Supervisor Recommendation Tool for Computer Science Projects

Gintare Zemaityte
School of Computer Science, University of St Andrews
St Andrews, Scotland, UK
gz25@st-andrews.ac.uk

Kasim Terzić*
School of Computer Science, University of St Andrews
St Andrews, Scotland, UK
kt54@st-andrews.ac.uk

ABSTRACT

In most Computer Science programmes, students are required to undertake an individual project under the guidance of a supervisor during their studies. With increasing student numbers, matching students to suitable supervisors is becoming an increasing challenge. This paper presents a software tool which assists Computer Science students in identifying the most suitable supervisor for their final year project. It does this by matching a list of keywords or a project proposal provided by the students to a list of keywords which were automatically extracted from freely available data for each potential supervisor. The tool was evaluated using both manual and user testing, with generally positive results and user feedback. 83% of respondents agree that the current implementation of the tool is accurate, with 67% saying it would be a useful tool to have when looking for a supervisor. The tool is currently being adapted for wider use in the School.

CCS CONCEPTS

• **Applied computing** → **Education**; • **Information systems** → *Information retrieval*;

ACM Reference Format:
Gintare Zemaityte and Kasim Terzić. 2019. Supervisor Recommendation Tool for Computer Science Projects. In *Computing Education Practice (CEP '19), January 9, 2019, Durham, United Kingdom.* ACM, New York, NY, USA, Article 39, 4 pages. https://doi.org/10.1145/3294016.3294030

1 INTRODUCTION

All undergraduate and taught postgraduate programmes in the School of Computer Science at the University of St Andrews require completion of an individual project under the guidance of a member of staff. At the moment, there are about 40 staff members who can supervise projects, and about 100 undergraduate projects, and 100 MSc dissertations completed every year. In order to match students to supervisors, we have a project blog on which supervisors advertise available projects, and students are encouraged to contact supervisors directly, or propose their own projects.

Currently there are over 200 project topics advertised on the blog, which means that it can take a long time for students to find suitable supervisors and interesting projects. Most of our MSc students are only with us for a little over a semester by the time they start looking for dissertation supervisors, so they do not have a complete overview of the research background of every staff member, so they tend to concentrate on the subset of staff who have taught them during the semester.

The tool presented in this paper provides students with a shortlist of suitable supervisors based on a list of keywords.

2 RELATED WORK

We are not aware of existing algorithms for the particular problem of supervisor application though we suspect that various approaches have been implemented in different departments around the country. There is, however, a wealth of of relevant literature on matching in the context of publish-subscribe systems, expertise retrieval, and matching conference papers to reviewers.

The main principle of *publish/subscribe* systems is matching events to subscriptions. The matches are more relevant when they are conditioned on particular attributes and value ranges. An example would be a classified ad subscription which generated alerts for all new listings for all items of type "car" with a price attribute lower than "£10000" [2, 6]. The complexity of matching large number of events to subscriptions is addressed by using trees [2] various indexing approaches [13] and optimised database queries [3]. Although our problem deals with smaller quantities of data, we make use of tree-based structures to speed up the search in our work.

Matching quality can be improved by introducing semantic similarity [11], such as incorporating ontologies [1] and qualifiers for exact and inexact predicate matches [12]. For example, Paolucci et al. employ a hierarchical ontology and flexible matches exhibiting a degree of similarity [10]. In our work, we apply these ideas by looking for semantically similar keywords using the WordNet ontology.

While not phrased explicitly as a matching problem, expertise search and discovery is relevant because it attempts to identify a list of people who are knowledgeable about a given topic. A study by Balog et al. used a corpus of heterogeneous publicly available data from their own academic institution to determine the probability of a given academic being an expert in an identified topic area [5]. Using keyword frequency for this task is a common approach [8], and although recent work has produced more complex Bayesian models [4, 9], the simpler metric *term frequency-inverse document frequency (TF-IDF)* used by Balog et al. have proven useful [5]. In our work, we use TF-IDF for ranking suggested supervisors.

*This is the corresponding author

CEP '19, January 9, 2019, Durham, United Kingdom

ACM ISBN 978-1-4503-6631-1/19/01…$15.00
https://doi.org/10.1145/3294016.3294030

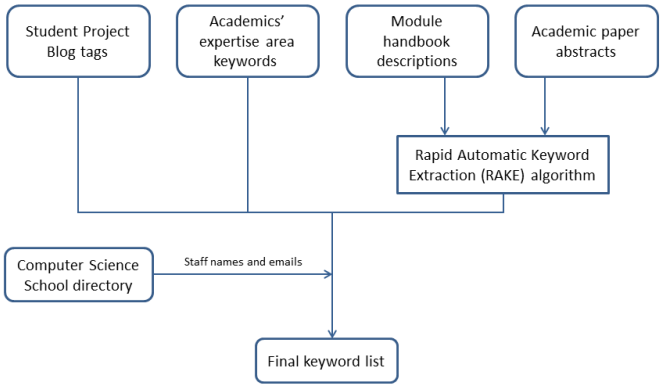

Figure 1: The keyword extraction process.

