CVExplorer: Identifying Candidate Developers by Mining and Exploring Their Open Source Contributions

Gillian J. Greene and Bernd Fischer
CAIR, CSIR Meraka, Computer Science Division, University of Stellenbosch, South Africa
ggreene@cs.sun.ac.za, bfischer@cs.sun.ac.za

ABSTRACT

Open source code contributions contain a large amount of technical skill information about developers, which can help to identify suitable candidates for a particular development job and therefore impact the success of a development team. We develop CVExplorer as a tool to extract, visualize, and explore relevant technical skills data from GitHub, such as languages and libraries used. It allows non-technical users to filter and identify developers according to technical skills demonstrated across all of their open source contributions, in order to support more accurate candidate identification. We demonstrate the usefulness of CVExplorer by using it to recommend candidates for open positions in two companies. A video demonstration of the tool is available at https://youtu.be/xRxK-wa7PME

CCS Concepts

•**Software and its engineering** → *Open source model; Programming teams;*

Keywords

Identifying candidate developers, Developer skills identification, Mining software repositories

1. INTRODUCTION

Poor hiring decisions are a well-known risk factor to the success of a software project [25]. DeMarco and Lister observe that work quality is more dependent on the involved team member than on how the work is done [12]. In industries such as Software Engineering with many open positions and not as many qualified candidates, identifying (or *sourcing*) candidates with the right combination of skills is crucial to the success of a software project because these candidates may not actively be applying for jobs themselves [22]. Developers' open source contributions have been suggested as a mechanism to determine suitability for a particular job [21, 19]. Open-source contributions allow us to infer information about the developer's interests (e.g., what they program in their free time) and technical skill sets (e.g., programming languages they are using) which can both be used to improve the quality of candidate sourcing. There are even claims that "GitHub is a developer's new CV" [13, 26].

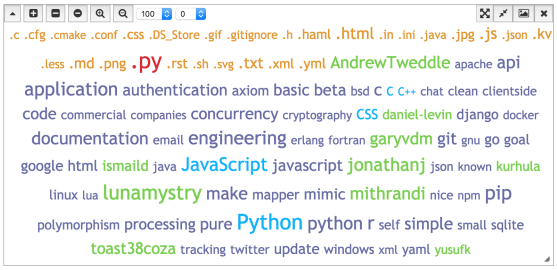

Figure 1: Skills (purple), filetypes (orange), and developers (green) making changes to Python files (selected, red) for Cape Town developers on GitHub.

CVExplorer (available at http://recruit.conceptcloud.org) presents technical skills extracted from GitHub data in an intuitive, interactive tag cloud interface. Our approach is novel in that it allows users to explore skill sets of a large number of developers *simultaneously* and narrow the developer pool to only those possessing relevant skills. However, our interface can also function as a skill aggregator: once a candidate has been identified, the individual candidate's full set of skills are displayed, which can be used to personalize contact with the candidate.

Business-oriented social networks such as LinkedIn [5], where profiles are self-authored, are commonly used for online recruitment. However, individuals may exaggerate their skill-set or, conversely, omit some skills in a self-authored CV. Therefore, diversifying the candidate search to developer-oriented sites, such as GitHub is beneficial to the recruitment process. Additionally, as noted by Capiluppi et al. [11], a skills-based identification could provide equal opportunities to skilled developers with no formal qualifications.

There are already sites (such as Coderwall [1], Masterbranch [6], Open Hub [7], Stack Overflow Careers [8] or TalentBin [9]) which aggregate developers' skills across a site or on various platforms, but these typically assist in evaluating individual candidates that have already been sourced through other means or have actively applied for a position. Therefore, CVExplorer serves a *different purpose* to profile aggregation which only allows evaluation of the skills of one

ASE'16, September 3–7, 2016, Singapore, Singapore
© 2016 ACM. 978-1-4503-3845-5/16/09...$15.00
http://dx.doi.org/10.1145/2970276.2970285

developer at a time and does not allow multiple developers to be filtered to those that have suitable skill sets.

CVExplorer supports *identification of suitable technical candidates* from a large pool of developers as opposed to aggregation of individual developers' skills across platforms. We mine and aggregate *technical skills* across all of a developer's commits and present these in an interactive tag cloud (see Figure 1) to facilitate skills browsing and identification of suitable developers. Users can filter the developer pool by selecting a combination of tags comprising relevant skills.

