Generating An Optimal Interview Question Plan Using A Knowledge Graph And Integer Linear Programming

Soham Datta, Prabir Mallick, Sangameshwar Patil
d.soham, mallick.prabir, sangameshwar.patil

Indrajit Bhattacharya, Girish Palshikar
b.indrajit, gk.palshikar, @tcs.com
TCS Research

Abstract

Given the diversity of the candidates and complexity of job requirements, and since interviewing is an inherently subjective process, it is an important task to ensure consistent, uniform, efficient and objective interviews that result in high quality recruitment. We propose an interview assistant system to automatically, and in an objective manner, select an optimal set of *technical* questions (from question banks) personalized for a candidate. This set can help a human interviewer to plan for an upcoming interview of that candidate. We formalize the problem of selecting a set of questions as an integer linear programming problem and use standard solvers to get a solution. We use knowledge graph as background knowledge in this formulation, and derive our objective functions and constraints from it. We use candidate's resume to personalize the selection of questions. We propose an intrinsic evaluation to compare a set of suggested questions with actually asked questions. We also use expert interviewers to comparatively evaluate our approach with a set of reasonable baselines.

1 Introduction

A large multi-national IT company added roughly 70,000 employees in FY2018-19.[1] Assuming an average interview time of 30 minutes, 3 interviewers in each interview, and 4 candidates interviewed for every position, implies approximately 420,000 person-hours were spent in one year just on conducting the interviews. Given the diversity of the candidates and complexity of job requirements, and considering that interviewing is an inherently human and subjective process, it is a mammoth task to ensure consistent, uniform, efficient and objective interviews that result in high quality recruitment. AI and ML technologies are increasingly playing important roles in helping improve recruitment quality, e.g., Faliagka et al. (2012), Javed et al. (2015), Palshikar et al. (2017), although ethical issues are emerging.[2] In this paper, we consider one particular way to assist human interviewers in improving the quality of their interviews and in reducing subjectivity.

Before conducting an interview, an interviewer typically studies the candidate's resume, noting salient points about her education, skills, job history, roles, projects, tasks handled etc. The interviewer also notes the apparent strengths and weaknesses of the candidate, as also the extent to which she matches (and does not match) the job profile for which she will be interviewed. In short, the interviewer builds an *a priori* rough mental profile of the candidate, and prepares a mental *plan* of how to interview her. Such a plan includes preparing an unordered set of questions that the interviewer would like to ask the candidate. In this paper, we propose an interview assistant system to automatically, and in an objective, unbiased manner, build such a set of questions for a human interviewer, which can be part of a plan for an upcoming interview. We assume we have question banks from where questions can be selected.

Note that such a plan is *static*, and the actual sequence of questions asked by an interviewer may diverge from the static plan, due to dynamic and contextual reasons observed during the flow of the interview. Such reasons include (i) mismatch between the interviewer's prior impression about the strengths of the candidate and the quality of the candidate's actual answers; (ii) the questions asked by other interviewers, if they are present. Nevertheless, such a plan generated by the system is still useful, as it reduces the cognitive load on the interviewer, and brings some standardization

[1]https://www.tcs.com/content/dam/tcs/investor-relations/financial-statements/2018-19/ar/annual-report-2018-2019.pdf

[2]https://hbr.org/2019/04/the-legal-and-ethical-implications-of-using-ai-in-hiring

Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 1996–2005
June 6–11, 2021. ©2021 Association for Computational Linguistics

and objectivity to an interview. Having a system-suggested set of questions, personalized for a particular candidate, before starting the interview is useful for the interviewer. The questions help in getting to a good start and also give ideas about where to focus during the interview.

The novel contributions of this paper are as follows. We formalize the problem of selecting a set of questions as an integer programming optimization problem and use standard solvers to get a solution. We use knowledge graph as background knowledge, and formulate our objective functions and constraints from it. To our knowledge, this is the first paper to address the problem of creating an optimal interview plan. We report experiments on a real dataset of candidates and interview questions and compare against a state-of-the-art baseline using both intrinsic and human study based evaluation.

The rest of the paper is organized as follows. Section 2 summarizes related work, Section 3 formulates the optimization problem, Section 4 gives details of our novel evaluation measure, Sectionr̃efsec:nlp describes the use of NLP techniques to build the requisite resources, Section 6 describes the baselines used for comparison, Section 7 describes the our experimental results, and Section 8 concludes the paper.

2 Related Work

Work in this paper is close to the field of *computerized adaptive testing (CAT)* (Van der Linden and Glas, 2000), where the task is to select questions (also called *items*) *on-the-fly* from a question bank, depending on how the student has answered the questions so far (i.e., adjusting to her ability level), with goals of creating shorter tests that yield better differentiation among students. CAT techniques are used in large-scale general online examinations like GRE, and in specialized medical licensing or certification examinations, e.g., in clinical pathology, emergency medicine, and pharmacy. We are not aware of any work on applying CAT techniques to interviews.

