Transformer-based solutions using transfer learning and instruction fine-tuning conditional on context input data for downstream NLP tasks in the domain of job application pain points

Stanford CS224N Custom Project

Aris Aristorenas
Department of Computer Science
Stanford University
`aaristorenas@stanford.edu`

Abstract

The increasing availability of pretrained transformer models has made transfer learning a more viable solution towards downstream NLP tasks. Fine-tuning a pretrained model on data examples directly related to the downstream task can result in better performance with less labels because of the size of the corpus the pretrained model was trained under (typically in the range of several billions of tokens). In this study, we assess this claim by considering a set of downstream NLP tasks within the domain of job application pain points. In identifying these pain points, five NLP tasks were developed with the underlying goal to offload the manual effort job applicants face in the end-to-end job application process. Tasks 1 and 2 were solved using a masked language model (BERT and DistilRoBERTa) with respective accuracies of 78.3%, and 48% obtained in the fine-tuned models. Tasks 3, 4, and 5 were solved using instruction fine-tuning with FLAN-T5. The models produced semantic textual similarity scores of 0.34436, 0.14566, and 0.43527 respectively. Tasks 1 and 2 showed an improvement over the baseline models because of the inclusion of niche domain-specific words that could be accounted for in the overall vocabulary, while Tasks 3, 4, and 5 did not show an improvement over the baseline (gpt-3.5-turbo) most likely because of the few training examples used during fine-tuning which could not compete with GPT's large language model.

1 Key Information

- External collaborators: N/A
- External mentor: N/A
- Sharing project: N/A

2 Introduction

This project uses modern NLP transformer-based solutions to solve problems within the domain of the job application process. From the job applicant's point of view, the main pain points experienced during a typical job search include: 1) searching for a job posting to apply to, 2) extracting important skills keywords from that job posting, 3) modifying an existing resume to include the important skills keywords in order to tailor the resume towards a specific job posting, 4) writing a cover letter which builds on the resume, and is also specific to a job posting, 5) answering additional web form questions when submitting the application online, and 6) preparing for the interview if shortlisted.

The solution to all these pain-points can adequately be addressed through the use of transfer learning and pretrained models with fine-tuning on the downstream NLP task using data collected within the domain of that downstream task. In total, there are 5 NLP tasks this project addresses: Task 1: Named Entity Recognition (NER) for extracting important skills keywords from job postings, Task 2: Masked Language Modeling (MLM) for automating the modification of existing resumes to include skills keywords from job postings, and Task 3: instruction fine-tuning to autoregressively generate a cover letter conditional on inputs.

Task 3 was divided into three NLP sub-tasks: Task 3a to generate a cover letter conditional only on a job posting as input, Task 3b to generate a cover letter conditional on a job posting, but also demographic information such as the applicant's education history, and full name from the resume. Task 3c takes the cover letter generation one step further by conditioning on the same inputs as Task 3a, and 3b, but in addition, conditions on resume bullet points from the applicant's employment history (what companies they have previously worked for, their job title, and bullet point descriptions of what sort of tasks they were previously in charge of).

Tasks 4, and 5 again involved instruction fine-tuning, but this time for a question-answer (Q/A) task: Task 4 is aimed at automating the completion of web form questions on a company's job application page. Examples of these questions include "Why do you want to work here?" or "Explain a time when you when above and beyond for a customer". Task 5 is also question-answer task, but involves generating a question-set conditioned on the resume. This question-set can be used by the applicant to help them anticipate potential job interview questions that might arise around their claimed resume experiences during the real interview. Lastly, the social motivation behind this project also aims at providing job applicants equal opportunities: ideally, a job applicant will be able to apply to more jobs, in a shorter amount of time, with a higher success rate.

3 Related Works

The first paper which motivated the solution to Task 2 was by Liu, et al. (2019) on RoBERTa: an optimized version of BERT. The authors replicated the original BERT pretraining study [2] by Delvin, et al. (2019), and with some configuration changes (such as training the model longer, using bigger batches, removing the next sentence prediction objective, and dynamically changing the mask pattern), they were able to obtain improvements to the published BERT results on various GLUE tasks, as well as on SQuAD and RACE [1]. RoBERTa also makes use of dynamic masking (as a pose to BERT which uses static masking). This is the process by which different parts of sentences are masked for different epochs. This was related to my project with Task 2 regarding the keyword insertion problem. Using RoBERTa for this task fine-tuned on resume data allowed for keywords to be more correctly inserted into sentence locations that preserved grammatical correctness.

The second paper which motivated the solutions to Tasks 3, 4, and 5 was by Chung, et al. (2022) on FLAN-T5 and instruction fine-tuning [3]. Instruction fine-tuning is the problem by which a model is trained on a variety of tasks and a collection of datasets, through phrased instructions as the input to the model. Originally, the T5 model was proposed by Raffel, et al. (2020), which was an encoder-decoder pretrained on a mixture of supervised, and unsupervised tasks [4]. The FLAN-T5 is an augmented version of T5. The paper by Chung, et al. in particular helped to solve a crucial component of this project related to NLP tasks, which was conditioning on context input data such as resumes, and job postings. By fine-tuning FLAN-T5 on downstream tasks, I was able to achieve this component of the project.

Lastly, the paper from the project proposal by Gururangan, et al. (2020) motivated the data collection efforts to construct domain-specific corpora [5]. The authors showed the results of domain-adaptive pretraining (DAPT) and task-adaptive pretraining (TAPT), and how improvements over the RoBERTa baseline were possible. This paper related to my project because in the TAPT experiments, it was shown how acceptable results were still possible with smaller datasets, and less compute: the authors were able to match the performance of DAPT using TAPT.

4 Approach

The overarching problem is to investigate transformer-based solutions to offloading the manual effort within the job application process. The project approach was as follows: 1) first validate the major pain points job applicants face during the job application process, 2) identify what sort of NLP solutions might be feasible to solve these major pain points, 3) collect the data needed for the downstream NLP tasks, 4) write the neural network training loop to solve for and save the model weights, 5) evaluate the model on the test data set, and 6) deploy the model in such a way that it can be used for inference in a production environment.

Validate the major pain points job applications face: At the start of the course, preliminary background research was conducted using a mix of literature review, informal question-answer interviews conducted virtually with job applicants on Zoom, and reviewing existing market solutions. The results were aggregated and coded into 6 main themes which were taken to be the major pain points: 1) searching and filtering for job postings online, 2) extracting important skills keywords for jobs which the applicant is qualified for, 3) using those keywords to modify their existing resume, 4) writing a customized cover letter against the job posting, 5) submitting the application online while sometimes having to answer additional web-form questions, and 6) preparing for both behavioural, and technical job interviews. The reason for pain points 2, and 3 is because of the increasing sophistication of applicant tracking systems (ATS) that companies prevalently now use.

Identify candidate NLP solutions: several candidate solutions were developed at the start of the course, but none of them were viable. Details pertaining to the development of these preliminary solutions can be found in Appendix A.

Data collection: The data collection activities are explained in further detail in section 5, but essentially aim to collect examples related to each downstream NLP task.

Training loop and fine-tuning: The procedure for all 5 downstream tasks were the same: find a suitable pretrained model for the given NLP task, and fine-tune it using the examples (also related to the NLP task) obtained during data collection.

Model evaluation: For Tasks 1, and 2, the evaluation metric was accuracy, and top-k accuracy. Top-k accuracy for Task 2 (as it was defined in the CS 224n project milestone), is the ratio of predictions where the keyword, $w_i \in W$ to be inserted in sentence $s \in S$ appears in the top-k scores output of the MLM, to the total number of examples. For Tasks 3, 4, and 5, since the output is also a string of text, the evaluation metric of semantic textual similarity (STS) was used.

Model deployment: The best solution found so far for model inference in a production environment far was to make use of LinkedIn URLs, where a job applicant provides their personal URL to trigger this end-to-end pipeline where Tasks 1 - 5 are performed. The use of LinkedIn was due to convenience because of how data was collected (see Section 5).

Baselines and main approaches: The baselines are also discussed in more detail in Section 5. At a high-level, for Task 1, the baseline was GPT-3, for Task 2, the baseline was pretrained DistilRoBERTa. For Tasks 3, 4, and 5, the baseline model was GPT-3.5-turbo. The main approach for Task 1 was BERT (fine-tuned on skills keywords in job postings), while for Task 2 the main approach was DistilRoBERTa (fine-tuned on software resumes). The main approaches for Tasks 3, 4, and 5 made use of FLAN-T5 and instruction fine-tuning on the textual data for each downstream task.

5 Experiments

In this section, the data collection procedure is discussed in full detail, along with the evaluation methods, and experimental details. We also present the quantitative results.

5.1 Data collection

The data collection procedure started during the early weeks of the course, beginning with job posting data. This phase of the project (along with fine-tuning) proved to be the most time-consuming. The following sections describe the data collection and pre-processing steps for each task.

Job posting data for Task 1: The source for collecting job posting data for Task 1 was Google Jobs. This is a public job board accessible through search engine, which lists job postings based on some query (such as "software" or "nlp" jobs) and geographical location. Each job posting typically links to another source such as LinkedIn, or the job posting company's website. The posting is rich with detail, and the text data of interest is within the job description. Details in the header such as company descriptions, salary information, or other logistics were not needed, therefore the scraping procedure ignored this. Out of all the data collection steps, the scraping of job posting data was quickest and easiest. In total, 500 postings were scraped using the keyword "software", and another 500 postings were scraped for the keywords of "data science". For both, a geographical location in the U.S (San Francisco, CA) and Canada (Toronto, ON) were used. These were chosen to match the LinkedIn profiles which were scraped in Task 2 (also in San Francisco Bay area, and Toronto). The pre-processing steps involved the removal of non-alphanumeric characters. Figures 1 and 2 in Appendix B contain screenshot of a typical Google Jobs posting (job highlights in 1 and job description in 2, while Figure 3 contains a screenshot of the data frame object for which the job posting data was stored, and eventually converted into a CSV file.

Resume data for Task 2: There were no realistic resume data sets that could be found publicly. Resume data is typically confidential. Companies store this data securely and privately. Furthermore, generating synthetic resume data is also not ideal, nor realistic. The closest representation of resume data was found to be with LinkedIn profiles, which are public. A typical LinkedIn profile consists of a header description, a list of work experiences, a list of education and degrees, a list of skills and endorsements, a list of publications, and several other attributes. It turns out that these attributes closely resemble what a job applicant will list on their resume (LinkedIn can be used to apply for jobs after all). To collect resume data, the fields of header description, work experience, education, and skills were scraped. The header description, and skills list were both just strings. The schema for work experience, and education are similar: they are both lists which differ in length for each user. Each element of the list is a dictionary. For work experience, the dictionary keys are company name, date range, job title, and job description. The values are all strings. The job description string consists of sentences of what that user did at their company. This is akin to resume job description bullet points. For Tasks 2, 3, 4, and 5, this information is used as input to the models. As of this final report submission, over 1000 LinkedIn profiles have been successfully scraped, and stored into a CSV file. To adhere to privacy, personal identifiers such as the user's first name, last name, and LinkedIn URL were not stored. Figure 4 in Appendix B contains a snippet of a typical LinkedIn public profile work experience section, while Figure 5 shows a screenshot of the data frame object this data was stored in. The fields of 'jobs', and 'schools' are both lists of dictionaries each of which contains keys such as 'companyName', 'school', 'degree', and 'description'.

