information

Article

Deep Learning Approaches for Big Data-Driven Metadata Extraction in Online Job Postings

Panagiotis Skondras [1,*], Nikos Zotos [2], Dimitris Lagios [1], Panagiotis Zervas [1], Konstantinos C. Giotopoulos [2] and Giannis Tzimas [1]

[1] Data and Media Laboratory, Department of Electrical and Computer Engineering, University of Peloponnese, 22131 Tripolis, Greece; d.lagios@go.uop.gr (D.L.); p.zervas@uop.gr (P.Z.); tzimas@uop.gr (G.T.)
[2] Department of Management Science and Technology, University of Patras, 26334 Patras, Greece; nzotos@upatras.gr (N.Z.); kgiotop@upatras.gr (K.C.G.)
* Correspondence: eced2024@go.uop.gr

Abstract: This article presents a study on the multi-class classification of job postings using machine learning algorithms. With the growth of online job platforms, there has been an influx of labor market data. Machine learning, particularly NLP, is increasingly used to analyze and classify job postings. However, the effectiveness of these algorithms largely hinges on the quality and volume of the training data. In our study, we propose a multi-class classification methodology for job postings, drawing on AI models such as text-davinci-003 and the quantized versions of Falcon 7b (Falcon), Wizardlm 7B (Wizardlm), and Vicuna 7B (Vicuna) to generate synthetic datasets. These synthetic data are employed in two use-case scenarios: (a) exclusively as training datasets composed of synthetic job postings (situations where no real data is available) and (b) as an augmentation method to bolster underrepresented job title categories. To evaluate our proposed method, we relied on two well-established approaches: the feedforward neural network (FFNN) and the BERT model. Both the use cases and training methods were assessed against a genuine job posting dataset to gauge classification accuracy. Our experiments substantiated the benefits of using synthetic data to enhance job posting classification. In the first scenario, the models' performance matched, and occasionally exceeded, that of the real data. In the second scenario, the augmented classes consistently outperformed in most instances. This research confirms that AI-generated datasets can enhance the efficacy of NLP algorithms, especially in the domain of multi-class classification job postings. While data augmentation can boost model generalization, its impact varies. It is especially beneficial for simpler models like FNN. BERT, due to its context-aware architecture, also benefits from augmentation but sees limited improvement. Selecting the right type and amount of augmentation is essential.

Keywords: metadata extraction; online job postings; big data; web crawling; data preprocessing; ChatGPT; deep learning; embeddings; labor market analysis

Citation: Skondras, P.; Zotos, N.; Lagios, D.; Zervas, P.; Giotopoulos, K.C.; Tzimas, G. Deep Learning Approaches for Big Data-Driven Metadata Extraction in Online Job Postings. *Information* **2023**, *14*, 585. https://doi.org/10.3390/info14110585

Academic Editor: Haridimos Kondylakis

Received: 19 September 2023
Revised: 19 October 2023
Accepted: 23 October 2023
Published: 25 October 2023

1. Introduction

The evolution of the internet has given rise to a plethora of online job portals and internet-based labor market platforms. This digital transformation has brought about an era rich in data. However, this vast influx of data brings forth challenges, particularly in extracting valuable insights due to the ever-changing nature of job postings, which necessitates daily extraction and continuous monitoring. The further integration of social networks into recruitment processes adds another layer of complexity to data collection and analysis. It is vital to understand the significance of accurate job posting classification in the digital age, as it can influence job seekers, recruiters, and even labor market analysts to make informed decisions.

Considering these challenges, this article zeroes in on the multi-class classification of online job postings. Our focus is on data sourced exclusively from dedicated job posting

portals, excluding social networks, which inherently exhibit the characteristics of big data. One cannot overemphasize that the success of machine learning algorithms for job posting classification is intricately tied to the quality and volume of training data. By leveraging the capabilities of large language models (LLMs) for text generation and understanding, we can counterbalance the scarcity of annotated data.

Our comprehensive strategy addresses not just the classification problem but also delves deeply into the intricacies of data extraction and preprocessing, along with the vast challenges presented by the sheer volume of data. We have developed specialized web crawlers to adeptly gather job posting data from various online portals. Building on that, we employ natural language processing (NLP) techniques to ensure data cleaning and preprocessing, guaranteeing the consistency and quality of the collected data.

Utilizing the advanced capabilities of the OpenAI API [1] and the GPT4All [2] framework, we are positioned to efficiently create high-quality job descriptions paired with their respective job title annotations. Models like text-davinci-003 [3,4], Falcon [5–7], Vicuna [8], and Wizardlm [9] (used in our approach), when guided by carefully structured prompts [10–15], produce annotated data for us. This boosts our supervised learning techniques, guaranteeing the creation of reliable and precise annotated datasets—crucial for efficient metadata extraction and analysis.

The synthetic datasets we curate are pivotal to our methodology, serving two distinct purposes: (a) Training in the absence of real data: Here, we address scenarios hampered by a lack of genuine data. Relying solely on real data can constrict a model's comprehension and performance. Our synthetic data, designed to mirror real job postings, serves as an independent training dataset, enabling model building and training where sourcing genuine data might be challenging. (b) Data augmentation [16–20] for imbalanced categories: Within datasets, certain categories may be inadequately represented. This can lead to biases in predictions. Our synthetic data act as a remedy, amplifying these less-represented job title categories and ensuring a balanced training set, crucial for the development of accurate and universally applicable models.

Our methodologies are validated via experiments utilizing two different learning strategies: (a) leveraging USE4 [21] to craft embeddings from the textual data, which subsequently serve as input for a feedforward neural network architecture, and (b) deploying a pretrained BERT [22] model to derive embeddings, which are then fine-tuned.

The previous work section refer to research on job postings using multi-class classification, especially those studies that use machine learning. In the methodology section, we delve into the methods, datasets, and preprocessing steps. We also outline the deep learning structures and training procedures in the data collection and manipulation section. In the results section, we discuss our experimental findings. The article concludes with a discussion section on potential future research directions in this field.

2. Previous Work

In recent years, many studies have been made in relation to job postings and how we may extract useful information or classify them per category. Recommendation systems have been developed that match resumes with job postings, and extensive analyses have been made of the texts of the job postings in order to extract skills, education level, and other useful information.

Nasser I. and Alzaanin A. H. [23] investigated the problem of text classification using multiple machine learning classifiers such as multinomial naive Bayes, support vector machines, decision trees, K nearest neighbors, and random forest. The data they used contained real and fake job postings. Zaroor A. et al. [24] developed the JRC—a job posting and resume classification system that makes use of an integrated knowledge base for performing the classification task. The innovation of the system lies in the search for matching job postings and resumes. Unlike other systems, the JRC matches resumes that only fall under their relevant occupational categories. The experiments were conducted using a real-world dataset.