We first outline the key differences between our work and CAT, which stem from the obvious differences between interviews and examinations. Interviews are not really examinations, but are human, face-to-face, oral, and two-way interactions among a single candidate and possibly multiple interviewers. There is no set question paper, no rigid time-limit and interview questions need short free-form textual or spoken answers whereas CAT deals mostly with examinations administering multiple-choice questions. Unlike an examination, the interactions are two-way; e.g., the candidate can ask for clarification about a question. Goals of an interview are different from that of an examination, e.g., assessing fitment for job position requiring multiple skills, rather than assessing depth and breadth in a fixed subject. Students are anonymous in CAT, whereas interviewers have detailed knowledge about the candidate; e.g., through her resume. CAT is about a dynamically assembled, personalized, sequence of questions which are dependent on the student's answers so far, whereas in this paper we deal with a static one-time selection of interview questions, with no dynamic adjustment as per the candidate's answers i.e., in this paper we cannot estimate the candidate's latent abilities, since we do not have her answers. This prevents a direct comparison of our work with most CAT techniques.

See Han (2018) for a review of CAT research. The key aspects of CAT are: item selection criteria, content balancing (ensuring coverage of all sub-areas in the subject) and item exposure control (using randomization to prevent excessive item reuse across multiple examinations). Many item selection approaches are formulated using item response theory and use information-theoretic criteria; e.g., Fisher information (Weiss, 1982), efficiency balanced information (EBI) (Han, 2012), Kullback-Liebler information (Chang and Ying, 1996). Various item exposure control methods have been proposed to reduce overuse of "good" items; see Stocking and Lewis (2000) for a survey of early methods.

While some CAT systems use the above 3 aspects separately, the *automated test assembly (ATA)* approaches use them together in an optimization framework such as linear programming or mixed integer programming, where content balancing criteria are constraints and item selection criteria are objective functions; e.g., Theunissen (1986), der Linden and Boekkooi-Timminga (1989), Stocking and Swanson (1993), Swanson and Stocking (1993). For a comparative analysis of such optimization approaches, see der Linden (2005) and Luo (2020). Again, it is difficult to directly compare our work with these optimization-based approaches, because of our static setting in interviews where no answers are available to estimate candidate proficiency (unlike CAT). In most CAT approaches, the information available for each item is rather limited: subject, sub-area, difficulty level, discrimination level etc. We have used knowledge graphs to create a semantically rich and detailed characterization of questions in terms of concepts. Our optimization formulation uses the knowledge graph to generate novel constraints (including content balancing) and objective functions for item selection.

CAT algorithms cannot be directly used as baselines, because (i) they output an ordered *sequence* of questions (we output an unordered *set* of questions); and (ii) they need candidate answers, which are not available to us.

DuerQuiz (Qin et al., 2019) starts from job descriptions and resumes of candidates, and considers the problem of recommending a set of questions using a skill graph. It uses knowledge of resumes of candidates who have been hired in the past. It additionally considers the task of extracting skills from resumes and job descriptions, and construction of the skill graph, which are not our primary focus. For the actual task of question selection for a specific resume, DuerQuiz initializes weights of concepts based on the job description, historical resumes and the focus resume, and then dissipates those weights over descendant concepts in the skill graph. Finally, the weights determine the number of questions selected from a concept. It does not consider the notion of question difficulty, or relations between questions.

$$\text{maximize} \quad f_1 : \sum_{i=1}^{|Q|} x_i \cdot |\psi(q_i)| +$$

$$f_2 : \sum_{i=1}^{|Q|} \sum_{j>i}^{|Q|} x_i \cdot x_j \cdot is\_qgraph\_edge(q_i, q_j) +$$

$$f_3 : \sum_{i=1}^{|Q|} \sum_{j>i}^{|Q|} x_i \cdot x_j \cdot \neg is\_qgraph\_path(q_i, q_j) +$$

$$f_4 : \sum_{i=1}^{|Q|} x_i \cdot (\psi(q_i) \cap \Phi(W_R) \neq \emptyset) +$$

$$f_5 : \sum_{i=1}^{|Q|} x_i \cdot (\psi(q_i) \cap \Phi(W_J) \neq \emptyset)$$

$$\text{such that} \quad \mathrm{C1} : \sum_{i=1}^{|Q|} x_i \cdot T_{\delta(q_i)}(q_i) \leq T$$

$$\mathrm{C2(k)} : \sum_{i=1}^{|Q|} x_i \cdot (\delta(q_i) == k) \leq (m_k \cdot (\sum_{i=1}^{|Q|} x_i))$$

$$\mathrm{C5} : \sum_{i=1}^{|Q|} x_i \cdot \delta(q_i) \geq h_0 \cdot \sum_{i=1}^{|Q|} x_i$$

Figure 1: The integer programming problem

3 Problem Formulation

For concreteness, in this paper we focus on candidates in the IT domain. We start by noting that an interviewer asks different *types* of questions. **Technical questions** explore the breadth and depth of the candidate's understanding of a particular technical skill. Other than technical questions, interviewers also ask *techno-experience questions* (e.g. about skills in projects), *methodological questions*, *behavioural questions*, among others. For concreteness, in this paper we focus only on technical questions about skills in the IT domain. We focus on entry-level candidates (freshers or those with less than 1 year experience), because for more experienced candidates the interviewers tend to move quickly to techno-experience questions. We also assume the questions are such that they require short answers, typically containing up to 5-6 sentences.