There were many challenges during this data collection stage. The first challenge was with the inaccessibility of some LinkedIn profiles. LinkedIn has a notion of degree-of-connection: browsing some LinkedIn profiles was not possible using a public search. In order to view the contents of that profile, a LinkedIn account with a 2nd degree connection or higher was needed. To overcome this, my personal LinkedIn profile was used. Prior to data collection, several network connection requests were made to users in Toronto, and San Francisco in order to improve the scraping success rate. This substantially improved the results from 60% to 97% (for example, for every 100 profiles, only 3 were not accessible when using my own LinkedIn account to browse).

The second challenge was with LinkedIn's network limitation rate. Scraping too many profiles in a single day initially led to a block, and the requirement to submit government issued ID to LinkedIn to unlock the account. The ideal number of profiles to scrape on a given day is between 50-80. The third challenge was that this number should be spread throughout the day time hours, and at any point in time, no more than 10 profiles should be scraped. Ultimately, this is what led to having only collected 1000+ LinkedIn profiles as of this final report. Data pre-processing steps were applied only to the job description bullet points. The most important step was to first split this string by the newline '
n' delimiter, and store the result into a Python list. This was repeated for each user's employment history. The next step was to remove bullet points and numbered lists: some users precede their job description sentences either with a bullet, or a number. We cannot just remove all numbers, as this would remove quantitative metrics appearing later on in the sentence. We also cannot just remove non-alphanumeric characters, as this would remove apostrophes, slashes, and hyphens.

One final challenge with this stage was how some users did not list any job description bullet points in their LinkedIn profiles: they only listed the company, their job title, and date range of employment. These could not be omitted during the scrape jobs, however during pre-processing, these profiles could be identified. To handle them later on, their job description bullet point set was assigned to be an empty string. Therefore to skip over them in future, we simply check if the attribute 'description' is empty.

Cover letter data for Task 3: To train a model to autoregressively generate cover letters conditioned on a resume, and job posting, cover letters are needed in addition to the LinkedIn profiles (acting as a proxy for resumes) that have been collected. The closest representation of cover letters which correspond to each LinkedIn profile is the header. Unfortunately, most users do not list a header, and for those that do, the header is not as rich and detailed as a real cover letter. Again, real cover letter data is confidential, and furthermore, we had already devoted substantial amount of time to collect the LinkedIn profiles. Ideally, we would want a representation of a cover letter related in some way to each individual LinkedIn profile. The next best approach that was implemented was to generate cover letters from the LinkedIn profiles and job postings using the gpt-3.5-turbo API. This step also had substantial challenges.

The first challenge was having to write a loop to call the gpt-3.5-turbo for each resume, and job posting combination. A few test run calls to gpt-3.5-turbo conditioning on the resume, and job postings showed impressive results. The issue was now with scaling this data collection job: calls to gpt-3.5-turbo costed \$0.002 per 1K tokens, and so mistakes could be costly. Also, with APIs, some calls were likely to fail (because of widespread adoption of chatGPT), and so some re-start mechanism should be implemented so that we would never have to start over again with resume 1, and job posting 1. The best solution found was to persist loop variables to an outside CSV file stored locally, and write the data collection loops in an atomic way such that API calls which fail, do not persist or store any data (despite generating the cover letter). This is an "all or nothing" type of operation.

The second challenge was with automating this data collection job. The solution was to write the main data collection loop to process in batches specified by the number of resumes, and number of jobs. For example, if the resume batch was set to 3, and the job posting batch was set to 10, then $3 * 10 = 30$ many cover letters would be generated by gpt-3.5-turbo. The third challenge was related to keeping track of costs, and execution times. For this, a separate analytics data frame and CSV was created to keep track of the cumulative costs, just to make sure they were within reason.

The fourth challenge was with linking all of the CSV tables. We have a table for resumes (collected from LinkedIn), job postings (collected from Google Jobs), and now cover letters (generated by gpt-3.5-turbo), as well as an analytics CSV file. These tables needed to be linked relationally in order for creating the Hugging Face dataset object later on as input to model training. To perform this linking, primary and foreign keys were created within each table. The final result of this data collection step was a set of cover letters containing resume bullet point sentences from LinkedIn, keywords from the job posting (Task 3a), and skills keywords from the user's resume (Task 3b), and keywords from both the job posting, and resume (Task 3c). Figure 6 in Appendix B shows a screenshot of a comparison between cover letters generated for Task 3b, and Task 3c. The user's name is not shown for privacy reasons. The cover letter on the right (green) shows a more personalized cover letter generated around the user's resume bullet points from their LinkedIn profile. Figure 7 in shows a screenshot of the data frame object used to store the generated cover letters, while Figure 8 shows a screenshot of the analytics data frame for the cover letters.

Web-form question-answer data for Task 4: Again, to train a model to autoregressively answer web-form questions on a company's job application website, examples consisting of ((question, resume, cover letter), answer) tuples were needed, where 'answer' is the label. Two of these, we have already obtained from the previous data collection steps: resumes and cover letters. The most accurate way to obtain the questions is to go to each company's website and scrape the web-form questions. The issue is that most company websites require a login, and so this would not be feasible from the perspective of time. The next best approach found was to generate a question set consisting of a mix between gpt-3.5-turbo, and public websites which provide a list of similar questions, such as "why do you want to work here?". In doing this, 101 questions were generated. The next step was now to generate answers to each of these 101 questions per resume, conditioned on the resume job description points, but this time, also conditioned on the user's work experience: company name, and job title. An example response to the previous question from the collected data can be found in Appendix C.1. As seen in this example, the answer correctly integrates the user's company (Deloitte Digital), job title (Senior Consultant), and a variety of skills (React, Java, JavaScript, Node.js). The challenges with this step also involved having to write batch processing loops, and persistent and atomic CSV writes. An analytics file was also written to save the gpt costs. The resulting question-answer CSV file was linked to the other tables in the current database under development through resume id, and gpt id (to link to the question-answer analytics table). Figure **??** shows what a typical web-form question box looks like on a job application website, while Figure **??** shows a screenshot of the data frame object used to store the question-answer pairs for training, and 11 shows the data frame object to track analytics such as execution time, and cost.

Interview data for Task 5: In Task 4, the labels or targets were autoregressively generated answers. Task 5 performs a similar task but instead generates a question set given some answer. The answers are contained in documents such as the resume, cover letter, and job posting. The goal here is to generate a question set in order to anticipate potential interview questions that might be asked. In Task 5, the job applicant is further downstream in the job application process: they have searched for a job posting, customized their resume and cover letter towards that job posting, submitted their application online, and have been shortlisted and invited to an interview. In order to generate the question set, gpt-3.5-turbo was again used, this time feeding in job description, and education description bullet points from the LinkedIn resume, as well as the skills attribute previously collected. Many job interviews will often question the candidate about the skills they claim on their resume, so these are a good input to condition on when generating the question set. Most of the data generation scripts followed the same procedure as with Task 4: examples were generated in batches, with the loop variables saved into a CSV file on the hard disk (to prevent incomplete data being written to the data frame and corpus in case API calls failed). Again, an analytics file was written to keep a log on variables like execution time, the gpt-3.5-turbo tokens used, and the cost. Figure 12 shows the data frame used to store the answer and interview question-set pairs, while Figure 13 shows the analytics data frame. The field of 'questions' is a list of the generated interview questions. Appendix C.2 shows a print out of two example interview questions generated conditioned on the applicant's resume.

5.2 Evaluation method:

For Task 1, the evaluation method for the main approach was accuracy. Unfortunately, there were too many job postings to manually label each one. So in order to autogenerate the labels, gpt-3 was used. This produced tuple pairs of (job description, list of skills keywords), where the list of skills keywords was the label. Fine-tuning BERT on this task produced a model which we could then feed in unseen job posting data from the test data set, and check if it outputted the correct skills keywords. For example, if the model outputted at least 75% of the skills keywords, we take this as a successful prediction. Therefore, accuracy defined here is the ratio of successful predictions (as previously defined) to the total number of examples.

To evaluate Task 2, the evaluation method was top-k accuracy as previously defined in the project milestone. Again, after training the model using MLM on the resume data (randomly masking out tokens, with the objective of reconstructing them), for inference, we pass in an unseen resume from the test data set. Recall that this resume is just a sentence set of bullet points collected from LinkedIn profiles. For a given placement of a <mask> token in a sentence, we produce the top-k predictions using the model, and compare this to the word list. If any word appears, we perform the insertion, and remove from the word list. If multiple words appear as candidate solutions, we take the top scoring one. Top-k accuracy is therefore the ratio of predictions where the keyword, $w|i \in W$ to be inserted in sentence $s \in S$ appears in the top-k scores output of the MLM, to the total number of examples. For Tasks 3, 4, and 5, the evaluation method was with semantic textual similarity. Since the output is pure text data (cover letter, an answer, and a question set), and the target is also pure text data, we can calculate the STS scores.

5.3 Experimental details:

Each task was completed in sequence, often using data collected or generated from the previous task as a feed in to the next task. To run these experiments, Google Colab was used with a Pro subscription, along with the the provided AWS credits. For each task, after the data was collected, pre-processed, and stored into a CSV file. This was then uploaded to Google Drive, and read into a data frame in the Colab notebook. From here, a step was needed to transform the data frame into a JSON line-delimited data set. This step was needed in order to create Hugging Face dataset objects (regular JSON files did not work). A temporary Hugging Face dataset was created with splits of either 60-40 or 75-25. The resulting test set was then split 50-50 into a final test set, and a validation test set. A final Hugging Face data set object was then created from the train, test, and validation split. Only features required for training were included in this data set. The features here contain the example features, along with the target label. The dataset object was saved to Google Drive for use in model evaluation later on. The next step was to tokenize the examples. For many of the tasks, the RobertaTokenizer was used with the 'codet5-small' model. For Tasks 3, 4, and 5, a prefix was appended to the model inputs before tokenization. The prefix typically consisted of the instruction, such as "generate a cover letter condition on ____", along with the conditional data such as the resume, and/or job posting. The max input lengths, and max target lengths were set to be 3000, and 1500 respectively, because some of the prompts were very long. The labels were also tokenized (no prefix was needed here). This was repeated for all examples in the train, test, and validation Hugging Face dataset objects using the map() function. The last step before fine-tuning was to specify the batch sizes using DataLoader from torch.utils.data, and setting the format. Each batch was a dict object with the keys of the tokenized input_ids, the attention mask, and the labels. For fine-tuning, I made use of a class object based on PyTorch Lightning. This was only script which I did not write. This script was taken from Hugging Face, and defined the forward method, training step, validation step, and test step methods, along with a function to configure the optimizer. AdamW was used. In the training loop, EarlyStopping was used, along with learning rates of 5e-5, 15 training epochs, and 1000 warmup steps. The max epochs was set between 50 and 100 for most tasks. Training was done using 1 GPU with the Premium Class, and High-RAM options in Google Colab. When finished, the model was saved to Google Drive storage. During inference for each task, the model was loaded, along with the same tokenizer. The model.generate() function was used to generate predictions for Tasks 3, 4, and 5, and the result was decoded back and compared with the target label. For Task 2, a separate script was required for inference which involved tokenizing the unseen data, and obtaining the mask token index and top-k prediction IDs from the logits. The top-k prediction token IDs were then decoded, from which the candidate word list and scores could be outputted for model evaluation.