Several methodologies were proposed to address the job posting classification problem. One of them [25] aims to identify four big dataset job families, recognize homogeneous groups of big dataset skills (skill sets), and characterize each job family with the appropriate level of competence required within each big dataset skill set using machine learning algorithms. The aim of a structured classification of job families and skill sets was to help establish a common dictionary to be used by HR recruiters and education providers. Another approach [26] uses a transformer model with zero-shot and few-shot classification settings, which aims to classify skills through job postings. This study used the European Skills, Competences, Qualifications, and Occupations (ESCO) taxonomy to obtain fine-grained classes for skills, and its target was the Danish labor market. Goindani M. et al. [27], in their study, developed a system that classifies the industry of an employer using job posting data and corrects possible errors in the former classifications. They used two models for classification: support vector machines (SVMs) and gradient-boosted decision trees (GBDTs). They observed that while both models perform similarly in classifying job employers, GBDT is more effective than SVM in identifying job employers that were wrongly computed. Varelas G. et al. [28] also used SVM, random forests, KNN, SGD (stochastic gradient descent), and MLP neural network classifiers as voting algorithms for the classification of job postings according to the ISCO Occupation Codes.

Although these approaches hold promise for improving classification accuracy, new models and algorithms were developed. Nevertheless, among others, one of the main challenges that remains is obtaining large and annotated datasets. Moreover, careful evaluation is necessary to prevent biases toward specific job postings. The development of effective strategies for collecting and annotating high-quality data is critical to facilitating the development of accurate and unbiased models for the multi-class classification of job postings.

3. Methodology

3.1. Overview

Our adopted methodology consists of the following steps: We initiated the process with the collection of job posting data from online sources. This involved crawling websites to gather raw data, preprocessing these data to make them consistent and clean, and finally shaping the data into a standard comma-separated value format for easier handling and analysis. Subsequently, we leveraged large language models to create synthetic job postings. For this purpose, we utilized tools such as the OpenAI API and the GPT4All framework, which are known for their text-generation capabilities.

The next phase focused on forming training data tailored for two specific use cases, using synthetic data for (a) training in the absence of real data and (b) augmentation of imbalanced categories. With the data in place, we moved on to the implementation of the training process. Here, two distinct approaches were employed: the first combined USE4 embeddings with a feedforward neural network (FFNN) architecture, while the second approach focused on using BERT embeddings and fine-tuning the BERT model for our specific task. The implementation leveraged Hugging Face's transformers library [29].

To ensure the robustness of our models, we defined an experimental procedure that would guide our evaluation phase. Lastly, we determined specific metrics that would be used to gauge the performance of our models, ensuring accuracy, reliability, and relevance to our research goals. These steps are depicted in Figure 1 and are analyzed in the next subsections.

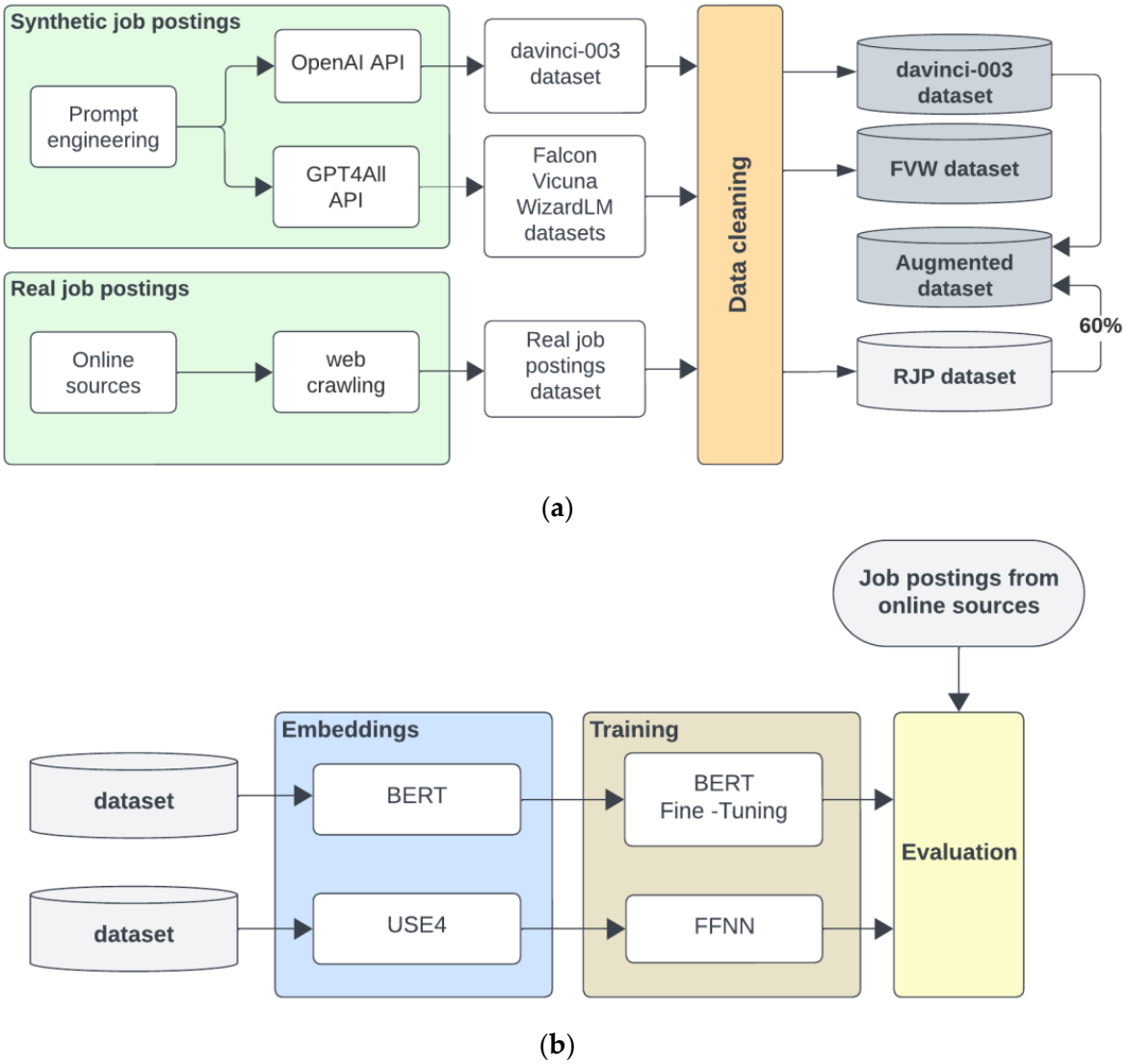

Figure 1. (**a**) Data collection, preprocessing, and dataset creation pipeline; (**b**) embedding generation, training, and evaluation pipeline.