Given a candidate resume R, a technical skill s, and a question bank QB_{s} about that skill, the task we address is: how to select the "best" questions from QB_{s}, which maximize some objective functions and meet the required constraints? The questions need to selected from QB_{s} and need to be highly *personalized* for the candidate in the sense that they should be closely related to the candidate's background mentioned in R. The complete optimization formulation is given in Fig. 1.

We use the term *skill* to refer to a broad technical area; examples: Python, Machine_Learning, Algorithms, Networking etc. Let s denote a given skill. Let C_{s} (or just C if the skill is clear) be a set of *concepts* related to a given skill. Relationships such as IS-A, HAS-A (inverse of IS-PART-OF) hold between pairs of concepts; e.g., array IS-A data_structure and class HAS-A method. We represent the concepts in a particular skill and their inter-relationships as a *knowledge graph* $G = (C, E, \eta)$, where the vertices are concepts, E is the set of directed edges linking pairs of concepts, and $\eta : E \rightarrow REL$ is the edge labeling function that associates a relationship name with every edge. Fig. 3 shows a small part of a concept graph for the skill Python; here, each vertex corresponds to a concept, the edges show relationships between concepts and the edge label is shown on each arrow.

Neighbourhood of a concept $u \in C$, denoted $\phi(u)$, is the set of concepts directly connected to u in the knowledge graph, along with u itself. For simplicity, we ignore the edge direction and edge label when computing $\phi(u)$. Example: $\phi(11) = \{4, 11, 12, 15\}$. *Neighbourhood* of a set of concepts $B = \{u_1, \ldots, u_k\}$ is the union of their neighbourhoods: $\Phi(B) = \phi(u_1) \cup \ldots \cup \phi(u_k)$.

We assume we have a question bank, which is a set of questions about a particular skill, along with some other information with each question. Formally, a *question bank* for a skill s is $QB_{\mathsf{s}} = (Q, \delta, \lambda)$, where Q is a set of questions about s, the function $\delta(q)$ associates a *difficulty level* with every question $q \in Q$, and the function $\lambda(q)$ associates a non-empty subset of concepts with every question $q \in Q$.[3] A *difficulty level* is 0 (easy), 1 (medium), 2 (hard); a more nuanced scale can be easily incorporated. Fig. 2 shows a small question bank containing 13 questions for the skill Python; it also shows the difficulty level and the subset of concepts (from the knowledge graph of Fig. 3) associated with each question. In order to prevent the same subset of questions being identified by our solution for very similar resumes, we could either shuffle the questions in Q, or use only a random subset of Q as input.

Coverage of a question q, denoted $\psi(q)$, is the set of concepts $\lambda(q)$ associated with q, along with the concepts at 1-hop from each concept in $\lambda(q)$. For simplicity, we ignore the edge direction and edge label when computing $\psi(q)$. Example: $\psi(q_2) = \{20, 21, 17, 24\}$. Let $x_i, 1 \le i \le |Q|$ be the set of Boolean variables, where if $x_i = 1$ the question q_i is included in a set of questions, and not included if $x_i = 0$. Then the first term f_1 in our objective function selects a set of questions which has the maximum coverage.

Different candidates can take different amounts of time $T(q)$ for answering any particular question q in the QB. This time is candidate specific and unknown *a priori*. We have a simple model to accommodate this time: a candidate takes time $T_i(q)$ minutes to answer a question q having difficulty level $\delta(q) = i$; for concreteness, we assume $T_0(q) = 1, T_1(q) = 2, T_2(q) = 3$. This simplified model predicts the same time, for all candidates, for all questions having a particular difficulty level; a more nuanced approach would be, for example, to learn the time distribution from data of past interviews. Interviewers often have an informal time-limit (*budget*) T on the time to spend on a particular skill. So we have constraint C1: the total estimated time taken to answer the selected questions must be at most T.

In order to prevent selection of only easy questions, we introduce constraints {C2(j)} for $j \in 0, 1, 2$ that force a more equitable user-specified distribution of difficulty levels in selected questions. The user-specified constants $0 \le m_0, m_1, m_2 \le 1, m_0 + m_1 + m_2 = 1$ give control to the user to generate questions that "suit" a particular "style"; e.g., setting $m_0 = 0.2, m_1 = 0.2, m_2 = 0.6$ will tend to select more hard questions.

Questions asked in an interview are often related to another question, indicating exploration of the depth of a candidate's knowledge. Given a set of questions $A = \{q_1, \ldots, q_k\}$, we define a *question graph* G_A, whose vertices are the questions in A and two questions q_i, q_j have an undirected edge if $\lambda(q_i) \cap \lambda(q_j) \neq \emptyset$. In general, G_A may be a disconnected graph. A path of length 1 or more indicates a sequence of inter-related questions. Fig. 4 shows the question graph for the questions in Fig. 2. A path P in a graph is a *longest path* (or *chain*) if P is not a sub-path of any other path in the graph. Now we have another term f_2 in our objective function: maximize the sum of the lengths of all longest paths in G_A. Since this is computationally expensive, we can use as an approximation the number of edges in G_A, since an edge indicates a *sequence* of two questions. Note that this term has a quadratic form, which can be easily linearized taking advantage of the fact that the decision variables are all binary (we omit this reformulation).

