown in Figure 2.

Assuming that most documentation is nowadays published on-line we can deduce that the **number of pages containing an *exact phrase* required by the license is an estimation of the number of deployments of a project** and therefore its popularity.

$$M_2(p) = \sum \begin{pmatrix} \text{pages containing a phrase required} \\ \text{by the project's license} \end{pmatrix}. \quad (2)$$

1) Advantages: Very accurate — it is highly unlikely that the same exact phrase required by the license is used in any other meanings than to denote its deployment.

2) Disadvantages:

- Not all projects require a reference to its name. In fact, as observed on many open source licenses, they are copied verbatim from their template, often with untouched, blank form fields. This estimation method is unavailable for projects of this kind.
- In certain cases, the license refers to the software house rather then the project (see Figure 2). For these projects, this method will be unusable.
- Open source projects are commonly used *without* fulfilling all license terms. In practice, developers seem to look only at the "compatibility" between various kinds of licenses, grouping them into GPL-compatible, BSD-compatible etc. Additional sections of the license file are often neglected.

```
project-acknowledgement
In addition, we request that you include in the end-user documentation
provided with the redistribution and/or in the software itself an
acknowledgement equivalent to the following:
"This product includes software developed by the Egothor Project.
http://egothor.sf.net/"
```

```
os-vendor-acknowledgement
 * 3. The end-user documentation included with the redistribution, if
 *    any, must include the following acknowlegement:
 *       "This product includes software developed by the
 *        Apache Software Foundation (http://www.apache.org/)."
 *    Alternately, this acknowlegement may appear in the software itself,
 *    if and wherever such third-party acknowlegements normally appear.
```

Fig. 2. Open source licenses: Egothor (above) requires specific acknowledgement, Apache projects (below) only demand that the ASF is referenced properly.

3) Discussion of estimation quality: If only applicable, the estimation of the number of deployments given by M_2 is perhaps the best possible using automated methods. Let us point out that the number of references to the phrase required by the project's license is always an underestimation of the total number of deployments. It can be therefore considered a lower bound for the project's popularity.

C. Method 3 (M_3): backlinks counting

So far we only utilized information about the presence of a certain keyword, or a phrase in the Web resources. The third proposal takes into account the link structure present between Web pages. That this structure is a viable source of information is obvious — the same kind of information has long been used to *rank the importance* of pages matching a query [2], [1].

There are a few options to derive the popularity of a project from the link structure:

- We could acquire an already calculated importance score for the home page of a project (PageRank in case of Google). Unfortunately, this kind of information is not provided by search engines.
- Count the number of *backlinks* of the project's home page. Backlinks are hypertextual references pointing from some location on the Web to the page in question.

The number of backlinks is clearly correlated with the popularity of a project, even though it does not distinguish between deployments and other types of citations. Most modern search engines provide ways of querying for backlinks of a given page. A potential problem is that the number of backlinks will include all pages from the project that internally link to its home page. We introduce two measures. The first one is a plain backlink count. In the second measure we substract the total number of pages found in the project's domain from the total backlink count. This adjustment should deal with the pessimistic scenario of every page linking back to the main one.

$$M_3(p) = \sum(\text{pages that link to hp}(p)) - \sum(\text{pages in the domain of hp}(p)), \quad (3)$$

$$M_4(p) = \sum(\text{pages that link to hp}(p)). \quad (4)$$

1) Advantages:

- High relevance — a backlink to the main page of the project is a confident indication of external interest in the project. It is impossible to guess the nature of this reference, but we measure the popularity in general, so this is not important.
- Broad applicability — the method applies to all kinds of projects. The only requirement is that the analyzed project must have a Web site.

2) Disadvantages: The method counts all backlinks and it should rather count external *domains* those backlinks originate from. One can imagine a situation when a full mirror of JavaDoc documentation of the project is pointing back to its home page. This would give an unjustified boost to the popularity score. At the moment we have no means of detecting this kind of situations.

