Using Full-text Academic Articles and Wikipedia to Find Alternative Free Bioinformatics Software

Shutian Ma, Chengzhi Zhang[*]
Nanjing University of Science and Technology, Nanjing, China, 210094

Given the importance of scientific software in academic research, more and more software are arriving on market each year and most powerful and user-friendly ones can be really expensive (Smith, 2014). Simultaneously, it turns to be difficult for researchers, teachers and students to choose suitable software which contain the required functions at reasonable cost or even no cost. Taking bioinformatics software as a case study, this paper wants to find free software which are similar with commercial ones and have potential to be alternatives. Content and network information are applied for preference-oriented results, which encapsulates how people describe them in wiki and how people use them in research.

In order to obtain bioinformatics software list, we searched the Wikipedia category named *Bioinformatics Software*[1]. Except categories starting with characters *List of*, we crawled 143 specific bioinformatics software and 20 of them are commercial ones according to licenses. 114,510 *PLOS ONE* papers are downloaded in XML format and 11,013 articles contain bioinformatics software after exact string matching. To calculate the similarity between software, we firstly project them into different vector spaces based on software knowledge graph generated by info-box from Wikipedia and textual content from papers and Wikis.

According to data source, we use following content information to represent software: paper full text, Wikipedia text and Wikipedia info box. When learning software vectors from full text of literature and wiki pages, we applied LSI (Deerwester, Dumais, Furnas, Landauer, & Harshman, 1990), LDA (Blei, Ng, & Jordan, 2003) and Doc2Vec (Le & Mikolov, 2014), vector dimensions are all set to be 100. When using Wikipedia, info box in wiki pages inspired us to build knowledge graph. For instance, info box in *MODELLER* wiki page[2] describes basic profile, such as original authors, developers, operation system and so on. Therefore, we construct a software knowledge graph consisting of seven kinds of nodes: developed by what kind of team (university, company and person), year of stable release (14 different years), written in what kind of programming language (17 languages), operation system (Linux, Unix, Windows and MacOS), applied platform (6 kinds), available language (English or cross-language) and software type (44 types). Then, Node2Vec (Grover & Leskovec, 2016), Diff2Vec (Rozemberczki & Sarkar, 2018) and struc2vec (Ribeiro, Saverese, & Figueiredo, 2017) are applied to learn node embedding, vector dimensions are all set to be 128. After representing software based on contextual and network information, we calculate cosine similarities between using these vectors and combine similarities to generate different preference-oriented software results, which are: Wiki-Content, Paper-Content, Wiki-Graph, Wiki-All, Paper Content+Wiki Content, Paper Content+Wiki Graph and Wiki+Paper. Specifically, when doing linear combinations, all weights are set to be one.

Aiming at constructing ground truth, we measure of the information overlap between two software. Firstly, queries are made on Google by searching software names one by one and all sets of free and commercial software pair in format of *Software A + Software B* one by one. Then, we convert results number returned from search engine into normalized pointwise mutual information (Bouma, 2009) using following formula:

[*] Corresponding author: Chengzhi Zhang, Email: zhangcz@njust.edu.cn.
[1] Available at: https://en.wikipedia.org/wiki/Category:Bioinformatics_software
[2] Available at: https://en.wikipedia.org/wiki/MODELLER

$$i_n(A,B) = (\log\frac{p(A,B)}{p(A)}p(B))/\log p(A,B)$$

where $p(A,B)$ is results number returned when searching query of *Software A* + *Software B*, $p(A)$ and $p(B)$ are the results number returned when searching query of *Software A* and *Software B* separately. In evaluation, we calculate precision, recall, mean average precision (MAP) and normalized discounted cumulative gain (nDCG) to compare different methods, which are shown in Table 1.

Table 1. Evaluation Results about Different Reference-Orientied Software

Oriented Preference	nDCG@15	nDCG@20	MAP@15	MAP@20	P@15	R@15
Wiki-Content	0.4312	0.4512	0.1217	0.1624	0.9933	0.1220
Paper-Content	0.3999	0.4237	0.1187	0.1584	0.9767	0.1198
Wiki-Graph	0.4112	0.4456	0.1013	0.1369	0.8933	0.1098
Wiki-All	0.4654	0.4915	0.1228	0.1638	1.0000	0.1228
PaperContent+WikiContent	0.4038	0.4219	0.1197	0.1593	0.9833	0.1207
PaperContent+WikiGraph	0.4313	0.4565	0.1219	0.1622	0.9933	0.122
Wiki+Paper	0.4268	0.447	0.1223	0.1626	0.9967	0.1224

Based on the results, we found that using Wiki content to characterize software itself is better than using full-text paper content, since most of paper-based results are worse than Wiki-based ones. By combining all the information learned within each data source and both data sources, performance isn't getting much higher as expected. One of the possible reason might be the similarity calculated based on scientific papers are not as accurate as software descriptions provided in Wikipedia. Other conclusions can also be found. Graph-based information can help to rich semantic information when only using textual context. Since scientific papers are more relevant to certain research topic, it's not suitable to use such kind of full-text publication data set to represent entities like software or some others in research.

To sum up, considering about our free software finding pipeline, the proposed method is easy to conduct and can obtain good performance, which can be applied to provide scientific users potential free software choices as alternatives. Since current results are simply from preliminary investigation, revealing different preference-oriented results. More work should be done in the future, such as finding more domains of free scientific software to users and measuring our performance based on real user experience.

Acknowledgement

This work is supported by Major Projects of National Social Science Fund (No.17ZDA291).

Reference

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research*, *3*(Jan), 993–1022.

Bouma, G. (2009). Normalized (pointwise) mutual information in collocation extraction. *Proceedings of GSCL*, 31–40.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, *41*(6), 391–407.

Grover, A., & Leskovec, J. (2016). node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 855–864). ACM.

Le, Q., & Mikolov, T. (2014). Distributed representations of sentences and documents. In *International Conference on Machine Learning* (pp. 1188–1196).

Ribeiro, L. F. R., Saverese, P. H. P., & Figueiredo, D. R. (2017). struc2vec: Learning node representations from structural identity. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 385–394). ACM.

Rozemberczki, B., & Sarkar, R. (2018). Fast Sequence-Based Embedding with Diffusion Graphs. In *International Workshop on Complex Networks* (pp. 99–107). Springer.

Smith, D. R. (2014). Buying in to bioinformatics: an introduction to commercial sequence analysis software. *Briefings in Bioinformatics*, *16*(4), 700–709.