Finally, we note the increasing use of tools such as the Toronto Paper Matching System [7], but it requires a manually self-assessed list of topics for each potential reviewers, which increases the work when setting up the system for a new conference.

3 AUTOMATED MATCHING TOOL

3.1 Automated Data Gathering and Profile Generation

We begin by scraping publicly available sources and extracting useful text and keywords from them. It is accepted that a researcher's body of work is a good indicator of their expertise and interest areas [7], so the sources we use include:

- existing projects advertised on our project blog,
- existing webpages, including the list of potential PhD supervisors, which come with keywords,
- our online student handbook which includes descriptions of our taught modules and the associated teaching staff,
- abstracts from staff papers, available from University pages.

The scraping process stores the relevant text in a directory structure which is then processed by the RAKE algorithm [**?**] to automatically extract keywords.

The entire process takes about 20 minutes and is automated so it can be repeated often to take new publications into account. The result is a keyword database which related keywords to academics, and which can be used in the matching process. The current database contains about 6000 key terms. Since it has been noted that manually curating a list of keywords and keeping it up-to-date is a tedious process [4], the ability to automatically update our database (e.g. when new staff members join the school) is an important feature.

3.2 Ranking

In order to rank supervisors, we calculate the TF-IDF (term frequency-inverse document frequency) measure:

$$\text{TF-IDF} = \text{TF} \times \text{IDF}, \text{where} \tag{1}$$

$$TF = \frac{\#\text{ occurrences of a term in the document}}{\#\text{ all words in the document}}, \text{and}$$

$$IDF = \log\left(\frac{\#\text{ all documents}}{\#\text{ documents containing the term}}\right).$$

Prior to calculating the scores, we normalise the words by splitting multi-phrase terms into single words and stemming the words using the Porter stemmer available in the NLTK library. The term frequency for each stemmed word for each researcher is calculated in relation to the rest of the terms associated with that researcher.

For the inverse document frequency, the total list of documents is equal to the number of entries in the dictionary, and the search requires checking which other researchers are associated with the same word. We store the final TF-IDF score in a dictionary structure for each term for every researcher in the database. Precalculating and storing these values means that any subsequent lookup will be fast.

3.3 Matching algorithm

Our matching algorithm accepts a list of keywords and tries to match each of them to a list of supervisors in turn. The algorithm we employ is a simplified version of the semantic matching algorithm proposed by Paolucci et al. [10]. We start with an empty list of matches and then, for each keyword in turn, search a hash-based dictionary comprising 6000 keywords for matches. Each of these keywords is associated with a list of supervisors. We then look up each supervisor in a second hash-based dictionary structure in order to find the TF-IDF measure for this keyword/supervisor combination. This is then used to rank the proposals.

We apply several heuristics to filter out undesired results. This includes matching only whole words for keywords shorter than three characters (i.e. a search for "AI" would not return keywords "aim" or "contain"), and matching parts of words only if they start a word (so searching for "data" will return matches for "database"). If a keyword is not found, we obtain a list of ynonyms using NLTK's WordNet corpus, and look for those. If there are still no matches, the algorithm splits multi-word keywords into single words and looks for those. So if there are no matches for "data intensive computation", the algorithm would look for "data", "intensive" and "computation".

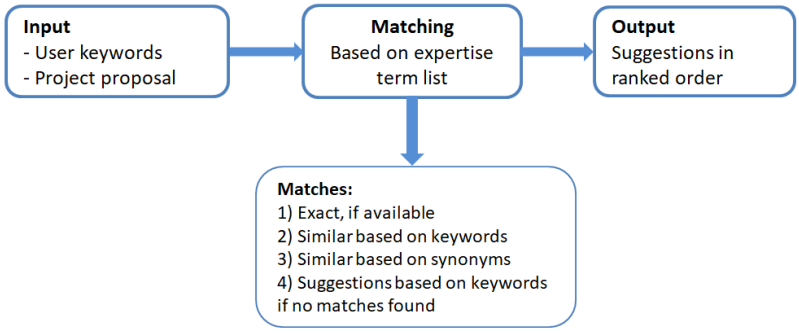

Figure 2: The matching process.