3.2. Data Collection from Online Sources and Preprocessing

The process followed for the creation of the real job posting dataset is shown in Figure 2.

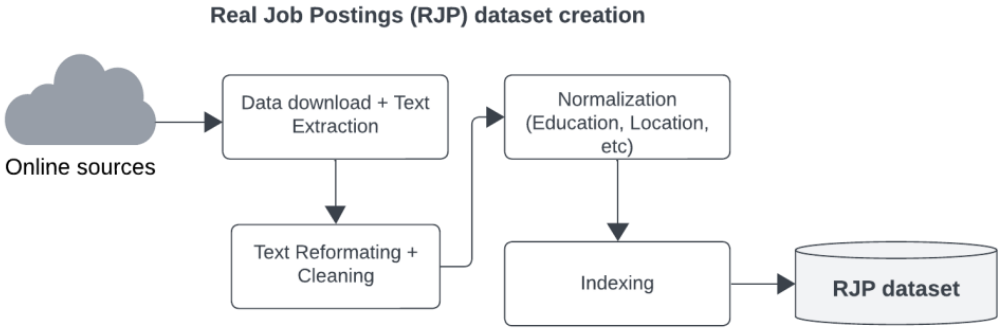

Figure 2. An overview of the creation procedure of the real job posting dataset.

Online Sources: As depicted in Figure 1a, the real job posting (RJP) dataset comprises job postings collected exclusively from job advertisement sites. These sites serve as trusted

resources for gaining insights into the dynamics of the labor market. They encompass various industries and job categories, allowing for an in-depth understanding of labor market patterns, skill requirements, and other key factors influencing employment. Crawling data from job postings during the 2020 time period revealed potential biases stemming from the disproportionate impact of the COVID-19 pandemic on certain industries. This resulted in an underrepresentation of specific job sectors in our dataset.

Data Download and Text Extraction: For the data download task, we employed advanced web crawling techniques, utilizing tools such as Scrapy [30], Requests [31], and Beautiful Soup [32] in the Python language. The web crawling process involved automatically traversing websites dedicated to job postings and navigating the intricate network structure of these platforms to pinpoint and extract relevant job postings. Using Beautiful Soup, we also extracted plain text from HTML structured data, ensuring that only pertinent content was retained.

Text Reformatting and Cleaning: Throughout our data collection and processing, emphasis was placed on maintaining the consistency and accuracy of the data. This phase involved:

- Correcting typographical errors.
- Removing duplicate records.
- Stripping special characters like   (HTML element), \r, and _.
- Managing empty lines within a text block.
- Using regular expressions to identify and exclude URLs embedded in job descriptions.
- Deleting site-specific prefix keywords (e.g., 'Job Description').

Such procedures were critically implemented to ensure a dataset that is both representative and free from noise or inaccuracies.

Normalization: A significant aspect of our data treatment was the standardization of job postings' locations and educational qualifications. Preliminary analysis revealed that different job advertisement platforms often had inconsistent naming conventions. We addressed this by:

- Constructing detailed inventories for geographical names linked to companies and ensuring unified location representations.
- Creating mappings for educational titles, recognizing that the same qualifications might be described differently across job boards.

Indexing: Using a MariaDB [33] database, we designed a schema with two main columns: "key" and "value". The "key" column recorded the original descriptor from job postings, while the "value" column logged its standardized equivalent. This indexing was carried out using a blend of automated algorithms and periodic manual verifications.

3.3. Utilization of LLMs for Job Postings Generation

In the pursuit of generating high-quality synthetic job postings, we employed both the OpenAI API and the GPT4All framework. OpenAI's API provides a robust environment for text generation. Conversely, the GPT4All framework, with its diverse model portfolio, brings forth adaptability suitable for various computational setups. From the OpenAI repertoire, we chose 'text-davinci-003' owing to its proven efficacy in generating contextually appropriate and coherent content. Within the GPT4All framework, we specifically chose the Falcon, Vicuna, and Wizard models. Their reduced parameter count allows them to operate on consumer-level PCs with GPUs. By leveraging this combination, we aimed to achieve a wide variety in our job posting responses, striving to be on par with the text-davinci-003 model.

Our methodological approach was firmly rooted in the creation of dynamic queries meticulously tailored for the selected LLMs. A crucial facet of our strategy revolved around the intricate tuning of model parameters. Beyond the standard considerations like token limits, we delved deep into parameters [34,35], such as temperature and top_p, which play pivotal roles in influencing the randomness and diversity of text generation,

respectively. By adjusting the temperature, we were able to modulate the balance between randomness and determinism in the generated content. Simultaneously, fine-tuning the top_p parameter allowed us to control the diversity, ensuring that the generated postings were neither too generic nor too niche. In our final implementation, we opted for a token count representative of conventional job postings, maintaining an average of 350 tokens.

The depth of our prompts encompassed a vast spectrum of job titles and their corresponding synonyms and spanned a range of educational backgrounds and work experiences. Combinations of educational levels, English proficiency, and styles that were incongruous with the job title (e.g., a lawyer with a high school degree) were either proactively prohibited by the LLMs, which issued specific warning messages, or were selectively excluded during the manual annotation phase (addressed in the data cleaning step shown in Figure 1a). This meticulous attention to detail ensured our synthetic outputs were congruent with real-world job postings, all while maintaining strict adherence to predetermined criteria.

Measuring Similarity of Job Postings

To thoroughly understand the content and variety of job postings within each dataset and draw meaningful comparisons, we employed the following methodological approach. Initially, we converted the textual content of job descriptions into numerical vectors through the utilization of term frequency-inverse document frequency (TF-IDF) vectorization [36]. This approach serves to quantify the significance of each term within a given job description relative to the entirety of the dataset. To measure the similarity between these vectorized descriptions, we leveraged the cosine similarity metric [37]. To facilitate a more intuitive grasp of the similarity scores between job descriptions, we present our findings through the utilization of a heatmap visualization. This graphical representation employs color gradients to convey the extent of similarity between pairs of job descriptions. To make the visual representation more understandable and organized, we arranged the order such that similar jobs are positioned next to each other. Thus, clusters of job titles with similar characteristics were visually juxtaposed, rendering the inherent relationships and patterns more readily discernible.

Real job postings (RJPs): As regards real job posting data, the heatmap clearly shows that job postings with identical titles exude a high degree of similarity. This reflection of consistency underlines the standardized nature of how certain roles are described across different platforms or sources. At the same time, the heatmap highlighted areas where distinct job titles shared substantial content overlap. Such patterns could hint at roles that share common domains, responsibilities, or skills. Conversely, there were job titles that distinctly stood apart, emphasizing the specialized or niche nature of certain roles.

