RESUMEFLOW: An LLM-facilitated Pipeline for Personalized Resume Generation and Refinement

Saurabh Bhausaheb Zinjad
Arizona State University
Tempe, Arizona, USA
szinjad@asu.edu

Amrita Bhattacharjee
Arizona State University
Tempe, Arizona, USA
abhatt43@asu.edu

Amey Bhilegaonkar
Arizona State University
Tempe, Arizona, USA
abhilega@asu.edu

Huan Liu
Arizona State University
Tempe, Arizona, USA
huanliu@asu.edu

ABSTRACT

Crafting the ideal, job-specific resume is a challenging task for many job applicants, especially for early-career applicants. While it is highly recommended that applicants tailor their resume to the specific role they are applying for, manually tailoring resumes to job descriptions and role-specific requirements is often (1) extremely time-consuming, and (2) prone to human errors. Furthermore, performing such a tailoring step at scale while applying to several roles may result in a lack of quality of the edited resumes. To tackle this problem, in this demo paper, we propose RESUMEFLOW: a Large Language Model (LLM) aided tool that enables an end user to simply provide their detailed resume and the desired job posting, and obtain a personalized resume specifically tailored to that specific job posting in the matter of a few seconds. Our proposed pipeline leverages the language understanding and information extraction capabilities of state-of-the-art LLMs such as OpenAI's GPT-4 and Google's Gemini, in order to (1) extract details from a job description, (2) extract role-specific details from the user-provided resume, and then (3) use these to refine and generate a role-specific resume for the user. Our easy-to-use tool leverages the user-chosen LLM in a completely off-the-shelf manner, thus requiring no fine-tuning. We demonstrate the effectiveness of our tool via a video demo and propose novel task-specific evaluation metrics to control for alignment and hallucination. Our tool is available at RESUMEFLOW.

CCS CONCEPTS

• **Applied computing → Document management and text processing**; • **Human-centered computing**;

KEYWORDS

Large Language Models, Automated Resume Generation, Information Extraction, Personalization, Prompt Engineering, AI Persona

1 INTRODUCTION

One of the most important steps for a candidate on the job market to focus on is crafting a good quality, impactful resume. Condensing years of education, projects and unique experiences into a short 1 or 2 page document while still highlighting the uniqueness and individuality of the candidate is a quite challenging task. Given the increasing number of candidates applying to jobs, employers often rely upon automated applicant tracking systems (ATS) and resume filtering systems that filter out candidates based on the degree of match to a job opening. In order to deal with these automated systems, candidates now need to spend time and effort crafting and tailoring their resume to the specific job posting, such as retaining only relevant experience and skills, highlighting job-specific projects, highlighting achievements, etc. This is a challenging and extremely time-consuming step and many candidates struggle with identifying and using impactful keywords, keeping the resume concise, etc [14, 21]. While there are some automated tools for analyzing and retrieving information from candidate resumes, there is currently no automated solution via which a job applicant can tailor their resume to some specific job they are interested in. To solve this issue, in this work we propose a tool called RESUMEFLOW that enables the user to simply use their general-purpose resume and tailor it to a specific job that the user is interested in, thereby potentially saving hours of effort for the applicant.

Recent advancements in language modeling and particularly in the direction of instruction-tuned Large Language Models (LLMs) have demonstrated the vast capabilities of such models. Alongside improved performance on standard natural language tasks [6], LLMs have also been shown to have impressive performance in many challenging use-cases [4, 5]. Inspired by the impressive instruction following capabilities [26] of recent LLMs, such as OpenAI's GPT-3.5, GPT-4 [1] and Google Deepmind's Gemini [20], we aim to leverage the textual understanding, instruction following, and generation capabilities of these recent LLMs to facilitate the

arXiv:2402.06221v2 [cs.CL] 8 May 2024

Saurabh Bhausaheb Zinjad, Amrita Bhattacharjee, Amey Bhilegaonkar, and Huan Liu

task of resume tailoring. Therefore, we design and develop ResumeFlow as an end-to-end pipeline that leverages the power of Large Language Models for retrieval, document processing, and text generation, especially in the context of user resumes. Our tool is available at https://job-aligned-resume.streamlit.app and a demo video is also available at this YouTube link.

