Learning to Ask Screening Questions for Job Postings

Baoxu Shi* Shan Li* Jaewon Yang Mustafa Emre Kazdagli Qi He
LinkedIn
{dashi,shali,jeyang,ekazdagli,qhe}@linkedin.com

ABSTRACT

At LinkedIn, we want to create economic opportunity for everyone in the global workforce. A critical aspect of this goal is matching jobs with qualified applicants. To improve hiring efficiency and reduce the need to manually screening each applicant, we develop a new product where recruiters can ask screening questions online so that they can filter qualified candidates easily. To add screening questions to all 20M active jobs at LinkedIn, we propose a new task that aims to automatically generate screening questions for a given job posting. To solve the task of generating screening questions, we develop a two-stage deep learning model called Job2Questions, where we apply a deep learning model to detect intent from the text description, and then rank the detected intents by their importance based on other contextual features. Since this is a new product with no historical data, we employ deep transfer learning to train complex models with limited training data. We launched the screening question product and our AI models to LinkedIn users and observed significant impact in the job marketplace. During our online A/B test, we observed +53.10% screening question suggestion acceptance rate, +22.17% job coverage, +190% recruiter-applicant interaction, and +11 Net Promoter Score. In sum, the deployed Job2Questions model helps recruiters to find qualified applicants and job seekers to find jobs they are qualified for.

KEYWORDS

screening question generation, applicant screening

ACM Reference Format:
Baoxu Shi* Shan Li* Jaewon Yang Mustafa Emre Kazdagli Qi He. 2020. Learning to Ask Screening Questions for Job Postings. In *SIGIR '20: 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, July 25–30, 2020, Xi'an, China.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

LinkedIn is the largest hiring marketplace in the world, hosting over 20 million active job postings that are created across various channels, including LinkedIn's on-site recruiting products and integrations with external hiring products.

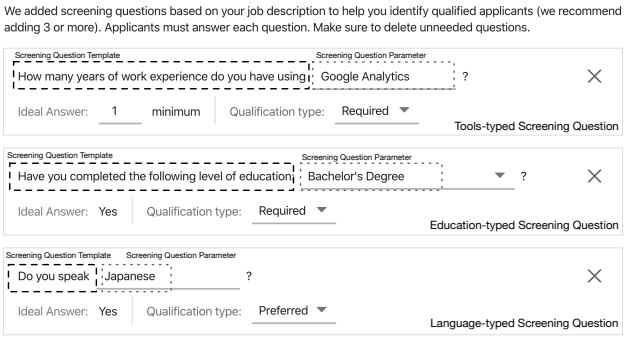

Figure 1: Screenshot of screening questions suggested to a job posting being posted on LinkedIn.

In hiring, interviewing applicants is costly and inefficient. Therefore, recruiters typically screen the applicants in the pool by their profile and conduct additional phone screenings before sending out interview invites. According to our user research study, approximately 70% of phone screenings end up finding out the applicant is missing basic qualifications such as work authorization/visa, minimum years of experience, or degree requirements. Also, the majority of the applications for coveted jobs disappear in the hiring funnel because recruiters do not have time to review them all.

To address such hiring inefficiency, researchers have proposed models to ease the manual workload by estimating person-job fit automatically. Existing methods aim to match job postings based on the members' experiences [32] or based on members' profile attributes [8, 13, 26, 42]. However, these models heavily rely on the assumption that applicants' online profile and resume are always up-to-date and contain all the information that hiring companies need. As we show in Sec. 6.4, the member profile is not the perfect source for modeling applicants because 1) members do not update their profiles promptly, and 2) there is often a gap between what members present in their profile and what employers want to know. Moreover, in Sec. 6.4 we also find that job posting text is sub-optimal for modeling job qualifications due to trivial and unnecessary requirements.

Based on the above observations, we decide to design a new Screening Question (SQ) -based online screening product for LinkedIn to assess job applicants automatically. To be specific, we proactively ask job-specific questions as shown in Fig. 1 to applicants and assess them using the answers they provide when applying for the job. Compared to member profiles, the answers collected by SQs are most recent and contain all the facts employers want to learn.

There are two significant product challenges for designing a successful SQ-based online screening product. Firstly, the product should provide an easy way to add SQs to jobs. If we ask recruiters to reformulate their job postings into SQs manually, such an excessive need for human efforts will forbid us to add SQs to all 20 million

* These authors contributed equally.

SIGIR '20, July 25–30, 2020, Xi'an, China

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

jobs on LinkedIn. Secondly, SQs should help recruiters identify qualified applicants quickly. If we use unstructured text questions [3, 9] to present SQs, one job requirement may have many different expressions and hence hard for recruiters or AI models to interpret the intent of the SQ, group SQs with same intent together, and categorize applicants based on their answers to SQs.

In this work, we address the above product challenges by designing and productionizing a Screening Question Generation (SQG) model called Job2Questions, which automatically generates structured SQs for a given job posting. By developing a machine learning SQG model, we no longer rely on human input and can apply the SQG model to generate SQs for all 20M jobs on LinkedIn. By generating structured SQs in the format of (*template*, *parameter*) as illustrated in Fig. 1, SQs will have a unified internal representation that describes SQs' intent (template) and focus (parameter) precisely. To ensure SQ quality, we asked hiring experts to design and review the SQ templates and corresponding parameter lists. Using structured representation instead unstructured text avoids SQ ambiguity and discrepancy across different jobs. This also makes it easier for AI models and recruiters to group and screen candidates based on specific intent such as education, language, and others.

Although researchers have studied the Question Generation (QG) task extensively, SQG cannot be viewed as a simple application of QG methods because it poses many unique challenges as follows.

Diversified input styles and topics. Unlike QG datasets which are often shorter passages focusing on a few specific topics, the input of SQG are lengthy text having both different narrative styles across different industries and also various topics ranging from company introduction, requirements, to benefits. As shown in Tab. 1, the average number of words and sentences per LinkedIn job posting is larger than other common QG datasets. We believe a good SQG model needs to be able to process long text and general enough to handle job postings from different industries.

Job marketplace domain-specific. Majority of the QG methods [11, 37, 41] are designed to generate questions to test the cognitive skills of readers [4]. To generate QGs that represent important job qualifications, a good SQG model needs to be domain-specific and have deep understanding of the job marketplace. In fact, generic QG models yield embarrassing results for the SQG task. Given the text of a Staff Software Engineer job posting, QG methods return *What is to enable others to derive near-limitless insights from LinkedIn's data?* [15] or *what does Experience stand for?* [9], which are not SQs as they do not represent job qualifications.

Low online inference latency. Lastly, QG models are designed without explicit latency constraints. As shown in Tab. 2, QG methods usually have 20+ms latency. However, SQG model has a more strict latency requirement because recruiters expect SQG model to provide screening questions right after they entered the job description. To avoid sluggish performance and poor customer experience, a good SQG model needs to have a simple yet effective architecture in order to keep the inference latency within an acceptable range.

With all above challenges in mind, here we propose a two-step SQG model named Job2Questions that given the content of a job posting, first generates all possible structured SQ candidates using a deep learning model, and then ranks and identifies top-k screening questions as the model output.