GitHub contributions are already compared to a traditional CV as serving a summary of the developer's skills and therefore, we stick to the terminology of a "CV". Note that this CV is not in the traditional form of activities in chronological order, but is similar to a portfolio of the developers' skills. However, code contributions constitute the actual portfolio and so we extract the portfolio's meta-information.

To determine the effectiveness of CVExplorer we used it to recommend candidates for positions at two companies; we received positive feedback on the candidates' suitability for interviews.

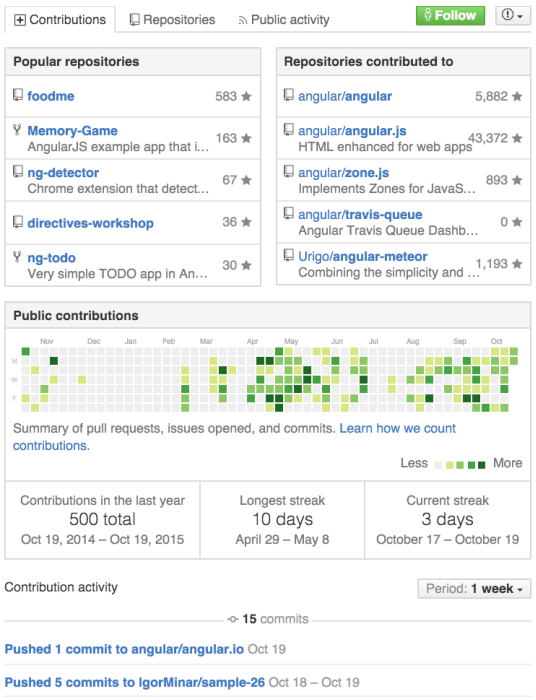

Figure 2: GitHub profile example showing project contributions (top) and commit activity (bottom)

2. APPROACH

2.1 Evaluating Skills via GitHub's Interface

A developer's GitHub profile (Figure 2) indicates the number of followers they have, their repositories and public activity (e.g., following another developer or starring a project). GitHub provides an activity chart indicating how many commits a user has been making over the previous year. This information is prominent on a user's profile but information about the *content* of the changes a developer has made is significantly more difficult to obtain. GitHub profiles link all projects that the user has contributed to or forked but do not directly provide additional information about the contribution. In this format the profile can only be considered as a signal [21] for the developer's interests or expertise. Inexperienced GitHub users might not be aware that the information on the user's profile needs to be verified by examining the user's commits to identify their contributions.

There are a number of issues that make it difficult to use GitHub profiles directly to assess a developer's skills. In order to get an accurate representation of a developer's skills via the GitHub interface, a user needs to select all projects that the developer has contributed to and verify the contributions. Programming languages for all projects then need to be manually aggregated to get an overview of the developer's skills set. However, even a developer contributing to a Ruby project may not have changed any Ruby code in his contributions and so the developer's skills need to be manually verified on an individual commit level.

On GitHub it is also non-trivial to find developers using a particular programming language. GitHub provides an advanced search feature that allows users to find developers that use a particular programming language, however this does not indicate how much experience a developer has in a language and what other languages they are experienced in.

2.2 Mining Developer Contributions

We mined developer profiles according to the location provided on their GitHub profiles so that we could aggregate the profiles of developers in a specific area where there is an open position. We used the GitHub API [4] (more specifically the Eclipse EGit GitHub reader [3]) in order to obtain the most up-to-date information for 1000 developers in a specific location and to extract a list of each developer's repositories. We extracted the first 1000 developers that were available from the API (which limits search results to 1000 responses). We then automatically cloned all repositories to which those 1000 developers contributed and identified those developers' individual commits. We extracted from each commit, the commit message and the changed files directly from the Git repositories using the Eclipse JGit library [2].

The author's commit name which identifies them in the commit logs, is not necessarily their GitHub username. We therefore automatically match the commit name to either the user's GitHub username or the name listed on their profile, so that if the author does not use their GitHub username in the commit logs we could still identify their commits.