Questions asked in an interview are often unrelated to another question, indicating exploration of the breadth of a candidate's knowledge. We define two questions q_i, q_j as *unrelated* if there is no path between them in G_A i.e., q_i is unreachable from q_j and vice versa. Analogously, we define

[3]A more realistic setting would associate an ordered sequence of concepts with a question, ranked in terms of the decreasing relevance of the concepts to the question.

two paths P_1, P_2 in G_A as *unrelated* if no vertex in P_2 is reachable from any vertex in P_1 and vice versa. Now we have another term f_3: maximize the number of pairs of paths which are unrelated. Since this is computationally expensive, we can use as an approximation the number of pairs of vertices which are unreachable via paths from each other. Note that this objective also has the quadratic form, which we linearize.

An interview often includes questions about concepts related to a given skill which are mentioned in the candidate's resume R. Let $W_R = \{w_1, w_2, \ldots, w_\ell\}$ denote the ℓ concepts mentioned in the candidate's resume; e.g., in the descriptions of her jobs, projects, trainings etc. A reasonable interview must include questions related to as many concepts in the neighbourhood of W_R as possible, giving us another objective function term f_4. We can further refine this objective to specifically consider questions *directly* related to W_R, giving us an additional term f_4^d, with W_R instead of $\Phi(W_R)$ and $\lambda(q)$ instead of $\psi(q)$. We could refine this even further, if we could estimate from the resume that the candidate has different proficiency levels in different concepts; e.g., if a candidate has worked in Flash in 1 project of 3 months and in numpy in two projects for 11 months, then she is clearly stronger in numpy than in Flash.

Analogously, let W_J denote the set of concepts relevant to a given job description for which the candidate is being interviewed. A reasonable interview must include questions related to as many concepts in the neighborhood W_J as possible, giving us term f_5. As for f_4, here as well we consider a *direct* version f_5^d, with W_J replacing $\Phi(W_J)$ and $\lambda(q)$ replacing $\psi(q)$.

Suppose we have some idea of the proficiency level $\eta(\mathsf{s})$ that a particular candidate has in a given skill s. This estimate could be generated from the information in the resume (projects, tasks, trainings) or from other sources, such as the scores in a prior written test. Suppose the estimated proficiency level in a skill is an integer from 0 (does not know) to 4 (expert). We should take this input into account in order to adjust the difficulty level of selected questions; e.g., a candidate with proficiency level $\eta(\mathsf{s}) = 3$ should be asked fairly difficult questions. This gives us constraint C5, which says that the average difficulty level of selected questions should be above a user-specified

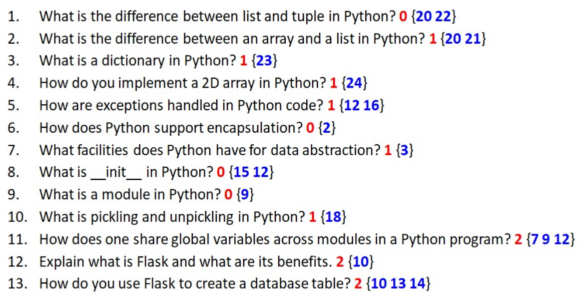
1. What is the difference between list and tuple in Python? 0 {20 22}
2. What is the difference between an array and a list in Python? 1 {20 21}
3. What is a dictionary in Python? 1 {23}
4. How do you implement a 2D array in Python? 1 {24}
5. How are exceptions handled in Python code? 1 {12 16}
6. How does Python support encapsulation? 0 {2}
7. What facilities does Python have for data abstraction? 1 {3}
8. What is __init__ in Python? 0 {15 12}
9. What is a module in Python? 0 {9}
10. What is pickling and unpickling in Python? 1 {18}
11. How does one share global variables across modules in a Python program? 2 {7 9 12}
12. Explain what is Flask and what are its benefits. 2 {10}
13. How do you use Flask to create a database table? 2 {10 13 14}

Figure 2: A small question bank for the skill Python.

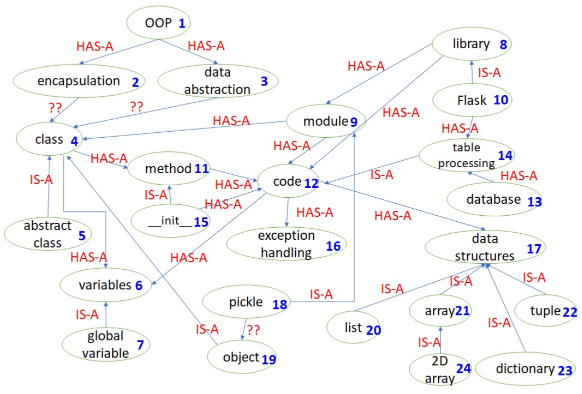

Figure 3: A knowledge (sub)graph for skill Python.

constant h_0, which can be derived from the proficiency level $\eta(\mathsf{s})$ of the candidate in skill s.