Table 1: Statistics of popular question generation datasets.

Dataset	Avg. words/doc	Avg. sents/doc
SQuAD [34]	135	5
RACE [24]	323	18
LinkedIn	**584**	**40**

Table 2: Empirical CPU inference time per sentence. J2Q-TC-DAN is our current in production model.

	Rule-based	Seq2Seq	Our SQG Models		
Latency	H&S [15]	NQG [9]	BOW-XGB	**J2Q-TC-DAN**	J2Q-TC-BERT
(ms)	24ms	62ms	4ms	**9**ms	100ms

In candidate generation, we divide job postings into sentences and generate all SQ candidates by converting each SQ-eligible sentence to structured (*template*, *parameter*) pairs. To get the template of the sentence, we solve multiclass classification in which one sentence is classified into one of the predefined templates. The challenge is to develop a deep, fast model that can understand the semantic meaning of the job posting text with a small number of labeled examples. We apply deep transfer learning [6] with Deep Averaging Network [18] to achieve both speed and accuracy. In terms of parameter entities, we used an in-house entity linking system to tag out mentions in the sentence and link them to the corresponding entities. For question ranking, we build an XGBoost pairwise ranking model to sort screening questions using extensive job and question features.

The contributions of this work are summarized as follows:

- To the best of our knowledge, this is the first work on the Screening Question Generation (SQG) task, which generates structured screening questions to help assess job applicants.
- We proposed and deployed the first SQG model, Job2Questions, to production to help millions of jobs finding qualified applicants and help hundreds of millions of members to identify qualified jobs.
- During offline evaluation, the proposed Job2Questions model improved the AUROC of both template classification and question ranking by 178% and 27.4%, respectively.
- Job2Questions significantly improved the online SQ suggestion quality by +53.10% acceptance rate and +22.17% job coverage. Jobs adopted SQ suggestions yielded 190% more recruiter-applicant interactions. These improvements increase the Net Promoter Score [35] by 11 points for recruiters who use Job2Questions.
- We conducted extensive analyses of the Job2Questions results and obtained exciting insights about the quality of member profile, requirements mentioned in the job posting, and different applicant screening focuses across the job marketplace.

2 RELATED WORK

Rule-based Question Generation. Rule-based models usually transform and formulate the questions based on the text input using a series of hand-crafted rules. ELIZA [39] generates question responses for conversations using human-made, keyword-based rules.

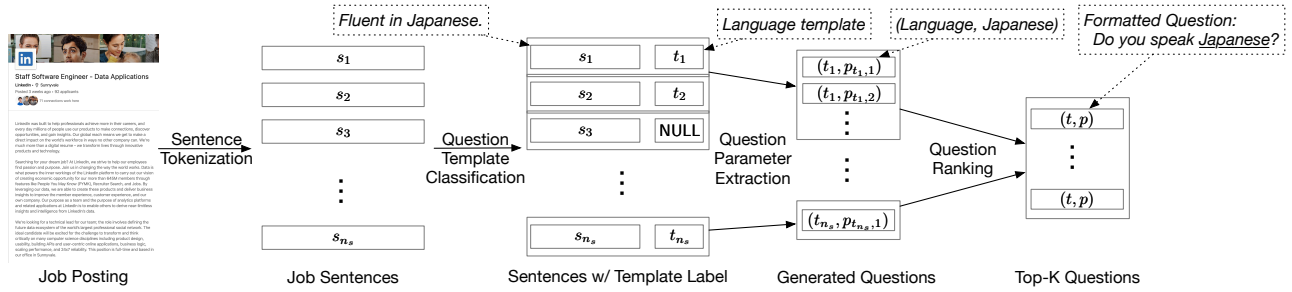

Figure 2: Overview of the Screening Question Generation task and its sub-tasks.

Mitkov uses language patterns and WordNet to create multiple-choice questions [29]. Heilman introduced a rule-based model to generate reading comprehension questions [15]. Despite the high precision, these rule-based models are not scalable and require immense human efforts and domain-specific expertise, which is not suitable for our case where it is hard to conclude patterns for jobs from different industries.

Context-only Neural Network Question Generation. Recently, neural network models that generate questions based on the given context have shown promising results with little human intervention. Du et al. [9] and Chali et al. [3] employed Seq2Seq model with attention mechanism for question generation tasks and trained the model in an end-to-end fashion. These methods are not designed for generating screening questions from job posting text for applicant assessment, which is different from traditional reading comprehension question generation tasks. Moreover, the unstructured, freeform questions generated by Seq2Seq models often do not have clear-cut-intent and too vague for job posters and downstream machine learning models to interpret and categorize. Lastly, the latency of Seq2seq models is also less ideal for fast online inference.

Answer-aware Neural Network Question Generation. To simplify the question generation task and handle cases where one input sentence maps to multiple questions, researchers proposed answer-aware question generation models where both the context and the answer of the target question is known. Tang et al. [37] improved the Seq2Seq model with a global attention mechanism that leverages additional answer information in the postprocessing step by replacing out-of-vocabulary words with the word in the answer having the highest relevance score among the attention distribution. Zhou et al. [41] proposed a Seq2Seq model that takes both the context and answer as input and generates questions using answer words-copying [12]. Sun et al. [36] further empower the QG model by explicitly leveraging the answer embedding and modeling the distance between the answer and the context. Gao, et al. [11] extend the traditional QG problem to Difficulty-controllable Question Generation (DQG) in which questions were generated based on the difficulty level designated given one pair of the context and the answer. However, the answer is not available for the screening question generation task, which means these methods are not applicable to our case.

Person-Job fit. Unlike recommender systems [16, 21, 31] that recommend items by predicting member actions, our work is more related to person-job fit, which aims at identifying if an applicant is qualified for the job. Recently, DuerQuiz [33] is proposed to create in-depth skill assessment questions to test if the applicant is good at certain hard skills such as Machine Learning. But it does not consider other general qualifications that are crucial for applicant screening, such as education background, work authorization, year-of-experience, or soft skills. APJFNN [32] is developed to predict person-job fit by comparing the job description and applicantsâĂŹ work experience in their resume. APJFNN does not have good explainability because it does not attribute the decision to a single requirement. Other resume-based methods [26, 42] may return sub-optimal assessment if certain information is missing in the resume. In general, none of these person-job fit models have studied the task of modeling person-job fit by generating explicit screening questions from job postings.

3 PROBLEM STATEMENT

The Screening Question Generation (SQG for short) task aims to generate screening questions from the job posting text. Job applicants will provide their answers to these questions during the job application. Based on the answers the applicants provide, the recruiters will identify qualified candidates. LinkedIn will also recommend other jobs to the applicants by matching their answers to the screening questions of other jobs and identify the ones that the applicants are qualified for.

