Synthetic Data Fine-Tuning for Effective Team Formation in Enterprises

Guilherme Drummond, Adriano Veloso
[1]Instituto Kunumi, Belo Horizonte, Brazil
[2]Department of Computer Science, Universidade Federal de Minas Gerais, Brazil
{guilherme.lima, adriano}@kunumi.com

Abstract

We evaluate the effectiveness of synthetic data fine-tuning for Semantic Search in a real-world Enterprise Team Formation problem scenario. In this problem, we aim to retrieve the best employee for a given task, given their information regarding abilities, experiences, and other aspects. We evaluate two synthetic data generation strategies: (1) augmenting real-world data with synthetic labels and (2) generating synthetic profiles for employees tailored to specific tasks. To measure the impact of these strategies, we fine-tune a pretrained text embedding model using LoRA and Rank Aggregation techniques. We evaluate the model performance against current SOTA algorithms on a human-curated dataset. Our experiments indicate that training a model that uses a combination of both Synthetic data generation strategies outperforms already established pre-trained models on the Team Formation task, improving the ranking metrics by an average of 30% in comparison to the best-performing pre-trained model.

1 Introduction

A semantic search system processes text queries to retrieve and rank related documents. This ranking process is based on extracting underlying semantic relationships between the queries and the content of the documents by using text embeddings (Mikolov et al., 2013; Pennington et al., 2014) to calculate the query-document similarities. This technique enables a more nuanced understanding of user intent and context. Search-based Information Retrieval systems often use this type of algorithm for reranking (Nogueira et al., 2019; Ma et al., 2023a) and vector-based search (Johnson et al., 2019).

The advent of Word Embeddings allowed search systems to measure semantic similarity between vectors rather than a naive lexical overlapping. The arrival of deep learning and transformer-based models like BERT (Devlin et al., 2019) further revolutionized the field, enabling embeddings to capture the meanings of words and their contextual usage within sentences. Decoder-based LLMs have recently been fine-tuned to perform dense retrieval tasks (Ma et al., 2023a; Xiao et al., 2023; Lee et al., 2025; Wang et al., 2024a), achieving the current state-of-the-art (SOTA) results.

Although semantic search algorithms are typically trained on open general-purpose datasets (Wang et al., 2024a), this widely-used approach demonstrates limited effectiveness when applied to specialized domains. One straightforward solution to this problem is to fine-tune pre-trained models on specialized datasets for the specific domain. However, building a training dataset can become expensive, as field specialists often need to label large amounts of data for effective fine-tuning.

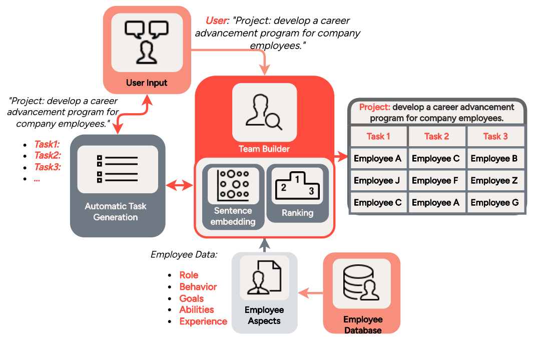


Figure 1: Schematic for the full team formation application pipeline. To create the team, we extract employee data from an internal database and use a sentence embedding model to create the rankings for each task.

With LLMs, generating high amounts of high-quality textual data became possible. This opens possibilities of using synthetic textual data to fine-tune Semantic Search Models for specific contexts. This work will focus on one domain-specific task that uses semantic search: the **Enterprise Team Formation** problem. This problem involves selecting the best employee for a given task based on their abilities, experiences, objectives, and other aspects. Figure 1 shows a schematic diagram of our developed Enterprise Team Formation framework. We start with an end-user inputting a project. Then, the system breaks it into tasks that the user can edit. When the user is satisfied with the tasks, we use all available employee data to calculate the best rankings for each task.

To create high-quality data for this domain, we need a specialist involved in multiple enterprise contexts who is familiar with all employees' positive and negative aspects. To tackle this dependence on a specialist for labeling, we propose two different strategies for synthetic data generation:
Augmenting real-world data with synthetic labels: We generate synthetic tasks and use a large language model (LLM) to label employees as relevant or irrelevant for those tasks.
Generating synthetic employee profiles: We use an LLM to generate the ideal employee curriculum for each of the generated tasks.

We fine-tune a Qwen2-based (Li et al., 2024) semantic search model (Stella-400M (Zhang, 2024)) using Low-Rank Adapters (Hu et al., 2022). Our experiments compare the fine-tuned models against current SOTA pre-trained models across multiple ranking metrics. The results show that our best fine-tuned model achieved a relative improvement of over 35% in nDCG and 30% in Average Precision compared to strong baselines.

The data used in our study was obtained with a leading Brazilian company specialized in wood paneling, ceramic tiles, and bathroom fixtures. The company has more than 10,000 employees, of which more than 1,000 are employees on strategic roles (Senior-level +). Another challenge is the wide variety of contexts within the same enterprise, such as factories, office, sales, and many others. For this project, we focus on creating strategic teams composed of only senior and management-level employees to tackle strategic projects.

2 Related Work

2.1 Improving Language Models with Synthetic Data

Recent advances have shown that synthetic data can significantly boost the performance of language models across various NLP tasks. For example, synthetic data can be used to create improved general-purpose text-embedding models (Wang et al., 2024a), substituting a human evaluator in preference optimization (Guo et al., 2024; Dong et al., 2024), or even to prevent language models to hallucinate (Jones et al., 2024).