3) Discussion of estimation quality: In spite of its disadvantages, we think backlink counting is a relevant method of calculating popularity of a project (see Section V-B for discussion of potential abuse of this measure). It is more conservative compared to simple word counting and less strict than license-reference counting. We believe the following invariant should always hold between the presented measures: $M_1 \gg M_4 > M_3 > M_2$.

IV. Experimental evaluation

A. Goal of the experiment

We designed a practical experiment to verify our intuition and prove the concept feasible. Specifically, we tried to provide answers to the following questions:

- Is it always possible to calculate all the introduced measures? How do they relate to each other?
- Which measure is most useful as a predictor of popularity of a project?
- Are all our measures consistent with some other measure of open source projects' success?

B. Data for the experiment

In the experiment we had to analyze popularity of real, existing open source projects. As part of the work on this paper, we crawled and extracted [3] actual project names, addresses of home pages and numerous other features from the largest open source hosting facility in the world — SourceForge. We chose SourceForge hoping to have a good (large and diverse) population of projects to choose from. As it turned out, we could also utilize the *activity factor* SF assigns to its projects as a reference measure of success we could compare our method against.

From the entire collection of over 80 thousand projects, we assembled three distinct test sets of 200 randomly chosen samples from larger groups of projects satisfying the following conditions:

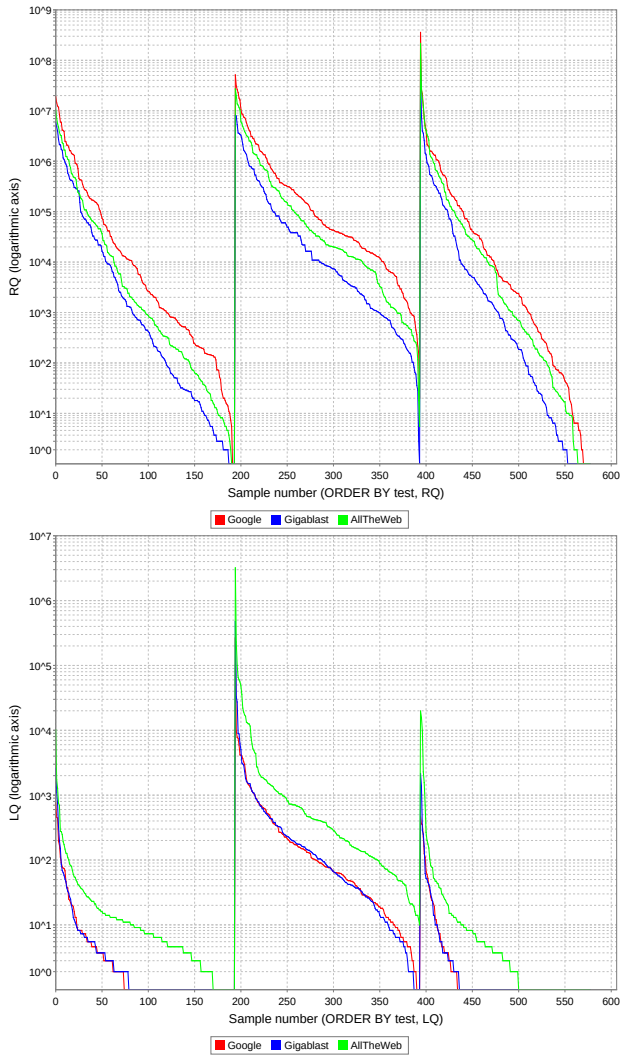

Fig. 3. Distribution of values of RQ (top) and LQ (bottom). Projects were sorted by test first (from the left on each chart: FAILED, GOOD, RANDOM and then by RQ or LQ. Note logarithmic Y axis on the charts.