We normalize the terms in the objective function so that these take values in $[0, 1]$. Further, we take a weighted sum (instead of the plain sum) of the terms: $w_1 \cdot f_1 + \ldots + w_5 \cdot f_5$, where $w_1, \ldots, w_4, w_4^d, w_5, w_5^d$ are user-given positive real weights. The weights will allow the interviewer to change the relative importance of the terms.

Interview Plans for a Set of Candidates: The optimization program discussed so far is useful to generate an interview plan for one particular candidate. However, in many situations (such as campus interviews), there is a sequence of interviews for multiple candidates and the system is required to generate a system plan for each of them. There are additional constraints on the set of interview plans generated in such a situation. For example, the repetition of questions across multiple candidates should be minimized i.e., different candidates (even those having a similar background) should by-and-large get different sets of questions.

Let N_0 denote the number of candidates to be interviewed in the current campaign. We assume that a single question bank QB_s for skill s (or, just Q for simplicity) will be used to se-

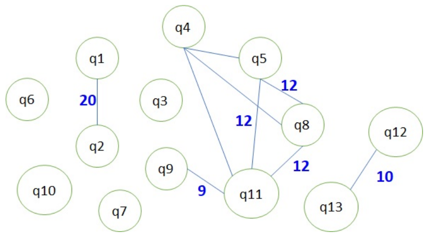

Figure 4: Question graph for the example questions.

lect questions for each of the N_0 candidates. Let $gen\_interview\_plan$ denote the above optimization program for selecting questions for a given candidate. Output of this program is a Boolean vector sol, where $sol(i) = 1$ if question $q_i \in Q$ is to be included in the interview plan for the given candidate; 0 otherwise. We extend our earlier definition of a question bank by adding another property for each question viz., *novelty count* $\beta(q_i)$, which is the $N_0 -$ count(q_i), with count(q_i) being number of times the question $q_i \in Q$ has been used so far. Initially count$(q_i) = 0$ and $\beta(q_i) = N_0$ for each question in Q. We sequentially call $gen\_interview\_plan$ for each of the N_0 candidates and use the output Boolean vector sol to increment $\beta(q_i)$ for each question in Q; see algorithm $gen\_interview\_plan\_set$.

```
input : ${R_1, \ldots, R_{N_0}}, Q$
output: $sol\_set$
$sol\_set := \emptyset$;
foreach $i$ in $1 \ldots |Q|$ do
    $\beta(q_i) := N_0$;
end
foreach $j$ in $1 \ldots N_0$ do
    $sol := gen\_interview\_plan(R_j, Q)$;
    $sol\_set := sol\_set \cup \{(j, sol)\}$;
    foreach $i$ in $1 \ldots |Q|$ do
        $\beta(q_i) := \beta(q_i) - sol(i)$;
    end
end
```
Algorithm 1: $gen\_interview\_plan\_set$

The optimization program $gen\_interview\_plan$ is same as earlier, except that we have added a new term f_6 to the objective function that maximizes the sum of novelty counts of the questions.

$$f_6 : \sum_{i=1}^{|Q|} x_i \cdot \beta(q_i) \quad (1)$$

Again, we normalize this to ensure that the value is between 0 and 1. Note that the novelty counts are updated for all questions after each call to the optimization program.

Integer programming being an NP-Hard problem, exact solvers often take a lot of time. Instead, we resorted to the LP rounding approximation, where we relax the integer constraints $x_i \in \{0, 1\}$ to $x_i \in [0, 1]$ and used available LP solvers (CBC solver in python Pulp) to solve the resultant linear programming problem. Then we rounded the x_i values to $\{0, 1\}$ using a threshold.

4 Intrinsic Evaluation

While we do use human experts to compare two sets of questions, here we propose an automated intrinsic evaluation method to judge the quality of the set of questions (e.g., selected by our optimization model, or by any baseline method discussed later) for a particular candidate, by comparing this set with the set of questions actually asked to this candidate in a real interview. Let A_i denote the set of actual questions asked to i-th candidate, ignoring the order of asking; we can get A_i from interview transcripts. Let Q_i be the set of questions recommended by some algorithm for the i-th candidate. In general, $|A_i| \neq |Q_i|$. We assume that both Q_i and A_i are about the given skill s.

Suppose we have a Boolean function is_qsim (discussed shortly) that returns $TRUE$ if two given questions are “highly similar” and $FALSE$ otherwise. In *forward* evaluation, we compare each question $q_j \in Q_i$ with questions in A_i and define a forward Boolean score such that $b_f(q_j) = 1$ is there is at least one question $a_k \in A_i$ such that $is\_qsim(q_j, a_k) = TRUE$ and 0 otherwise. The quality of Q_i is evaluated based on the number of questions in Q_i having score 1.

$$qual_f(Q_i) = \frac{1}{|A_i|} \cdot \sum_{j=1}^{|Q_i|} b_f(q_j) \quad (2)$$

Enforcing an additional constraint that a question in A_i is “matched” to at most one question in Q_i, ensures that score $qual_f(Q_i)$ is between 0 and 1.

In *backward* evaluation, we compare each question $a_j \in A_i$ with questions in Q_i and define a backward Boolean score such that $b_b(a_j) = 1$ if there is at least one question $q_k \in Q_i$ such that $is_qsim(q_k, a_j) = TRUE$ and 0 otherwise. The quality measure $qual_b(A_i)$ is defined analogously.