2.2 Domain-specific Language Modeling

Domain-specific Language Modeling aims to efficiently adapt pre-trained models to specific domains without losing generalization on general-purpose tasks. MixDA (Diao et al., 2023) addresses this by decoupling the feed-forward networks of Transformers into frozen pre-trained components and dynamic, domain-specific adapters. The adapter networks improve the model performance in out-of-domain and knowledge-intensive tasks while maintaining the performance across in-domain tasks.

Complementary approaches focus on the selective integration of synthetic data for domain adaptation. QVE (Yue et al., 2022) uses synthetic data to improve Question-Ansewring models in low-resource settings. Meanwhile, MathGenie (Lu et al., 2024) proposes a pipeline that combines iterative solution augmentation, question back-translation, and verification-based filtering to generate reliable math problems.

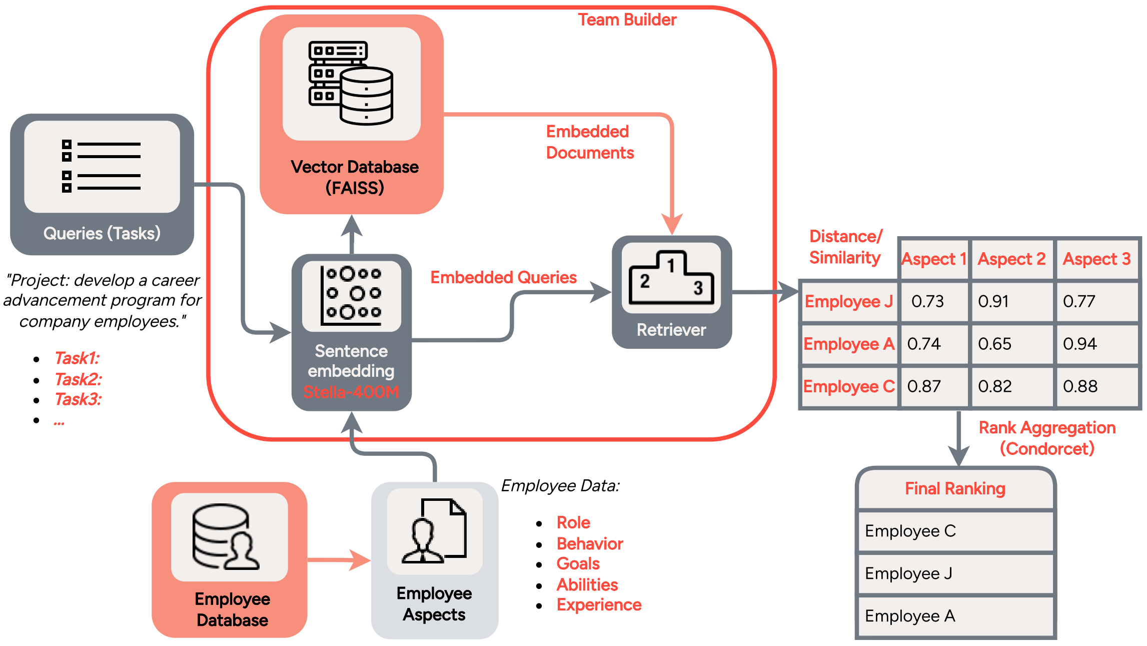
3 Problem Statement

We model the Enterprise team Formation problem as a Semantic Search task. This task revolves around finding the most relevant **documents** to a **query** by calculating the semantic similarity between their vector representations in a shared space. To create an effective team for a given project, we assume a different set of tasks that are tied to that project. These tasks can be viewed as queries in the following format:

$$q_{i,j} = \text{“Project: } \{p_i\}\text{. Task: } \{t_{i,j}\}\text{"} \qquad (1)$$

where each query $q_{i,j}$ is the concatenation of the parent project p_i and a corresponding task $t_{i,j}$.

We model the documents as the available employee data in the enterprise's internal dataset. This data contains employee aspects such as their abilities, past experiences, and goals. We can create an extensive document that is the textual concatenation of all these aspects, or we can treat them separately, aggregating the multiple generated ranks into a single consensus ranking. Formally, given a query text q and a set of N documents $D = \{d_1, d_2, ..., d_n\}$, we map each document d_i into a vector representation v_{d_i} using an embedding function f_d, where $v_{d_i} = f_d(d_i)$. Similarly, we get



	Aspect 1	Aspect 2	Aspect 3
Employee J	0.73	0.91	0.77
Employee A	0.74	0.65	0.94
Employee C	0.87	0.82	0.88

Final Ranking
Employee C
Employee J
Employee A

Figure 2: The complete semantic search pipeline at inference. First, we use a trained sentence embedding model to create the vector embeddings for both queries (tasks) and documents (employees). We generate multiple rankings (one for each employee aspect) based on the cosine similarity between query and documents. Finally, we use a rank aggregation algorithm to create the final Ranking.

the query vector representation v_q of the original query q by using a embedding function f_q, where $v_q = f_q(q)$.

Often, f_q and f_d are the same function but can be distinct in some cases (e.g., an asymmetric Dual-Encoder). In our implementation, we use the Siamese dual-encoder (SDE) architecture, as it is simpler to train (shared weights) and generally outperforms other dual-encoder architectures (Dong et al., 2022). The ranking for a task $q_{i,j}$ is then determined by the pairwise similarities between the query and documents ($\phi(q_{i,j}, d_k)$), ordered from best to worst scores.