2 RELATED WORK

With the evolution of various natural language and text mining methods, there have been interesting applications of new methods in tasks such as resume analysis and classification, information retrieval from resumes, etc. Several of these efforts try to parse out and extract information [10, 19], predict skills from resumes [9], perform job matching [24], etc. Authors in [7] propose a two-step approach for information extraction from resumes, by identifying the blocks of information first. However, to the best of our knowledge, there has been no work on automatically generating job-aligned resumes. In the direction of using LLMs as instruction-following assistants, there has been recent work in the direction of using LLMs for information retrieval [23, 27], along with interesting explorations in the field of education [2, 13], text analysis and research [15, 17], etc. Inspired by these works and the overall potential of recent state-of-the-art LLMs, we propose to leverage LLMs in our ResumeFlow pipeline. A recent work that is closest to ours is [18] where the authors generate structured and unstructured resumes using ChatGPT. However, they simply use it as a data generation/augmentation step for a downstream classification task, and there is no aspect of tailoring the resumes to specific jobs.

3 RESUMEFLOW: HOW IT WORKS

In this section, we describe how our ResumeFlow tool works in the backend and describe each of the components in detail. Figure 1 presents an overview of ResumeFlow, our proposed LLM-facilitated pipeline for personalized resume generation. The system comprises three core components:

- **User Data Extractor:** This module transforms a user-provided baseline resume, in the form of a PDF document, into a structured JSON format suitable for subsequent processing.
- **Job Details Extractor**: This module analyzes the job description, provided as a string by simply copy-pasting the job posting, to extract keywords, requirements, etc., in the form of a JSON, and will be used in the next step for informing the content and tailoring of the generated resume.
- **Resume Generator**: This component is where the bulk of the processing takes place. Leveraging the structured user data and extracted job requirements, this module generates a personalized resume.

The following sections will introduce details of the pipeline.

3.1 User Data Extractor

Our tool allows the user to input their (structured or unstructured) resume as a PDF, along with the job description of the job they are applying to. The user also has the option to choose which LLM to be used for the entire pipeline. In this component, we first convert the resume PDF to a text string and then use the user-chosen LLM as an information extraction model to extract key

sections and important information from the user's resume text. To do this, we simply use the chosen LLM in an off-the-shelf manner without additional training, by using carefully structured prompts that facilitate information extraction. To perform this step, we use a prompt such as (entire prompt is truncated for brevity): "You are a software developer working on a resume parsing application. Your task is to design a system that takes a resume in text format as input and converts it into a structured JSON format."

This allows us to have a structured dictionary of the user's information as extracted by the LLM. We refer to this dictionary as **UserData** in the following sections, for ease of understanding.

3.2 Job Details Extractor

The "Job Details Extractor" accepts job descriptions in text format: the user is instructed to copy-paste the job description from their desired job listing into the textbox on our tool. This component in our pipeline aims to extract the key points from the job description such as: job title, required and preferred qualifications, etc. In order to do this via the LLM, we use two carefully crafted prompts: the system prompt that is meant to induce the 'persona' of an experienced resume writer into the LLM, followed by the task prompt to extract the important job details. More specifically, for the system prompt, we describe a persona specifically that mimics the tone, style, and focus of professional resumes, such as: "You are a seasoned career advising expert in crafting resumes and cover letters, boasting a rich 15-year history dedicated to mastering this skill...". We empirically see that priming the LLM via this prompt greatly improves the performance of the resulting system, since it enables the outputs to adhere to the context and task-specific conventions and language.

Then the task prompt is designed to guide the LLM to extract the important job details from the provided job description and output these as a structured JSON. By combining these prompts, the LLM accurately extracts and structures key information, presented in a structured JSON format. Accurately extracting this data is crucial for the next step of the pipeline - that is the resume tailoring step. This extracted data typically includes essential elements like title, keywords, purpose, responsibilities, qualifications (both required and preferred), company name, and company information. We refer to this JSON of job details as **JobDetails** in the remainder of this paper.