ChatGPT job postings (davinci-003): In the dataset generated by the text-davinci-003 model, job descriptions corresponding to identical titles demonstrated a high degree of intra-category homogeneity, showcasing the model's proficiency in maintaining content consistency for defined roles. However, inter-category overlaps were observed, indicating the model's perception of shared responsibilities across distinct roles. Specific descriptions also manifested as outliers, representing niche or specialized roles, further underscoring the model's granularity in content generation. Given the observed consistency and diversity in the synthesized dataset, it is inferred that the model accurately captures both the overarching and detailed nuances of the professional landscape.

GPT4All models job postings (FVW): In this case, our findings also revealed that AI-generated job descriptions bearing the same title show strong similarity, affirming the model's adeptness in generating consistent content for specific roles. Yet, intertwined within this consistency were regions where different job titles converged, hinting at the model's perception of roles with intertwined responsibilities or overlapping domains. These confluences offer a glimpse into the model's synthesized understanding of the complex world of professions. In contrast, there were descriptions that maintained their distinct identity, emphasizing roles that the model discerned as specialized or uniquely positioned.

The analysis reveals that advanced AI models, like text-davinci-003, exhibit a commendable capability to replicate the intricate details and themes present in genuine job descriptions. Thus, it is evident that the AI-generated data present a potential asset for classifier training, promising a comprehensive dataset that encompasses both general themes and specific job role intricacies. The subsequent figures present the job posting similarities in the constructed datasets. Figure 3 presents job posting similarity for the RJP dataset.

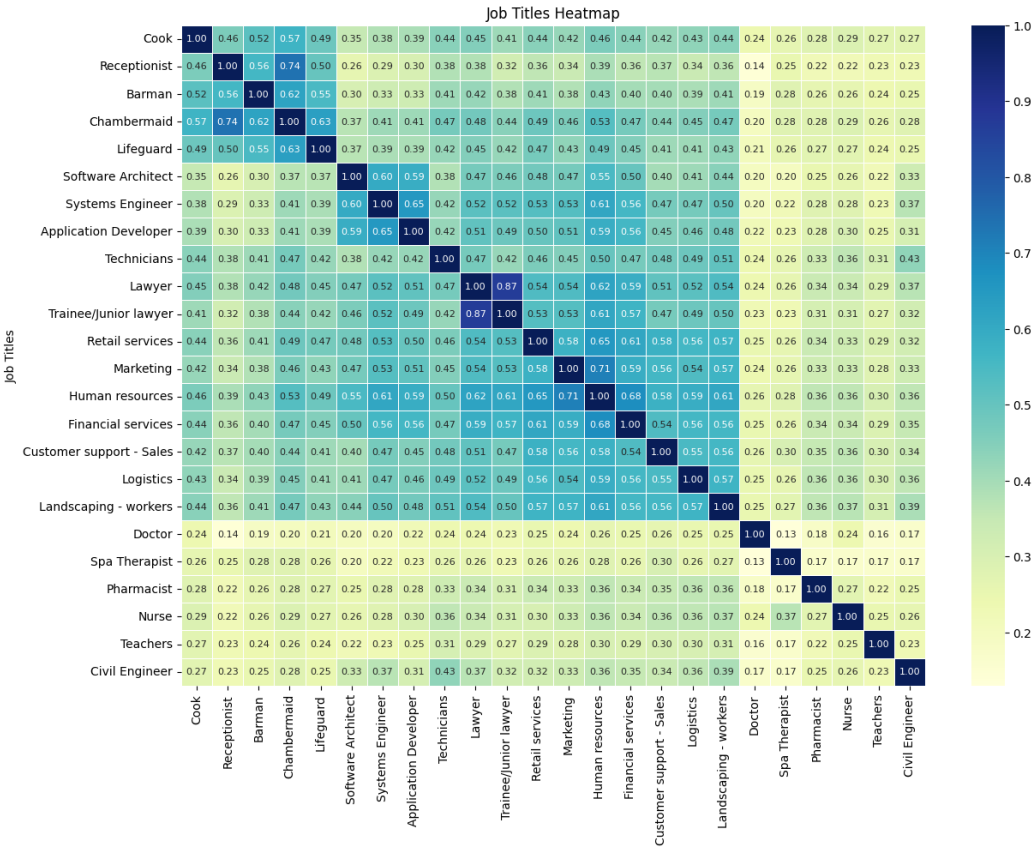

Figure 3. Similarity of job postings in the RJP dataset.

As shown in Figure 3, despite the varied nature of different job roles (e.g., cooks and doctors, lifeguards, and marketing), an average similarity score of approximately 0.2 is observed across job postings, suggesting overlapping language usage. This overlap might stem from the predominance of generic content in postings, with only a minor emphasis on specific job skills, resulting in a foundational similarity even among unrelated professions.

Within the dataset generated by the text-davinci-003 model, we observed a notable level of internal consistency among job descriptions associated with the same job titles. This observation underscores the model's competence in ensuring content coherence for specific roles. However, we also noted instances of cross-category convergence, suggesting that the model recognizes shared responsibilities among distinct job positions. Figure 4 presents the similarities of job postings in the davinci-003 dataset.

The generated FWV dataset also presented consistency among job descriptions associated with the same job titles and high similarities in job postings, particularly in occupations such as systems engineer, software architect, and application developer. The similarity matrix for the FWV dataset is presented in Figure 5.