Using multiple textual features means a query may generate several rankings. Rank Aggregation algorithms leverage this set of rankings, aggregating them into a single final consensus rank (Dwork et al., 2001; Wang et al., 2024b). Formally, given a set of N items to be ranked $U = \{u_1, u_2, ..., u_N\}$, we define an arbitrary ranking $R^t = \{u_i > u_j > ... > u_k\}, i \neq j \neq k$, with $R^t(u_i)$ denoting the position of item u_i in the ranking. Thus, if $R^t(u_i) < R^t(u_j)$, then u_i is more relevant than u_j under R^t. Given a set of M different basic rankings $\mathcal{R} = \{R^1, R^2, R^3, ...R^M\}$, we denote an aggregated consensus ranking R^* as $R^* = f(\mathcal{R})$, where f is a Rank Aggregation function. Here, we focus on the Condorcet method (de Condorcet, 1785), which has interesting properties such as granting the choice of the overall best item among all ranks when possible (Young and Levenglick, 1978).

Figure 2 shows the complete pipeline for generating the final ranking for a given task. First, we use a Sentence Embedding model to calculate the embeddings for a query and all documents. The document embeddings are pre-calculated and stored in a vector database. Then, we calculate the similarities between query and aspect documents, generating multiple rankings, one for each aspect. These multiple rankings are then aggregated using the Condorcet Rank Aggregation algorithm (de Condorcet, 1785) to create the final Rank for that given task.

4 Method

4.1 Data

The data we used includes employee information from a large Brazilian company, specifically focusing on individuals in senior management roles within the enterprise, which are strategically important to the company. This dataset includes 835 employees. We also create a task-employee human feedback dataset from scratch for our proposed methods to use as an evaluation dataset.

Employee Data: We collect multiple textual features for each employee, which are referred to as *aspects* in this work. These aspects refer to abilities, past working experiences, behavior, and goals, and next we detail each of these aspects.

Role: employees' role in the company alongside a description of their responsibilities.

Behavior: The employee's most recent behavior evaluation. The direct boss of each employee performs this evaluation yearly.

Goals: The established goals for each employee. This aspect has a mixture of personal goals and goals established by the bosses.

Abilities: Relevant professional abilities.

Experience: Employee's previous professional experiences within or outside the company.

This dataset includes 835 employees with strategic roles, such as coordinators and managers. Although these individuals represent a small portion of the company, optimizing team performance in these roles has an outsized impact on the organization's overall success. In total, there are 280 Coordinators, 247 Senior employees, 222 Supervisors, 83 Managers, and 3 Directors.

Human Feedback: To validate our model's performance on real-world applications, we have established a human-curated evaluation dataset alongside our synthetic training data. We built this curated dataset using two distinct applications where domain experts could build teams for their desired project or label the relevance of specific employees for a given task. In total, we collected over 500 labeled query-document pairs.

4.2 Synthetic Data Generation

To fine-tune our model, we test two different approaches to generating synthetic examples. In the first approach, we use an LLM to generate synthetic labels for a set of query-document pairs. In the second approach, we leverage all queries, which are tasks within projects, and generate an ideal employee for each task. For both approaches, we first need to generate diverse projects and tasks relevant to the company's context. For all data generation tasks, we use OpenAI's GPT-4o-mini API (OpenAI, 2024b,a).

Synthetic Tasks: To generate the synthetic tasks, we build a two-step generation process. First, we prompt the LLM to brainstorm a pool of projects. To ensure the relevance and diversity of projects, we provide a contextualizing text to the prompt, presenting the enterprise areas and main corporate activities. We also sample a set of abilities present in our employee dataset to ensure that the LLM generates projects for the various kinds of abilities present in the company. This brainstorming process is done multiple times, with different abilities each time. We ensure the output format is a Python list that can be used in further steps. For the second step, we feed the projects into a second prompt to extract a set of tasks for each generated project. We guide the generation process by presenting a one-shot example from a human-written project (extracted from our user application).

Synthetic Curriculums: Our initial idea with the Synthetic curriculums was to generate both positive and hard negative examples for each task. However, we found that the LLMs struggle to generate negative examples, as stated in previous work (García-Ferrero et al., 2023; Hossain et al., 2020; Truong et al., 2022). Therefore, we changed our prompt to generate only positive examples, and during training, we used the in-batch examples as negatives, which is shown to be a strong alternative to labeled hard negative examples (Chen et al., 2020; Ye et al., 2019; Doersch and Zisserman, 2017). We generated one synthetic curriculum for each of the generated tasks, totaling $\approx$ 32000 curriculums.

Synthetic Labeling: The synthetic labeling process comprises two key steps: task and employee sampling, followed by the application of a Chain-of-Thought (CoT) prompt (Wei et al., 2022) to generate the labels. Using the synthetic tasks generated previously, we use Stella-400M to generate the 20 best-ranked employees for each task. To create the task-employee pairs, we first randomly sample the tasks. The associated employee has a 50% chance of being sampled from the top 20 and a 50% chance of being sampled from the whole database.

After generating the pairs, we feed them to the LLM with the labeling prompt. Similar to the task generation process, we add the enterprise context to the start of the prompt. Then, we guide its generation process with strict analysis guidelines, such as discussing the positive and negative aspects of that employee regarding the task, followed by a competency analysis. In the conclusion, the LLM must give the employee a score of 0 (irrelevant) or 1 (relevant). Using this process, we generated $\approx$ 30000 labeled task-employee pairs.