3.3 Resume Generator

This is the main component of our framework, where the processing of the resume takes place and thereby a modified version of the user's resume is generated after being tailored towards the specific job. The two inputs to this component are the **UserData** and the **JobDetails** from the previous two steps. Now, the **UserData** is processed section by section: say a typical resume has the following sections: {Personal Details, Education, Work Experience, Skills, Achievements, Projects}, etc. We choose to process the resume by dividing it into sections since it follows the natural structure of how a standard resume is formatted and organized. Additionally, LLMs have restrictions on the context length they can handle, and feeding the entire resume text at once would exceed the context window.
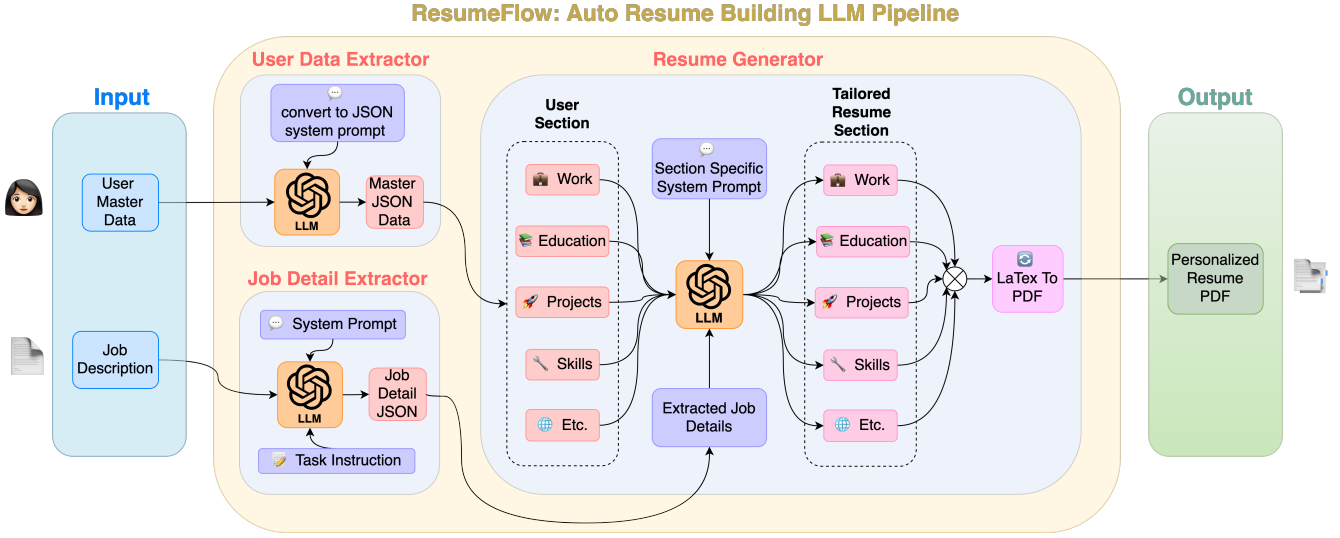
Figure 1: Our overall RESUMEFLOW pipeline for LLM-facilitated generation of job-aligned resumes.

Furthermore, LLMs have been shown to struggle to extract information from very long contexts, and often overlook information that is in the middle of a long context [12]. Therefore processing each section at a time alleviates all these issues. We iterate over each of the **UserData** sections as follows: first, the personal details section is extracted without change, since we do not want the LLM to change any factual information that is in this section, such as phone number, address, or name. Next, we loop over the sections in **UserData**, prompt the chosen LLM with a section-specific prompt along with the **UserData** section data and the **JobDetails**. For each section, the LLM is tasked with re-creating the section after analyzing the user's resume and retaining, emphasizing, or removing points to make that section more aligned with the specific job. At the end of processing each of these sections, we collect the response of the LLM. Finally, these are combined into a JSON.

The system prompt we use here contains instructions on how to craft the resume based on the job description in **JobDetails**. To do this we use best practices as highlighted by career counselors and professional resume writers as in [3, 16]. More specifically, we carefully instruct the LLM to follow these principles:

(1) *Focus*: Craft relevant achievements aligned with the job description.
(2) *Honesty*: Prioritize truthfulness and objective language.
(3) *Specificity*: Prioritize relevance to the specific job over general achievements.
(4) *Style*:
   (a) *Voice*: Use active voice whenever possible.
   (b) *Proofreading*: Ensure impeccable spelling and grammar.

These carefully tailored instruction-style prompts leverage the instruction-understanding capabilities of these state-of-the-art LLMs and strive to emulate the skill and style of professional resume writers who are experienced in this task.