It remains now to define is_qsim. Several similarity measures have been designed in the literature for short text similarity. For efficiency, we consider a simple measure: two questions are similar if their coverages share at least a required minimum number of concepts k_0 (which is a user-specified constant):

$$is\_qsim(q_i, q_j) = \begin{cases} T & \text{if } |\psi(q_i) \cap \psi(q_j)| \geq k_0 \\ F & \text{otherwise} \end{cases} \tag{3}$$

For example, questions 12 and 13 in Fig. 2 are similar (i.e., $is\_qsim(12, 13) = TRUE$), assuming $k_0 = 3$, since concepts 8, 10, 14 are common in $\psi(12)$ and $\psi(13)$.

5 NLP for Interview Resource Creation

While natural language processing does not play a direct role in the optimization formulation for the selection of interview questions, it is crucial for the creation of the prerequisite resources.

The first task is the annotation of questions to identify concepts. These concepts are used to determine question coverage and to construct the question graph based on their coverage. We also use these annotations to construct the knowledge graph for skills, as we explain below. There is a rich body of literature in both mention detection and named entity disambiguation (Ferragina and Scaiella, 2012; Ganea and Hofmann, 2017; Sakor et al., 2020). Since we focus on skill-graphs which are largely sub-graphs of DBPedia, we use the publicly available APIs from TAGME (Ferragina and Scaiella, 2012). However, this resulted in two types of errors. Many mentions were annotated with concepts irrelevant for our skills. Secondly, many mentions relevant for our skills were left unannotated. We performed manual curation on the TAGME annotations to correct these two types of errors.

The second task is extraction of skills from resumes. We use an information extraction system called RINX (Pawar et al., 2017), which uses gazetteer-based, linguistic patterns based machine learning methods (e.g., CRF, BiLSTM) to extract mentions of various entity types and relations from resumes. For example, RINX extracts mentions of SKILL (e.g., `Machine_learning`, `Python`, CONCEPT (e.g., `Activation function`, `Maximum margin`), ROLE (e.g., `Developer`, `DBA`, `Test_Engineer`), TASK (e.g., `Developed vendor master`, `Performance Tuning`), among other entity types. The extracted skills are again annotated according to concepts by using TAGME, and manually curated to correct errors.

The next task is construction of the knowledge graph for different skills, based on the concept annotation of the questions and the extracted resume skills. This problem has not received enough attention in the computational linguistics community. We first identified a subgraph of DBPedia (Faralli et al., 2018) using the question concepts and the resume skill concepts as positive seeds. We then curated this knowledge graph manually to correct any errors.

The next task is assigning difficulty levels to questions. This problem has also received very little attention (Padó, 2017). We use the following simple approach. Use any automatic answer extraction technique to extract an answer text A for q, from a suitable corpus like Wikipedia or a textbook. Let $\lambda(A)$ be the set of concepts associated with A. The degree $d(u)$ of a concept vertex u in the knowledge graph, ignoring edge directions, is a good approximation of the "complexity" of that concept; a concept is complex, if it is directly related to many other concepts. Thus the sum of the complexities of the individual concepts in the answer to a question is a good measure of the complexity of that question: $\gamma(q) = \sum_{u \in \lambda(q)} d(u)$. We can now write simple rules to assign a difficulty level to each question: **if** $\gamma(q) \leq c_0$ **then** $\delta(q) = 0$ **else if** $c_0 < \gamma(q) \leq c_1$ **then** $\delta(q) = 1$ **else** $\delta(q) = 2$ (c_0, c_1, c_2 are user-specified constants). More complex approaches, such as applying machine learning to predict difficulty levels for questions, are possible.

Finally, we identify similar questions for our evaluation. The is_qsim() function (Eqn.3) uses overlap between annotated concepts for simplicity. Clearly, there is a need for more sophisticated approaches, for example using paraphrase detection (Galbraith et al., 2017).

6 Baselines

To compare against out integer programming approach (IP), we use the following baselines for selecting questions for a candidate having resume R:

- **BR1**: Select n_q questions randomly from QB_s, where n_q is same as the number of questions in the optimal plan.
- **BR2**: Let $F_R(\mathsf{s})$ denote the set of concepts related to skill s mentioned in resume R. Select n_q questions randomly from QB_s, where coverage of each selected question q has at

least one concept common with the neighbourhood of the concept set $F_R(\mathsf{s})$ i.e., $\psi(q) \cap \Phi(F_R(\mathsf{s})) \neq \emptyset$.

- **BR3**: same as **BR2**, but ensures even distribution of question difficulty levels.
- **DuerQuiz** (Qin et al., 2019): discussed in Section 2. Since no implementation is publicly available, we implemented our own version of this baseline. Since we do not use historical resumes or skill graph edge labels in our framework, we adapted DuerQuiz in the best possible way for out setting. We ignore the terms corresponding to historical resumes in the weight assignment to concepts. We further approximate descendants of concepts as their direct neighbors in the skill graph, both for weight initialization and weight propagation.