To get this prompt, we perform an **optimization process** using the Eureka framework (Ma et al., 2023b). Briefly, Eureka is an evolutionary search

	Avg. Prec	nDCG@1	nDCG@5	Hit@5	AUC
BM25 (Robertson et al., 1994)	0.081	0.020	0.076	0.183	0.670
TF-IDF (Salton and McGill, 1983)	0.169	0.101	0.172	0.305	0.683
e5-large-v2 (Wang et al., 2022)	0.141	0.060	0.117	0.243	0.740
Sentence-T5-large (Ni et al., 2021)	0.160	0.044	0.144	0.280	0.742
OpenAI-text-embedding-3$_{\text{small}}$ (OpenAI, 2023)	0.288	0.165	0.305	0.481	0.747
OpenAI-text-embedding-3$_{\text{large}}$ (OpenAI, 2023)	0.296	0.160	0.303	0.460	0.782
bge-large-en-v1.5 (Xiao et al., 2023)	0.195	0.105	0.188	0.322	0.703
gte-Qwen2-1.5B-instruct (Li et al., 2023)	0.236	0.143	0.220	0.340	0.734
Stella-400M (Zhang, 2024)	0.284	0.150	0.285	0.481	0.758
Stella-LoRA$_{\text{Condorcet}}$ + full data	**0.386**	0.217	**0.418**	**0.620**	**0.798**
with synthetic curriculums only	0.296	0.140	0.323	0.544	0.758
with synthetic labels only	0.358	**0.260**	0.393	0.581	0.771
Concat Model	0.352	0.220	0.364	0.565	0.734

Table 1: Overall results of model performance on our evaluation dataset.

that leverages an LLM to optimize prompts for specific tasks. We model the relevance labeling as a prompt refinement task, providing a base handcrafted prompt as input. To effectively guide this optimization, we provided evaluation metrics, including accuracy, precision, and recall scores from the best-performing prompt, along with representative examples spanning TPs, TNs, FPs and FNs. Appendix A provides a detailed summary of this process and the resulting prompts.

Limitations of synthetic supervision: Since synthetic relevance labels are generated using candidates pre-ranked by the base encoder, the resulting supervision may partially reflect the inductive biases of the initial model. Accordingly, the observed gains should be interpreted as improved domain alignment for the chosen encoder rather than model-agnostic retrieval improvements. Future work will explore whether similar benefits hold across alternative embedding architectures and enterprise contexts.

4.3 Modeling

We use Stella-400M (Zhang, 2024) as our pretrained sentence embedding encoder and use LoRA (Hu et al., 2022) for a parameter-efficient fine-tuning for the Enterprise Team Formation task. Stella-400M, derived from gte-Qwen2-1.5B-instruct (Li et al., 2023), uses *Knowledge Distillation* (Hinton et al., 2015). To obtain the sentence-level embedding, we apply mean pooling over the token embeddings from the final transformer layer. We choose Stella-400M due to it being a very cost-effective model, obtaining one of the top rankings on the MTEB Benchmark (Muennighoff et al., 2023) for sentence embedding tasks while having a low number of parameters. We set LoRA $r = \alpha = 16$, and apply adapters to all linear layers, setting approximately 8 million trainable parameters.

We use a Contrastive Learning approach. The queries represent the anchor embedding, while the employee data (documents) represent the positive (for the relevant example) or negative (for irrelevant/random in-batch examples) embeddings.

Given a positive pair of query-document (q^+, d^+), we apply a simple instruction template to the queries:

$$q^+_{instr} = \text{“Instruct: \{tmpl\} \textbackslash n Query: ”} + q^+ \quad (2)$$

where *“tmpl"* represents the text embedding related task. In our case, we use the text as an instruction template: *“In an enterprise context, given a project and an associated task, retrieve relevant employees that fill that role."*. To train the embedding model, we employ the InfoNCE loss function (van den Oord et al., 2019), which operates on both positive and negative examples, where negative samples can be either hard negatives or in-batch examples. The loss is defined as:

$$\mathbb{L} = -\log \frac{\exp\left(\frac{\phi(q,d^+)}{\tau}\right)}{\exp\left(\frac{\phi(q,v_{d^+})}{\tau}\right) + \sum_{n_i \in \mathcal{N}} \exp\left(\frac{\phi(q,n_i)}{\tau}\right)} \quad (3)$$

where ϕ represents a similarity measure, in our case, the cosine similarity, $n_i \in N$ represents a negative document, and τ is a temperature parameter that controls the separation between positive and negative examples, set as 0.1 in our experiments.

	Avg. Prec	nDCG@1	nDCG@5	Hit@5	AUC
Model performance on removed aspect					
Stella-LoRA$_{\text{Condorcet}}$ $- \textit{Role}$	0.381	**0.260**	0.407	0.60	0.771
Stella-LoRA$_{\text{Condorcet}}$ $- \textit{Behavior}$	0.352	0.245	0.379	0.58	0.762
Stella-LoRA$_{\text{Condorcet}}$ $- \textit{Goals}$	0.381	0.245	0.403	0.58	0.784
Stella-LoRA$_{\text{Condorcet}}$ $- \textit{Abilities}$	0.382	0.250	**0.412**	**0.62**	**0.779**
Stella-LoRA$_{\text{Condorcet}}$ $- \textit{Experience}$	0.373	0.256	0.397	0.58	0.773
Condorcet (Aggregation)	**0.385**	0.250	0.407	0.605	**0.779**

Table 2: Ablation results. At each step, we remove an aspect from training.

Rank Aggregation: Our model generates one rank for each aspect data at inference time. For example, the Abilities aspect creates a rank of employees different from the Experiences aspect. This approach necessitates an aggregation method to combine these individual rankings into a final rank. A standard practice to skip this aggregation process is to concatenate all text and perform the ranking process using the embeddings generated with the full text. However, this approach presents significant limitations when applied to our employee dataset, such as information loss due to text truncation.

In this work, we use the Condorcet algorithm, a method based on pairwise comparisons between items. For each pair of items, we calculate how many times one item "won" against each other item, creating a pairwise matrix $\mathcal{M}$ of size (N, N). To calculate the Condorcet scores for each item, we simply sum the column values of each line ($R^*(u_i) = \texttt{sum}(\mathcal{M}[i])$). In the end, the item with the highest Condorcet score is the best item, also known as **Condorcet winner**.

5 Experiments

We compare our three synthetic data fine-tuning strategies (label-based, curriculum-based, and their combination) against already established models. In sequence, to analyze the impact of each aspect, we conduct ablation studies where we remove one aspect at a time from training.