For each repository we also extracted the programming language as it was listed on GitHub (using the GitHub API) and the project's ReadMe file. ReadMe files contain details about the project such as the type of technologies it makes use of, installation instructions and the project's dependencies. Therefore, the ReadMe file can provide an overview of the project and indicate what skills a developer contributing to the project is likely to possess.

We associated a project (and its extracted information) to a developer only if the developer had made more than five commits to the project, in order to ensure that the developer was not only a one-off contributor.

2.3 Aggregating Developer Contributions

We aggregate all of a developer's commits to all of their repositories on GitHub and identify in particular the file types that a developer has changed. We also process the

(a)

(b)

Figure 3: (a) Developers changing Java projects sized according to the number of commits (b) Developers changing .java files in Java projects

ReadMe files of all a developer's repositories in order to extract skills that a contributor to the project is likely to posses (e.g., experience with a web framework or database).

We constructed a white list of skills (available at http://conceptcloud.org/hiring_from_github) from the Wikipedia [10] lists of Programming Languages, Web Frameworks, Platform Independent GUI Libraries, Ajax Frameworks, and Object-Relational Mapping Frameworks as well as the ACM Classification specifications. This white list allows us to extract as skills both the technologies used as well as high level phrases such as "Machine Learning" from the ReadMe files of projects that a developer has contributed to. We run each project's ReadMe file through the white list in order to generate a bag of words containing the extracted skills that appear on our white list for each project.

We also ran the commit message through our skills white list in order to remove words not directly referencing a technology or concept (e.g.,"added"). We removed punctuation and spaces when matching skills on our white list to text in the ReadMe files in order to ensure that we pick up all possible matches. For phrases in the white list containing multiple words, we also matched the text in the ReadMe file against all separate words in the phrase to account for cases when the phrase did not appear in the ReadMe file in full. We then aggregated the identified skills for each commit and presented the information in a browsable format.

Note that we include the changed file types for each commit because these provide a stronger indication for a developer's experience in a particular programming language than considering only the language of the project the developer has been contributing to, because the developer might only have been making changes to documentation. Hence,

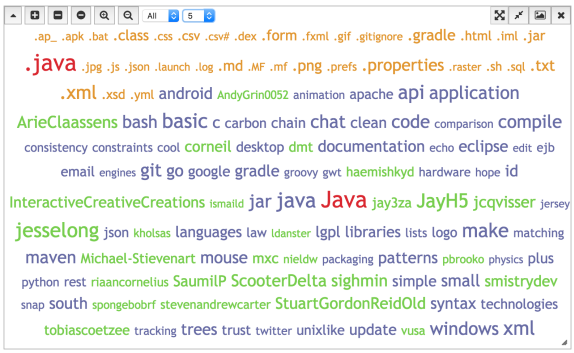

Figure 4: Largest tags from commits in which `.java` files have been changed in Java projects.

the user can verify, for example, that a developer has been changing Java files before the developer is considered to have Java as a skill. Figure 3(a) shows the tag cloud of developers working on Java projects (where the language on GitHub is indicated as Java); however, when we add the tag for `.java` files in Figure 3(b) we see that the number of developers is reduced significantly, indicating that not all developers that have contributed to a Java project have actually always been changing `.java` files (see for example evanx or stigglePogletX, whose tags disappear or become much smaller).

2.4 Presenting Developer Contributions in a Browsable Format

We visualize developers' contributions and skills in an *interactive* tag cloud. Tag clouds [20] are a simple visualization method for textual data where the importance of each tag (typically its frequency) is reflected in its size. When users do not yet know what information they are looking for (e.g., when they are looking for job candidates with an unknown combination of skills) or have no previous knowledge of the information they are browsing, the user's task becomes one of exploratory search [27]. Tag clouds support exploratory search tasks and have been found to be effective when the information discovery task is wide [23].

Tags in our tag clouds are colored according to the type of information that they represent. Figure 4 shows a tag cloud of the largest tags from our sample of developers in Cape Town that have changed `.java` files in Java projects. The developer tags are green. We see that some skills associated with changes to Java files are "Android", "Maven" or "Apache". We also see that developers changing `.java` files have also changed HTML, Gradle and XML files. Selecting tag "Android" in the cloud would further restrict the cloud to showing only developers that have changed Java files in a project that mentions Android in its ReadMe file, and other skills exhibited in the changes to these projects. While both the extracted words from the commit messages and from the ReadMe files are presented as skills, these are indicated as separate categories of information in the browser so that users have the option of restricting their selections to only skills present in the ReadMe file or the commit message.

