ՀՏԴ 519.85

Content-based Candidate Recommendation System for Jobs

Narek Hovsepyan

***Key words**: candidate recommendation, full-text search, natural language processing, Solr, job parsing, candidate-job matching*

This paper presents a content-based candidate recommender system which is used to find the best candidates for the specified job. A variant of this recommender system is currently used by Teamable, a San Francisco-based software company that offers an intelligent employee referral platform. Our approach mainly has two phases, indexing candidate information (title, skills, location, etc.), then parsing job description/information to form a query, which helps to determine the best candidates and suggest them accordingly.

Our primary study shows that this approach produces promising results.

Introduction. Today, recommender systems are being used in a number of spheres. However, the type of recommendations provided, may be different according to the domain of its use. For a candidate recommendation system, we need to find a way to rank all the candidates for a specified job, having the job description. The candidate recommender will help job providers to find and reach out the best potential candidates.

Most of candidate recommendation systems [4, 1959-1965; 7, 169; 9, 1544-1553; 11, 2943–2951] use collaborative filtering (CF) [3, 12-32; 8, 291-324; 10, 1-19] and only small amount of approaches [2, 215-220; 12, 37-45] include some content-based approaches, which however are just used to solve so called cold-start problem of CF approach and, in general, they are all heavily CF based approaches.

Our solution focuses only on content-based recommendations which perform better than existing solutions.

Datasets. We use two main datasets, skills/responsibilities dataset to extract information from job description and job2candidates dataset for evaluation purposes. Job2candidates dataset contains jobs and their suggestions from candidates set, which is collected with the use of some open-source datasets and partly our own data at Teamable.

Skills/responsibilities dataset. We have collected a big dataset of skills/responsibilities which covers all job industries. Main sources for the dataset were Stack Exchange websites, O*NET dataset [5] and LinkedIn skills dataset. We’ve crawled all tags, skills from Stackexchange websites and LinkedIn, which are used more than 100 times.

Job2candidate dataset. We have collected a job-candidate dataset, where job information contains title and description and the candidate information contains title, skills and bio (which also includes previous experience). Using job and candidates sets, we’ve manually created the job2candidate dataset, where for each pair of job and candidate we have label 1 if the candidate is a good enough match for suggesting and 0 otherwise. Then merged it with our internal dataset of “offers and hires”, which contains job and candidate pairs, where the candidate hired or got an offer for that specific job. As a result, dataset contains 3000+ job-candidate pairs with label 0 or 1. We’ve used this dataset to evaluate solutions ability to filter out bad candidates and suggest only good enough ones.

Candidate recommender for jobs. Our approach contains several parts as shown in figure 1. At first, we index candidates in Solr [1] using all existing information. Solr is an open-source enterprise-search platform, written in Java and has some major features including full-text search, hit highlighting, faceted search, real-time indexing, dynamic clustering, database integration, NoSQL features and rich document handling.

Also, we extract information from job descriptions and then we use that information to form a Solr query, where we have already indexed our candidates’ information in a structured way.

Forming query from job information. Figure 1 also shows the main components and steps for query forming. Query format is the same Solr query format, which will help us just easily query Solr and get a response candidates list with respective scores.

Requirements classifier. This module is responsible for classifying and extracting the requirements section of the job description. It’s pretty obvious that the main part of useful terms is located in the requirements section of the job description. Here we use a simple keyword based approach to determine the requirements section. At first, we divide the job description into paragraphs, then with the use of two simple conditions try to find the paragraphs which are including requirements. The first condition is requirements-like keywords existence in a paragraph, the second one with the help of skills/responsibilities dataset we can determine how many skills are included in a paragraph and if that number is above the specified threshold that will indicate the paragraph is about requirements.