At LinkedIn, we decide to use structured screening questions in the form of (*template*, *parameter*) instead of freeform text. For example, we use (*How many years of work experience do you have using*, *Java*), with the first part of the pair being template and the second part being parameter. We do it for the following reasons:

- **Structured questions ensure question quality**. By predefining the question types and possible parameters, we can ensure the screening question is unambiguous and reduce the chance of introducing inappropriate questions;
- **Structured questions have clear intent**. Unlike freeform text questions, the intent of structured questions are strictly defined by the question template. Therefore, job posters can easily group and screen candidates based on certain intent, e.g. education background, experience in multiple industries, or the list of tools they are familiar with.
- **Structured questions standardize questions across jobs**. By limiting screening questions to have pre-defined templates and parameters, questions from different jobs will have exactly the same representation. This property makes it possible for us to

recommend jobs that the applicants may be qualified for by comparing their answers to other jobsâĂŹ screening questions.

Based on the above three reasons, here we define the SQG task as inferring structured screening rather than free form questions from job posting text.

Definition 1. Screening Question Generation. Given the text of a job posting $j = \{w_1, \ldots, w_{n_w}\}$, where w represents words in the job and n_w denotes the total number of words in the job. **Screening Question Generation (SQG)** returns k top-ranked structured screening questions $\{(t, p)|t \in \mathbf{T}, p \in \mathbf{P}_t\}$, where $\mathbf{T}$ is a set of pre-defined templates, and $\mathbf{P}_t$ is a set of pre-defined parameters used by template t.

For example, given a job posting of Staff Software Engineer - Data Applications posted by LinkedIn, the Screening Question Generation model should return a list of screening questions in the format of template and parameter pairs such as (*Have you completed the following level of education*, *BachelorâĂŹs Degree*) and (*How many years of work experience do you have using*, *Java*).

However, designing a SQG model that can generate screening questions using the whole job posting as input is challenging. The job postings are longer and much noisier compared to the SQuAD [34] dataset, where each passage is relatively short and only focuses on one topic. As we shown in Tab. 1, the average length of a job posting is four times longer than the SQuAD passages. Moreover, job postings usually cover a wide range of topics, including company description, job functions, benefits, compensation, schedules, disclaimers, and many others that are not related to the screening question generation process.

Inspired by the fact in Question Generation (QG for short) tasks that the majority (99.73% [9]) of the questions could be derived from a single sentence, we hypothesize that the SQG problem can also be modeled by a sentence-level model.

Based on the above assumption, we propose a four-component sentence-level SQG framework and illustrate it in Fig. 2. As shown in Fig. 2, we first tokenize the given job posting into sentences, and then we run a question template classification model to detect the most probable template for each sentence. For every sentence that has a valid, non-NULL template, we first use the template-dependent parameter extractor to extract possible parameters, and then construct a list of screening question candidates using the extracted parameters and the template. Lastly, we aggregate all question candidates and use a question ranking model to pick k top-ranked template-parameter pairs as the final suggested screening questions for the given job posting. Next, we will provide the formal definition of the sentence-level SQG task and its sub-tasks.

Definition 2. Sentence-level Screening Question Generation. Given job posting $j = \{s_1, \ldots, s_{n_s}\}$, **Sentence-level Screening Question Generation** extracts a list of screening question candidates $\mathbf{Q}_j = \bigcup_{s_i \in j}\{(t, p)|t = \mathrm{TC}(s_i), p \in \mathrm{PE}(s_i, t)\}$, where TC is some question template classification model, and PE is some parameter extraction model, and output k top-ranked structured screening questions $\{(t, p)|t \in \mathbf{T}, p \in \mathbf{P}_t\} = \mathrm{QR}(\mathbf{Q}_j, j, k)$, where QR is some question ranking model.

In Def. 2, we enforce a one-to-one mapping between the sentence and the question template. Based on our observation, we find that the ratio of sentences that map to only one question is 88.9%. The rest 11.1% sentences, on the other hand, usually maps to multiple questions with the same template but different parameters. For example, "*4+ years experience programming experience in Java and C/C++*" can be converted into two screening questions with the same template (*How many years of work experience do you have using*, *Java*) and (*How many years of work experience do you have using*, *C/C++*).

Table 3: Screening Question Generation dataset statistics.

Task	Train	Test
Question Template Classification	7,053	3,648
Question Ranking	88,354	22,055

Next, we will formally define three sub-tasks of the sentence-level SQG task, namely question template classification (TC), template parameter extraction (PE), and question ranking (QR).

Definition 3. Question Template Classification. Given a sentence $s = \{w_1, \ldots, w_{n_w}\}$ from job posting j where w are word tokens, **Question Template Classification** (TC) predicts the question template $t_s \in T$ of s or NULL if s does not match any template.

Definition 4. Template Parameter Extraction. Given a sentence $s = \{w_1, \ldots, w_{n_w}\}$ from job posting j and predicted template t_s, **Template Parameter Extraction** (PE) extracts a list of possible parameter values $\mathbf{P}_{s,t_s}$ from s with respect to template t_s.

Note that for a given sentence s, Def. 3 may return NULL if s should not be converted into any screening question. Note that SQG is different from the traditional QG settings where the input passage always maps to one or more questions [10, 40]. In SQG, a large portion of the sentences in the job posting is irrelevant to the qualification evaluation of an applicant, and therefore should not be converted into screening questions.

After getting the screening question candidate set $\mathbf{Q}_j$, we use a question ranking model to rank all the questions and return the top-k as the generated screening questions of job posting j.

Definition 5. Question Ranking. Given a list of screening question candidates $\mathbf{Q}_j$, **Question Ranking** (QR) ranks them into an ordered list based on $\Pr(\text{accepted}|(j, t, p))$, the probability that job posters will add screening question (t, p) to job j.

In the following sections, we will describe the data collection strategy and the model design of our proposed sentence-level SQG model, Job2Questions.

4 DATA PREPARATION

In this section, we will describe two methods we used, namely crowdsourcing and user feedback, to collect training and high-quality evaluation data for the template classification and question ranking tasks. Note that we leverage our existing, in-house entity linking system as the parameter extractor, therefore we will omit the data preparation for the parameter extraction component in this section. The statistics of the two datasets we collected in this section is described in Tab. 3.

Table 4: Examples of the crowd sourcing annotation task.

Is the given sentence from job description directly related to the given screening question?
Sentence from job description: Post-graduate or PhD in Computer Science or Machine Learning related degree with a focus on NLP; **Screening Question**: Have you completed the following level of education: Ph.D.?
Is the given sentence from job description directly related to the given screening question?
Sentence from job description: Performing annual and periodic Fair Lending and UDAAP analysis and reporting utilizing CRA Wiz and R Studio . **Screening Question**: How many years of work experience do you have using R?

4.1 Question Template Classification

Given a sentence s, the first step of SQG is to predict its question template t. In order to train a template classification model, we need to collect sentence-template (s, t) pairs.

We performed a crowdsourcing task to collect labeled sentence-template pairs. Tab. 4 shows two examples of the crowdsourcing annotation task we designed to collect labeled data. The sentences and screening questions are generated as follows: we first recognize entities from all sentences, and collect a list of sentences that contain valid parameter entities; then for each sentence-parameter (s, p) pair, we generate a screening question (t, p) for s, where p can be used as t's parameter ($p \in P_t$); lastly, we randomly sample a subset of these generated (s, t, p) triples, convert them into the format shown in Tab. 4, and ask human annotators to label these sentence-question pairs. We consider the sentence-template pair (s, t) pair is positive if the human labeler labels at least one triple from $\{(s, (t, p))|p \in \mathbf{P}_t\}$ as directly related. Otherwise we consider that sentence s maps to NULL template (s, NULL).