Skills tags are sized according to the *number of commits* in which the developer has exhibited the skill (such as changing a `.java` file) as opposed to the *number of files* they have changed in one particular commit, e.g., if a developer changes multiple `.java` files in one commit `.java` is still only

associated to the commit once. Therefore, a larger sized tag indicates that a skill is *consistently exhibited* over many commits as opposed to a skill that is exhibited many times in a single commit, which could be caused be a task such as refactoring where many files are touched in a single commit. Note that a skill exhibited in 100 commits to the same project will be sized the same as a skill tag that is exhibited in individual commits across 100 projects. However, since the project name is present as a tag in the cloud as well, it is easy to identify whether the developer has exhibited a skill in multiple projects (multiple project tags, cf. Figure 5(c) where project tags for the selected developer are indicated in pink) or only one (a single project tag).

Particular web frameworks are associated with a group of file types (e.g., Ruby on Rails with `.erb, .rb, .html` and `.css`) therefore, multiple file types can be selected in the tag cloud to reveal developers that use a particular framework.

Figure 5 shows how CVExplorer can be used to identify and evaluate a Java developer. In Figure 5(a) we select the tag for Java file types (`.java` indicated in red) to identify which developers have been changing Java files. From this tag cloud we see the developers working on Java files sized according to the number of commits in which they have changed Java files. We then add the tag for a specific developer in 5(b) to see what other skills are associated with his changes to `.java` files. We now also see in which projects he has changed Java files (indicated in pink). We can see that this developer uses Java in more than one project and that most of these projects are listed on GitHub as Java projects (Java Repository language tag indicated in light blue) but that a small number of these changes are made in a project which is listed as a JavaScript project on GitHub (JavaScript language tag indicated in light blue). In Figure 5(c), we remove the tag for `.java` to show all tags associated with the developer and see what other programming languages he has experience in: we see that this developer is working on projects that are in other languages (Ruby, Python etc.) and also changing other filetypes which indicate that he is programming in a variety of languages (not just Java). Selecting a further tag for a particular skill (such as Ruby) indicates in which projects the developer has exhibited the skill. Note that tags can also be de-selected in a different order in which they were selected, which supports exploration of the underlying data.

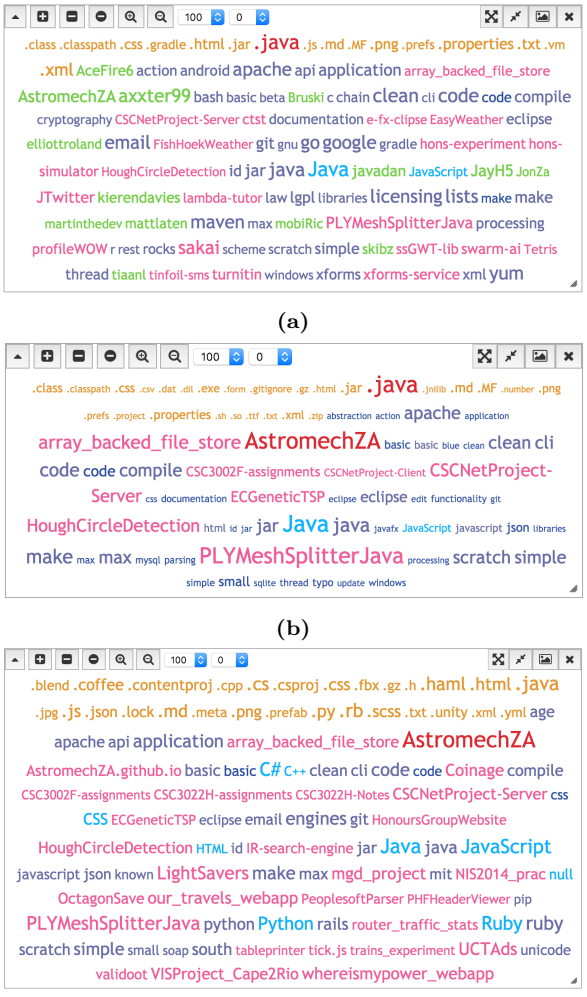

Figure 5: Top 100 tags from developers, skills, files and projects associated with a) changing `.java` files, b) developer selection editing `.java` files, and c) developer only selection.