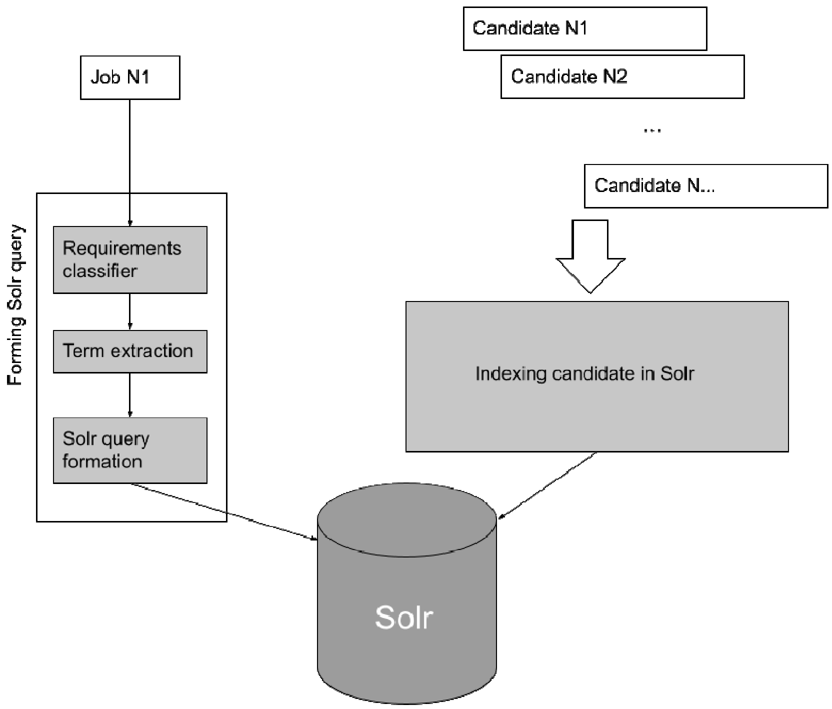

Figure 1: Candidate recommender system for jobs.

Search term extractor (STE). We use a skills/responsibilities dataset to match and extract from the requirements section of the job description. It's a simple keyword search algorithm which provides us with a list of useful terms used to form a Solr query. The main disadvantage of this approach is that we need to keep the dataset up to date to have new skills and responsibilities from all industries.

Query formation. We need to form a Solr query which will be executed on Solr engine and return candidates for suggestions sorted by Solr score. As candidates have title, bio and skills field, Solr can be used to form subquery for each field.

We have used job title and job description extracted terms joined with OR operator as a subquery for candidate title and bio fields, while for the skills field only the job description extracted terms joined with OR operator. Also, we assign weights to each subquery, which helps to prioritize some fields over the others for the term matching score. We've used grid-search algorithm to find the best matching weights, and the results of the grid-search indicated that titles need to have more priority than skills, and skills need to have more priority than bios, which is pretty intuitive.

Seniority level extraction. Above described solution works pretty fine, but has one issue connected with seniority levels. For example, if we need Senior python engineer with skills [python, Django, databases], the same exact terms can be matched in junior developers' profiles, but we don't have to suggest them. To fix this issue, it made sense to have an extra feature seniority level, which helps us to match candidates not only with terms, but also with seniority level. For jobs, we use keyword search in title trying to extract experience requirements from job description. For candidates, the same keyword search in title and calculating experience years from previous work history. We've defined 5 seniority levels, as shown in table 1.

Table 1: Seniority levels.

Level 0	Interns, juniors, …
Level 1	Mid-level, assistants, …
Level 2	Senior-level, …
Level 3	Managers, team leads, …
Level 4	Directors, senior managers, …
Level 5	C-level executives, owners, founders, …

After detecting the seniority level for candidates, we index them in Solr. And the seniority level of the job is used in Solr query to suggest candidates only with matching seniority levels.

Evaluation. As we mentioned above, we've used job2candidate dataset for evaluation, but it's only the first stage of our evaluation system. The metrics, which we used in the first stage, are simply accuracy, recall and precision.

The second stage of evaluation is customer feedback/usage, where we have defined a metric called Top10-referrals. Top10-referrals metric is calculated using referral/hire events in our platform. Top10-referrals metric shows how many percent referrals/hires made by the customer from our top 10 suggestions.