3.4 Matching based on written proposals

In addition to searching by a list of keywords, the tool also allows a student to upload a short proposal and match it to existing supervisor. In this case, the RAKE algorithm will first extract a list of keywords from the proposal, and then match them to supervisors as described above.

4 EVALUATION

Our prototype was implemented in Python, using NLTK and the existing implementation of the RAKE algorithm. There is a rudimentary GUI for entering keywords and displaying the results in order to test the tool with actual users.

4.1 User study

We conducted a small study on 12 MSc student volunteers from the department, all of whom were currently writing their dissertation. They were asked to attempt to attempt to find a supervisor for their current MSc dissertation, and to complete a survey afterwards.

The feedback was generally positive. Ten out of twelve people found the tool accurate and eight out of nine found it useful (see Fig. 3). With seven out of twelve users, the tool recommended their current supervisor, which is promising, but shows potential for improvement. However, ten out of twelve people reported that the tool recommended a suitable supervisor they did not originally consider (see Fig. 4).

The two users who did not find the tool accurate did not engage much with the tool and only tested a very small number of keywords. However, this indicates that the tool requires a certain number of descriptive keywords, which could pose a limit to adoption. This can lead to problems with new topics such as "blockchain" and "ethereum". Since there are no experts on this topic in the school, the tool will fail to recommend a supervisor, although some staff members may be able to supervise such a dissertation.

4.2 Manual testing

The traditional way to find suitable supervisors is to ask the project coordinator (in this case, the second author). In addition to the user study, we tested the tool by trying all the keywords found on our project blog and school pages and verifying that the recommendations made by the tool are sensible.

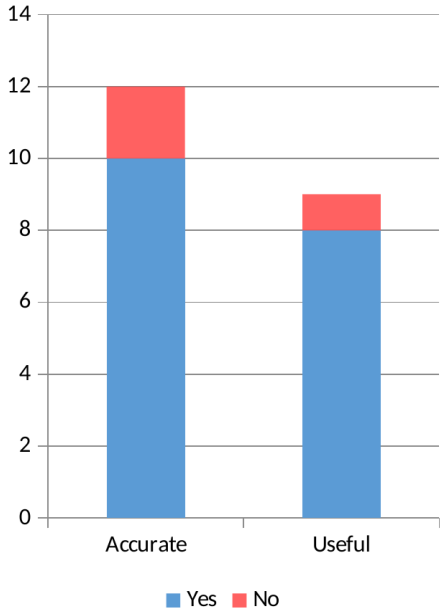

Figure 3: 10 out of 12 users in our study found the tool accurate. 8 out of 9 found it useful.

We also tested the tool by uploading a number of project proposals and looking for potential supervisors. To evaluate this, we downloaded 32 existing project proposals which were hidden (and thus not found by the scraping process). These hidden project proposals correspond to past projects which were successfully completed and are no longer advertised to students. In each of these cases, we tested whether our tool would recommend the supervisor who originally wrote the proposal. Fig. 5 shows the results. Although our tool extracted the keywords from the uploaded proposal automatically, our tool successfully recommended the original author in 50% of the cases. In most cases, at least one of the suggested supervisors was a suitable choice for supervising the proposed project. While there is obviously still room for improvement, the ability of the tool to process natural text and suggest suitable supervisors is already promising.

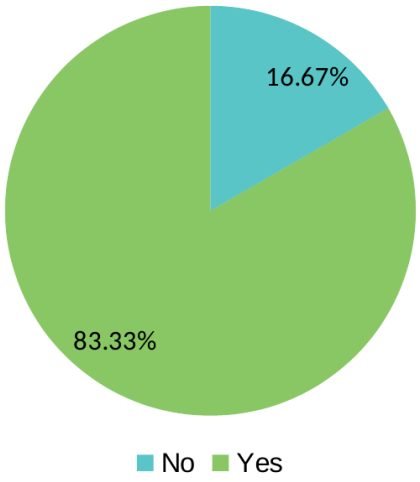

Figure 4: 83% of the users in our study found that the tool recommended a suitable supervisor they would not have otherwise considered.

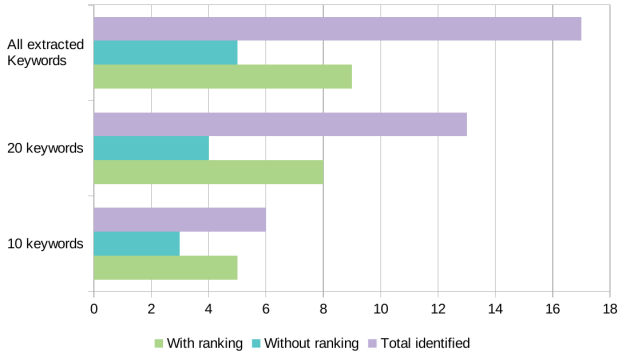

Figure 5: We tested our tool by uploading 32 project proposals which correspond to old, completed projects from previous years and automatically extracting desired keywords from the text of the proposal. In 50% of the cases, the tool suggested the original supervisor. Even when we only matched based on the first 10 extracted keywords, it suggested the original author in 17% of the cases.