Our tool provides the user the option to choose which LLM to use in the pipeline. Currently, we have support for OpenAI models
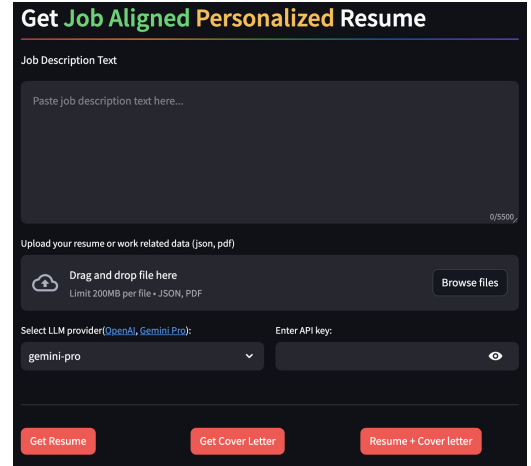

Figure 2: Our easy-to-use User Interface.

(default is `gpt-4-1106-preview`), and Google Deepmind's Gemini [20] [1] (default version is `gemini-pro`). For the `gpt-4` models we use the ChatCompletion backend and we use the `json_object` output format in order to easily handle LLM outputs programmatically, by simply casting these JSON outputs into a Python dictionary. For `gemini`, we use a parsing function we defined to parse the generated output into a JSON format. Finally, we use a LATEXengine to convert this JSON to a visually appealing and professionally formatted PDF document, allowing for customization based on various resume templates. Alongside this resume generation, our tool also gives the user the option to generate a cover letter that is aligned with the job as well.
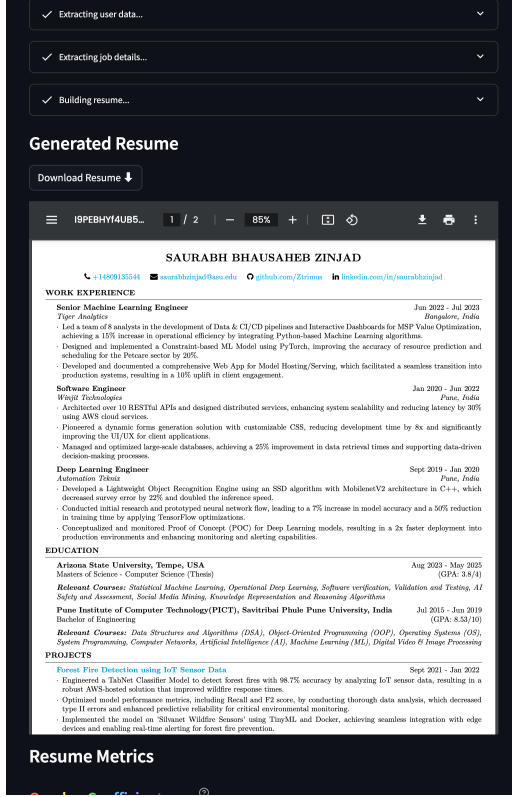
---

[1]https://deepmind.google/technologies/gemini/

Saurabh Bhausaheb Zinjad, Amrita Bhattacharjee, Amey Bhilegaonkar, and Huan Liu

Figure 3: Truncated screenshot of RESUMEFLOW generating a job-aligned resume.

4 EVALUATION

In this section, we describe how we evaluate our system on the basis of *goodness* of the generated resumes. Unlike standard natural language processing tasks that have well-defined benchmarks and evaluation metrics, automated evaluation of such a system solving this novel task is quite challenging. However, we introduce two metrics to evaluate generated resumes on two dimensions: *job_alignment* and *content_preservation*. We formulate these metrics as:

job_alignment: This metric measures the alignment between the generated resume and the job details. Ideally, we would want a higher score since this would mean the resume generated by our tool is well-tailored to the specific job the user is applying to. We measure this in both the token space and the latent space. In the token space, we define $job\_alignment_{token}$ as the overlap coefficient [22] between the unique words in the generated resume and the unique words in the job description. In the latent space, we define $job\_alignment_{latent}$ as the cosine similarity between vector embeddings of the generated resume and the job description text. We use Gemini and OpenAI embeddings[2] to embed both the texts. We formulate these as:

$$job\_alignment_{token} = \frac{|\mathcal{W}(\mathcal{D}^{gen}) \cap \mathcal{W}(\mathcal{J})|}{min(|\mathcal{W}(\mathcal{D}^{gen})|, |\mathcal{W}(\mathcal{J})|)} \quad (1)$$

where $\mathcal{D}^{gen}$ refers to the resume generated by our tool, $\mathcal{J}$ refers to the job description, $\mathcal{W}(x)$ refers to the set of unique words in x.