Since tags for multiple developers are present in the tag cloud (unless an individual developer is selected) the tags are sized according to their occurrence in the full list of commits for *all* the developers. However, when an individual developer is selected the list of commits is refined to only commits from that developer and so the tags will be sized only according to the number of commits in which the selected developer has exhibited the skill. Therefore, when no developer tags are selected the tag sizes provide an indication of what skills are common in the full set of developers in that particular location. However, when an individual developer is selected the tag sizes will indicate in which languages the selected developer has worked the most.

3. IMPLEMENTATION

CVExplorer has been built using our ConceptCloud Browser Framework which provides a generic framework for browsing semi-structured data [17, 16] (available at www.conceptcloud.org). ConceptCloud has previously been used to examine the history of software projects in either Git or SVN repositories. Here we use the framework to build CVExplorer that displays information from multiple repositories and to include technical skills and programming languages extracted directly from GitHub. We process the developer's commits over multiple repositories off-line and load this into CVExplorer to be visualized as cloning multiple repositories can be too time consuming to do on-line (e.g. our Cape Town tag cloud took roughly six hours to compute).

Figure 6 shows the process of constructing a tag cloud from extracted GitHub information. Navigation in the tag cloud is achieved through an underlying concept lattice [14], which provides structure for the data. An intermediate step in the construction of a concept lattice is the construction of a formal context table. In order to construct a context table we need to make a distinction between attributes and objects in the data, where the choice of object will determine which attributes are feasible. We extract information from the ReadMe, GitHub programming language and check-in

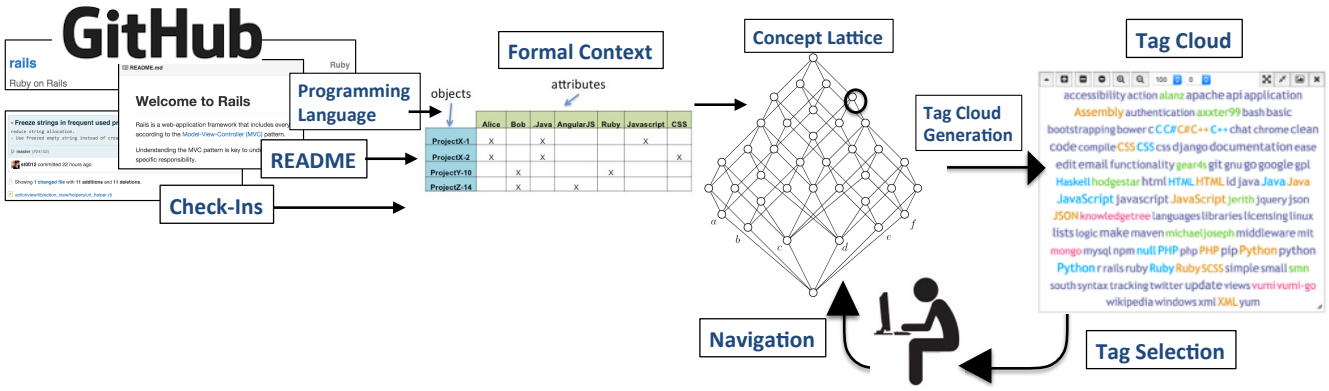

Figure 6: Context table and concept lattice construction from GitHub ReadMe and check-ins. Navigation is driven by tag selections which update the focus in the lattice generating a new tag cloud.

information (changed file types and commit messages) to construct the formal context. As objects (rows) in the context we use the project's name followed by the number of the check-in, such as "projectX-1". We then associate all skills derived from the project and commit messages as well as changed files from the check-in with the object in the context table; these form attributes (columns) in the table. We therefore examine the information at an individual commit level, so that we can use as attributes all files types changed in a particular commit. Whenever a particular commit to a project is associated with a skill, changed file or developer we indicate this with an X in the context table. A concept lattice can then be generated automatically from the context table. For the lattice construction, we use a method based on the Colibri Java library [15] which constructs concepts on the fly, so that we never need to compute the full lattice and are able to render an initial tag cloud quickly.