5.4 Results

Using the evaluation methods outlined in section Section 5.2, the following results in Table 1 were obtained.

Task	evaluation metric	baseline	fine-tuned
1	accuracy	73.7%	78.3%
2	top-k accuracy	37%	48%
3	STS	0.65284	0.34436
4	STS	0.61836	0.14566
5	STS	0.68131	0.43527

Table 1: Results of fine-tuning domain NLP tasks compared to baselines

As we can see, for Tasks 1, and 2, the main approach using fine-tuning improved the results over the baseline, while for Tasks 3, 4, and 5, the main approach could not outperform the baseline. There are a few possible explanations for this. First, for Tasks 1, and 2, the baseline models were gpt-3, and pretrained DistilRoBERTa respectively. Although these models were already very good, it was found that some niche software related terms could not be re-produced in the MLM task, nor identified in the NER task. Even with a few examples (1000 job descriptions, and 1000 LinkedIn profiles), an improvement in the downstream NLP tasks was realized. With Tasks 3, 4, and 5 however, there was not an improvement obtained with fine-tuning. The problem of small example size here was more prevalent: comparing the autoregressively generated output sometimes produced very different results to the target, for example with the cover letters, and interview question sets. Some cover letters also were not as coherent. The other issue was that gpt-3.5-turbo was already a very large language model, and was difficult to complete with using the transfer learning approach here. For future, it might be worth while to attempt to fine-tune gpt-3.5 or experiment with gpt-4, which was released in the middle of this project.

6 Analysis

As mentioned in the results, one limitation of this project is with the small examples collected, and generated. The target number of LinkedIn profiles to scrape was between 6000 and 10,000, but it was not anticipated how strict LinkedIn's rate limitation rules would be. This likely led to some undesirable side-affects, namely with grammatically incorrect sentences in the cover letter. Despite these imperfections, the models seem to indeed condition on the input data: for Task 5 where we generated potential interview question sets conditioned in a user's resume, and job posting they would have applied to, the model did quite well at generating question sets based on this input. Two examples can be found in Appendix C.2 which show content from the resume conditional input appearing in the question set.

For Task 2, in the two algorithms that were developed to identify <mask> token positions, and the top-k candidate words, there was a performance bottleneck: the algorithm for determining the position as of this final report submission uses a simplistic brute force approach, where all positions are checked until an insertion is made. This algorithm therefore has an $O(n^3)$ type of time complexity. To be exact, the complexity is approximately $O(|S|\text{max}_i s_i|W|)$ where $|S|$ is the number of sentences in the resume (made up of job description bullet points), $\text{max}_i s_i$ is the maximum word length of all the sentences in S, and $|W|$ is the size of the skills keyword list obtained from Task 1 (NER). Upon running this brute force search algorithm on some examples, the run time was under 25 seconds each time. In production, this could still be acceptable. However, to improve the search, some heuristics were developed (but not implemented due to time constraints). The most plausible heuristic involved building a topic sentence classifier, and running this classifier first on the resume bullet points. The output of this classifier is a word representing the topic of some sentence. For example, a bullet point describing some tasks a user performed at their previous job related to Jenkins might output "DevOps" as the topic for that resume bullet point sentence. Then, for each keyword in W, we first check to see whether any of the keywords closely resemble "DevOps". This closeness can be represented using cosine similarity. The highest cosine similarity score can then be obtained, and the original insertion algorithm can be applied to only that sentence. Since the topic sentence classifier only has to run once for each resume, we can cache the results. This would reduce the time complexity to approximately $O(\text{max}_i s_i|W|)$.

The last point to be made is on the comparison between the transfer learning methods here, and large language models (LLM) like GPT-3.5 or GPT-4. As of this final project submission, these LLMs are extremely fast, and cost effective to consume. In each task where data generation was used, the run time was less than 60 seconds, and the costs were astonishingly small. Most outputs only consumed between 4000 to 7000 tokens, and the cost of each API call was therefore between \$0.006 - \$0.014. Based on this, if one's goal is to build a production-ready product and deploy as fast as possible, it might be better to consume the GPT APIs. The only way to really complete with these LLMs in the context of the domain problem of this project, is to collect data the LLMs almost have no chance at seeing or being trained on, such as the LinkedIn profiles. As mentioned, other plausible solutions might involve fine-tuning on the GPT LLMs, which is possible with gpt-3.5-turbo.

7 Conclusion

In conclusion, this project outlined how transformer-based transfer learning solutions could be used to solve downstream tasks in the domain of the job application process. What this project achieved was the collection of real-data, and its post-processing in order to fine-tune pretrained models for solving the downstream problems. The result is a proof-of-concept which could legitimately offload the manual efforts job applicants face. For future work, more data can be collected, in addition to implementing the insertion algorithm for Task 2 to improve run-time, using the proposed heuristic discussed in Section 6. Some additional work could also be devoted to the deployment of the model, and architecting a Front-End and Back-End solution in which real users could consume the models developed here for Tasks 1-5. The way they consume the models could also impact future NLP solutions. Lastly, there was an initial Task 6, and Task 7 related to dialog, and sentiment analysis respectively. This would have involved a task where, after the interview questions were generated from Task 5, the applicant could then engage in a two-way conversational dialog with a simulated interviewer.

References

[1] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L. and Stoyanov, V., 2019. Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692.

[2] Devlin, J., Chang, M.W., Lee, K. and Toutanova, K., 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

[3] Chung, H.W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., Li, E., Wang, X., Dehghani, M., Brahma, S. and Webson, A., 2022. Scaling instruction-finetuned language models. arXiv preprint arXiv:2210.11416.

[4] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W. and Liu, P.J., 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. The Journal of Machine Learning Research, 21(1), pp.5485-5551.

[5] Gururangan, S., Marasović, A., Swayamdipta, S., Lo, K., Beltagy, I., Downey, D. and Smith, N.A., 2020. Don't stop pretraining: Adapt language models to domains and tasks. arXiv preprint arXiv:2004.

Appendix A

The first of the candidate solutions was with the use of vector embeddings, and cosine similarity to solve Task 2 on skills keyword insertion. The algorithm was simple: given a set of sentences, S, and word list, W, for $w_i \in W$, define: s_{ij} as the new sentence that is formed when inserting w_i into position j of original sentence $s \in S$. We then take the solution, s_i' as:

$$s_{i,l}' = \frac{\mathbf{s_{ij}[j-1] \cdot w_i}}{||\mathbf{s_{ij}[j-1]}||||\mathbf{w_i}||} \tag{1}$$

$$s_{i,r}' = \frac{\mathbf{w_i \cdot s_{ij}[j+1]}}{||\mathbf{w_i}||||\mathbf{s_{ij}[j+1]}||} \tag{2}$$

$$s_i' = \arg\max_j \frac{s_{i,l}' + s_{i,r}'}{2} \tag{3}$$

The expression (3) just maximizes the average cosine similarity between w_i and both the left, and right neighbouring words for a given position, j. This solution did not produce good results because it resulted in several new sentences s_i', which were not grammatically correct. The next approach built on top of this by using a language judgement acceptability task whereby we iteratively, and exhaustively try all possible positions j in a sentence for which we insert w_i. For each candidate sentence, we then run a grammar acceptability check: if the new sentence is grammatically correct, we store it as a candidate solution. The problem with this algorithm however, is both computational expense, and in the end, we usually still have "too many" solutions. When CS 224n got into language models, the third approach was to consider the use of $n-gram$ language models. The algorithm developed at that stage was to insert word w_i into position j for which the average conditional probability of:

$$\frac{p(w_i|s_{j-1,w_i}) + p(w_i|s_{j+1,w_i})}{2} \tag{4}$$

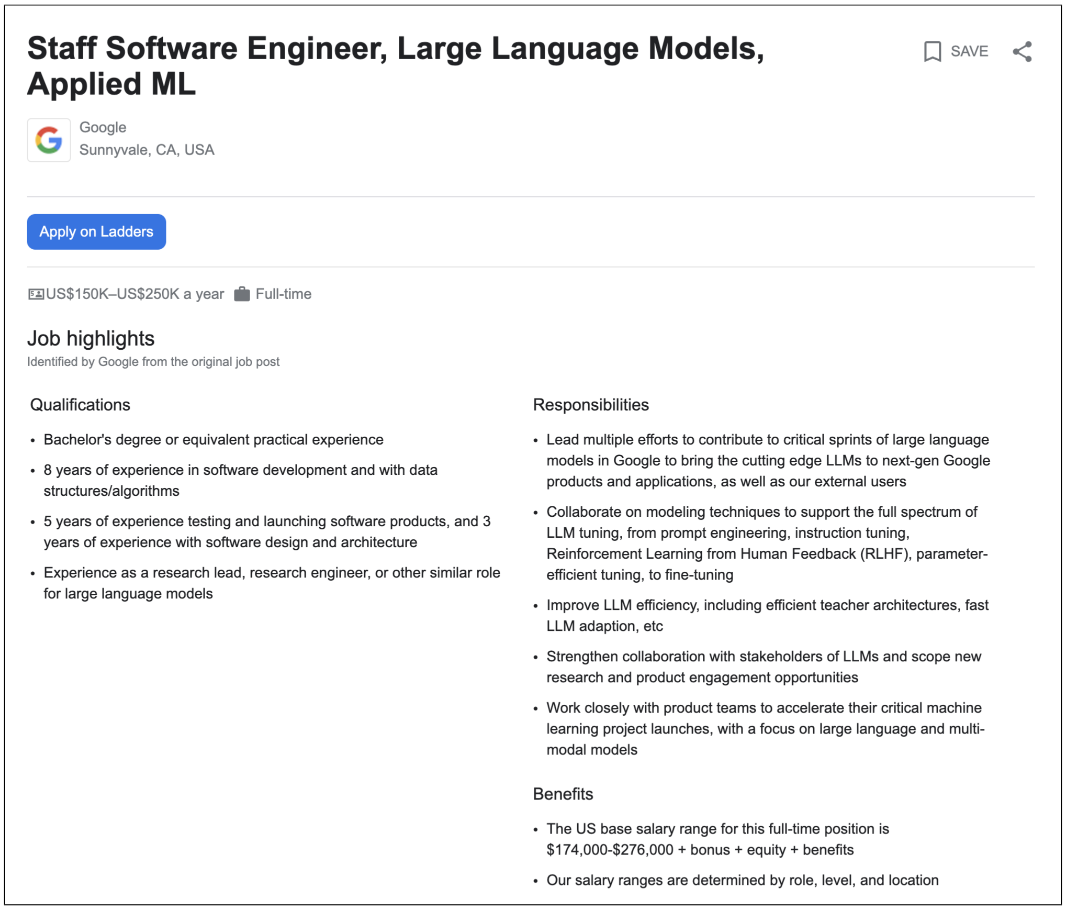
was largest, where:

$$p(w_i|s_{i-1,w_i}) = \frac{\text{counts}(s_{i-1,w})}{\text{counts}(w_i)} \tag{5}$$

$$p(w_i|s_{i+1,w_i}) = \frac{\text{counts}(s_{i+1,w})}{\text{counts}(w_i)} \tag{6}$$

Again, this approach failed because depending on the corpus and vocabulary, one of the counts expressions in the numerator or denominator of (5) or (6) could be zero for some $n-gram$ model. In the last few weeks of the course, the most promising approach so far was found to be with transformer-based solutions which circumvent the need for recurrence and convolutions through implementation of self-attention. This led to the first plausible solution to task 2: the use of masked-language models (MLM) to predict mask tokens inserted into sentences in very much the same way as the 3 previous algorithms developed here. The other factor which made this approach the most feasible was through the availability of pretrained models. Using one of these pretrained models solved issues related to grammatical correctness, but the issue of words not appearing in the vocabulary persisted. This was solved by taking a fine-tuning approach, where the pretrained model was fine-tuned on examples related the the domain NLP task. It was in fact this same approach of finding an appropriate pretained model, and fine-tuning it, that also led to solutions within the other NLP tasks (Task 1, 3, 4, and 5).