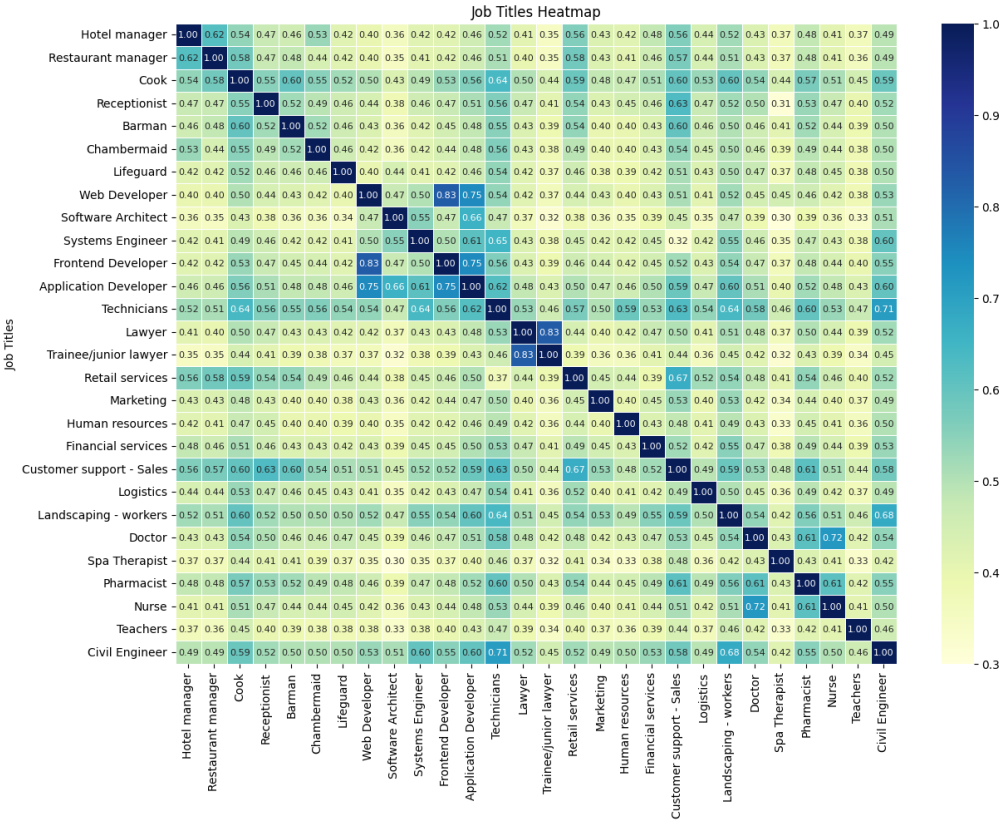

Figure 4. Similarity of job postings in the davinci-003 dataset.

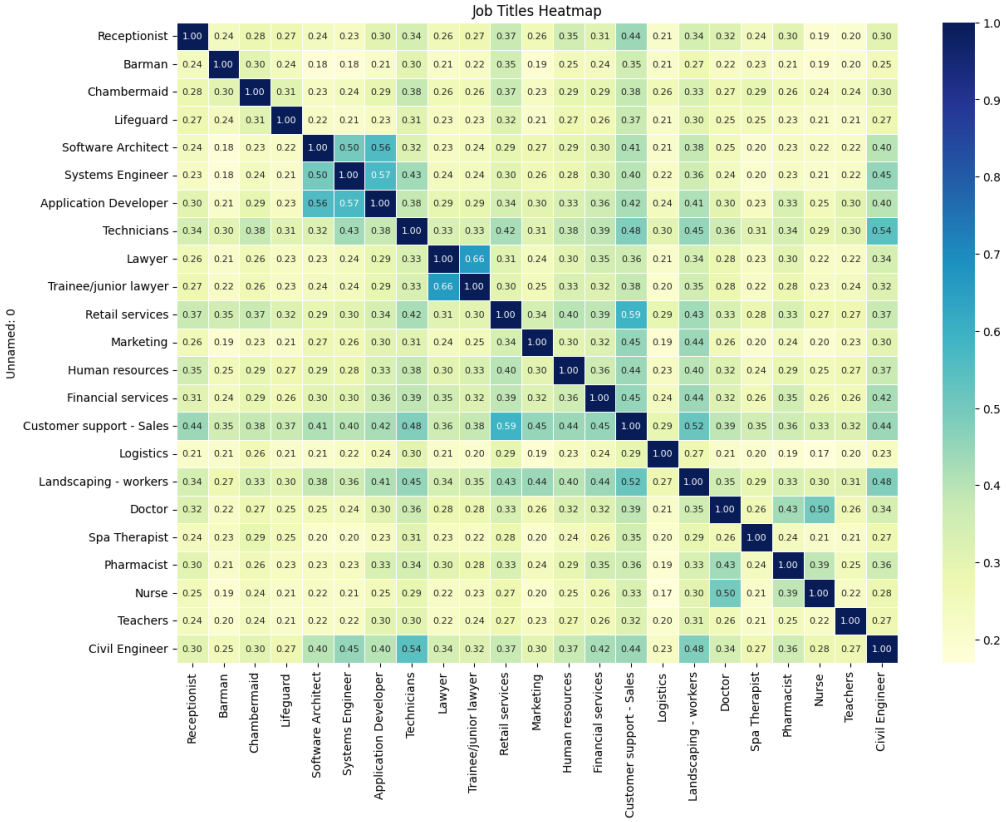

Figure 5. Similarity of job postings in the FVW dataset.

3.4. Training Dataset Creation and Use Cases

To balance out the absence of annotated data for performing the multi-class classification task, four datasets were created,

- one dataset with real-world data from online sources (RJP),
- two synthetic datasets from the OpenAI API (davinci-003) and the GPT4All framework models Falcon, Vicuna, and Wizardlm (FVW),
- an augmented dataset (augmented), which is a composition of instances from davinci-003 for augmenting underrepresented classes, and 60% of the RJP dataset ('davinci-003' was selected due to its diverse and linguistically rich responses, resulting in a higher quality of augmentation). This augmented dataset was then combined with the RJP_train (60% of the original RJP dataset),
- RJP_evaluation dataset (evaluation) derived from the remaining data of the RJP dataset (approximately 40%). To ensure both a substantial training set and a robust evaluation, we allocated 60% of the data for training and 40% for evaluation. The selection of instances for each subset was conducted randomly, ensuring against a biased distribution or the exclusion of difficult-to-learn/classify instances.

The dataset details are tabulated in Table 1.

Table 1. Detailed job posting numbers are represented per job title.

Job Category	RJP	FVW	Davinci-003	Augmented	RJP_ Evaluation
Lawyers	367	153	250	275	147
Trainee lawyers	42	150	150	175	32
Teachers	499	155	150	299	200
Human resources	498	155	150	299	199
Marketing	497	255	250	298	199
Customer support—sales	495	268	250	297	198
Retail services	495	155	150	297	198
Cook	500	205	200	300	200
Logistics	494	154	150	296	198
Financial services	492	219	200	295	197
Civil engineers	484	160	150	290	194
Receptionist	65	170	232	271	44
Hotel manager	68	170	232	272	58
Barman	67	170	232	271	28
Lifeguard	64	160	160	198	26
Restaurant manager	67	180	232	272	25
Chambermaid	67	180	232	270	63
Landscaping—workers	471	189	150	283	188
Technicians	469	255	250	281	188
Systems engineer	47	170	252	280	19
Web developer	27	170	250	266	21
Software architect	31	170	262	280	13
Front end developer	163	180	179	280	62
Application aeveloper	122	182	210	280	52
Pharmacist	64	169	242	280	26
Doctor	47	186	252	280	19
Nurse	65	183	180	219	26
Spa Therapist	84	191	230	280	34
Total	6851	5104	5827	7684	2854

Table 1 presents all classes (28) contained in each dataset. Some classes act as a superset (parent) of similar or specialized types of jobs (children). For example, the class Cooks contains job postings that refer to Cook A, Cook B, and Baker. Table 2 presents the classes that embody similar or specialized types of jobs.