Our navigation maintains a lattice element called the *focus concept*, which is an individual element (concept) in the concept lattice. Each tag cloud is generated directly from this concept. The focus is recalculated after each new tag selection, which causes the tag cloud to be updated. Our approach is designed to support exploratory search of the dataset, and while concept lattices support navigation well, a large Hasse diagram of a lattice is difficult to visualize and interpret and so we therefore make use of a tag cloud interface to present the information intuitively. For more details about the underlying technology see our previous work [16].

4. PRELIMINARY EVALUATION

We preliminarily evaluate CVExplorer by using it to recommend candidates for two companies and requesting feedback on whether the candidates are suitable to interview. We are currently in the process of conducting a long-term case study in which we provide a national recruitment team with access to CVExplorer and document the number of suitable developers that are found using the tool.

4.1 Recommendation of Developers for an Open Position

We have obtained a generic job description for a Software Developer from a large South African company in the financial sector (Company A) and from a smaller company working on social development applications (Company B). We used keywords from their job description in order to match open source developers to these particular positions. We recommended 33 candidates to company A and eleven to company B. We then asked for feedback on the recommendations as to whether the companies would interview the candidates or not, which was provided in free text by the companies. The companies were able to evaluate the candidates using both their GitHub and LinkedIn profiles.

4.1.1 Suitability of Candidates for Company A

We initially identified twelve candidates for mobile development positions at Company A. One of the candidates that we identified was already in the interview process and had been sourced through other means. All candidates were marked as "typically people we would look to recruit" and "great matches" by a member of the recruitment team. We were then asked by the company to recommend candidates for a further position in a different location (Johannesburg) in South Africa. We recommended a further 21 candidates that had experience in Java, Python, C# or Ruby. The candidates were again marked as suitable to interview, with the C# candidates in particular indicated as exactly they type of developers they were trying to find. The recruitment team are currently in different stages of the recruitment process with all of the recommended candidates.

4.1.2 Suitability of Candidates for Company B

We identified five candidates for a front-end development position, of which two were deemed suitable to interview, one was deemed unsuitable due to factors other than technical skills and the LinkedIn profiles of the other two candidates provided too little information to judge their suitability. We also identified six candidates for an Android development position, out of which two were deemed suitable to interview and the other four were inconclusive because of a lack of information provided on their LinkedIn profiles.

4.1.3 Summary

We were able to identify candidates for open positions at two companies and received positive reports from the recruitment representatives of both companies. Successful identification of candidates relied only on selecting a combination of tags that were mentioned in the job specification. In some cases not all tags could be selected at the same time as no single user had all the listed skills but various com-

binations could be tried to find users that closely matched the job specification. This scenario indicates the value of an exploratory search approach which includes human involvement so that the search terms can potentially be altered to gather meaningful results. If the list of developers was extracted in report form according to specified tags then the user would not be able to alter the selected skills in order to return the best available selection of developers.

5. RELATED WORK

Hauff and Gousios [18] propose an approach to match job advertisements to developers based on the information available on their GitHub profiles. They use the DBPedia Ontology in order to extract relevant information from job advertisements and from ReadMes of the user's GitHub projects in order to match users to jobs. This work is aimed at developers actively applying for their own jobs as opposed to recruiters targeting developers. We also process ReadMe files in order to extract skills for developers. However, we do not make use of the DBPedia Ontology so that we can obtain the skills in the lowest granularity possible.

Singer et al. [24] investigate the use of profile aggregators (e.g., Masterbranch [6] and Coderwall [1]) in the assessment of developers' skills by developers and recruiters. However, while Coderwall and Masterbranch aggregate skills for individual developers, it is unclear how they support the identification of relevant developers from a large pool of candidates.

6. CONCLUSIONS AND FUTURE WORK

We have mined developers' skill sets from GitHub data to allow the filtering of a large pool of developers by relevant skills in our CVExplorer tool. Our approach supports recruiters in identifying relevant candidates using the GitHub dataset which can result in more accurate identification of developers. We used CVExplorer to recommend candidates for open positions at two companies, where one of the companies requested recommendations for a further position.