Appendix B


Staff Software Engineer, Large Language Models, Applied ML

Google
Sunnyvale, CA, USA

US$150K–US$250K a year Full-time

Job highlights
Identified by Google from the original job post

Qualifications

- Bachelor's degree or equivalent practical experience
- 8 years of experience in software development and with data structures/algorithms
- 5 years of experience testing and launching software products, and 3 years of experience with software design and architecture
- Experience as a research lead, research engineer, or other similar role for large language models

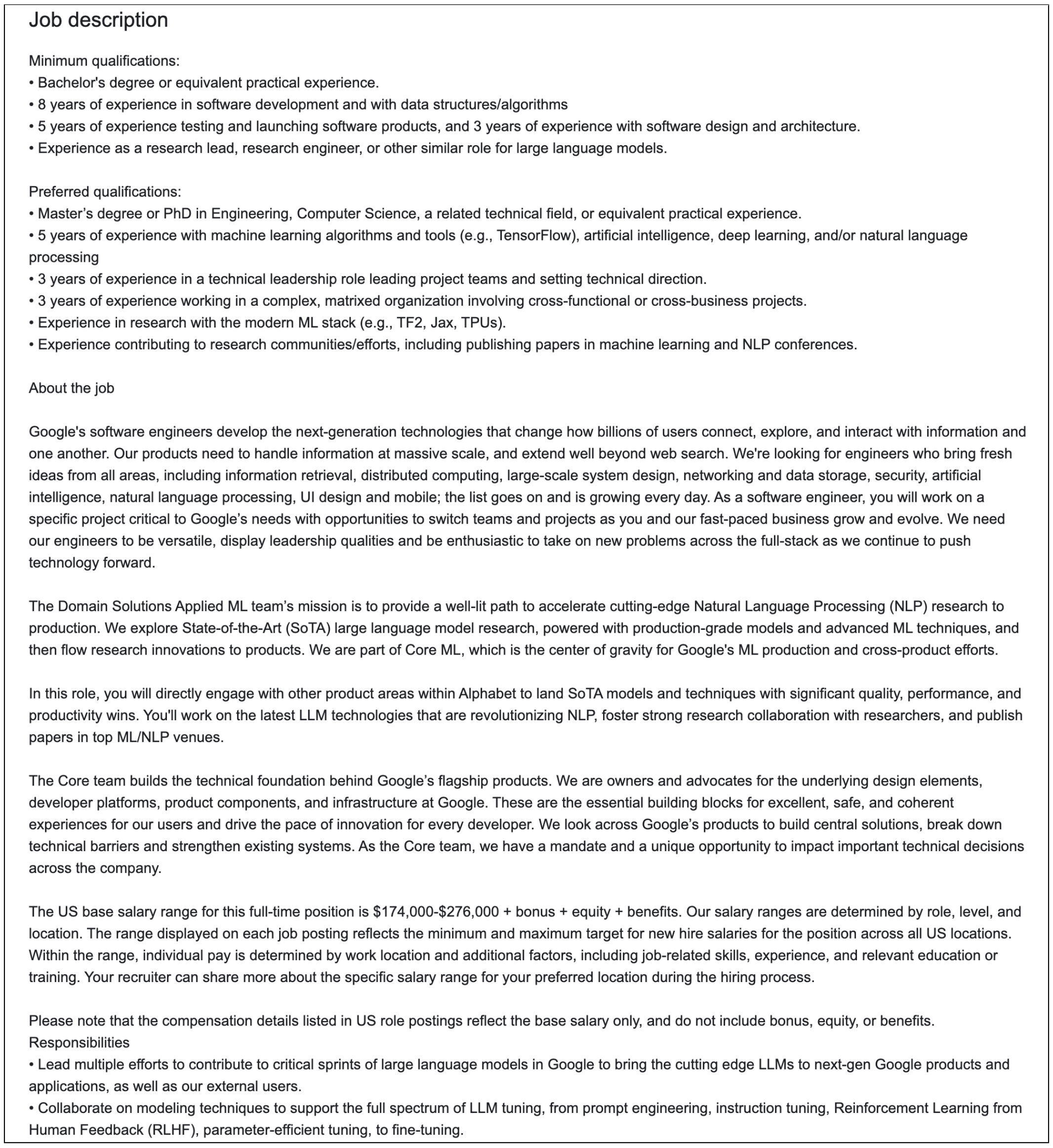
Responsibilities

- Lead multiple efforts to contribute to critical sprints of large language models in Google to bring the cutting edge LLMs to next-gen Google products and applications, as well as our external users
- Collaborate on modeling techniques to support the full spectrum of LLM tuning, from prompt engineering, instruction tuning, Reinforcement Learning from Human Feedback (RLHF), parameter-efficient tuning, to fine-tuning
- Improve LLM efficiency, including efficient teacher architectures, fast LLM adaption, etc
- Strengthen collaboration with stakeholders of LLMs and scope new research and product engagement opportunities
- Work closely with product teams to accelerate their critical machine learning project launches, with a focus on large language and multi-modal models

Benefits

- The US base salary range for this full-time position is $174,000-$276,000 + bonus + equity + benefits
- Our salary ranges are determined by role, level, and location

Figure 1: Typical job posting on Google Jobs board: this figure shows the Job Highlights portion of the posting, which was not included in the job posting corpus.


Job description

Minimum qualifications:
• Bachelor's degree or equivalent practical experience.
• 8 years of experience in software development and with data structures/algorithms
• 5 years of experience testing and launching software products, and 3 years of experience with software design and architecture.
• Experience as a research lead, research engineer, or other similar role for large language models.

Preferred qualifications:
• Master's degree or PhD in Engineering, Computer Science, a related technical field, or equivalent practical experience.
• 5 years of experience with machine learning algorithms and tools (e.g., TensorFlow), artificial intelligence, deep learning, and/or natural language processing
• 3 years of experience in a technical leadership role leading project teams and setting technical direction.
• 3 years of experience working in a complex, matrixed organization involving cross-functional or cross-business projects.
• Experience in research with the modern ML stack (e.g., TF2, Jax, TPUs).
• Experience contributing to research communities/efforts, including publishing papers in machine learning and NLP conferences.

About the job

Google's software engineers develop the next-generation technologies that change how billions of users connect, explore, and interact with information and one another. Our products need to handle information at massive scale, and extend well beyond web search. We're looking for engineers who bring fresh ideas from all areas, including information retrieval, distributed computing, large-scale system design, networking and data storage, security, artificial intelligence, natural language processing, UI design and mobile; the list goes on and is growing every day. As a software engineer, you will work on a specific project critical to Google's needs with opportunities to switch teams and projects as you and our fast-paced business grow and evolve. We need our engineers to be versatile, display leadership qualities and be enthusiastic to take on new problems across the full-stack as we continue to push technology forward.

The Domain Solutions Applied ML team's mission is to provide a well-lit path to accelerate cutting-edge Natural Language Processing (NLP) research to production. We explore State-of-the-Art (SoTA) large language model research, powered with production-grade models and advanced ML techniques, and then flow research innovations to products. We are part of Core ML, which is the center of gravity for Google's ML production and cross-product efforts.

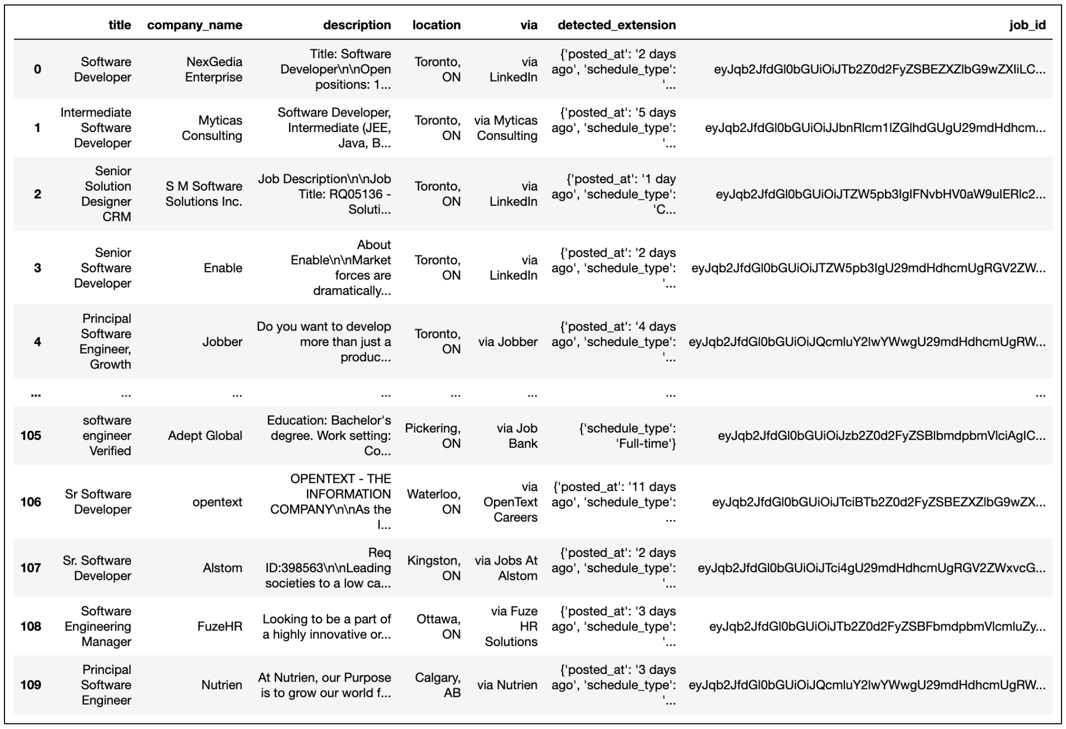
In this role, you will directly engage with other product areas within Alphabet to land SoTA models and techniques with significant quality, performance, and productivity wins. You'll work on the latest LLM technologies that are revolutionizing NLP, foster strong research collaboration with researchers, and publish papers in top ML/NLP venues.