$$job\_alignment_{latent} = cosine\_sim(\mathcal{E}(\mathcal{D}^{gen}), \mathcal{E}(\mathcal{J}))$$

where $cosine\_sim(\cdot, \cdot)$ refers to the cosine similarity, $\mathcal{E}(\cdot)$ refers to the encoding used to embed the text.

content-preservation: Since hallucination is a known issue even in state-of-the-art LLMs, we would want to verify how well our RESUMEFLOW pipeline is able to preserve content between the user-provided resume and the generated one. Ideally, we would want high values for this, since low values of content preservation might imply hallucination by the LLM. Similar to the *job_alignment* metric, we measure in both the token space and the latent space. In the token space, we formulate this as the overlap coefficient between the user's original resume and the RESUMEFLOW generated resume:

$$content\_preservation_{token} = \frac{|\mathcal{W}(\mathcal{D}^{gen}) \cap \mathcal{W}(\mathcal{D}^{user})|}{min(|\mathcal{W}(\mathcal{D}^{gen})|, |\mathcal{W}(\mathcal{D}^{user})|)}$$

where $\mathcal{D}^{user}$ refers to the original resume input by the user to the RESUMEFLOW tool. Similarly, in the latent space, we consider the cosine similarity between the generated resume and the original resume:

$$content\_preservation_{latent} = cosine\_sim(\mathcal{E}(\mathcal{D}^{gen}), \mathcal{E}(\mathcal{D}^{user}))$$

Note that this metric is loosely inspired by summarization metrics such as the ROUGE score commonly used in NLP [11] and also bears similarity with recently proposed hallucination detection metrics [8] such as AlignScore [25].

The combination of these two metrics: *job_alignment* and *content_preservation* provide an idea of how good the generated resume is. Low values of *content-preservation* coupled with high values of *job-alignment* is particularly undesirable since it implies that the LLM has possibly hallucinated major parts of the generated resume in order to make it very suitable for the job description. This particular case might be considered unethical since it may seem dishonest from the user's side. We report these scores on the user interface as well, so the end user gets an idea of how well the generated resume is, before directly using it.

5 CONCLUSION & FUTURE WORK

In this work, we introduce an easy-to-use pipeline RESUMEFLOW to help a job applicant tailor their resume to a specific job posting by simply providing their resume and the corresponding job description to our LLM-facilitated pipeline. While our tool currently supports two state-of-the-art models: GPT-4 and Gemini, future improvements to the tool could incorporate more LLMs, preferably open-source ones. With any LLM-powered system, the users of the system need to be aware of the phenomenon of hallucination and how to mitigate instances of hallucinations. To that end, future work may utilize emerging concepts such as retrieval augmented generation and generation using knowledge graphs.

---

[2]`models/embedding-001` for Gemini, `text-embedding-ada-002` for OpenAI.

REFERENCES

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774* (2023).

[2] Ibrahim Adeshola and Adeola Praise Adepoju. 2023. The opportunities and challenges of ChatGPT in education. *Interactive Learning Environments* (2023), 1–14.

[3] LinkedIn Advice. [n. d.]. What are the most common resume writing challenges? https://www.linkedin.com/advice/1/what-most-common-resume-writing-challenges-skills-resume-writing-fyuje

[4] Amrita Bhattacharjee, Raha Moraffah, Joshua Garland, and Huan Liu. 2023. LLMs as Counterfactual Explanation Modules: Can ChatGPT Explain Black-box Text Classifiers? *arXiv preprint arXiv:2309.13340* (2023).

[5] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712* (2023).

[6] Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. 2023. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology* (2023).

[7] Jie Chen, Chunxia Zhang, Zhendong Niu, et al. 2018. A two-step resume information extraction algorithm. *Mathematical Problems in Engineering* 2018 (2018).