4.2 Question Ranking

As shown in Fig. 2, once we have a list of screening question candidates, the next step is to rank them and pick the top-k ranked questions as the output of the SQG model. The objective of question ranking is to predict the probability of a screening question (t, p) been added to job posting j by the job poster. To train such a ranking model, we need to collect the corresponding (j, t, p) triples for model training.

Although we can ask job posters to manually add screening questions to jobs, such approach only provides positive labeled triples. The challenge for question ranking data collection is the lack of negative labeled data. The random generated negative data are easy to separate and cannot help improve the model performance. Another approach is to randomly pick auto-generated screening questions that do not match manual-added questions from job posters as negative labeled data. Such negative labeled triples may have a high false-negative rate: job posters did not add such questions manually because they simply forgot about it.

Therefore, to collect high-quality question ranking data, we need to explicitly ask job posters to provide negative labeled data. In this work, we first designed a simplified sentence-level SQG model which is described in Sec. 6.1 as the BOW-XGB model, deployed it in production to provide screening question suggestions to job posters, and then collect the labeled question ranking triples using job posters' feedbacks. Namely if a job poster accepts a suggestion or adds a new screening question, we generate a positive labeled (j, t, p) triple. If job poster rejects a screening question suggestion, we generate a negative labeled (j, t, p) triple accordingly. We collected 110,409 labeled triples and group them into two triple sets $(j, t, p) \in D^+$ and $(j, t, p) \in D^-$, where D^+ contains all positive-labeled data and D^- is a triple set of negative-labeled data.

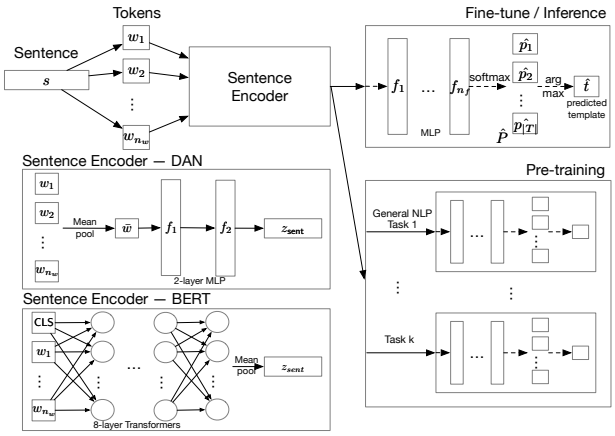

Figure 3: Question Template Classification model. We first pretrain Sentence Encoders (DAN/BERT) with general NLP tasks [7, 18], then conduct task-specific fine-tuning. During inference, the model takes sentence as input and use sentence encoder + task-specific MLP to predict template.

5 JOB2QUESTIONS

After describing the problem formulation and data preparation process for the sentence-level screening question generation (SQG) task, here we describe the detailed design of our production sentence-level SQG model, the Job2Questions model. We will describe its three core components as shown in Fig. 2, namely the question template classification, question parameter extraction, and question ranking.

5.1 Question Template Classification

As the first component of the Job2Questions model, question template classification takes a raw sentence as input and predicts its most probable template label, or NULL if it is not eligible. Here in this work, we treat this task as a multiclass classification task and consider template labels and the non-eligible NULL as classes. The overview of the question template classification component is shown in Fig. 3.

As shown in Fig. 3, for a given sentence s, we first tokenize it into word tokens $\{w_1, \ldots, w_{n_w}\}$, and then use a sentence encoder to convert the tokens into a sentence embedding vector z_{sent}. The generated sentence embedding vector z_{sent} is then sent to a neural network model to predict its most probable class label $\hat{t}$.

Because the training data set (around 8k) is relatively small compared to tens of millions of jobs posted on LinkedIn, it definitely does not contain all the words in the vocabulary and does not cover all the creative ways recruiters describe job requirements. To address this issue, we decide to utilize multi-task transfer learning [30] to pre-train the sentence encoding model with multiple natural language understanding tasks, and then use transfer learning to

fine-tune the trained model with our question template classification task. As shown in Fig. 3, we first pre-train the sentence encoder with general tasks, and then fine-tune the sentence encoder with the task-specific MLP using our template classification data.

Here we propose two methods to encode sentence into embeddings, a simple and fast Deep Average Network (DAN) [18] model and a more advanced Deep Bidirectional Transformers (BERT) [7] model. We choose DAN due to its simplicity and competitive performance compared to relative computational expensive models such as CNN [19] and LSTM [27] models. We also used BERT as our sentence encoder to see how advanced NLP model can help improve the performance in this specific task.

The DAN model first average the embedding of the input tokens into $\bar{w} = \frac{1}{n_w}\sum_i^k w_i$, and then pass it through two fully-connected layers to get the sentence representation $z_{\text{sent}} = \sigma(\sigma(\bar{w}W_1 + b_1)W_2 + b_2)$.

BERT, on the other hand, uses Transformer layer [38] to encode the input sentence to embedding. It is defined as follows

$$\begin{aligned}
\text{TFLayer}(h^{n-1}) &= \text{FC}(\text{MultiAttn}(h^{n-1}));\\
\text{FC}(x) &= \text{relu}(xW_1 + b_1)W_2 + b_2;\\
\text{MultiAttn}(h^{n-1}) &= \text{concat}(head_1(h^{n-1}), \ldots, head_k(h^{n-1}))W^O;\\
\text{head}_i(h^{n-1}) &= \text{softmax}\left(\frac{(h^{n-1}W_q^i)(h^{n-1}W_k^i)}{\sqrt{d_k}}\right)(h^{n-1}W_v^i).
\end{aligned} \tag{1}$$

where h^{n-1} is the output of the previous Transformer layer. Here we use a BERT model with 8 Transformer layers, and define the output sentence embedding z_{sent} as the meanpooling result of the last transformer layer's output. For simplicity, we omit batch normalization [17] and residual connections [14] in the equations.

After we obtain the sentence embedding z_{sent}, we then pass it through a multilayer perceptron network (MLP) where each fully-connected layer is defined as $f(x) = \text{relu}(xW + b)$, and the last layer of the MLP is defined as $\hat{P} = \text{softmax}(f(x)W + b)$, where the output $\hat{P}$ is the categorical probability distribution of each class. Finally, we pick the most probable class $\arg\max(\hat{P})$ as the final predicted template label. To train the model, we use a binary-cross entropy loss

$$\mathcal{L}(P, \hat{P}) = -\sum_i p^i \log \hat{p}^i, \tag{2}$$

where P is the ground truth, p^i and $\hat{p}^i$ are the ground truth and predicted probability of i^{th} template respectively. We use the Adam optimizer [23] to optimize the model parameters.

5.2 Question Parameter Extraction

Given a job posting, we first break it down into sentences and get each sentence's template label using the above question template classification model. For sentences that have a valid non-NULL template and require a parameter, we call our in-house entity linking system to detect the corresponding template parameter by tagging specific types of entities from the given sentence in real-time. Note this work is not about designing the entity linking system so we will only give a brief overview of the system below.

