New Graduate Candidate Selection via Combinatorial Pure Exploration

Jason Boyd, Rhys Davies, Alec Jackson
CS 5110
March 11, 2020

Abstract
How should employers in the software field narrow a list of new graduate candidates from a computer science department at a university to include only the best-suited candidates that meet their criteria? We propose implementing a centralized agent that uses combinatorial pure exploration (CPE) to perform costly exploration of a set of arms of new graduate candidates. Employers will specify minimum criteria to the centralized agent, which will in turn explore the set of arms with a given number of candidates. We will examine what number of candidates must be given to each individual agent (or arm) to produce a candidate which meets the criteria set by the employer. We will also examine what portion of agents find a candidate from the computer science department at Utah State University given that threshold, and conclude what portion of Utah State University computer science students are qualified for real-world software positions.

Introduction

The experiments found in our research explore how combinatorial pure exploration methods can be used to include more diversity in a pool of candidates [1]. We intend to use similar methods to find candidates that meet a criteria set by an employer. The system will act as an agent for each employer in producing suitable candidates, and will determine what threshold (or number of candidates) must be pulled by each arm until a suitable candidate is produced.

Employers will specify criteria in courses completed, years of experience, field of emphasis, internships, personal projects, frameworks and methodologies known, and programming languages in which candidates are proficient. We will simulate this data ourselves by examining real job postings for entry-level software engineers. We will also simulate the data for new graduates by creating a list of candidates with randomized backgrounds similar to what is currently offered by Utah State University computer science graduates. Each of the candidates will also be assigned a minimum desired salary based on experience and field of emphasis. The system will also act as an agent for the candidates by ensuring that, upon the final round of narrowing candidates for the employer, the pool of candidates does not include those whose minimum salaries are above the employers' threshold.

Using our simulated data, the centralized agent will utilize CPE methods to produce suitable candidates while minimizing resources. Once found, we will determine the minimum threshold needed to produce such candidates, what resources were minimized by the agents, and what proportion of candidates met the criteria from Utah State University graduates.

Background Information and Related Work

The problem of candidate selection is related to CPE in that it seeks to find a best fit for a job using as few resources as possible. Previous research explores the trade-offs between exploration and exploitation of utility, where the selector (in our case, the employer) may find candidates out of a pool of applicants by either selecting the arms with the top-*k* highest means, or pulling all arms above a certain mean utility threshold. The method of pulling the top-*k* highest means can be thought of as minimizing regret, while using a threshold may represent a resource limit of the employer, such as the time it takes to conduct interviews or a maximum salary allowance for a candidate [1][2][3].

Proposed Work

The purpose of our proposed work is to determine how well students at Utah State University match real-world software position criteria. Our research and implementation is narrowed into producing only the most suitable candidates for the position in question. We propose that this work focus on a mixed approach by utilizing the top-*k* method for initial selection, and the thresholding method to determine the final candidate.

Using realistic and randomly generated resumes for graduate candidate applicants specific to Utah State University, the system will ultimately return an optimal list of candidates for a specified position using top-*k* exploration. The possible data that the candidate resumes take from will be real university data. Classes, possible projects, frameworks and languages taught, and other characteristics unique to Utah State University will show up on a randomly generated graduate resume. Once the list of possible candidates and resumes are generated, the process of extracting the top candidates in the multiagent system begins using the criteria needed from an employer, being a centralized agent. A final decision representing salary negotiation will be made using a threshold, according to the employer's resource budget. To do this, we will generate a minimum salary requirement for each candidate based on their previous experience (if any) and typical salaries for an entry-level position in the employer's field of work.

As mentioned previously, employers will give specific criteria they are looking for in candidates. Employer needs and associated utility will come directly from real-world job descriptions from various job posting sites, such as Indeed.com. Based on the job description in question, utility scores will be assigned to required qualifications via a central agent. These will then be used to explore the candidates and find an optimal solution.

References

[1] Candice Schumann, Samsara N. Counts, Jeffrey S. Foster, and John P. Dickerson. 2019. "The Diverse Cohort Selection Problem." *In Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019)*, Montreal, Canada, May 13–17, 2019, IFAAMAS.

[2] Andrea Locatelli, Maurilio Gutzeit, and Alexandra Carpentier. “An optimal algorithm for the thresholding bandit problem.” *In Proceedings of The 33rd International Conference on Machine Learning,* pages 1690–1698, 2016.

[3] Lijie Chen, Anupam Gupta, Jian Li, Migda Qiao, Ruosong Wang, 2017. “Nearly Optimal Sampling Algorithms for Combinatorial Pure Exploration.” *In Proceedings of Machine Learning Research,* vol: 65:1-53*,* 2017.