Let's denote each referred/hired candidate with x_i. Define k_i , which is equal to 1 if x_i is in the top 10 suggestions of recommendation, otherwise 0.

Top10-referrals metric is calculated as shown below:

$$y = \frac{\sum_{i=1}^{N} k_i}{N}$$

where N is the number of referred/hired candidates.

Here are the results of experiments during several months by some existing algorithms and our approach.

Evaluation stage one results are shown in Table 2:

Table 2: Evaluation stage one, results.

Solution name	Accuracy	Precision	Recall
CF (without cold start solution)	68.91%	61.66%	74.12%
CF + content-based solution	71.32%	67.37%	77.01%
Our solution (without seniority level)	65.01%	63.63%	70.09%
Our solution (with seniority level)	76.01%	75.33%	78.50%

Our solution outperforms existing CF based solutions on the first stage of evaluation. For the second stage of evaluation we've deployed our solutions for 100+ customers for several months (10000+ referred/hired candidates) in Teamable platform. The results are shown in Table 3.

Table 3: Evaluation stage two, results.

Solution name	Top10-referrals metric
CF (without cold start solution)	9.20%
CF + content-based solution	10.01%
Our solution (without seniority level)	8.51%
Our solution (with seniority level)	18.88%

The whole evaluation process shows that our solution significantly outperforms other existing solutions. Also, our solution has no cold start problem, which is a huge problem with new customers.

Conclusion. In this paper, we have described a new approach which allowed us to outperform existing CF based approaches. Also, it has no cold start problem and performs better both with or without existing data. The only disadvantage of our approach is the necessity of keeping skills and responsibilities dataset up to date to cover newly formed industries and skills. From the execution time perspective, our solution works pretty fast, even with more than 1 million candidates, the advantage mainly follows from Solr search engine, thus the solution is scalable.

The next step for this approach is to handle candidate career pivots and changes [6, 325–328], for example, a QA engineer's career path can become Software engineer, and detecting this kind of career continuations and changes can help to suggest more matching candidates for jobs.

References

1. Apache Software Foundation, Solr, https://solr.apache.org/, retrieved: 20.08.2021.
2. Belsare R. G., Deshmukh D. V. "Employment Recommendation System using Matching, Collaborative Filtering and Content Based Recommendation". Int. J. Comput. Appl. Technol. Res 7.6 (2018): 215-220.
3. Jie L., Dianshuang W., Mingsong M., Wei W., Guangquan Z., Recommender system application developments: a survey, Decis. Support Syst. 74 (2015) 12-32.
4. Koh M. F., Yew C. C.. "Intelligent job matching with self-learning recommendation engine. Procedia Manufacturing 3 (2015): 1959-1965.
5. "O*NET OnLine Help: The Database." O*NET OnLine, National Center for O*NET Development, www.onetonline.org/help/onet/database. Accessed 23 August 2021.
6. Paparrizos I., Barla C., Aristides G.. "Machine learned job recommendation". Proceedings of the fifth ACM Conference on Recommender Systems. 2011: 325-328.
7. Patel R., Santosh K. V. "An Efficient Approach for Job Recommendation System Based on Collaborative Filtering". ICT Systems and Sustainability: Proceedings of ICT4SD 2019, Volume 1 1077 (2020): 169.
8. Schafer J. B., et al. "Collaborative filtering recommender systems". The adaptive web. Springer, Berlin, Heidelberg, 2007. 291-324.
9. Shalaby, Walid, et al. "Help me find a job: A graph-based approach for job recommendation at scale." 2017 IEEE international conference on big data (big data). IEEE, 2017: 1544-1553.
10. Su X., Taghi M. K. "A survey of collaborative filtering techniques". Advances in artificial intelligence 2009 (2009): 1-19.
11. Yan X., et al. "Social skill validation at linkedin". Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2019: 2943–2951.
12. Yang S., et al. "Combining content-based and collaborative filtering for job recommendation system: A cost-sensitive Statistical Relational Learning approach". Knowledge-Based Systems 136 (2017): 37-45.