First, we evaluate the overall impact of our three proposed synthetic data approaches on model performance. We compare models trained only on synthetic curriculums, only on synthetic labels, and on a combination of both, using the same training arguments (LoRA $r = 16$, LoRA $\alpha = 16$ and batch size of 24). To validate our results, we compare our models against current SOTA algorithms in text-embedding tasks. We also compare against classic unsupervised models such as BM25. To measure the impact of the Rank Aggregation algorithm, we separately train a model on the concatenation of all employee aspects (concat model).

The overall results are shown in Table 1. As the table shows, the synthetic fine-tuning is highly effective, achieving a performance improvement of over 30% across all ranking metrics in comparison to the best-performing baseline. We also see that the best strategy overall for fine-tuning was the full data approach, where we use a combination of both curriculum and synthetic labeling data.

Separately, the synthetic curriculum approach had marginal improvements over the Stella-400M baseline, with an average improvement of 10% on the ranking except for $NDCG@1$, where there was a performance decrease of around 9%. In contrast, the synthetic labeling data shown significant improvements over the base model, achieving an averege improvement of 39% over Stella-400M, with a 73% improvement on $NDCG@1$. In conclusion, while both data approaches seem to be complementary for the final result, the synthetic labeling has a bigger performance improvement when comparing both approaches separately.

In Table 1, we can also measure the impact of the Rank Aggregation approach. When comparing the "full data" approach against the Concat Model, we see an average improvement of around 6% on the tested ranking metrics. This shows considerable improvements of the Condorcet algorithm over the classic text concatenation approach.

Ablation Test: To leverage the individual importance of each aspect to model performance, we conduct an ablation study. At each test, we remove one aspect from the training and aggregation process and calculate the metrics for each resulting model. We then compare the ablation results against the full aggregation and the Concatenation models.

Table 2 shows the ablation results. For this experiment, we used the Global adapter model introduced in the previous section. The table shows that by removing the *Abilities* aspect, the overall performance of the model increases marginally when compared to the full Aggregation model. This suggests that the *Abilities* aspect does not contribute with useful information to the model and may even introduce noise or redundancy. The other aspects presented a performance decrease overall when removed from training, with some exceptions regarding the $nDCG@1$ when discarding *Role* and *Experience* aspects, which had a slight increase in performance compared to the full aggregation.

6 Conclusion

In this paper, we present a Synthetic data Fine-tuning approach to improve the performance of the Enterprise team Formation task. We design this task as a Semantic search problem, where the projects and tasks are modeled as queries, and the employee with their aspects (e.g., skills, experience, and behavior) are modeled as documents. We evaluate our model on a curated human-labeled dataset and conduct a series of experiments in order to validate our proposed approach.

Future work will focus on evaluating the robustness and transferability of the proposed synthetic data fine-tuning strategies across multiple enterprise contexts. In particular, we plan to collaborate with organizations from different industries and organizational structures to assess how synthetic supervision adapts to varying employee profiles, task distributions, and domain-specific constraints. While privacy considerations limit public data release, cross-enterprise validation would provide stronger evidence of the general applicability of data-centric synthetic supervision for enterprise semantic search.

7 Ethical Considerations

This work was done under the supervision of an internal committee to ensure that the project followed the Brazilian Personal Data Protection Law (translated from Lei Geral de Proteção de Dados Pessoais—LGPD). All annotators performed their annotations during their working hours within the company.

This is an experimental project done for a limited set of employees within a company. Potential risks, such as bias in the rankings, will be monitored before deploying the system to production.

Throughout this work, we used Github Copilot as a coding assistant and Grammarly for spellcheck and punctual writing improvements.

Limitations

We do not release the evaluation data, as they contain sensitive information about the partner company's employees, which limits the reproducibility of our results. Due to data limitations, our approach does not consider inter-employee relationships when creating the team. For future work, we plan to collect data that leverages the affinity between employees and build a system that considers both aspects and relationships when creating the team for a given project. Our experiments are also limited to a single company. We plan to test this approach in different enterprise contexts.

References

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1597–1607. PMLR.

Marquis de Condorcet. 1785. *Essai sur l'application de l'analyse à la probabilité des décisions rendues à la pluralité des voix*. Imprimerie Royale, Paris.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Shizhe Diao, Tianyang Xu, Ruijia Xu, Jiawei Wang, and Tong Zhang. 2023. Mixture-of-domain-adapters: Decoupling and injecting domain knowledge to pre-trained language models' memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5113–5129, Toronto, Canada. Association for Computational Linguistics.

Carl Doersch and Andrew Zisserman. 2017. Multi-task self-supervised visual learning. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2070–2079.

Qingxiu Dong, Li Dong, Xingxing Zhang, Zhifang Sui, and Furu Wei. 2024. Self-boosting large language models with synthetic preference data. *Preprint*, arXiv:2410.06961.

Zhe Dong, Jianmo Ni, Dan Bikel, Enrique Alfonseca, Yuan Wang, Chen Qu, and Imed Zitouni. 2022. Exploring dual encoder architectures for question answering. In *Proceedings of the 2022 Conference on*

Empirical Methods in Natural Language Processing, pages 9414–9419, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Cynthia Dwork, Ravi Kumar, Moni Naor, and D. Sivakumar. 2001. Rank aggregation methods for the web. In *Proceedings of the 10th International Conference on World Wide Web*, WWW '01, page 613–622, New York, NY, USA. Association for Computing Machinery.

Iker García-Ferrero, Begoña Altuna, Javier Alvez, Itziar Gonzalez-Dios, and German Rigau. 2023. This is not a dataset: A large negation benchmark to challenge large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8596–8615, Singapore. Association for Computational Linguistics.