To find template parameters, the system first tag possible entity mentions from sentences. We utilize an in-house, comprehensive entity taxonomy that contains large sets of entity surface forms to identify possible entity mentions from the given text. In our current production model, we support four types of entities, namely education degrees, tool-typed skills, spoken languages, and credentials (certifications and licenses).

After we identified entity mentions from job sentences, we then use a feature-based regression model to link the mention to an entity in the taxonomy. Besides global features such as mention frequency, we also employed many contextual features such as POS tag, context n-grams, and the cosine similarity between the FastText [20] embeddings of the mention and its context. These contextual features help our model to identify invalid mentions such as Bachelor's degree in "*We provide bachelor party supplies*" or Chinese language in "*Our clients include European and Chinese companies*". Here we choose FastText instead of other methods such as LSTM [25] or charCNN [28] because FastText has a lower latency and works reasonably well for identifying and linking the four entity types we listed above.

Finally, entity mentions with a confidence score that passes the given threshold will be considered as template parameters of the given sentence s and template t_s.

5.3 Question Ranking

After we get all the question candidates in the format of template label and parameter pairs (t, p) from the given job posting, the next step is to rank the candidates and find questions that are helpful for the hiring process. Because determining whether or not a screening question is helpful for the hiring process is non-trivial, in this work we rely on recruiters and job posters to label what screening questions are best for the hiring process. Hence, we use the screening questions added and rejected by them as the ground truth labels and define the question ranking objective as predicting the likeliness of a job poster adds a screening question candidate (t, p) to a job posting j.

$$\Pr(\text{accepted}|j, t, p) = sigmoid(f(x_{j,t,p})), \tag{3}$$

where f is the scoring function, $x_{j,t,p}$ is the feature vector with respect to the given job j, template label t, and parameter p.

The features we used to construct $x_{j,t,p}$ can be group into three groups, job-side features, question-side features and job-question interactive features.

Job-side features: Job attributes such as job's title, industry, company, location, and others. We use 27 different features to represent jobs.

Question-side features: Screening question attributes such as question template type, parameter value, template classification score, and entity linking system's confidence score. We use 5 features to represent questions.

Job-Question interactive features: We generate interactive features by computing the Pointwise Mutual Information (PMI) between job- and question-side features. The PMI is defined as follows:

$$\text{PMI}(F_j; F_q) = log\frac{\Pr(F_j, F_q)}{\Pr(F_j)\Pr(F_q)}, \tag{4}$$

where F_j and F_q are the job- and question-side categorical features respectively. Here we use PMI value to quantify the discrepancy between the probability of correspondence of a job-side and a question-side event given both joint and individual distribution. In total we use 135 interactive features in our question ranking model.

Table 5: Question template classification offline evaluation.

Model	Overall Acc.	NULL		Work Auth		Sponsorship		Education		Language		Credential		Tools	
		Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.
BOW-XGB	0.3547	0.1336	**0.5988**	0.4118	0.1386	0.5000	0.0051	0.5644	0.8519	**0.9888**	0.3346	-	-	0.5329	0.4863
J2Q-TC-CNN	0.8640	0.5408	0.2536	0.9397	0.9639	0.9488	0.9915	0.9199	0.8857	0.9328	0.9680	0.6865	**0.9227**	0.9037	0.8151
J2Q-TC-NNLM	0.8765	0.6250	0.2871	0.9425	0.9716	0.9514	0.9915	0.8984	0.9343	0.9598	0.9710	0.7019	0.9040	0.9193	0.8374
J2Q-TC-DAN	0.8798	0.6330	0.3301	0.9333	**0.9742**	0.9538	0.9887	0.9056	0.9314	0.9648	0.9564	0.7227	0.8827	0.9068	0.8664
J2Q-TC-BERT	**0.9138**	**0.6688**	0.4928	**0.9592**	0.9691	**0.9592**	**0.9944**	**0.9282**	**0.9600**	0.9655	**0.9767**	**0.8564**	0.8747	**0.9197**	**0.9465**

After describing the feature vector $x_{j,t,p}$, next we present the scoring function f. Here we use XGBoost as the scoring function and therefore rewrite Eq. 3 as

$$\Pr(\text{accepted}|j,\ t,\ p) = sigmoid\left(\sum_k f_k(x_{j,t,p})\right) \tag{5}$$

where f_k is the k^{th} tree of the model. We use the following loss function to optimize the question ranking model

$$\begin{aligned}\mathcal{L} = &- \sum_{\langle j,t,p\rangle \in \mathrm{D}^+} \log\left(\sum_k f_k\left(x_{j,t,p}\right)\right) \\ &- \sum_{\langle j,t,p\rangle \in \mathrm{D}^-} \log\left(1 - \sum_k f_k\left(x_{j,t,p}\right)\right) + \sum_k \Omega(f_k),\end{aligned} \tag{6}$$

where D^+ and D^- are the positive and negative (j,t,p) triple sets collected using the job postersâĂŹ feedback described in Sec. 4.2. f_k represents the k^{th} tree in the boosted-tree model, $\Omega(f_k) = \gamma T + \frac{1}{2}\lambda||\mathbf{w}||^2$ is the regularization term that penalizes the complexity of tree f_k, in which T denotes the number of leaves in tree f_k, $\mathbf{w}$ is the leaf weights, γ and λ are the regularization parameters.

6 EXPERIMENTS

In this section, we conducted extensive evaluations on the proposed Job2Questions (J2Q for short) model. The promising offline and online A/B test results demonstrate the effectiveness of the proposed Job2Questions model in terms of providing high quality screening question suggestions, helping recruiters identify qualified applicants, suggesting qualified jobs to members, and boosting recruiter-applicant interactions. The large-scale case studies on Job2Questions result also reveal many interesting insights that help us better understand the job marketplace.

6.1 Experiment Setting

The evaluated question template classification models are:

- **BOW-XGB**: A non-neural network baseline which tokenize the input sentence into bag-of-word vectors and then trained an XGBoost [5] model to predict the template label.
- **J2Q-TC-NNLM**: J2Q template classification model which uses a simple feed-forward neural network language model (NNLM) [1] as the sentence encoder.
- **J2Q-TC-DAN**: J2Q template classification model with deep averaging networks [18] as the sentence encoder.
- **J2Q-TC-CNN**: J2Q template classification model with CNN-based universal sentence encoder [19] as the sentence encoder.
- **J2Q-TC-BERT**: J2Q template classification model with BERT [7] as the sentence encoder.

All above models are trained using the same dataset as described in Tab. 3. For neural network J2Q-TC-⋆ models, we use the public-available pre-trained models [2, 7] as initialization and fine-tune them accordingly. For J2Q-TC-{NNLM, DAN, CNN}, we set the learning rate to $1e-3$, batch size to 256, and drop-out rate to 0.4. For J2Q-TC-BERT, we further truncate the input sentence to 32 tokens and set the learning rate to 5e-5. All models are trained for at most 100 epochs with a 3-layer MLP.