The Core team builds the technical foundation behind Google's flagship products. We are owners and advocates for the underlying design elements, developer platforms, product components, and infrastructure at Google. These are the essential building blocks for excellent, safe, and coherent experiences for our users and drive the pace of innovation for every developer. We look across Google's products to build central solutions, break down technical barriers and strengthen existing systems. As the Core team, we have a mandate and a unique opportunity to impact important technical decisions across the company.

The US base salary range for this full-time position is $174,000-$276,000 + bonus + equity + benefits. Our salary ranges are determined by role, level, and location. The range displayed on each job posting reflects the minimum and maximum target for new hire salaries for the position across all US locations. Within the range, individual pay is determined by work location and additional factors, including job-related skills, experience, and relevant education or training. Your recruiter can share more about the specific salary range for your preferred location during the hiring process.

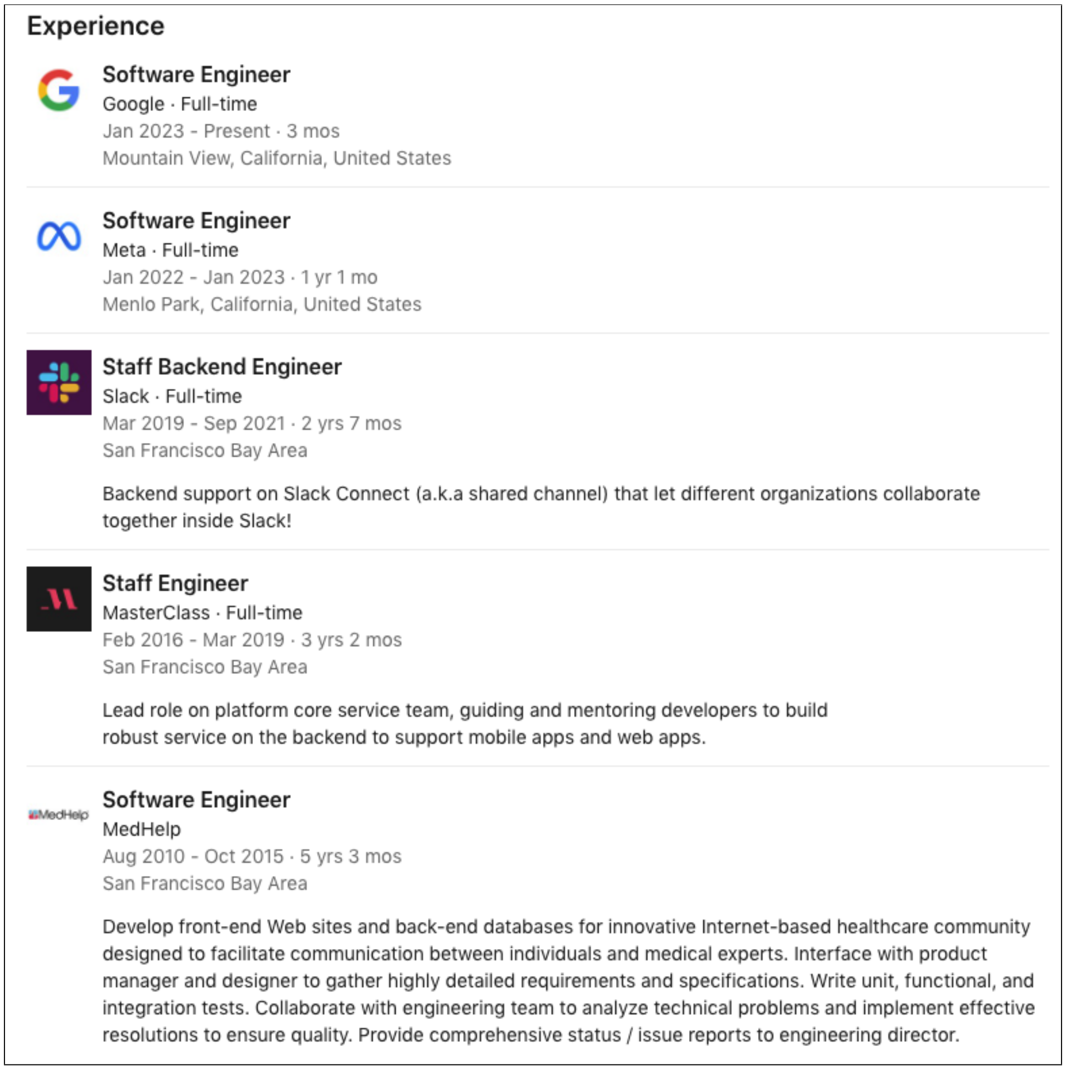
Please note that the compensation details listed in US role postings reflect the base salary only, and do not include bonus, equity, or benefits.
Responsibilities
• Lead multiple efforts to contribute to critical sprints of large language models in Google to bring the cutting edge LLMs to next-gen Google products and applications, as well as our external users.
• Collaborate on modeling techniques to support the full spectrum of LLM tuning, from prompt engineering, instruction tuning, Reinforcement Learning from Human Feedback (RLHF), parameter-efficient tuning, to fine-tuning.

Figure 2: Job description portion of the same posting in Figure 1. This is the body of text which was added to the corpus for each posting.



	title	company_name	description	location	via	detected_extension	job_id
0	Software Developer	NexGedia Enterprise	Title: Software Developer\n\nOpen positions: 1...	Toronto, ON	via LinkedIn	{'posted_at': '2 days ago', 'schedule_type': '...	eyJqb2JfdGl0bGUiOiJTb2Z0d2FyZSBEZXZlbG9wZXIiLC...
1	Intermediate Software Developer	Myticas Consulting	Software Developer, Intermediate (JEE, Java, B...	Toronto, ON	via Myticas Consulting	{'posted_at': '5 days ago', 'schedule_type': '...	eyJqb2JfdGl0bGUiOiJJbnRlcm1lZGlhdGUgU29mdHdhcm...
2	Senior Solution Designer CRM	S M Software Solutions Inc.	Job Description\n\nJob Title: RQ05136 - Soluti...	Toronto, ON	via LinkedIn	{'posted_at': '1 day ago', 'schedule_type': 'C...	eyJqb2JfdGl0bGUiOiJTZW5pb3IgIFNvbHV0aW9uIERlc2...
3	Senior Software Developer	Enable	About Enable\n\nMarket forces are dramatically...	Toronto, ON	via LinkedIn	{'posted_at': '2 days ago', 'schedule_type': '...	eyJqb2JfdGl0bGUiOiJTZW5pb3IgU29mdHdhcmUgRGV2ZW...
4	Principal Software Engineer, Growth	Jobber	Do you want to develop more than just a produc...	Toronto, ON	via Jobber	{'posted_at': '4 days ago', 'schedule_type': '...	eyJqb2JfdGl0bGUiOiJQcmluY2lwYWwgU29mdHdhcmUgRW...
...	...	...	...	...	...	...	...
105	software engineer Verified	Adept Global	Education: Bachelor's degree. Work setting: Co...	Pickering, ON	via Job Bank	{'schedule_type': 'Full-time'}	eyJqb2JfdGl0bGUiOiJzb2Z0d2FyZSBlbmdpbmVlciAgIC...
106	Sr Software Developer	opentext	OPENTEXT - THE INFORMATION COMPANY\n\nAs the I...	Waterloo, ON	via OpenText Careers	{'posted_at': '11 days ago', 'schedule_type': ...	eyJqb2JfdGl0bGUiOiJTciBTb2Z0d2FyZSBEZXZlbG9wZX...
107	Sr. Software Developer	Alstom	Req ID:398563\n\nLeading societies to a low ca...	Kingston, ON	via Jobs At Alstom	{'posted_at': '2 days ago', 'schedule_type': '...	eyJqb2JfdGl0bGUiOiJTci4gU29mdHdhcmUgRGV2ZWxvcG...
108	Software Engineering Manager	FuzeHR	Looking to be a part of a highly innovative or...	Ottawa, ON	via Fuze HR Solutions	{'posted_at': '3 days ago', 'schedule_type': '...	eyJqb2JfdGl0bGUiOiJTb2Z0d2FyZSBFbmdpbmVlcmluZy...
109	Principal Software Engineer	Nutrien	At Nutrien, our Purpose is to grow our world f...	Calgary, AB	via Nutrien	{'posted_at': '3 days ago', 'schedule_type': '...	eyJqb2JfdGl0bGUiOiJQcmluY2lwYWwgU29mdHdhcmUgRW...

Figure 3: Data frame object showing the fields of scraped job postings from Google Jobs. Shown here are job postings in Toronto, ON, Canada using the query of "software".



Experience

Software Engineer
Google · Full-time
Jan 2023 - Present · 3 mos
Mountain View, California, United States

Software Engineer
Meta · Full-time
Jan 2022 - Jan 2023 · 1 yr 1 mo
Menlo Park, California, United States

Staff Backend Engineer
Slack · Full-time
Mar 2019 - Sep 2021 · 2 yrs 7 mos
San Francisco Bay Area

Backend support on Slack Connect (a.k.a shared channel) that let different organizations collaborate together inside Slack!

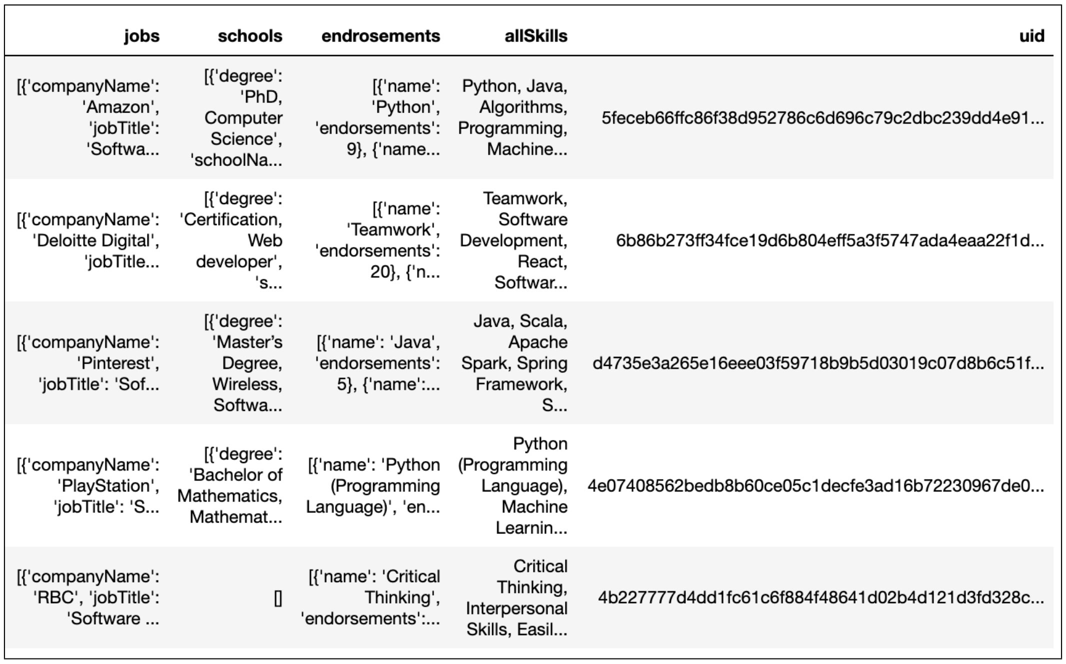
Staff Engineer
MasterClass · Full-time
Feb 2016 - Mar 2019 · 3 yrs 2 mos
San Francisco Bay Area

Lead role on platform core service team, guiding and mentoring developers to build robust service on the backend to support mobile apps and web apps.