Shangmin Guo, Biao Zhang, Tianlin Liu, Tianqi Liu, Misha Khalman, Felipe Llinares, Alexandre Rame, Thomas Mesnard, Yao Zhao, Bilal Piot, Johan Ferret, and Mathieu Blondel. 2024. Direct language model alignment from online ai feedback. *Preprint*, arXiv:2402.04792.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *Preprint*, arXiv:1503.02531.

Md Mosharaf Hossain, Venelin Kovatchev, Pranoy Dutta, Tiffany Kao, Elizabeth Wei, and Eduardo Blanco. 2020. An analysis of natural language inference benchmarks through the lens of negation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9106–9118, Online. Association for Computational Linguistics.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547.

Erik Jones, Hamid Palangi, Clarisse Simões Ribeiro, Varun Chandrasekaran, Subhabrata Mukherjee, Arindam Mitra, Ahmed Hassan Awadallah, and Ece Kamar. 2024. Teaching language models to hallucinate less with synthetic tasks. In *The Twelfth International Conference on Learning Representations*.

Chankyu Lee, Rajarshi Roy, Mengyao Xu, Jonathan Raiman, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. 2025. Nv-embed: Improved techniques for training llms as generalist embedding models. *Preprint*, arXiv:2405.17428.

Chaofan Li, MingHao Qin, Shitao Xiao, Jianlyu Chen, Kun Luo, Yingxia Shao, Defu Lian, and Zheng Liu. 2024. Making text embedders few-shot learners. *Preprint*, arXiv:2409.15700.

Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. 2023. Towards general text embeddings with multi-stage contrastive learning. *arXiv preprint arXiv:2308.03281*.

Zimu Lu, Aojun Zhou, Houxing Ren, Ke Wang, Weikang Shi, Junting Pan, Mingjie Zhan, and Hongsheng Li. 2024. MathGenie: Generating synthetic data with question back-translation for enhancing mathematical reasoning of LLMs. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2732–2747, Bangkok, Thailand. Association for Computational Linguistics.

Xueguang Ma, Liang Wang, Nan Yang, Furu Wei, and Jimmy Lin. 2023a. Fine-tuning llama for multi-stage text retrieval. *Preprint*, arXiv:2310.08319.

Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023b. Eureka: Human-level reward design via coding large language models. *arXiv preprint arXiv: Arxiv-2310.12931*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. 2023. Mteb: Massive text embedding benchmark. *Preprint*, arXiv:2210.07316.

Jianmo Ni, Gustavo Hernández Ábrego, Noah Constant, Ji Ma, Keith B. Hall, Daniel Cer, and Yinfei Yang. 2021. Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models. *Preprint*, arXiv:2108.08877.

Rodrigo Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. 2019. Multi-stage document ranking with bert. *Preprint*, arXiv:1910.14424.

OpenAI. 2023. Text embedding 3 model. `https://openai.com/index/new-embedding-models-and-api-updates/`. Accessed: 2024-12-29.

OpenAI. 2024a. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

OpenAI. 2024b. Gpt-4o system card. *Preprint*, arXiv:2410.21276.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Stephen Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. 1994. Okapi at trec-3. pages 0–.

Gerard Salton and Michael J. McGill. 1983. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA.

Thinh Hung Truong, Yulia Otmakhova, Timothy Baldwin, Trevor Cohn, Jey Han Lau, and Karin Verspoor. 2022. Not another negation benchmark: The NaN-NLI test suite for sub-clausal negation. In *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 883–894, Online only. Association for Computational Linguistics.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2019. Representation learning with contrastive predictive coding. *Preprint*, arXiv:1807.03748.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2024a. Improving text embeddings with large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11897–11916, Bangkok, Thailand. Association for Computational Linguistics.

Siyi Wang, Qi Deng, Shiwei Feng, Hong Zhang, and Chao Liang. 2024b. A survey on rank aggregation. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, IJCAI '24.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc.

Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. 2023. C-pack: Packaged resources to advance general chinese embedding. *Preprint*, arXiv:2309.07597.

Mang Ye, Xu Zhang, Pong C. Yuen, and Shih-Fu Chang. 2019. Unsupervised embedding learning via invariant and spreading instance feature. *Preprint*, arXiv:1904.03436.

H. P. Young and A. Levenglick. 1978. A consistent extension of condorcet's election principle. *SIAM Journal on Applied Mathematics*, 35(2):285–300.

Xiang Yue, Ziyu Yao, and Huan Sun. 2022. Synthetic question value estimation for domain adaptation of question answering. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1340–1351, Dublin, Ireland. Association for Computational Linguistics.

Dun Zhang. 2024. Stella english 400m v5. https://huggingface.co/dunzhang/stella_en_400M_v5/tree/main. Accessed: 2024-12-29.

A Prompts

In this section, we provide all prompts used during the modeling process.

A.1 Task Generation

The complete two-stage task generation process is highlighted in Figure 7.

A.2 Synthetic Labeling

For the synthetic labeling process, we have three prompts. The first prompt is a base handcrafted prompt that we use as a starting point in the Eureka optimization (Figure 3). The Eureka algorithm uses a review prompt that uses the current best-performing prompt as input to suggest new prompts based on a given fitness metric (in our case, f1, accuracy, precision, and recall). This prompt can be seen in Figure 4.

Machine Feedback base prompt (Eureka inital prompt)

{enterprise_context}

You are responsible for the department of coordinators and managers. This team consists of people from various sectors, such as manufacturing, sales, human resources, and others. Your task is to determine whether an employee is relevant for a specific task in a project.

You will receive as input a project description, an associated task, and an employee's resume. You must indicate whether the employee is suitable for that task.