We also evaluated the following question ranking models:

- **Rule-based**: A non-machine learning baseline. It sorts questions based on the template classification model score and re-ranks them using business rules, e.g. education and work authorization questions always rank at the top.
- **J2Q-QR-LR**: A logistic regression model that ranks candidates using job and question side features.
- **J2Q-QR-XGB-pointwise**: The proposed question ranking model trained using a pointwise loss.
- **J2Q-QR-XGB-pairwise**: The proposed question ranking model trained using a pairwise loss.

The J2Q question ranking model is trained with 110,409 labeled (j,t,p) triples collected via job posters feedback. We divide the data into $70-20-10$ training, evaluation, and validation sets. We explored and anchored the XGBoost hyper-parameters as follows: number of trees is 100, depth is 5, η is 0.7, and γ is set to 0.

6.2 Offline Evaluation

6.2.1 Question Template Prediction. We evaluate the question template classification performance on the crowdsourced dataset described in Tab. 3, which contains 6 different template labels and a special label NULL for sentences that cannot be convert into SQ. We report the precision/recall of each template label and the overall accuracy in Tab. 5. We found that all deep learning models outperformed the baseline by a large margin, and BERT yields the best overall accuracy, outperforming the second best DAN model by 3.9%. However, as shown in Tab. 2, the CPU inference time of BERT (100ms) is 10-times longer than the DAN model (9ms), making it less practical for online CPU inference.

6.2.2 Hyper-parameter Test. To understand how different hyper-parameters affect model performance, we tested different configurations on the learning rate, dropout rate, batch size, and the number of MLP layers. As shown in Fig. 4, the performance of the J2Q-TC-DAN model is mostly insensitive to its hyper-parameters besides the dropout rate, which is expected. Using a larger batch size slightly hurts the performance of our model due to overfitting [22]. Lastly, we found that using three-layer MLP works the best for our question template classification task.

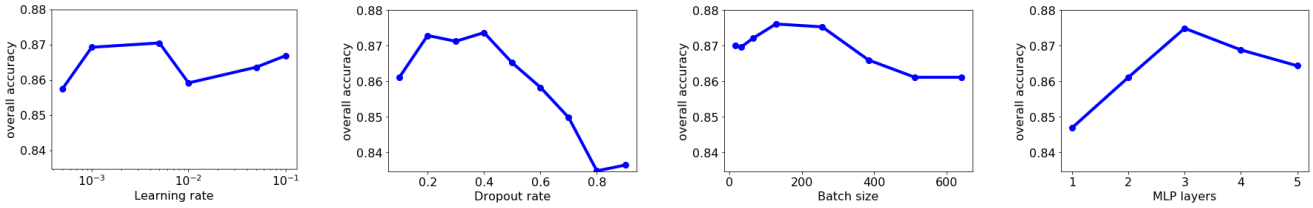

Figure 4: Hyper-parameter sensitivity test for the in-production J2Q-TC-DAN model.

Table 6: Question ranking offline evaluation.

Model	AUROC	Precision		Recall		NDCG	
		@1	@3	@1	@3	@1	@3
Rule-based	0.6408	0.6795	0.4505	0.5499	0.8967	0.6795	0.8075
J2Q-QR-LR	0.8008	0.8205	0.4864	0.6719	0.9629	0.8205	0.9078
J2Q-QR-XGB-Pointwise	**0.8282**	0.8325	0.4876	0.6818	0.9650	0.8325	0.9136
J2Q-QR-XGB-Pairwise	0.8164	**0.8428**	**0.4897**	**0.6910**	**0.9672**	**0.8428**	**0.9194**

Table 7: Question ranking feature ablation study.

Model	AUROC	Precision		Recall		NDCG	
		@1	@3	@1	@3	@1	@3
J2Q-QR-XGB-Pairwise	0.8164	**0.8428**	**0.4897**	**0.6910**	**0.9672**	**0.8428**	**0.9194**
– no job feat.	**0.8170**	0.8374	0.4889	0.6861	0.9661	0.8374	0.9167
– no question feat.	0.8077	0.8287	0.4858	0.6801	0.9622	0.8287	0.9104
– no interaction feat.	0.8148	0.8414	0.4879	0.6889	0.9650	0.8414	0.9168

6.2.3 Question Ranking. We evaluated four question ranking models listed in Sec. 6.1 using 22,055 (job, template, parameter) triples from 6,675 jobs and report the Area Under the Receiver Operating Characteristic curve (AUROC), Precision@k, Recall@k, and Normalized Discounted Cumulative Gain at k (NDCG@k). As shown in Tab. 6 the proposed J2Q-QR-XGB-pairwise model outperforms other baselines with up to 24.03% improvement in NDCG. This significant improvement indicates the effectiveness of the proposed model on predicting job poster actions on screening questions. Based on the observation, we chose the pairwise J2Q model as our online question ranking model due to its great ranking performance.

6.2.4 Ranking Feature Ablation Study. In this experiment, we study which feature group contributes the most to the question ranking model. In Tab. 7, we reported the AUROC and NDCG@k of different J2Q-QR-XGB-pairwise models trained with one group of the features removed. The results show that all feature groups positively contribute to the model and should be retained. Note that adding interaction features only improve the performance marginally. This is because tree-based model can capture some feature correlations without explicit signal.

6.3 Online Evaluation

6.3.1 Screening Question Suggestions. When posting jobs on LinkedIn, posters can manually add screening questions to jobs. Here we deployed two template classification mdoels, BOW-XGB and J2Q-TC-DAN, online to provide SQ suggestions to posters. We ramped each model to 50% of LinkedIn's traffic for two weeks and compared them against a baseline model, which simply extracts parameters from job posting and create questions regardless the sentence intent. The metrics we tracked are as follows:

- **Acceptance Rate**: #accepted SQs / #suggested SQs,
- **Suggestion Rate**: #jobs receive SQ suggestions / #jobs on LinkedIn,
- **Adoption Rate**: #jobs accepted SQ suggestions / #jobs on LinkedIn.

Table 8: Screening question suggestion online A/B test.

Model	Acceptance Rate	Suggestion Rate	Adoption Rate
BOW-XGB	+31.06%	+40.86%	+14.26%
Job2Questions	**+53.10%**	**+59.92%**	**+22.17%**

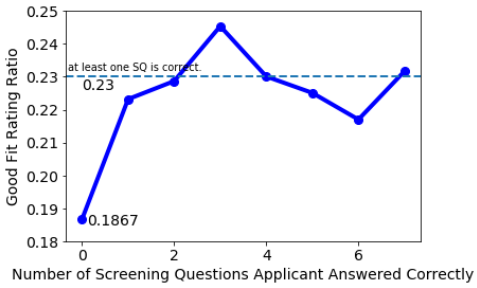

Figure 5: Applicants' good fit rating versus the number of screening questions they answered correctly. The horizontal line denotes the average good fit rating ratio of applicants who answered at least one question correctly.

As shown in Tab. 8, the proposed Job2Questions model significantly improves both precision and coverage. We believe the acceptance improvement is due to the expressive power of neural network models on modeling job posting semantics. The increase in coverage, on the other hand, is because the deep transfer learning gives good generalization ability to our model so it can handle job postings written in different styles.