1) **Test set name: GOOD** Each project had to have at least 15 bugs, more than 5 patches and SF activity indicator greater than zero. *Rationale*: projects with high number of bugs and patches were, at least at some point, developed actively. SF activity indicator is a measure of weekly activity used at SourceForge, we assume it can be used to mark popular, active projects;
2) **Test set name: FAILED** Each project had to have at least two or three bugs, registration date earlier than 2004, SF activity indicator not exceeding 5. *Rationale*: small number of bugs in a long time (more than one year) suggests that the project was unsuccessful at attracting user and developer community;
3) **Test set name: RANDOM** All project were initially included in this group.

We also assumed that recently registered projects have not had the time to become a popular subject in the Internet and only include projects registered prior to year 2004.

The test sets were manually inspected and cleaned from projects unrelated to computer science. Project names extracted automatically from SourceForge were often too verbose. We manually cleaned and simplified names to those actually used to refer to the project. In cases when the name was still too long and it had a sensible-looking abbreviation, this abbreviation was used (Java Weather Library → jweather, ANSI Common Lisp → CLISP). At this stage we also checked all home pages by following browser redirections or finding the home page of a project manually. Eventually, after all the pruning, the test data sets had the following sizes: FAILED: 194, GOOD: 200, RANDOM: 184.

Intuitively, by comparing values of measures calculated for these test sets, we should be able to notice a statistically significant difference between good, successful projects and the failed projects. From the random sample we should be able to find out about average values and properties of our defined measures.

C. Search engines

We selected three search engines as a data source for the experiment: Google (`www.google.com`), AllTheWeb (`www.alltheweb.com`) and Gigablast (`www.gigablast.com`). These three search engines have independent crawlers, indices and ranking formulas, which allowed us to cross-examine results.

For each project in the test data sets we queried[3] each search engine and collected the following parameters:

- RQ (*regular query*), number of pages containing all terms from the project's name,
- PQ (*phrase query*), number of pages containing an exact copy of the project's name as an ordered phrase,
- LQ (*link query*), number of pages with hyperlinks to the home page of the project,
- SQ (*site query*), number of pages on the home site.

Please note that our measures can be expressed in relation to these data as: $M_1 = \mathrm{RQ}$, $M_3 = \mathrm{LQ} - \mathrm{SQ}$, $M_4 = \mathrm{LQ}$. Obviously, we rely on the search engines to provide accurate numbers, which is not entirely true (in huge, distributed search engines, returned number of matching results is usually an approximation). We use three independent search engines to ensure the results are consistent among them.

V. Results and Discussion

A. Results of the experiment

The most valuable observations and conclusions from the experiment are provided below.

Data in Table I shows that the standard deviation for RQ is enormous. This is caused by projects with only common words in their names that boost the RQ, just as we expected. An extreme example: project "Show it!" had an $\mathrm{RQ} = 357000000$ and $\mathrm{LQ} = 0$! Using phrases does not help much with such projects and there were quite a few of them in our test data sets. This effectively renders measure M_1 unusable for projects with common names. LQ seems to be a much better indicator of real project's influence. It has lower standard deviation and sensible averages. This observation is confirmed by looking at Figure 3. Distribution of RQ's values for GOOD data set is on average higher compared to the other two data

[3]Description of the technical implementation of this process is quite complex and involves advanced tricks with query syntax. We omit the details here.

Search engine	Test set	RQ		PQ		LQ		SQ	
		$E(X)$	$D(X)$	$E(X)$	$D(X)$	$E(X)$	$D(X)$	$E(X)$	$D(X)$
AllTheWeb	failed	300740	1141744	240484	1105639	95	747	133	1377
AllTheWeb	good	842439	2995823	656591	2656649	20257	226251	530	3588
AllTheWeb	random	1711943	16074159	1469857	15731561	283	1992	35	275
Gigablast	failed	186569	703013	93854	426040	30	211	53	608
Gigablast	good	303173	1035229	217208	839435	3080	33612	79	509
Gigablast	random	553660	5346102	72013	381947	25	190	8	61
Google	failed	617705	2293558	505933	2210440	15	73	354	3148
Google	good	1468158	5276370	1414917	5269737	1797	18771	2090	11801
Google	random	2511927	26343093	385898	1970813	11	54	198	2011