You should explain your decision, but the last character of your response must be a score of 0 or 1, where:

0: irrelevant / insufficient
1: relevant / sufficient

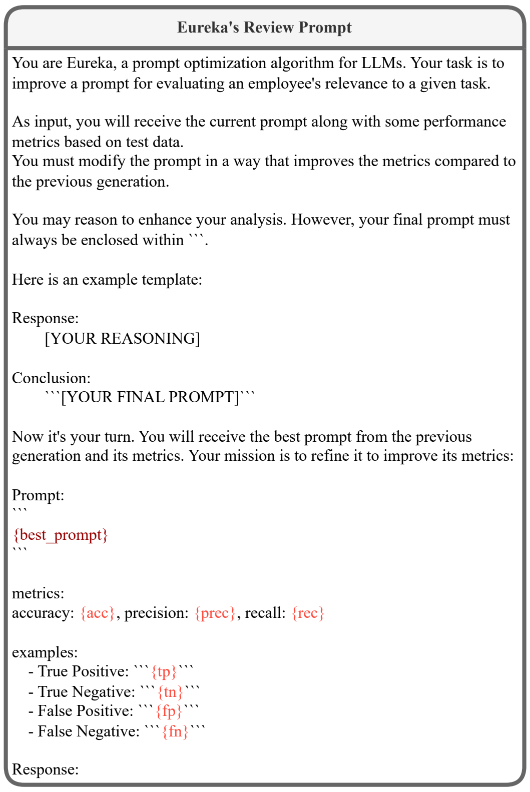
You must be very strict in your decision. Consider both the strengths and weaknesses of the resume in relation to the task.

Now it's your turn:

Project: **{project_description}**
Task: **{task_description}**
Resume: **{employee_resume}**

Your response:

Figure 3: Initial handcrafted machine feedback prompt. This is the starting point for the Eureka optimization process.



~~~
Eureka's Review Prompt

You are Eureka, a prompt optimization algorithm for LLMs. Your task is to
improve a prompt for evaluating an employee's relevance to a given task.

As input, you will receive the current prompt along with some performance
metrics based on test data.
You must modify the prompt in a way that improves the metrics compared to
the previous generation.

You may reason to enhance your analysis. However, your final prompt must
always be enclosed within ```.

Here is an example template:

Response:
      [YOUR REASONING]

Conclusion:
      ```[YOUR FINAL PROMPT]```

Now it's your turn. You will receive the best prompt from the previous
generation and its metrics. Your mission is to refine it to improve its metrics:

Prompt:
```
{best_prompt}
```

metrics:
accuracy: {acc}, precision: {prec}, recall: {rec}

examples:
   - True Positive: ```{tp}```
   - True Negative: ```{tn}```
   - False Positive: ```{fp}```
   - False Negative: ```{fn}```

Response:
~~~

Figure 4: Review prompt for the Eureka optimization process.

A.3 Synthetic Curriculums

Figure 6 shows the synthetic curriculums prompt.

B Prompt Optimization

Next, we describe the optimization process used to refine our prompts using the Eureka framework.

At each Eureka generation, an LLM generates candidate prompts (individuals) using the review prompt. Then, we evaluate each individual as a labeling prompt for our labeling LLM (GPT-4o-mini) by (re-)labeling our evaluation dataset. For each individual, we calculate its relevance classification metrics in the evaluation dataset. The best prompt is then passed to the next generation's review prompt. If neither individual outperformed the prompt, it is replicated to the next generation. Algorithm 1 summarizes all these steps.

Algorithm 1 EUREKA for labeling prompt optimization

```
Input: LLM, fitness function $F$, initial prompt prt
Output: $S_{Eureka}$
Hyperparameters: Search iteration $N$, number of
  samples $K$, elite size $R$
begin
    for $N$ Iterations do
        // Sample $K$ labeling prompts from LLM
        $S_i, ..., S_K \sim$ LLM(prt)
        // Evaluate candidates
        $s_i = F(S_i), ..., s_k = F(S_K)$
        // Reflection step
        prt := prt : $\text{Reflection}_{i=1}^{R}(S_{best_i}, s_{best_i})$
          where $best = R - argmax$
    end
    $S_{Eureka} := S_{best_i}$
end
```

In the algorithm, the fitness function F is a function that labels the evaluation dataset and calculates the accuracy, f1, precision, and recall metrics for a generated prompt S_k. The metric we use to optimize the prompt in the reflection step is the f1_score. The `Reflection` function use the best prompt globally to create the updated `prompt` for the next iteration.

C Data Collection

The evaluation data was collected through two distinct applications. The annotation process was conducted by a group of senior Human Resources specialists. For the team formation application, annotators were instructed to submit only those teams in which all task assignments within a given project were deemed satisfactory. Regarding the labeling application, annotators were required to assign labels exclusively to pairs for which they had absolute certainty (100% confidence). In cases of uncertainty, they were explicitly instructed to skip the pair.