Software Engineer
MedHelp
Aug 2010 - Oct 2015 · 5 yrs 3 mos
San Francisco Bay Area

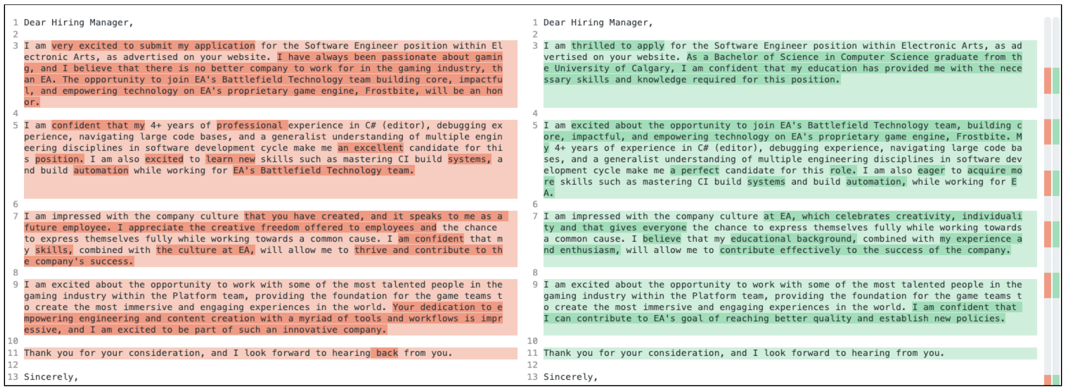
Develop front-end Web sites and back-end databases for innovative Internet-based healthcare community designed to facilitate communication between individuals and medical experts. Interface with product manager and designer to gather highly detailed requirements and specifications. Write unit, functional, and integration tests. Collaborate with engineering team to analyze technical problems and implement effective resolutions to ensure quality. Provide comprehensive status / issue reports to engineering director.

Figure 4: Typical Work Experience section of a LinkedIn profile. The text descriptions in for each job (if they are not empty) were assumed to resemble resume bullet points. This data was the core of the resume corpus.



jobs	schools	endrosements	allSkills	uid
[{'companyName': 'Amazon', 'jobTitle': 'Softwa...	[{'degree': 'PhD, Computer Science', 'schoolNa...	[{'name': 'Python', 'endorsements': 9}, {'name...	Python, Java, Algorithms, Programming, Machine...	5feceb66ffc86f38d952786c6d696c79c2dbc239dd4e91...
[{'companyName': 'Deloitte Digital', 'jobTitle...	[{'degree': 'Certification, Web developer', 's...	[{'name': 'Teamwork', 'endorsements': 20}, {'n...	Teamwork, Software Development, React, Softwar...	6b86b273ff34fce19d6b804eff5a3f5747ada4eaa22f1d...
[{'companyName': 'Pinterest', 'jobTitle': 'Sof...	[{'degree': 'Master's Degree, Wireless, Softwa...	[{'name': 'Java', 'endorsements': 5}, {'name':...	Java, Scala, Apache Spark, Spring Framework, S...	d4735e3a265e16eee03f59718b9b5d03019c07d8b6c51f...
[{'companyName': 'PlayStation', 'jobTitle': 'S...	[{'degree': 'Bachelor of Mathematics, Mathemat...	[{'name': 'Python (Programming Language)', 'en...	Python (Programming Language), Machine Learnin...	4e07408562bedb8b60ce05c1decfe3ad16b72230967de0...
[{'companyName': 'RBC', 'jobTitle': 'Software ...	[]	[{'name': 'Critical Thinking', 'endorsements':...	Critical Thinking, Interpersonal Skills, Easil...	4b227777d4dd1fc61c6f884f48641d02b4d121d3fd328c...

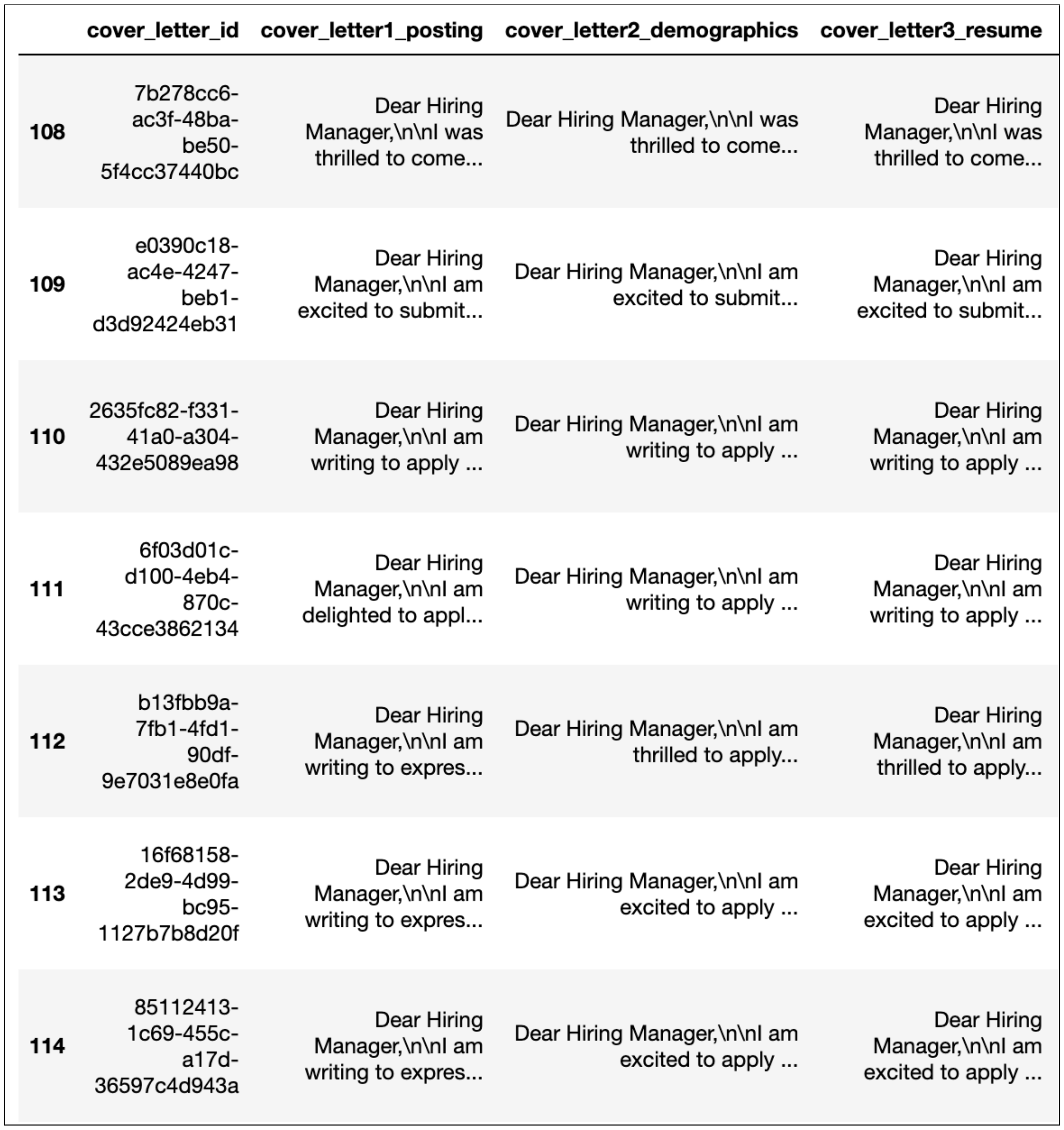
Figure 5: Data frame object showing the fields of scraped LinkedIn public profiles. The fields of 'jobs', and 'schools' contain lists of dictionaries for each job, and education the user has added to their profile. Each of these also contains a key called 'description'.



```
Dear Hiring Manager,

I am very excited to submit my application for the Software Engineer position within El
ectronic Arts, as advertised on your website. I have always been passionate about gamin
g, and I believe that there is no better company to work for in the gaming industry, th
an EA. The opportunity to join EA's Battlefield Technology team building core, impactfu
l, and empowering technology on EA's proprietary game engine, Frostbite, will be an hon
or.

I am confident that my 4+ years of professional experience in C# (editor), debugging ex
perience, navigating large code bases, and a generalist understanding of multiple engin
eering disciplines in software development cycle make me an excellent candidate for thi
s position. I am also excited to learn new skills such as mastering CI build systems, a
nd build automation while working for EA's Battlefield Technology team.

I am impressed with the company culture that you have created, and it speaks to me as a
future employee. I appreciate the creative freedom offered to employees and the chance
to express themselves fully while working towards a common cause. I am confident that m
y skills, combined with the culture at EA, will allow me to thrive and contribute to th
e company's success.

I am excited about the opportunity to work with some of the most talented people in the
gaming industry within the Platform team, providing the foundation for the game teams t
o create the most immersive and engaging experiences in the world. Your dedication to e
mpowering engineering and content creation with a myriad of tools and workflows is impr
essive, and I am excited to be part of such an innovative company.

Thank you for your consideration, and I look forward to hearing back from you.

Sincerely,
```

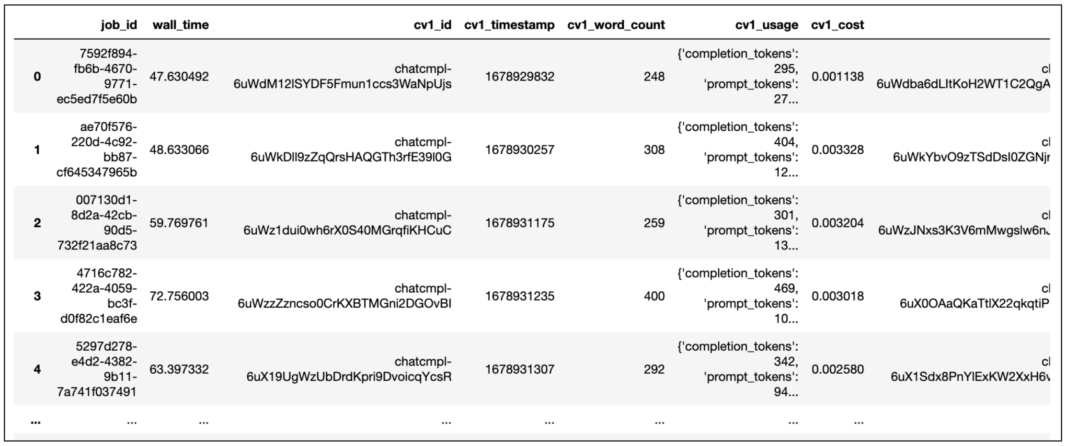
```
Dear Hiring Manager,

I am thrilled to apply for the Software Engineer position within Electronic Arts, as ad
vertised on your website. As a Bachelor of Science in Computer Science graduate from th
e University of Calgary, I am confident that my education has provided me with the nece
ssary skills and knowledge required for this position.

I am excited about the opportunity to join EA's Battlefield Technology team, building c
ore, impactful, and empowering technology on EA's proprietary game engine, Frostbite. M
y 4+ years of experience in C# (editor), debugging experience, navigating large code ba
ses, and a generalist understanding of multiple engineering disciplines in software dev
elopment cycle make me a perfect candidate for this role. I am also eager to acquire mo
re skills such as mastering CI build systems and build automation, while working for E
A.

I am impressed with the company culture at EA, which celebrates creativity, individuali
ty and that gives everyone the chance to express themselves fully while working towards
a common cause. I believe that my educational background, combined with my experience a
nd enthusiasm, will allow me to contribute effectively to the success of the company.

I am excited about the opportunity to work with some of the most talented people in the
gaming industry within the Platform team, providing the foundation for the game teams t
o create the most immersive and engaging experiences in the world. I am confident that
I can contribute to EA's goal of reaching better quality and establish new policies.

Thank you for your consideration, and I look forward to hearing from you.

Sincerely,
```

Figure 6: Comparison of generated cover letters between Task 3b (left), and Task 3c (right). The cover letter from Task 3c shows a more personalized cover letter generated around resume bullet points from the use.