[8] Grant C Forbes, Parth Katlana, and Zeydy Ortiz. 2023. Metric Ensembles For Hallucination Detection. *arXiv preprint arXiv:2310.10495* (2023).

[9] Kameni Florentin Flambeau Jiechieu and Norbert Tsopze. 2021. Skills prediction based on multi-label resume classification using CNN with model predictions explanation. *Neural Computing and Applications* 33 (2021), 5069–5087.

[10] XiaoWei Li, Hui Shu, Yi Zhai, and ZhiQiang Lin. 2021. A method for resume information extraction using bert-bilstm-crf. In *2021 IEEE 21st International Conference on Communication Technology (ICCT)*. IEEE, 1437–1442.

[11] Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*. 74–81.

[12] Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2023. Lost in the middle: How language models use long contexts. *arXiv preprint arXiv:2307.03172* (2023).

[13] Chung Kwan Lo. 2023. What is the impact of ChatGPT on education? A rapid review of the literature. *Education Sciences* 13, 4 (2023), 410.

[14] Resume Mansion. 2023. 10 challenges of creating a master resume. https://www.linkedin.com/pulse/10-challenges-creating-master-resume-resumemansion/

[15] Jesse G Meyer, Ryan J Urbanowicz, Patrick CN Martin, Karen O'Connor, Ruowang Li, Pei-Chen Peng, Tiffani J Bright, Nicholas Tatonetti, Kyoung Jae Won, Graciela Gonzalez-Hernandez, et al. 2023. ChatGPT and large language models in academia: opportunities and challenges. *BioData Mining* 16, 1 (2023), 20.

[16] Geneveive Northup. 2023. 10 Resume Writing Tips To Help You Land a Position. https://www.indeed.com/career-advice/resumes-cover-letters/10-resume-writing-tips

[17] Soumen Pal, Manojit Bhattacharya, Sang-Soo Lee, and Chiranjib Chakraborty. 2023. A domain-specific next-generation large language model (LLM) or ChatGPT is required for biomedical engineering and research. *Annals of Biomedical Engineering* (2023), 1–4.

[18] Panagiotis Skondras, George Psaroudakis, Panagiotis Zervas, and Giannis Tzimas. 2023. Efficient Resume Classification through Rapid Dataset Creation Using ChatGPT. In *2023 14th International Conference on Information, Intelligence, Systems & Applications (IISA)*. IEEE, 1–5.

[19] VV Satyanarayana Tallapragada, V Sushma Raj, U Deepak, P Divya Sai, and T Mallikarjuna. 2023. Improved Resume Parsing based on Contextual Meaning Extraction using BERT. In *2023 7th International Conference on Intelligent Computing and Control Systems (ICICCS)*. IEEE, 1702–1708.

[20] Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805* (2023).

[21] Adrienne Tom. 2022. 4 common resume writing challenges - with solutions! https://careerimpressions.ca/4-common-resume-writing-challenges-and-solutions/

[22] MK Vijaymeena and K Kavitha. 2016. A survey on similarity measures in text mining. *Machine Learning and Applications: An International Journal* 3, 2 (2016), 19–28.

[23] Xiang Wei, Xingyu Cui, Ning Cheng, Xiaobin Wang, Xin Zhang, Shen Huang, Pengjun Xie, Jinan Xu, Yufeng Chen, Meishan Zhang, et al. 2023. Zero-shot information extraction via chatting with chatgpt. *arXiv preprint arXiv:2302.10205* (2023).

[24] Xing Yi, James Allan, and W Bruce Croft. 2007. Matching resumes and jobs based on relevance models. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. 809–810.

[25] Yuheng Zha, Yichi Yang, Ruichen Li, and Zhiting Hu. 2023. AlignScore: Evaluating Factual Consistency with a Unified Alignment Function. *arXiv preprint arXiv:2305.16739* (2023).

[26] Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. 2023. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911* (2023).

[27] Yutao Zhu, Huaying Yuan, Shuting Wang, Jiongnan Liu, Wenhan Liu, Chenlong Deng, Zhicheng Dou, and Ji-Rong Wen. 2023. Large language models for information retrieval: A survey. *arXiv preprint arXiv:2308.07107* (2023).