Machine Feedback prompt

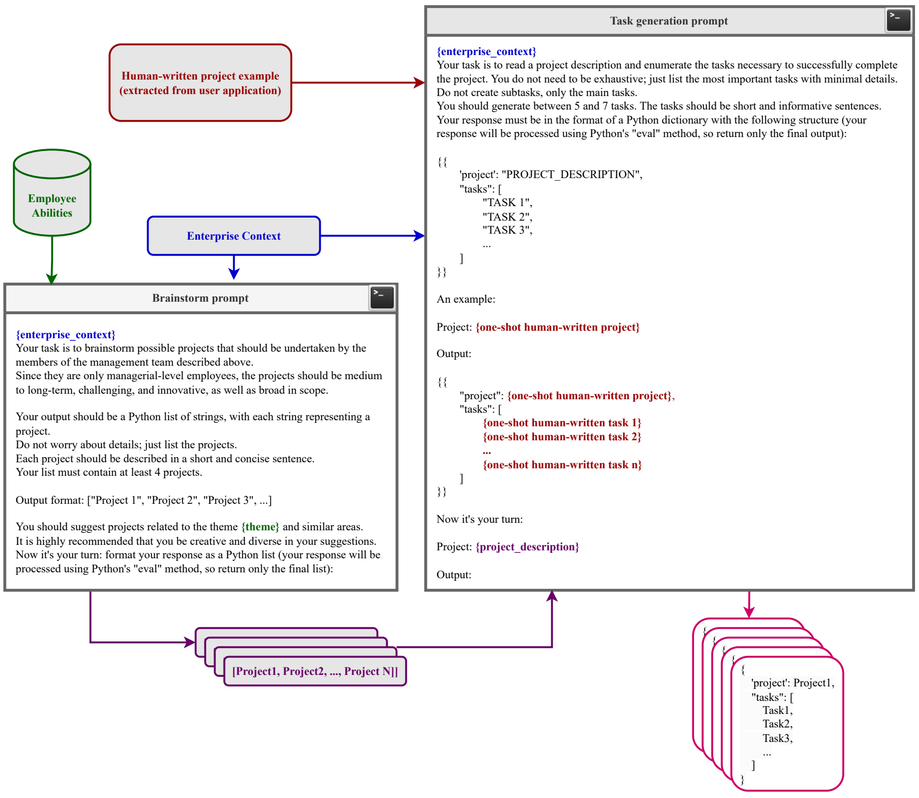
```
{enterprise_context}
You are responsible for the department of coordinators and managers. Your
team consists of professionals from various areas, including manufacturing,
sales, and human resources.
Your task is to evaluate whether an employee is relevant or not for a specific
task in a project.
You will receive the description of a project and a related task, along with the
resume of an employee. Critically analyze the information to determine if the
employee is suitable to perform the described task.

Please follow these guidelines in your response:

1. Briefly summarize the Project and Task, highlighting the importance of the
task for the project's success.
2. Analyze the employee's competencies directly related to the task,
considering both practical and theoretical experience. Ask yourself: "How
relevant are the employee's skills and experiences to the task in question?"
3. Discuss the positives and negatives of the resume concerning the task,
dividing your analysis into sections:
        - Competency Analysis
        - Strengths
        - Weaknesses
        - Conclusion
4. The last character of your response must be a score of 0 or 1, where:
        - 0: irrelevant/insufficient
        - 1: relevant/sufficient

Be rigorous in your evaluation and justify your decision clearly and concisely.

Now it's your turn:

Project: {project_description}
Task: {task_description}
Resume: {employee_resume}

Your response:
```

Figure 5: Machine-feedback prompt for the synthetic labeling process. We extract labels generated by an LLM to create synthetic relevance pairs between generated tasks and real employees.

Synthetic Curriculum prompt

```
{enterprise_context}

You are responsible for the department of coordinators and
managers. This team includes people from diverse sectors such as
manufacturing, sales, human resources, and more.
Your task is to generate an ideal resume for a specific task related
to a project.

The generated resume must follow this format:

"
Position: IDEAL_POSITION

Objective: IDEAL_OBJECTIVE

Behavior: IDEAL_BEHAVIOR

Experience: IDEAL_EXPERIENCE (previous professional
experiences relevant to the position)

Skills: IDEAL_SKILLS (skills relevant to the position)
"

Now it's your turn:

Project: {project_description}
Task: {task_description}

Ideal resume:
```

Figure 6: Prompt for the synthetic curriculum prompt. We use our generated synthetic tasks and generate an ideal curriculum for each task, creating similarity pairs that will be used during training.


Human-written project example
(extracted from user application)
Employee Abilities
Enterprise Context
[Project1, Project2, ..., Project N]]

Brainstorm prompt

```
{enterprise_context}
Your task is to brainstorm possible projects that should be undertaken by the
members of the management team described above.
Since they are only managerial-level employees, the projects should be medium
to long-term, challenging, and innovative, as well as broad in scope.

Your output should be a Python list of strings, with each string representing a
project.
Do not worry about details; just list the projects.
Each project should be described in a short and concise sentence.
Your list must contain at least 4 projects.

Output format: ["Project 1", "Project 2", "Project 3", ...]

You should suggest projects related to the theme {theme} and similar areas.
It is highly recommended that you be creative and diverse in your suggestions.
Now it's your turn: format your response as a Python list (your response will be
processed using Python's "eval" method, so return only the final list):
```

Task generation prompt

```
{enterprise_context}
Your task is to read a project description and enumerate the tasks necessary to successfully complete
the project. You do not need to be exhaustive; just list the most important tasks with minimal details.
Do not create subtasks, only the main tasks.
You should generate between 5 and 7 tasks. The tasks should be short and informative sentences.
Your response must be in the format of a Python dictionary with the following structure (your
response will be processed using Python's "eval" method, so return only the final output):

{{
    'project': "PROJECT_DESCRIPTION",
    "tasks": [
        "TASK 1",
        "TASK 2",
        "TASK 3",
        ...
    ]
}}

An example:

Project: {one-shot human-written project}

Output:

{{
    "project": {one-shot human-written project},
    "tasks": [
        {one-shot human-written task 1}
        {one-shot human-written task 2}
        ...
        {one-shot human-written task n}
    ]
}}

Now it's your turn:

Project: {project_description}

Output:
```

```
{
  'project': Project1,
  "tasks": [
    Task1,
    Task2,
    Task3,
    ...
  ]
}
```

Figure 7: Two-stage prompt for the synthetic tasks creation. On the left, we brainstorm a set of projects; on the right, we break the projects into tasks. The colored texts correspond to the contextual information provided to each prompt.