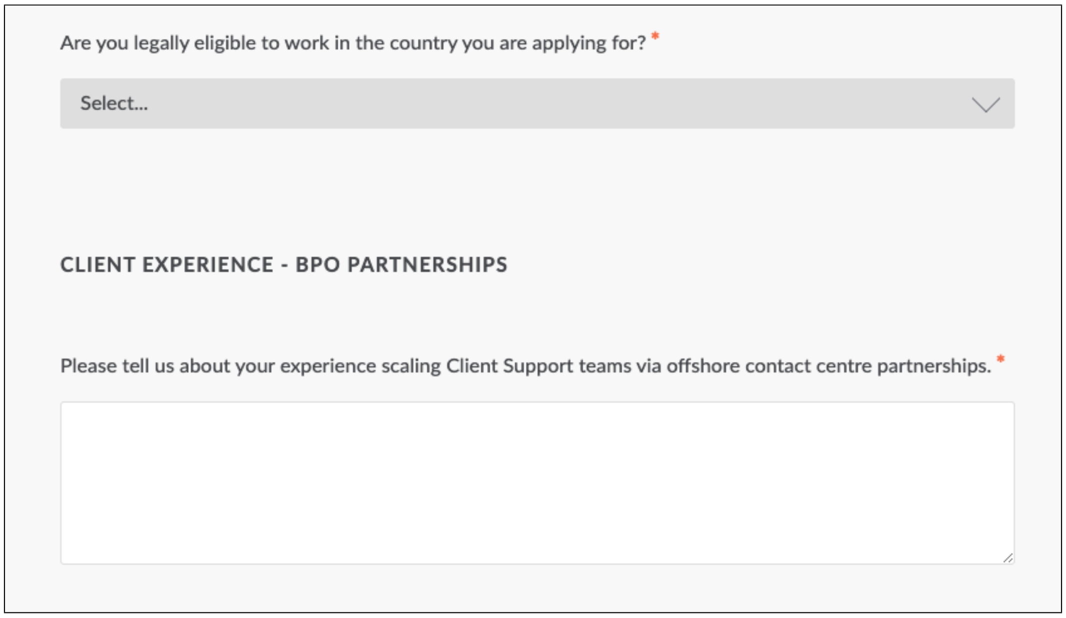
	cover_letter_id	cover_letter1_posting	cover_letter2_demographics	cover_letter3_resume
108	7b278cc6-ac3f-48ba-be50-5f4cc37440bc	Dear Hiring Manager,\n\nI was thrilled to come...	Dear Hiring Manager,\n\nI was thrilled to come...	Dear Hiring Manager,\n\nI was thrilled to come...
109	e0390c18-ac4e-4247-beb1-d3d92424eb31	Dear Hiring Manager,\n\nI am excited to submit...	Dear Hiring Manager,\n\nI am excited to submit...	Dear Hiring Manager,\n\nI am excited to submit...
110	2635fc82-f331-41a0-a304-432e5089ea98	Dear Hiring Manager,\n\nI am writing to apply ...	Dear Hiring Manager,\n\nI am writing to apply ...	Dear Hiring Manager,\n\nI am writing to apply ...
111	6f03d01c-d100-4eb4-870c-43cce3862134	Dear Hiring Manager,\n\nI am delighted to appl...	Dear Hiring Manager,\n\nI am writing to apply ...	Dear Hiring Manager,\n\nI am writing to apply ...
112	b13fbb9a-7fb1-4fd1-90df-9e7031e8e0fa	Dear Hiring Manager,\n\nI am writing to expres...	Dear Hiring Manager,\n\nI am thrilled to apply...	Dear Hiring Manager,\n\nI am thrilled to apply...
113	16f68158-2de9-4d99-bc95-1127b7b8d20f	Dear Hiring Manager,\n\nI am writing to expres...	Dear Hiring Manager,\n\nI am excited to apply ...	Dear Hiring Manager,\n\nI am excited to apply ...
114	85112413-1c69-455c-a17d-36597c4d943a	Dear Hiring Manager,\n\nI am writing to expres...	Dear Hiring Manager,\n\nI am excited to apply ...	Dear Hiring Manager,\n\nI am excited to apply ...

Figure 7: Data frame object of the generated cover letters for Task 3a (cover letter 1 posting), Task 3b (cover letter demographics), and Task 3c (cover letter resume).



	job_id	wall_time	cv1_id	cv1_timestamp	cv1_word_count	cv1_usage	cv1_cost	
0	7592f894-fb6b-4670-9771-ec5ed7f5e60b	47.630492	chatcmpl-6uWdM12lSYDF5Fmun1ccs3WaNpUjs	1678929832	248	{'completion_tokens': 295, 'prompt_tokens': 27...	0.001138	cl 6uWdba6dLItKoH2WT1C2QgA
1	ae70f576-220d-4c92-bb87-cf645347965b	48.633066	chatcmpl-6uWkDIl9zZqQrsHAQGTh3rfE39l0G	1678930257	308	{'completion_tokens': 404, 'prompt_tokens': 12...	0.003328	cl 6uWkYbvO9zTSdDsI0ZGNjr
2	007130d1-8d2a-42cb-90d5-732f21aa8c73	59.769761	chatcmpl-6uWz1dui0wh6rX0S40MGrqfiKHCuC	1678931175	259	{'completion_tokens': 301, 'prompt_tokens': 13...	0.003204	cl 6uWzJNxs3K3V6mMwgslw6nJ
3	4716c782-422a-4059-bc3f-d0f82c1eaf6e	72.756003	chatcmpl-6uWzzZzncso0CrKXBTMGni2DGOvBI	1678931235	400	{'completion_tokens': 469, 'prompt_tokens': 10...	0.003018	cl 6uX0OAaQKaTtlX22qkqtiP
4	5297d278-e4d2-4382-9b11-7a741f037491	63.397332	chatcmpl-6uX19UgWzUbDrdKpri9DvoicqYcsR	1678931307	292	{'completion_tokens': 342, 'prompt_tokens': 94...	0.002580	cl 6uX1Sdx8PnYlExKW2XxH6v
...	...	...	...	...	...	...	...	

Figure 8: Data frame object of the analytics collected for the cover letter generation jobs. Wall time is the execution time, usage stores the prompt tokens used by gpt-3.5-turbo, cv cost is the cost in dollars.



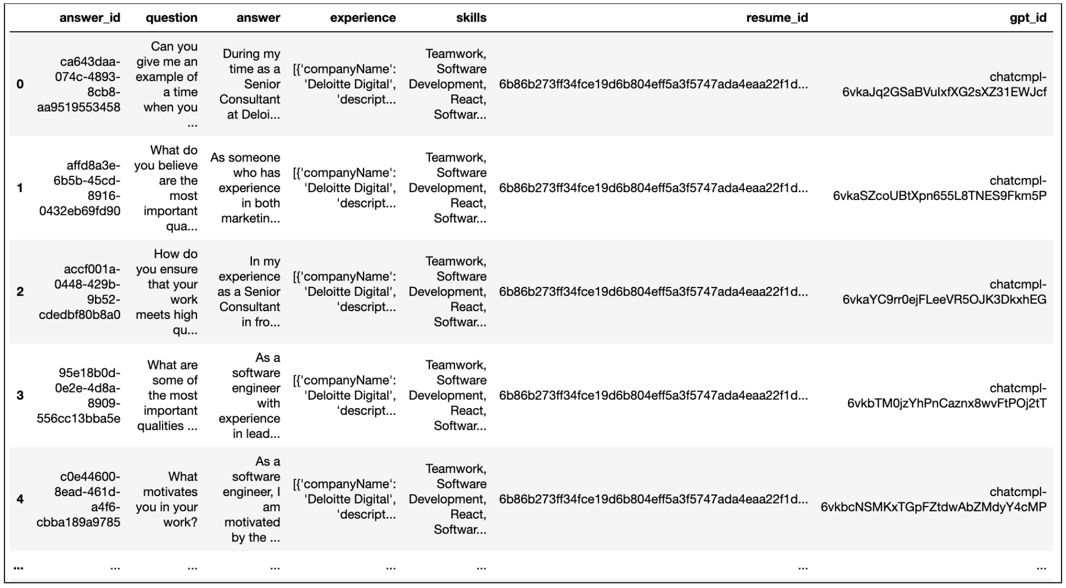
Are you legally eligible to work in the country you are applying for? *

Select...

CLIENT EXPERIENCE - BPO PARTNERSHIPS

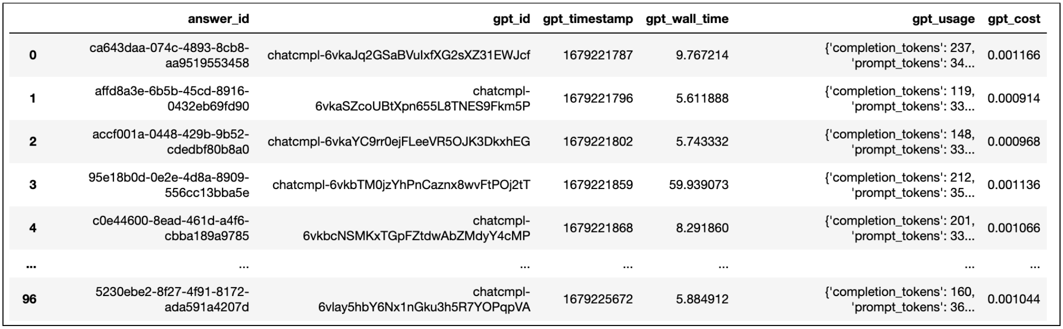
Please tell us about your experience scaling Client Support teams via offshore contact centre partnerships. *

Figure 9: Typical web form question box from a job application website (Wealth Simple, using Lever ATS). Task 4 is able to autogenerate responses to these types of questions which the applicant can insert in.