Աշխատատեղի համար թեկնածուներ առաջարկող համակարգ՝ հիմնված տեղեկատվության բովանդակության վրա

Նարեկ Հովսեփյան

Ամփոփում

Հանգուցային բառեր. *թեկնածուի առաջարկություն, ամբողջական տեքստի որոնում, բնական լեզվի մշակում, Solr, աշխատատեղի նկարագրության մշակում, թեկնածու-աշխատատեղ համապատասխանեցում*

Հոդվածում նկարագրված է տեղեկատվության բովանդակության վրա հիմնված թեկնածուներ առաջարկող համակարգ, որը օգնում է գտնելու լավագույն թեկնածուներին՝ տրված աշխատատեղի համար: Այս լուծումը օգտագործվում է Teamable ընկերությունում և օգնում է հաճախորդ ընկերություններին գտնելու իրենց թափուր աշխատատեղերի համար լավագույն թեկնածուներին: Ընդհանուր լուծումը բաղկացած է երկու հիմնական մասից, առաջինը՝ թեկնածուների տվյալների վերլուծություն, մշակում, ցուցչավորում, և երկրորդը՝ աշխատատեղի նկարագրության վերլուծություն, մշակում, կարևոր բանալի բառերի դուրս հանում ու հարցման կառուցում: Կառուցված հարցումն օգնում է գտնելու արդեն իսկ ցուցչավորված թեկնածուների բազմությունից լավագույններին:

Նկարագրված է նաև արդյունքների գնահատման երկփուլ մեխանիզմը և համեմատական վերլուծություն արդեն իսկ գոյություն ունեցող համակարգերի նկատմամբ: Գոյություն ունեցող հիմնական լուծումներն աշխատում են կոլլաբորատիվ զտման միջոցով, և այս մեթոդի հիմնական խնդիրներից մեկը նոր թեկնածուների և աշխատատեղերի դեպքում առաջարկների բացակայությունն է: Մեր առաջարկված լուծումը բովանդակության վրա հիմնված առաջարկող համակարգ է, և այն թույլ է տալիս ունենալ լավ առաջարկներ նույնիսկ նոր թեկնածուների և աշխատատեղերի դեպքում:

Հոդվածում նկարագրված լուծումն օգնում է գտնելու լավագույն թեկնածուներին և ցույց է տալիս լավ արդյունքներ:

Система рекомендаций кандидатов на работу на основе содержания информации

Нарек Овсепян

Резюме

Ключевые слова: *рекомендация кандидата, полнотекстовый поиск, обработка естественного языка, Solr, обработка должностных инструкций, кандидатов на работу*

В статье описывается основанная на содержании система рекомендаций кандидатов, помогающая найти лучших кандидатов на вакансию. Это решение используется Teamable и помогает компаниям-клиентам находить лучших кандидатов на свои вакансии. Общее решение состоит из двух основных частей: первая – это анализ, обработка, индексация данных кандидатов, а вторая – анализ информации описание работы, разработка, извлечение важных ключевых слов и построение запроса. Встроенный запрос помогает найти лучших из уже проиндексированных кандидатов.

Описывается также двухэтапный механизм оценки выборки и сравнительный анализ существующих систем. Существующие решения в основном работают через коллаборативную фильтрацию, и одна из основных проблем этого метода – отсутствие предложений для новых кандидатов и вакансий. Предлагаемое нами решение представляет собой систему рекомендаций, основанную на содержании, и позволяет получать хорошие рекомендации даже для новых кандидатов и новых вакансий.

Описанное в статье решение помогает найти лучших кандидатов и показывает хорошие результаты.

Ներկայացվել է 25.10.2021 թ.
Գրախոսվել է 27.09.2021 թ.
Ընդունվել է տպագրության 29.11.2021 թ.