For our MIP formulation, we use the following weights and hyper-parameters: $w_1 = 100$, $w_2 = 100$, $w_3 = 100$, $w_4 = 30$, $w_4^d = 70$, $w_5 = 30$, $w_5^d = 70$, $m_0 = 0.3$, $m_1 = 0.4$, $m_2 = 0.3$, $h_0 = 0.9$ and $T = 45$. These weights were not fine-tuned, aside for T for controlling the number of recommended question. For DuerQuiz, the paper does not recommend any thumb rule for setting hyper-parameters. We set the propagation weight $\alpha_c = 0.85$ by hand-tuning on one resume, and the smoothing weight $\beta_f = 0.001$.

7 Experimental Results

In this section, we describe our dataset derived from real interviews, and the experiments that we conducted using this dataset. We report comparisons of our proposed approach (IP) against the baselines defined in Section 6 using both actual interview questions and as well as an user-study for evaluation. We leave out BR2 and instead use its stronger version BR3.

All experiments were performed on an Ubuntu 18.04.5 LTS machine with 8-core Intel i7-8550U 1.80GHz processors, and 16 GB memory. For IP, generation of questions with 45min time budget ($\sim$ 25 questions) takes 155 secs on average.

Dataset: We constructed a knowledge graph of 714 concepts and 903 edges (avg. degree 2.51) from Machine Learning and Deep Learning. Our question bank consists of 549 questions from these two skills. Each question is annotated with concepts from the knowledge graph (1.18 concepts per question on average). Finally, we use real resumes of 40 candidates (mostly fresh IT graduates) interviewed by our organization over the last year. We identify knowledge graph concepts associated with their resumes (4.7 concepts per resume on average). For 20 of these candidates, we also have the actual questions asked to them during their interviews. Of these, we consider only the questions related to our two topics of interest. The average number of questions per candidate is 5.05.

Intrinsic Evaluation on Real Interviews: In our first evaluation, we compared the set of suggested questions with the set of actually asked questions. Fig. 5 shows a comparison of our optimization formulation with the three baselines, using the forward and backward quality measures (Section 4).

As seen, our approach is clearly better than all three baselines in both evaluations and for different values of k_0. The differences are large for backward evaluation. The improvement against BR1 shows the importance of focusing on the resume, rather than randomly selecting questions related to the skill. The improvement against BR3 shows that just focusing on questions related to the resume is not enough. Finally, the improvement against DuerQuiz, which combines aspects of both BR1 and BR3, shows the importance of the additional terms in our objective function. Also, our analysis shows that DuerQuiz is poor at balancing between high-degree and low-degree concepts in the knowledge graph. Depending on the value of its dissipation hyper-parameter (α_c), it either transfers all the weight of high-degree concepts to their neighbors, or does not transfer any weight from low-degree concepts to their neighbors. IP's trade-off using different terms and their corresponding weights works much better.

We further note that BR1 and BR3 perform better than DuerQuiz in terms of forward evaluation, which indicates that these generate fewer irrelevant questions. On the other hand, DuerQuiz is better than these baselines in terms of backward evaluation. This indicates that the questions generated by these baselines are more heterogeneous and lack diversity when compared against DuerQuiz to cover all questions asked during a real interview. Note that IP outperforms DuerQuiz in both directions.

Human Evaluation: Our second evaluation

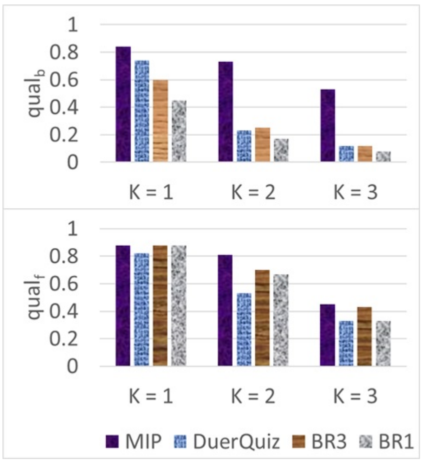

Figure 5: Backward and forward intrinsic evaluations.

compares question sets generated using pairs of algorithms by 3 experienced human interviewers E_1, E_2, E_3. We have 3 pairs of algorithms to compare: (IP, BR1), (IP, BR3), (IP, DuerQuiz). Note that to reduce the load on the human evaluators, we did not invest in comparing the baselines with each other. We randomly assign one of these pairs to each of the $N = 20$ candidates; e.g., 7 candidates got the pair (LP, BR1) and so forth. For each candidate, we generated two sets of questions, for the skill Machine Learning, using the algorithm pair assigned to it. Hiding the algorithm used to generate the question sets, we presented the two sets for the 20 candidates to each of the 3 experts, along with skills extracted from their resumes. For each candidate, each human expert gave a comparative ranking, indicating whether set 1 was better than set 2. We had not suggested any criterion for this comparison; each expert used her own intuition.