TABLE I

AVERAGES AND STANDARD DEVIATIONS OF RESULTS RETURNED BY SEARCH ENGINES.

sets, but still they overlap much. With LQ, this situation is much clearer — projects from GOOD have a much higher LQ. The difference between RANDOM and FAILED test sets is nicely highlighted too: RANDOM was a uniform sample from all SourceForge projects and it comes out *even worse* then those projects selected to FAILED (those had at least one bug and 83% projects on SourceForge has no recorded bugs [3]).

Data in Table I suggests that on average projects from GOOD test set had significantly larger websites compared to projects in RANDOM or FAILED (SQ parameter). The postulated relationship between M_1 and M_3 ($M_1 \gg M_3$) is confirmed in the experiment — only 26 projects out of 1734 had $M_1 < M_3$.

Reliability of results returned by search engines can be assessed by comparing numbers of results for identical queries returned from different search engines (Figure 3). The numbers returned from all of the search engines are highly correlated and consistent. Number of results for regular queries (RQ) reflects the size of the index: Google leads before AllTheWeb and Gigablast. A very intriguing thing happens, however, with the number of link queries (LQ) — AllTheWeb returns significantly *more* results than Google. This is against common sense, so we suppose that the two engines calculate this figure in a different way or have a completely different organization of their link database. We selected AllTheWeb as the reference point for further analysis.

In the second part of our analysis, we tried to establish correlation between our measures of popularity and several features of the projects in the test sets: number of bugs, patches and feature requests. To our surprise, no elements of such correlation could be found.

Next, we tried to compare values of our measures to the SourceForge's activity factor. There are a few problems with such comparison. First, activity factor is calculated on a basis of weekly history of projects' activity,[4] so the ranking it imposes often slightly changes. The activity factor is also not weighted — it is merely a reflection of a ranking made using another formula, so any kind of regression would be inappropriate. As a last resort we employed rank-order coefficients: Spearman's ρ and Kendall's τ. We ranked (sorted) all projects in the test sets using the activity factor and repeated the procedure for

[4]An exact formula is given at: `http://sourceforge.net/docman/display_doc.php?docid=14040&group_id=1`

	All data		**GOOD set only**	
	Kendall's τ	Spearman's ρ	Kendall's τ	Spearman's ρ
M_1	0.23	0.41	0.16	0.24
M_3	0.36	**0.69**	0.27	0.39
M_4	0.43	**0.78**	0.27	0.40

TABLE II

RANK-ORDER COEFFICIENTS FOR PROJECTS SORTED ACCORDING TO SOURCEFORGE'S ACTIVITY FACTOR TO ORDER IMPOSED BY M_1, M_3 AND M_4.

M_1, M_3 and M_4. The results in Table II demonstrate that in all cases there was a positive rank correlation between SourceForge's activity and our measures. This correlation was strongest for M_4.

Finally, we manually investigated the topmost results sorted according to the three measures M_1, M_3 and M_4. Just as expected from the previous observations, M_1's topmost results include many projects that were failures (see Table III). Their high score is caused by common terms in their names rather then their real popularity. Ranking created according to M_4 looks good at first glance — a few projects from the random data set were also included, but it was not an error (we checked manually: in spite of their zero SourceForge activity, these were active, popular projects). The adjustment of site size used in M_3 does not affect the top of the ranking, but results in many popular projects (like EXWIDGETS or GNUPLOT) pushed back from the top to the bottom of the list, which we consider the measure's weakness. This effect usually occurred when a project had a large on-line documentation and a relatively small number of backlinks.