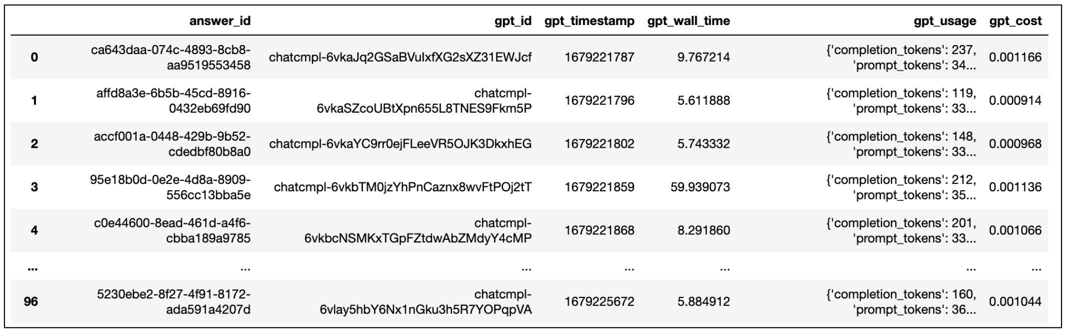
	answer_id	question	answer	experience	skills	resume_id	gpt_id
0	ca643daa-074c-4893-8cb8-aa9519553458	Can you give me an example of a time when you ...	During my time as a Senior Consultant at Deloi...	[{'companyName': 'Deloitte Digital', 'descript...	Teamwork, Software Development, React, Softwar...	6b86b273ff34fce19d6b804eff5a3f5747ada4eaa22f1d...	chatcmpl-6vkaJq2GSaBVulxfXG2sXZ31EWJcf
1	affd8a3e-6b5b-45cd-8916-0432eb69fd90	What do you believe are the most important qua...	As someone who has experience in both marketin...	[{'companyName': 'Deloitte Digital', 'descript...	Teamwork, Software Development, React, Softwar...	6b86b273ff34fce19d6b804eff5a3f5747ada4eaa22f1d...	chatcmpl-6vkaSZcoUBtXpn655L8TNES9Fkm5P
2	accf001a-0448-429b-9b52-cdedbf80b8a0	How do you ensure that your work meets high qu...	In my experience as a Senior Consultant in fro...	[{'companyName': 'Deloitte Digital', 'descript...	Teamwork, Software Development, React, Softwar...	6b86b273ff34fce19d6b804eff5a3f5747ada4eaa22f1d...	chatcmpl-6vkaYC9rr0ejFLeeVR5OJK3DkxhEG
3	95e18b0d-0e2e-4d8a-8909-556cc13bba5e	What are some of the most important qualities ...	As a software engineer with experience in lead...	[{'companyName': 'Deloitte Digital', 'descript...	Teamwork, Software Development, React, Softwar...	6b86b273ff34fce19d6b804eff5a3f5747ada4eaa22f1d...	chatcmpl-6vkbTM0jzYhPnCaznx8wvFtPOj2tT
4	c0e44600-8ead-461d-a4f6-cbba189a9785	What motivates you in your work?	As a software engineer, I am motivated by the ...	[{'companyName': 'Deloitte Digital', 'descript...	Teamwork, Software Development, React, Softwar...	6b86b273ff34fce19d6b804eff5a3f5747ada4eaa22f1d...	chatcmpl-6vkbcNSMKxTGpFZtdwAbZMdyY4cMP
...	...	...	...	...	...	...	...

Figure 10: Data frame object of the question-answer pairs corpus. Data generation works by generating an answer for each question, for each resume in the resume corpus (Figure 5). This is why the resume id is the same for the first four rows. Appendix C.1. shows a printout of the full answer for row 2.



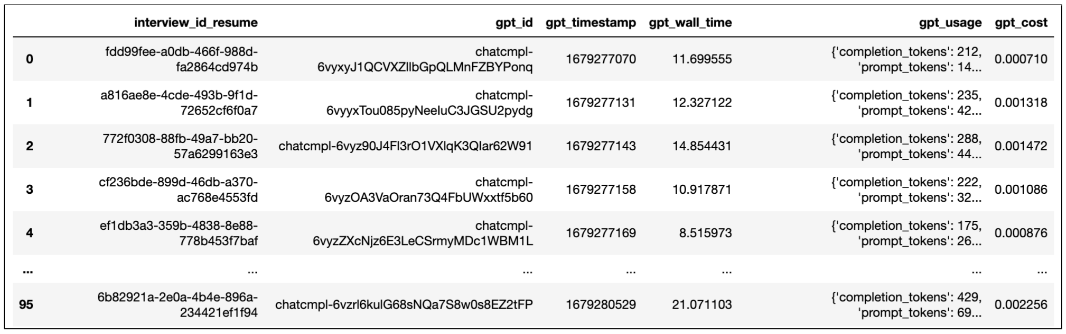
	answer_id	gpt_id	gpt_timestamp	gpt_wall_time	gpt_usage	gpt_cost
0	ca643daa-074c-4893-8cb8-aa9519553458	chatcmpl-6vkaJq2GSaBVulxfXG2sXZ31EWJcf	1679221787	9.767214	{'completion_tokens': 237, 'prompt_tokens': 34...	0.001166
1	affd8a3e-6b5b-45cd-8916-0432eb69fd90	chatcmpl-6vkaSZcoUBtXpn655L8TNES9Fkm5P	1679221796	5.611888	{'completion_tokens': 119, 'prompt_tokens': 33...	0.000914
2	accf001a-0448-429b-9b52-cdedbf80b8a0	chatcmpl-6vkaYC9rr0ejFLeeVR5OJK3DkxhEG	1679221802	5.743332	{'completion_tokens': 148, 'prompt_tokens': 33...	0.000968
3	95e18b0d-0e2e-4d8a-8909-556cc13bba5e	chatcmpl-6vkbTM0jzYhPnCaznx8wvFtPOj2tT	1679221859	59.939073	{'completion_tokens': 212, 'prompt_tokens': 35...	0.001136
4	c0e44600-8ead-461d-a4f6-cbba189a9785	chatcmpl-6vkbcNSMKxTGpFZtdwAbZMdyY4cMP	1679221868	8.291860	{'completion_tokens': 201, 'prompt_tokens': 33...	0.001066
...	...	...	...	...	...	...
96	5230ebe2-8f27-4f91-8172-ada591a4207d	chatcmpl-6vlay5hbY6Nx1nGku3h5R7YOPqpVA	1679225672	5.884912	{'completion_tokens': 160, 'prompt_tokens': 36...	0.001044

Figure 11: Data frame object of the analytics collected for the question-answer generation jobs. gpt wall time is the execution time, gpt usage stores the prompt tokens used by gpt-3.5-turbo, and gpt cost is the cost in dollars.



	answer_id	gpt_id	gpt_timestamp	gpt_wall_time	gpt_usage	gpt_cost
0	ca643daa-074c-4893-8cb8-aa9519553458	chatcmpl-6vkaJq2GSaBVulxfXG2sXZ31EWJcf	1679221787	9.767214	{'completion_tokens': 237, 'prompt_tokens': 34...	0.001166
1	affd8a3e-6b5b-45cd-8916-0432eb69fd90	chatcmpl-6vkaSZcoUBtXpn655L8TNES9Fkm5P	1679221796	5.611888	{'completion_tokens': 119, 'prompt_tokens': 33...	0.000914
2	accf001a-0448-429b-9b52-cdedbf80b8a0	chatcmpl-6vkaYC9rr0ejFLeeVR5OJK3DkxhEG	1679221802	5.743332	{'completion_tokens': 148, 'prompt_tokens': 33...	0.000968
3	95e18b0d-0e2e-4d8a-8909-556cc13bba5e	chatcmpl-6vkbTM0jzYhPnCaznx8wvFtPOj2tT	1679221859	59.939073	{'completion_tokens': 212, 'prompt_tokens': 35...	0.001136
4	c0e44600-8ead-461d-a4f6-cbba189a9785	chatcmpl-6vkbcNSMKxTGpFZtdwAbZMdyY4cMP	1679221868	8.291860	{'completion_tokens': 201, 'prompt_tokens': 33...	0.001066
...	...	...	...	...	...	...
96	5230ebe2-8f27-4f91-8172-ada591a4207d	chatcmpl-6vlay5hbY6Nx1nGku3h5R7YOPqpVA	1679225672	5.884912	{'completion_tokens': 160, 'prompt_tokens': 36...	0.001044

Figure 12: Data frame object of the interview question set generated for 3 different resumes. The 'target' is the original question, while the field 'questions' is a list of generated interview questions. Appendix C.2 shows a print out of two of these interview question examples.



	interview_id_resume	gpt_id	gpt_timestamp	gpt_wall_time	gpt_usage	gpt_cost
0	fdd99fee-a0db-466f-988d-fa2864cd974b	chatcmpl-6vyxyJ1QCVXZIlbGpQLMnFZBYPonq	1679277070	11.699555	{'completion_tokens': 212, 'prompt_tokens': 14...	0.000710
1	a816ae8e-4cde-493b-9f1d-72652cf6f0a7	chatcmpl-6vyyxTou085pyNeeluC3JGSU2pydg	1679277131	12.327122	{'completion_tokens': 235, 'prompt_tokens': 42...	0.001318
2	772f0308-88fb-49a7-bb20-57a6299163e3	chatcmpl-6vyz90J4Fl3rO1VXlqK3QIar62W91	1679277143	14.854431	{'completion_tokens': 288, 'prompt_tokens': 44...	0.001472
3	cf236bde-899d-46db-a370-ac768e4553fd	chatcmpl-6vyzOA3VaOran73Q4FbUWxxtf5b60	1679277158	10.917871	{'completion_tokens': 222, 'prompt_tokens': 32...	0.001086
4	ef1db3a3-359b-4838-8e88-778b453f7baf	chatcmpl-6vyzZXcNjz6E3LeCSrmyMDc1WBM1L	1679277169	8.515973	{'completion_tokens': 175, 'prompt_tokens': 26...	0.000876
...	...	...	...	...	...	...
95	6b82921a-2e0a-4b4e-896a-234421ef1f94	chatcmpl-6vzrl6kulG68sNQa7S8w0s8EZ2tFP	1679280529	21.071103	{'completion_tokens': 429, 'prompt_tokens': 69...	0.002256

Figure 13: Data frame object of the analytics collected for the interview question generation jobs. gpt wall time is the execution time, gpt usage stores the prompt tokens used by gpt-3.5-turbo, and gpt cost is the cost in dollars.

Appendix C

C.1 A printout of a cover letter response of Task 3 is shown below. Here, the cover letter is conditioned on the resume, and job posting, as evident through keywords of the applicant's previous work experience appearing in the generated cover letter:

"In my experience as a Senior Consultant in front-end engineering at Deloitte Digital, I have learned the importance of following rigorous quality standards. I ensure that my work is of high quality by utilizing my skills in Agile methodologies, software development using React, Java, JavaScript, Node.js, Cascading Style Sheets (CSS), Redux, and Git. My leadership skills also play a key role in coordinating with team members to ensure that we work together effectively to produce a high-quality end product. Additionally, my previous experiences as a behavioral aid for children with autism and as a quality assurance assistance intern at Computer Modelling Group Ltd. have honed my attention to detail and ability to maintain and automate test cases to ensure consistent, high-quality performance."

C.2 A printout of two interview questions generated conditional on the applicant's resume:

"1. Can you walk us through your experience working as a Senior Software Developer at Scotiabank?", and

"2. How have you utilized your skills in Linux and COMSOL to the University of Texas at Dallas?"