6.3.2 Job Applicant Quality. After the proposed Job2Questions model is deployed, we also analyzed how providing screening questions can help improve the hiring efficiency in terms of job applicant quality. We first look at how screening questions help identify qualified applicants in around 20M job applications. As shown in Fig. 5, we found that if the applicant does not answer any screening questions correctly, only 18.67% of the applications are rated as a good fit by the recruiter. But if the applicant answered at least one screening question correctly, the good fit rating increases to 23% (+23.19%).

6.3.3 Hiring Efficiency. Because recruiters often receive a large amount of applications per job, it is impractical for them to review all of them. Therefore they tend to sort the applicants first and only review the top-ranked candidates. Here at LinkedIn we sort the candidates by predicting if an applicant is a good fit. As an alternative, we developed another ranking model that simply sort applicants based on the favorableness of their answers to the screening questions. We conducted a 50-50 A/B test for one month and measured the good/bad fit rating that recruiters gave to the candidates they contacted. We found that ranking applicants by screening question

Table 9: Relationships between SQ answers and user profile.

	Edu.	Lang.	Tools
Matches Profile (%)	64%	29%	61%
Not in Profile (%)	33%	70%	37%
Conflicts with Profile (%)	3%	1%	2%

Table 10: Question rejection rate case study.

Question	Type	Rej. rate
How many years of work experience do you have using Fax?	Tool	99.10%
How many years of work experience do you have using Internet Explorer?	Tool	98.75%
Have you completed the following level of education: master's degree?	Edu.	77.68%

Table 11: Per-industry SQ type distribution.

Industry	Cred.	Edu.	Lang.	Sponsor	Tools	Work Auth.
Agriculture	3.46%	**36.88%**	**28.03%**	2.94%	21.47%	7.22%
Government	8.99%	30.70%	15.72%	**4.25%**	26.47%	13.86%
Technology	0.79%	3.82%	2.31%	0.73%	**90.81%**	1.54%
Transportation	**15.47%**	25.54%	19.95%	3.95%	20.94%	**14.14%**
Overall	4.96%	10.89%	6.39%	1.45%	72.47%	3.84%

answers can improve the applicant good fit rate by 7.45% and reduce the bad fit rate by 1.67%. This means the screening questions can help the recruiters surface qualified applicants and therefore improve the hiring efficiency.

6.3.4 Screening Question-based Job Recommendation. We also conducted applicant-side analysis to see if SQs can help applicants apply for jobs they are qualified for. We applied our Job2Questions model to all jobs on LinkedIn and retrained our job recommendation model (JYMBII [21]) using SQs and applicant answers as additional features. We observed that when LinkedIn members applying for jobs suggested via email, they are 46% more likely to get a good fit rating if the job is suggested by the JYMBII + SQ model.

6.3.5 Increased Interactions and Satisfactions. By providing SQ suggestions to recruiters, we observed boosted positive interactions. Namely jobs with SQs yield 1.9x more recruiter-applicant interactions in general and 2.4x more interactions with screening-qualified applicants. Moreover, the Net Promoter Score (NPS) [35] is 11 points higher for recruiters using SQs than those who donâĂŹt.

6.4 Case Studies and Insights

6.4.1 SQ Answers Complement Member Profile. To verify our hypothesis that member profile is not an ideal data source for job-applicant fit measurement, we compared the member profile and their screening question answers. We found that screening questions often contains information that members do not put in their profile. In Tab. 9, we can see that among members who answered screening questions, 33% of the members do not provide their education information in their profile. More specifically, people who hold secondary education degree are less likely to list that in their profile. As for languages, 70% of the members do not list the languages they spoke (mostly native speakers) in their profile. Lastly, 37% of the members do not include experience with specific tools, e.g. *Salesforce Sales Cloud*, *Adobe Design Programs*, or *Google Ads*, in their profile. In short, we suspect that when people composing their professional profile, they tend to overlook basic qualifications which recruiters value a lot during screening. Therefore, screening questions are much better, direct signals for applicant screening compared to member profile.

6.4.2 Job Postings Are Noisy. Because our Job2Questions model generate questions for explicitly mentioned requirements only, we can identify requirements that recruiters think trivial or unnecessary by finding suggested SQs with top rejection rate, i.e. requirements mentioned in the job posting but not important enough to be actual screening questions. Tab. 10 shows top-3 SQs with the highest rejection rate. We found that although recruiters explicitly mention requirements such as "*Access to computer with scanning, printing and **faxing** capabilities*" or "*Good working knowledge of **Internet Explorer***", more than 98% of the cases recruiters do not screen applicants based on these. We suspect recruiters do not update job postings frequently, therefore it sometimes contain outdated requirements that are too trivial to be used for screening. Another interesting finding is that job postings often contain unnecessary requirements such as degree requirements. Although job postings explicitly state requirements such as "*A Bachelor of Science or a Master Degree required*", recruiters usually screen applicants based on the lowest education requirement only. Based on these observations, we believe SQs are better than noisy job descriptions for modeling job requirements because they are more concise and reflect only the true needs of jobs.

6.4.3 SQ preferences across industries. Lastly, we found that different industries has different preferences or focus on screening candidates. In Tab. 11, we presented the overall SQ types used by all jobs posted on LinkedIn and per-industry breakdown of four example industries with interesting trends. For example, Agriculture industry is 4.4 times more likely to screening applicants based on language than other industries in general. Technology industry does not screening candidates based on education or language, instead 91% of the SQs are about tools they have used. Transportation industry does not require tools experience but are more likely to screen candidates by credentials such as driver's license or license to handle hazardous materials. Government and Transportation both ask a lot of work authorization and sponsorship questions probably because they usually do not sponsor working visas for foreigners but they do get a lot applicants who need sponsorship. By looking at SQ type distributions, we can better understand what each industry is looking for and how applicants can better position themselves by excelling in things employers value the most.

7 CONCLUSIONS AND FUTURE WORK

In this work, we proposed a novel Screening Question Generation (SQG) task that automatically generates screening questions for job postings. We also developed a general candidate-generation-and-ranking SQG framework and presented LinkedIn's in-production Job2Questions model. We provided design details of Job2Questions, including data preparation, deep transfer learning-based question template classification modeling, parameter extraction, and XGBoost-based question ranking. The extensive online and offline evaluations demonstrate the effectiveness of the Job2Questions model.

As for future work, we plan to infer SQs that are not explicitly mentioned in the job posting and investigate advanced question ranking methods to better model recruiter preferences. We also plan to investigate seq2seq models for template-free SQ generation..

REFERENCES

[1] Yoshua Bengio, RÃĳjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A Neural Probabilistic Language Model. *Journal of Machine Learning Research* 3, Feb (2003), 1137–1155.

[2] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Universal Sentence Encoder. *arXiv:1803.11175 [cs]* (April 2018). arXiv: 1803.11175.

[3] Yllias Chali and Tina Baghaee. 2018. Automatic opinion question generation. In *Proceedings of the 11th International Conference on Natural Language Generation*. 152–158.

[4] Guanliang Chen, Jie Yang, Claudia Hauff, and Geert-Jan Houben. 2018. LearningQ: A Large-Scale Dataset for Educational Question Generation. In *Twelfth International AAAI Conference on Web and Social Media*.