B. Intentional manipulation threat

There is a theoretical possibility of creating an intentional structure of links and Web pages in order to boost a given project's measure of popularity. One example technique to achieve this would be a *link farm* — cross-referenced set of Web sites where links are intentionally set up to boost one site's score. These practices are already in use, usually to promote sexual content and spam. Our measures of popularity partially rely on the fact that most search engines strongly discourage link farming (to the point of removal from search engine's index) and penalize sites involved with such activities.

Measure M_4

Project	SF Activity	Test	M_4
WordPress	88,56	good	3200000
MoinMoin	90,41	good	230000
NAnt	95,99	good	100000
ht://Dig	91,66	good	65600
PhpGedView	99,52	good	60100
FAQ-O-Matic	78,83	good	55700
Gaim	99,99	good	48300
JBoss	99,62	good	35800
7-Zip	99,91	good	22000
lmbench	62,68	random	19800
w3m	89,07	good	19100
PHP SysInfo	97,90	good	17300
POPFile	99,98	good	16500
JabberdPlus	0,00	random	15800
yappa-ng	96,92	good	13300
MinGW	99,90	good	13000
HLstats	92,88	good	12600
User-mode Linux kernel port	92,66	good	12500
htmlArea	97,12	good	11600
tcpslice	0,00	failed	10200
NetHack	97,93	random	9750
DOSBox	98,05	good	7790
Bochs	99,77	good	6170
netatalk	88,18	good	5170
SpamBayes	98,80	good	4880
CGI:IRC	92,16	good	4500
syn	94,51	good	4380
xbmc	99,81	good	2730
CDex	99,49	good	2700
Saxon	99,36	good	2290

Measure M_1

Project	SF Activity	Test	M_1
Show It!	0,00	random	216000000
Web Development Tools	0,00	random	28700000
WASTE	99,45	good	27000000
7-Zip	99,91	good	20000000
Lookup	0,00	random	17500000
SIM	99,60	good	13700000
Collective	99,05	good	12900000
Aviation++	72,65	random	12800000
modeling	45,51	good	11600000
Kate	0,00	failed	10400000
Outreach	72,47	good	10400000
fence	0,00	random	7480000
Silva	0,00	failed	7050000

Measure M_3

Project	SF Activity	Test	M_3
WordPress	88,56	good	3158900
MoinMoin	90,41	good	228290
NAnt	95,99	good	99775
PhpGedView	99,52	good	59839
FAQ-O-Matic	78,83	good	53010
Gaim	99,99	good	47981
ht://Dig	91,66	good	37300
JBoss	99,62	good	34190
7-Zip	99,91	good	21930
lmbench	62,68	random	19800
w3m	89,07	good	19094
PHP SysInfo	97,90	good	17293
POPFile	99,98	good	16454
yappa-ng	96,92	good	13300

TABLE III

LIST OF TOPMOST PROJECTS WHEN SORTED ACCORDING TO M_1, M_3 AND M_4 RESPECTIVELY.