Table 2. The distribution of types of jobs per class.

Job Category/Class	Similar/Specialized Type of Job
Human resources	HR Generalist, HR Assistant, HR Specialist, HR Payroll Officer, Junior HR Assistant, and HR Business Partner
Civil Engineer	Civil Engineer, Building Architect, Mechanical Technician, Natural Resources Engineer
Cooks	A' Cook, B' Cook, Chef, Baker, Sous Chef, Buffet/Bar, and Pastry Chef
Customer support—sales	Customer Support, Customer Service Supporter, Sales Executive, Commercial Representative, Customer Experience Specialist, E-commerce Site Specialist, Customer Service Supporter, and Customer Service Agent
Financial services	Economist, Accounting Officer, Accountant, Financial Advisor, Investment Advisor, Portfolio Manager, Financial Services Risk Management Advisor, Financial Services Consultant, Financial Analyst, Auditor, Credit Manager, and Financial Manager
Landscaping—workers	Agro Coordinator, Agronomist, Gardener, Market Development Agronomist, Landscape Environmental Manager, and Landscape Engineer
Logistics	Warehouse Worker, Warehouse Driver, Warehouse Officer, Warehouse Assistant, and Warehouse Staff
Marketing	Sales and Marketing, Marketing Executive, Marketing Officer, Digital Marketing, Marketing Manager, Performance Marketing Specialist, and Marketing Associate
Retail Services	Cashier, Store Manager, Food Delivery Driver, and Store Administrator
Software Engineer	Systems Engineer, Web Developer, Software Architect, Front end Developer, and Application Developer
Teachers	English Teacher, Italian Teacher, Spanish Teacher, German Teacher, French Teacher, Russian Teacher, and Arabic Teacher
Technicians	Plumber, Electrician, Refrigeration engineer, Ironworker, Bicycle technician, Bodyworker Car painter, and Cabinet maker

In the following paragraphs, we will give a description of the use cases.

Use Case 1—Training in the Absence of Real Data: In scenarios devoid of real data, synthetic data, birthed from AI models [38–43], emerges as an invaluable alternative. For the training in use Case 1, we utilized the davinci-003 and the FVW datasets. Both datasets were composed of the same classes as the RJP dataset, with a comparable number of instances for each class (Table 1). The performance of the models was compared with that of models trained with real data (the RJP dataset).

Use Case 2—Augmentation of Imbalanced Categories: In situations marked by an underrepresentation of specific job categories, augmentation becomes a crucial corrective measure. This approach guarantees a more balanced dataset, ensuring equal representation of all classes our model aims to classify, thereby preventing overfitting to any class. To train the classification model for use in Case 2, we employed an augmented version of the RJP dataset section that was originally used for training in the reference experiments. In specific, once we identified the classes that were not adequately represented, we augmented them by adding instances from the davinci-003 dataset, resulting in the augmented dataset (Table 1).

3.5. Models Architecture and Training

In this subsection, we delve into the intricacies of our methodology for training a state-of-the-art text classifier. Our approach embraces two distinct but powerful techniques: the first leverages the USE4 embeddings that serve as the input to a feedforward neural network (FFNN), embodying a synthesis of traditional and modern representations. The second employs BERT, utilizing its embeddings not merely as input but to fine-tune the entire BERT architecture for our specific task.

FFNN model: We have meticulously crafted and trained a feedforward neural network tailored to the demands of a multi-class classification assignment. The utilized architecture includes an input layer with 512 units, reflecting the model's anticipation of input data with 512 features, similar to those provided by the USE4 methodology. Further, we have introduced a hidden layer featuring 256 units coupled with the ReLU (Rectified Linear

Unit) activation function. This robust design choice enhances the network's capacity for capturing complex patterns within the data.

For the critical output layer, we have employed the softmax activation function, aligning with established best practices for multi-class classification tasks. The number of units in this layer corresponds to the previously calculated unique class count (28), ensuring comprehensive coverage of class predictions.

Our model instantiation was executed through the tf.keras.Model function, where we thoughtfully defined the input and output layers, establishing a coherent neural network architecture. The optimization strategy we employed is Adadelta, with a learning rate of 0.15 and a rho value of 0.95. These parameter values were thoughtfully chosen to balance the model's capacity for rapid convergence and effective minimization of loss.

To initiate the training process, we thoughtfully compiled the model, selecting 'sparse_categorical_crossentropy' as the loss function, a well-suited choice for multi-class classification. Additionally, we specified the earlier-established optimizer, and as an essential measure of performance, we designated 'accuracy' as the primary metric to be meticulously monitored.

The training procedure was initiated using a dataset as training data, accompanied by the corresponding class labels. Additionally, a validation dataset was thoughtfully provided to assess the model's performance on unseen data. Over the course of training, the model underwent 37 epochs, and each epoch's learning updates were processed with a modest batch size of 32, ensuring efficient utilization of computational resources and steady convergence. This rigorous training protocol allowed our model to reach its optimal performance potential.

A confusion matrix is constructed based on the predicted and true labels, and a function is used to visualize this matrix. The confusion matrix is also converted into an Excel spreadsheet for further analysis. A classification report is generated to summarize model performance by class, including metrics such as precision, recall, and F1-score.

BERT model: The implementation of building and training a BERT-based model ("BERT-base-uncased") for the multi-class classification task leveraged the Hugging Face's transformers library [29]. Data Splitting: We split the training data into training and evaluation datasets. This division helps assess the model's performance on unseen data. For testing, we employed 10% of the dataset as test size, ensuring that the data are randomly partitioned while maintaining class distribution balance. Tokenizer and Model: The tokenizer, obtained from the "BERT-base-uncased" pretrained model, is responsible for processing text input. The model, also pretrained on "BERT-base-uncased", is tailored for sequence classification and has a number of labels equal to the count of unique class labels (28) in the training data. This architecture is particularly well-suited for text classification tasks [44]. Data Preparation: For training and evaluation, we created two datasets, one for training and one for evaluation. These datasets include text inputs and their corresponding category labels (28), which are encoded as numerical values. The tokenizer is applied to tokenize and preprocess the text data, and a maximum sequence length of 256 tokens is imposed. Training Arguments Method: The training arguments, which were defined using a specific crafted method, included settings such as the output directory for model checkpoints. The batch sizes for training were eight, and for evaluation, they were also eight; the number of training epochs was three, and logging parameters were specified. We used the option GPU (no_cuda) to load the best model at the end of training. Trainer Configuration: The trainer is configured with the training arguments method and the prepared training and evaluation datasets. This setup streamlines the training and evaluation process. Training and Evaluation: We utilized the training using the train() method and evaluated the model's performance on the evaluation dataset using the evaluate() method. The results were stored for later analysis. Predictions on Test Data: The model is applied to predict categories (28) on a separate test dataset. Predicted labels and true labels are extracted, and the accuracy of the model's predictions is calculated using the method of accuracy score. Confusion Matrix and Visualization: A confusion matrix is constructed

based on the predicted and true labels, and a function is used to visualize this matrix. The confusion matrix is also converted into an Excel spreadsheet for further analysis. Metrics Report: A classification report is generated to summarize model performance by class, including metrics such as precision, recall, and F1-score.