There were $7 \times 3 = 21$ evaluations of (IP, BR1) pair, out of which IP "won" in 19. Using χ^2 test with 99.9% confidence, we reject the null hypothesis and accept that IP is better than BR1. Similarly, IP is better than BR3 in 14 out of 21 evaluations (χ^2 85% confidence). Unfortunately, IP is better than DuerQuiz in only 6 out of 21 evaluations. However, there was large disagreement among the experts in this case, and discussions showed that the experts' evaluation criteria were considerably simpler than the objective functions used in IP. For example, no expert considered the inter-linking of the questions in her evaluation, nor did they consider duplication of questions across different candidates as undesirable; but these are important factors in IP for choosing questions. In the future, we intend to perform a larger expert study with a more nuanced evaluation which compares specific quality aspects of the question sets.

8 Conclusions and Further Work

We have proposed an interview assistant system to automatically select an optimal set of technical questions (from a question bank) personalized for a candidate. We formalized the problem of selecting a set of questions from question banks as an integer programming problem, with multiple terms in the objective functions and multiple constraints. We used knowledge graph as background knowledge, and used the candidate's resume to personalize the selection of questions. We proposed a novel intrinsic evaluation to compare a set of suggested questions with actually asked questions in real interviews. We also used expert human interviewers to comparatively evaluate our approach with a set of reasonable baselines. Our comparisons against state-of-the-art and ablated baselines show the usefulness of our proposed approach.

References

H.H. Chang and Z. Ying. 1996. A global information approach to computerized adaptive testing. *Applied Psychological Measurement*, 20:213–229.

W.J. Van der Linden. 2005. A comparison of item-selection methods for adaptive tests with content constraints. 42:283–302.

W.J. Van der Linden and E. Boekkooi-Timminga. 1989. A maximin model for IRT-based test design with practical constraints. *Psychometrika*, 54:237–248.

Evanthia Faliagka, Kostas Ramantas, Athanasios Tsakalidis, and Giannis Tzimas. 2012. Application of machine learning algorithms to an online recruitment system. In *Seventh International Conference on Internet and Web Applications and Services (ICIW 2012)*, pages 215–220.

S. Faralli, I. Finocchi, S. Ponzetto, and P. Velardi. 2018. Efficient pruning of large knowledge graphs. In *IJCAI*.

P. Ferragina and U. Scaiella. 2012. Fast and accurate annotation of short texts with wikipedia pages. *IEEE Software*, 29(01):70–75.

Byron Galbraith, Bhanu Pratap, and Daniel Shank. 2017. Talla at SemEval-2017 task 3: Identifying similar questions through paraphrase detection. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*.

O. Ganea and T. Hofmann. 2017. Deep joint entity disambiguation with local neural attention. In *EMNLP*.

K.T. Han. 2012. An efficiency balanced information criterion for item selection in computerized adaptive testing. *Journal of Educational Measurement*, 49:225–246.

Kyung (Chris) Tyek Han. 2018. Components of the item selection algorithm in computerized adaptive testing. *Journal of Educational Evaluation for Health Professions*, 15(7):1–13.

F. Javed, Q. Luo, M. McNair, F. Jacob, M. Zhao, and T. S. Kang. 2015. Carotene: A job title classification system for the online recruitment domain. In *IEEE First International Conference on Big Data Computing Service and Applications*, pages 286–293.

X. Luo. 2020. Automated test assembly with mixed-integer programming: The effects of modeling approaches and solvers. *Journal of Educational Measurement*.

Ulrike Padó. 2017. Question difficulty – how to estimate without norming, how to use for automated grading. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*.

G.K. Palshikar, R. Srivastava, and S. Pawar. 2017. Analytics-led talent acquisition for improving efficiency and effectiveness. In *5th IIMA International Conference on Advanced Data Analysis, Business Analytics and Intelligence (ICADABAI 2017)*.

S. Pawar, P. Bhattacharyya, and G.K. Palshikar. 2017. End-to-end relation extraction using neural networks and markov logic networks. In *EACL*.

Chuan Qin, Hengshu Zhu, Chen Zhu, Tong Xu, Fuhzen Zuang, Chao Ma, Jingshuai Zhang, and Hui Xiong. 2019. Duerquiz: A personalized question recommender system for intelligent job interview. In *Proc. 25th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD'19)*, pages 2165–2173.

A. Sakor, K. Singh, A. Patel, and M. Vidal. 2020. Falcon 2.0: An entity and relation linking tool over wikidata. In *CIKM*, pages 3141–3148.

M.L. Stocking and C. Lewis. 2000. Methods of controlling the exposure of items in CAT. In *Computerized adaptive testing: theory and practice*, pages 163–182. Kluwer Academic Publishers.

M.L. Stocking and L. Swanson. 1993. A method for severely constrained item selection in adaptive testing. *Applied Psychological Measurement*, 17:277–292.

L. Swanson and M.L. Stocking. 1993. A model and heuristic for solving very large item selection. *Applied Psychological Measurement*, 17:151–166.

T.J. Theunissen. 1986. Some applications of optimization algorithms in test design and adaptive testing. *Applied Psychological Measurement*, 10:381–389.

W. J Van der Linden and C.A.W. (eds) Glas. 2000. *Computerized adaptive testing: Theory and practice*. Kluwer Academic Publishers.

D. Weiss. 1982. Improving measurement quality and efficiency with adaptive testing. *Applied Psychological Measurement*, 6:473–492.