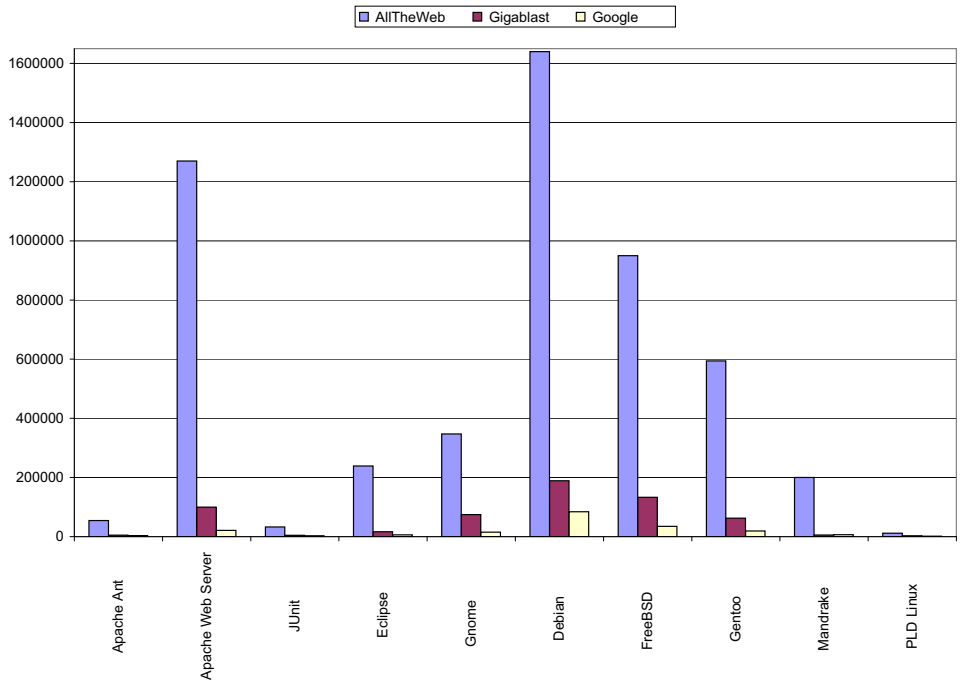

Fig. 4. Value of M_4 measure for several major open source projects (big and small). Link count for Google is considerable smaller, but consistent with AllTheWeb (as already pointed out in Section V-A).

C. Calculation of M_4 and M_1 for widely known projects

As a final, maybe a bit entertaining, element of this paper we allowed ourselves to measure the popularity of several widely known open source projects, including a few major Linux distributions. The results are presented in Figure 4. It is quite amazing to see the "difference" between major Linux distributions and ANT and JUnit — probably among the most widely known Java open source utilities. It also puts into perspective the actual impact of certain Linux distributions (here: PLD Linux) compared to the major ones.

VI. Summary and conclusions

The problem of defining and measuring success of open source projects is definitely not trivial. In this work we suggest a relationship between the success of a project and its popularity on the Web. We introduce three methods in which this popularity could be actually measured by mining knowledge existing in Web search engines and design an experiment that explicitly shows how this can be done. The experiment allows us to make the following conclusions about the presented measures:

- measure M_1 is not accurate enough due to problems with common terms in names of projects (but indeed does seem to be the upper bound for popularity of a project),
- measure M_2 is not practically applicable for most projects because in most cases licenses are merely generic templates and lack project-specific fragments,
- two measures seem to demonstrate relevant and sensible properties: M_3 and M_4. They also exhibit a significant rank order correlation to SourceForge's activity factor. We mention this fact, but would rather avoid making conclusions because the activity factor is an oddly calculated value and in our opinion is not a very good indication of success of open source projects.

Future work on the subject could refine the penalty component in M3 — currently it seems to be doing more harm than good. Additional work is also needed to make the measure less sensitive to apparently quasi-exponential distribution of values of the link query count component (LQ).

Acknowledgement. The author of this work would like to thank Maciej Hapke, Andrzej Jaszkiewicz and Krzysztof Kowalczykiewicz for fruitful discussions. This research has been funded by the European Commission via FP6 Co-ordinated Action Project 004337 in priority IST-2002-2.3.2.3 **CALIBRE** (`http://www.calibre.ie`).

References

[1] J. M. Kleinberg, "Authoritative sources in a hyperlinked environment," *Journal of the ACM*, vol. 46, no. 5, pp. 604–632, 1999. [Online]. Available: citeseer.ist.psu.edu/kleinberg99authoritative.html

[2] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web," Stanford Digital Library Technologies Project, Tech. Rep., 1998. [Online]. Available: citeseer.ist.psu.edu/page98pagerank.html

[3] D. Weiss, "A large crawl and quantitative analysis of open source projects hosted on sourceforge," Institute of Computing Science, Poznań University of Technology, Poland," Research Report RA-001/05, 2005.