[5] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16*. ACM Press, San Francisco, California, USA, 785–794.

[6] Yu-An Chung, Hung-Yi Lee, and James Glass. 2018. Supervised and Unsupervised Transfer Learning for Question Answering. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Association for Computational Linguistics, New Orleans, Louisiana, 1585–1594.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv:1810.04805 [cs]* (Oct. 2018). arXiv: 1810.04805.

[8] Mamadou Diaby, Emmanuel Viennet, and Tristan Launay. 2013. Toward the next generation of recruitment tools: an online social network-based job recommender system. In *2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2013)*. IEEE, 821–828.

[9] Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to Ask: Neural Question Generation for Reading Comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, 1342–1352.

[10] Nan Duan, Duyu Tang, Peng Chen, and Ming Zhou. 2017. Question Generation for Question Answering. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, 866–874.

[11] Yifan Gao, Lidong Bing, Wang Chen, Michael R Lyu, and Irwin King. 2019. Difficulty controllable generation of reading comprehension questions. In *Proc. 28th International Joint Conference on Artificial Intelligence (IJCAI)*.

[12] Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. *arXiv preprint arXiv:1603.08148* (2016).

[13] Viet Ha-Thuc, Ye Xu, Satya Pradeep Kanduri, Xianren Wu, Vijay Dialani, Yan Yan, Abhishek Gupta, and Shakti Sinha. 2016. Search by ideal candidates: Next generation of talent search at linkedin. In *Proceedings of the 25th International Conference Companion on World Wide Web*. International World Wide Web Conferences Steering Committee, 195–198.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. 770–778.

[15] Michael Heilman and Noah A. Smith. 2009. *Question Generation via Overgenerating Transformations and Ranking:*. Technical Report. Defense Technical Information Center, Fort Belvoir, VA.

[16] Chao Huang, Xian Wu, Xuchao Zhang, Chuxu Zhang, Jiashu Zhao, Dawei Yin, and Nitesh V. Chawla. 2019. Online Purchase Prediction via Multi-Scale Modeling of Behavior Dynamics. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '19)*. Association for Computing Machinery, Anchorage, AK, USA, 2613–2622.

[17] Sergey Ioffe and Christian Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv:1502.03167 [cs]* (March 2015). arXiv: 1502.03167.

[18] Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal DaumÃĺ III. 2015. Deep Unordered Composition Rivals Syntactic Methods for Text Classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Beijing, China, 1681–1691.

[19] Xiaoqi Jiao, Fang Wang, and Dan Feng. 2018. Convolutional Neural Network for Universal Sentence Embeddings. In *Proceedings of the 27th International Conference on Computational Linguistics*. Association for Computational Linguistics, Santa Fe, New Mexico, USA, 2470–2481.

[20] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of Tricks for Efficient Text Classification. *arXiv:1607.01759 [cs]* (Aug. 2016). arXiv: 1607.01759.

[21] Krishnaram Kenthapadi, Benjamin Le, and Ganesh Venkataraman. 2017. Personalized Job Recommendation System at LinkedIn: Practical Challenges and Lessons Learned. In *Proceedings of the Eleventh ACM Conference on Recommender Systems (RecSys '17)*. Association for Computing Machinery, Como, Italy, 346–347.

[22] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. 2017. On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima. *arXiv:1609.04836 [cs, math]* (Feb. 2017). arXiv: 1609.04836.

[23] Diederik P. Kingma and Jimmy Ba. 2017. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]* (Jan. 2017). arXiv: 1412.6980.

[24] Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. RACE: Large-scale ReAding Comprehension Dataset From Examinations. *arXiv:1704.04683 [cs]* (Dec. 2017). arXiv: 1704.04683.

[25] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural Architectures for Named Entity Recognition. In *NAACL-HLT*. ACL, San Diego, California, 260–270.

[26] Ran Le, Wenpeng Hu, Yang Song, Tao Zhang, Dongyan Zhao, and Rui Yan. 2019. Towards Effective and Interpretable Person-Job Fitting. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 1883–1892.

[27] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A Structured Self-attentive Sentence Embedding. *arXiv:1703.03130 [cs]* (March 2017). arXiv: 1703.03130.

[28] Xuezhe Ma and Eduard Hovy. 2016. End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, 1064–1074.

[29] Ruslan Mitkov. 2003. Computer-aided generation of multiple-choice tests. In *Proceedings of the HLT-NAACL 03 workshop on Building educational applications using natural language processing*. 17–22.

[30] Sinno Jialin Pan and Qiang Yang. 2010. A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering* 22, 10 (Oct. 2010), 1345–1359.

[31] Ioannis Paparrizos, B. Barla Cambazoglu, and Aristides Gionis. 2011. Machine learned job recommendation. In *Proceedings of the fifth ACM conference on Recommender systems (RecSys '11)*. Association for Computing Machinery, Chicago, Illinois, USA, 325–328.

[32] Chuan Qin, Hengshu Zhu, Tong Xu, Chen Zhu, Liang Jiang, Enhong Chen, and Hui Xiong. 2018. Enhancing person-job fit for talent recruitment: An ability-aware neural network approach. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. ACM, 25–34.

[33] Chuan Qin, Hengshu Zhu, Chen Zhu, Tong Xu, Fuzhen Zhuang, Chao Ma, Jingshuai Zhang, and Hui Xiong. 2019. DuerQuiz: A Personalized Question Recommender System for Intelligent Job Interview. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2165–2173.

[34] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, 2383–2392.

[35] Frederick F Reichheld. 2003. The one number you need to grow. *Harvard business review* 81, 12 (2003), 46–55.

[36] Xingwu Sun, Jing Liu, Yajuan Lyu, Wei He, Yanjun Ma, and Shi Wang. 2018. Answer-focused and position-aware neural question generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 3930–3939.

[37] Duyu Tang, Nan Duan, Tao Qin, Zhao Yan, and Ming Zhou. 2017. Question answering and question generation as dual tasks. *arXiv preprint arXiv:1706.02027* (2017).

[38] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, ÅĄukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.). Curran Associates, Inc., 5998–6008.

[39] Joseph Weizenbaum. 1966. ELIZA—a computer program for the study of natural language communication between man and machine. *Commun. ACM* 9, 1 (1966), 36–45.

[40] Yao Zhao, Xiaochuan Ni, Yuanyuan Ding, and Qifa Ke. 2018. Paragraph-level Neural Question Generation with Maxout Pointer and Gated Self-attention Networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Brussels, Belgium, 3901–3910.

[41] Qingyu Zhou, Nan Yang, Furu Wei, Chuanqi Tan, Hangbo Bao, and Ming Zhou. 2018. Neural Question Generation from Text: A Preliminary Study. In *Natural Language Processing and Chinese Computing (Lecture Notes in Computer Science)*. Springer International Publishing, Cham, 662–671.

[42] Chen Zhu, Hengshu Zhu, Hui Xiong, Chao Ma, Fang Xie, Pengliang Ding, and Pan Li. 2018. Person-Job Fit: Adapting the Right Talent for the Right Job with Joint Representation Learning. *ACM Transactions on Management Information Systems (TMIS)* 9, 3 (2018), 12.