The constructed model provides a comprehensive and systematic approach to training a BERT-based text classification model, evaluating its performance, and analyzing the results. The returned values include accuracy, a confusion matrix, and a metrics report for further insights into the model's classification performance.

Opting for a setup that employs both USE4 embeddings with a feedforward neural network and BERT embeddings for fine-tuning BERT is a strategic decision rooted in achieving a well-rounded analysis. The combination offers a chance to compare traditional methods with cutting-edge models, establish a foundational benchmark with the FFNN, and then gauge the incremental advantages of the more intricate BERT. Both USE4 and BERT embeddings are powerful in their own right, capturing diverse linguistic nuances; the former offers a broad semantic understanding, while the latter provides context-rich representations. BERT's design facilitates tailoring to specific tasks, leveraging both its pre-trained linguistic knowledge and the specifics of the current task through fine-tuning. By diversifying the approach, there's an inherent risk mitigation—if one method encounters challenges, the other might excel. Overall, this dual methodology fosters a deeper, more holistic grasp of text classification strategies, balancing both depth and breadth of understanding.

3.6. Metrics

For the evaluation of our trained models, we adopted the following metrics:

Total Accuracy: This is a foundational metric that calculates the proportion of all correctly predicted instances out of the total instances. It provides an aggregate measure of the model's performance, offering a general sense of its reliability.

Macro average: It calculates the metric for each class independently and then averages the results, treating all classes equally. Macro-averaging is useful for addressing class imbalances by giving equal weight to each class in evaluation.

Per Class Precision, Recall, and F-measure: While total accuracy offers an overview, it is crucial to assess the model's performance for individual classes, especially in imbalanced datasets. Precision measures the fraction of correctly predicted instances of a class against all instances that were predicted for that class. It provides insights into the model's ability to correctly identify positive cases. Recall (or sensitivity) assesses the fraction of correctly predicted instances of a class against all actual instances of that class. It showcases the model's ability to capture all potential positive cases. F-measure (or F1-score) is the harmonic mean of precision and recall, providing a balance between the two. It is particularly useful when there's a need to consider both false positives and false negatives.

4. Results

Within the scope of this study, we directed our attention to the examination and assessment of two distinct use cases defined in Section 3.4. Table 3 displays the total accuracy and micro-average metrics, which treat each class equally important, achieved by both the FFNN and BERT learning approaches for each training set. The "Use cases" column defines the purpose of each experiment, with "Ref" denoting the experiments used as reference points for comparison in Use Cases 1 and 2.

Table 3. Overall experiments with the prediction accuracy in each case.

Experiments	Use Cases	Model	Training Dataset	Evaluation Dataset	Accuracy	Macro Avg
1	Ref	FFNN	RJP_train	RJP_evaluation	0.84	0.62
2	Ref	BERT	RJP_train	RJP_evaluation	0.96	0.91
3	1	FFNN	davinci-003	RJP_evaluation	0.73	0.66
4	1	FFNN	FVW	RJP_evaluation	0.73	0.65
5	1	BERT	davinci-003	RJP_evaluation	0.86	0.79
6	1	BERT	FVW	RJP_evaluation	0.84	0.77
7	2	FFNN	Augmented	RJP_evaluation	0.89	0.83
8	2	BERT	Augmented	RJP_evaluation	0.97	0.94

The reference experiments, which utilized the RJP_train dataset, revealed a discernible difference in the performances of the two models. BERT demonstrated remarkable accuracy, registering at 0.96 and a macro average of 0.91, notably outperforming FFNN, which achieved scores of 0.84 and 0.62, respectively.

In the experiments labeled as Use Case 1, we employed two distinct training datasets: "davinci-003" and "FVW". Notably, FFNN's performance remained relatively consistent across these datasets, posting accuracies of 0.73 for both and macro averages of 0.66 and 0.65, respectively. In contrast, BERT showcased nuanced variations in its performance. Using the "davinci-003" dataset, it achieved an accuracy of 0.86 and a macro average of 0.79. Yet, when trained on the "FVW" dataset, a slight decline was evident, with the model registering an accuracy of 0.84 and a macro average of 0.77.

Our experiments for Use Case 2 highlighted the effectiveness of data augmentation. Both models, when trained on the augmented dataset, exhibited enhanced performance metrics. FFNN posted an accuracy of 0.89 and a macro average of 0.83. Concurrently, BERT achieved an accuracy of 0.97 and a macro average of 0.94, compared to the results of FFNN.

In summation, our results emphatically underscore BERT's superior performance over FFNN across a variety of training datasets. Furthermore, the tangible benefits of enhancing datasets with synthetic data augmentation were prominently observed, emphasizing its value as a best practice and highlighting its potential to shape future research and experimentation. In the subsequent subsections, we will delve into the per-class performance for each experiment.

4.1. Use Case 1 per Class Results

For Use Case 1, we evaluated the performance of FNN and BERT methods on three datasets: the RJP dataset, davinci-003 dataset, and the FVW dataset. Table 4 presents the average performance metrics, offering a comparison between these methods in a per-class manner.

Table 4. Model performance for the use of Case 1.

Dataset	Approach	Precision	Recall	F1-Score
RJP	FFNN	0.6546	0.6329	0.6161
RJP	BERT	0.9050	0.9111	0.9068
davinci-003	FFNN	0.6832	0.6889	0.6621
davinci-003	BERT	0.8861	0.7771	0.7886
FVW	FFNN	0.6886	0.6607	0.6546
FVW	BERT	0.8179	0.7700	0.7725

In the RJP dataset, BERT achieved a precision of 0.9050, outperforming FNN's 0.6546. BERT also recorded a recall of 0.9111, higher than FNN's 0.6329, resulting in an F1-score of 0.9068 for BERT and 0.6161 for FNN. However, on the synthetic datasets (davinci-003 and FVW), BERT's metrics displayed a decline compared to their performance on

the RJP dataset. FNN's performance remained relatively stable across all datasets. This consistency might be attributed to the evaluation dataset's origin being similar to the RJP data, potentially introducing bias that can influence model performance evaluation. We further analyzed individual categories to study the model's response to sparse true dataset instances when trained on synthetic data.