5 CONCLUSION

We have presented a new tool for suggesting project supervisors in a higher education setting. With the exception of the manual selection of data sources, it is almost completely automatic and easy to keep updated. Initial evaluation indicates that users find it accurate and useful, although it fails when used with very rare keywords.

In our school, we currently assign two batches of 100 students to project supervisors each year. Even small improvements to the process could save students and staff a lot of stress and time, and matching students to suitable supervisors early could help with student satisfaction and learning outcomes. The tool we have described in this paper is not ideal, but initial evaluation suggests that it could be useful for improving the allocation process.

5.1 Further Work

We are planning to adapt the existing code to provide a RESTful API which can serve a list of suggested supervisors. This can be integrated with our existing school webpages as an online tool for searching for supervisors. This would provide links to existing projects by suggested supervisors so students can focus on a subset of relevant projects without having to read all project proposals. We believe that this would make it interesting to other departments around the country.

REFERENCES

[1] W. Abramowicz, E. Bukowska, J. Dzikowski, A. Filipowska, and M. Kaczmarek. 2011. Semantically Enabled Experts Finding System âĂŞ Ontologies, Reasoning Approach and Web Interface Design. In *East-European Conference on Advances in Databases and Information Systems*. 157–166.

[2] M. K. Aguilera, R. E. Strom, D. C. Sturman, M. Astley, and T. D. Chandra. 1999. Matching Events in a Content-based Subscription System. In *ACM Symposium on Principles of Distributed Computing*. 53–61. https://doi.org/10.1145/301308.301326

[3] G. Ashayer, H. K. Y. Leung, and H. A. Jacobsen. 2002. Predicate Matching and Subscription Matching in Publish/Subscribe Systems. In *International Conference on Distributed Computing Systems Workshops*. https://doi.org/10.1109/ICDCSW.2002.1030823

[4] K. Balog, L. Azzopardi, and M. de Rijke. 2009. A Language Modeling Framework for Expert Finding. *Information Processing & Management* 45, 1 (2009), 1–19. https://doi.org/10.1016/j.ipm.2008.06.003

[5] K. Balog and M. de Rijke. 2007. Finding Similar Experts. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*. 821–822. https://doi.org/10.1145/1277741.1277926

[6] G. Banavar, T. Chandra, B. Mukherjee, J. Nagarajarao, R. E. Strom, and D. C. Sturman. 1999. Efficient Multicast Protocol for Content-Based Publish-Subscribe Systems. In *IEEE International Conference on Distributed Computing Systems*. 262–272. https://doi.org/10.1109/ICDCS.1999.776528

[7] L. Charlin and R. S. Zemel. 2013. The Toronto Paper Matching System: An automated paper-reviewer assignment system. In *ICML Workshop on Peer Reviewing and Publishing Models*.

[8] N. Craswell, D. Hawking, A. Vercoustre, and P. Wilkins. 2001. P@NOPTIC Expert: Searching for Experts not just for Documents. In *AusWeb 2001*. 21–25.

[9] H. Fang and C. Zhai. 2007. Probabilistic Models for Expert Finding. In *European Conference on Information Retrieval Research*. 418–430. https://doi.org/10.1007/978-3-540-71496-5_38

[10] M. Paolucci, T. Kawamura, T. R. Payne, and K. Sycara. 2002. Semantic Matching of Web Services Capabilities. In *International Semantic Web Conference (ISWC)*. 333âĂŞ347. https://doi.org/10.1007/3-540-48005-6_26

[11] M. Petrovic, I. Burcea, and H. A. Jacobsen. 2003. S-ToPSS: Semantic Toronto Publish/Subscribe System. In *International Conference on Very Large Data Bases*, Vol. 29. 1101–1104.

[12] K. Sycara, M. Klusch, and S. Widoff. 1999. Dynamic Service Matchmaking Among Agents in Open Information Environments. *ACM SIGMOD Record* 28, 1 (1999), 47–53. https://doi.org/10.1145/309844.30989

[13] T. W. Yan and H. Garcia-Molina. 1999. The SIFT Information Dissemination System. *ACM Transactions on Database Systems* 24, 4 (1999), 529–565. https://doi.org/10.1145/331983.331992

Received Oct 10 2018; accepted Nov 9 2018