The identification of candidates using GitHub data is based on their portfolio meta-data as opposed to a self-authored CV which may be inaccurate or fail to list skills in sufficient detail. Since GitHub profiles are generated from the user's activity they always contain up-to-date information about the developer's exhibited skills. More accurate identification of passive candidates can increase the quality of hiring decisions which can therefore decrease the risk that poor hiring decisions pose to the success of a software project [25].

We see several avenues for future work. Firstly, we plan to extend this approach by including additional data in order to provide a more unified view of a developer. Information about a developer's other activities on GitHub, such as pull request comments and merges etc. could potentially also be used as indication of wider skills. Secondly, we aim to better identify the locations of developers and to pre-process the free-text locations in order to obtain a more reliable developer to location matching. We could use the timezone provided in the developer's commit logs to verify their location. We are also interested in the potential of library extraction directly from a developer's code on GitHub repositories to gain a more accurate depiction of their technical activities.

We are currently conducting a long-term case study in which we have provided a national recruitment team with CVExplorer and are evaluating its usefulness by tracking the number of developers that are sourced using CVExplorer. We also plan to evaluate the accuracy of our skills extraction step, by comparing our extracted skills to those that can be manually identified from a developer's GitHub profile.

7. REFERENCES

[1] Coderwall. https://coderwall.com/.
[2] Eclipse jgit. http://www.eclipse.org/jgit/.
[3] Egit github. https://github.com/eclipse/egit-github/.
[4] Github api. https://developer.github.com/v3/.
[5] Linkedin. http://www.linkedin.com/.
[6] Masterbranch. https://www.masterbranch.com/.
[7] Open hub. https://www.openhub.net/.
[8] Stackoverflow careers. http://careers.stackoverflow.com.
[9] Talentbin. https://www.talentbin.com/.
[10] Wikipedia. http://wikipedia.org.
[11] A. Capiluppi, A. Serebrenik, and L. Singer. Assessing technical candidates on the social web. *Software, IEEE*, Jan 2013.
[12] T. DeMarco and T. Lister. *Peopleware: Productive Projects and Teams*. Pearson Education, 2013.
[13] D. Doubrovkine. Github is your new resume. http://code.dblock.org/2011/07/14/github-is-your-new-resume.html.
[14] B. Ganter and R. Wille. *Formal concept analysis - mathematical foundations*. Springer, 1999.
[15] D. N. Götzmann. Colibri/java. http://code.google.com/p/colibri-java/, 2007.
[16] G. Greene and B. Fischer. Interactive tag cloud visualization of software version control repositories. In *VISSOFT*, pages 56–65, Sept 2015.
[17] G. J. Greene and B. Fischer. Conceptcloud: A tagcloud browser for software archives. FSE, 2014.
[18] C. Hauff and G. Gousios. Matching github developer profiles to job advertisements. In *MSR*, 2015.
[19] A. W. Kosner. Software engineers are in demand, and github is how you find them. http://www.forbes.com/sites/anthonykosner/2012/10/20/software-engineers-are-in-demand-and-github-is-how- you-find-them/.
[20] S. Lohmann, J. Ziegler, and L. Tetzlaff. Comparison of tag cloud layouts: Task-related performance and visual exploration. In *INTERACT (1)*, pages 392–404, 2009.
[21] J. Marlow and L. Dabbish. Activity traces and signals in software developer recruitment and hiring. In *CSCW '13*, pages 145–156. ACM, 2013.
[22] P. McCuller. *How to Recruit and Hire Great Software Engineers: Building a Crack Development Team*. Apress, Berkely, CA, USA, 1st edition, 2012.
[23] J. Sinclair and M. Cardew-Hall. The folksonomy tag cloud: when is it useful? *JIS*, 34(1):15–29, 2008.
[24] L. Singer et al. Mutual assessment in the social programmer ecosystem: An empirical investigation of developer profile aggregators. CSCW '13.
[25] I. Sommerville. *Software Engineering*. Pearson Education, 2016.
[26] B. Weiss. Github is your resume now. http://anti-pattern.com/github-is-your-resume-now.
[27] R. W. White and R. A. Roth. Exploratory search: Beyond the query-response paradigm. *Lectures on Information Concepts, Retrieval, and Services*, 2009.