Our next steps involve examining specific classes with pronounced performance differences between synthetic and real data. We will emphasize the F1 score as a balanced metric. For the davinci-003 dataset, roles like "software architect" and "trainee-junior lawyer" performed better on synthetic than real data. In contrast, roles such as "chambermaid", "cook", and "hotel manager", among others, favored real data. A similar pattern was observed with the FVW synthetic dataset, notably with "trainee-junior lawyer" performing well on synthetic data, while roles like "application developer", "barman", and "chambermaid" preferred real data. The results are illustrated in Figures 6 and 7.

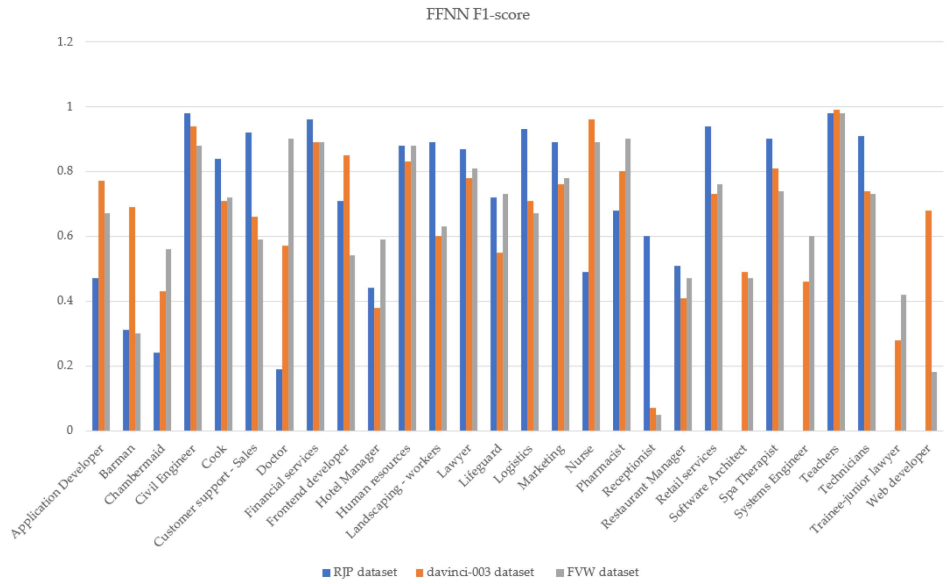

Figure 6. FFNN overall F1—score results for use Case 1.

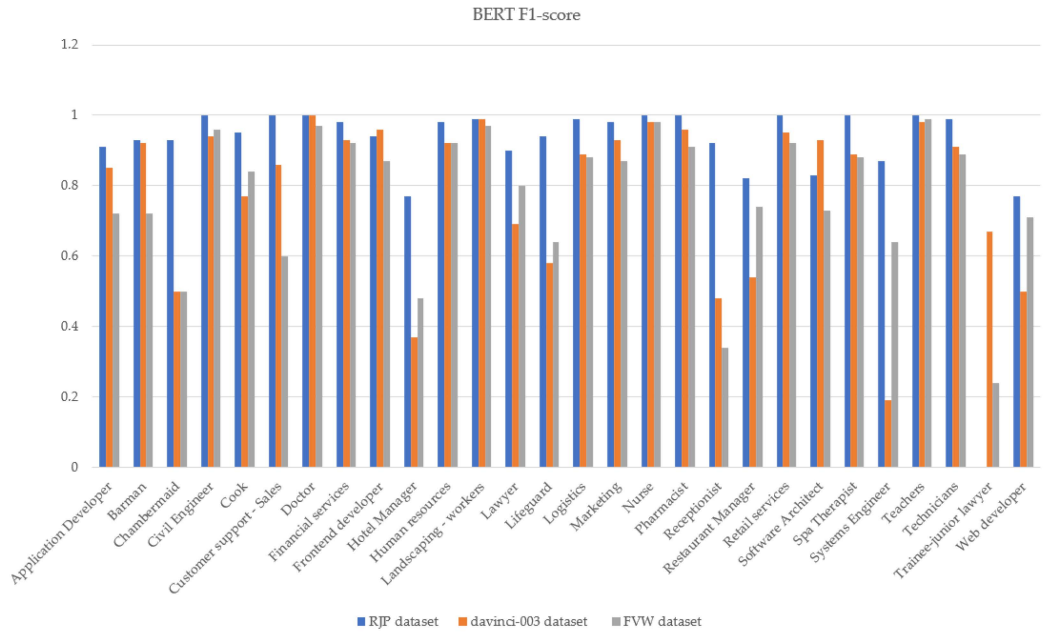

Figure 7. BERT overall F1—score results for use Case 1.

Figures 6 and 7 present a graphical comparison of F1 scores across various classes for the three datasets. From these figures, several observations can be made. Some classes, like "software architect" and "trainee-junior lawyer", show consistent performance across both real and synthetic datasets. However, there is a wide variance in performance among different classes. While some classes perform better with real data, others have comparable or even better F1 scores with synthetic data. On the whole, real data tend to yield better results, especially with the BERT method, whereas FNN's results are more comparable between dataset types. On a class-by-class basis, the effectiveness of synthetic data is not uniform, suggesting that certain classes might benefit from more genuine data or improved dataset generation techniques.

These observations highlight the potential of synthetic data for specific job roles and underscore the need for further research into advanced synthetic dataset generation methods. The possibility of augmenting real dataset classes with synthetic samples to enhance performance is also suggested. This discussion sets the stage for our exploration of Use Case 2 in the subsequent section.

4.2. Use Case 2 per Class Results

As explained in Section 3.4, Use Case 2 involves experiments focused on evaluating the use of synthetic data to augment underrepresented categories of job titles. In pursuit of this objective, we assess the augmented dataset (refer to Section 3.4) using experiments 1 and 2 as references. The dataset-level average performance metrics, Table 5, reveal interesting insights into the comparative performance of these approaches.

Table 5. Model performance for use Case 2.

Dataset	Approach	Precision	Recall	F1-Score
RJP	FFNN	0.6546	0.633	0.6160
RJP	BERT	0.9050	0.9111	0.9068
Augmented	FFNN	0.8625	0.8207	0.8279
Augmented	BERT	0.9411	0.9504	0.9432

In the following Figures 8 and 9, the results of the F1-score are presented for use in Case 2 for the FFNN and BERT models.

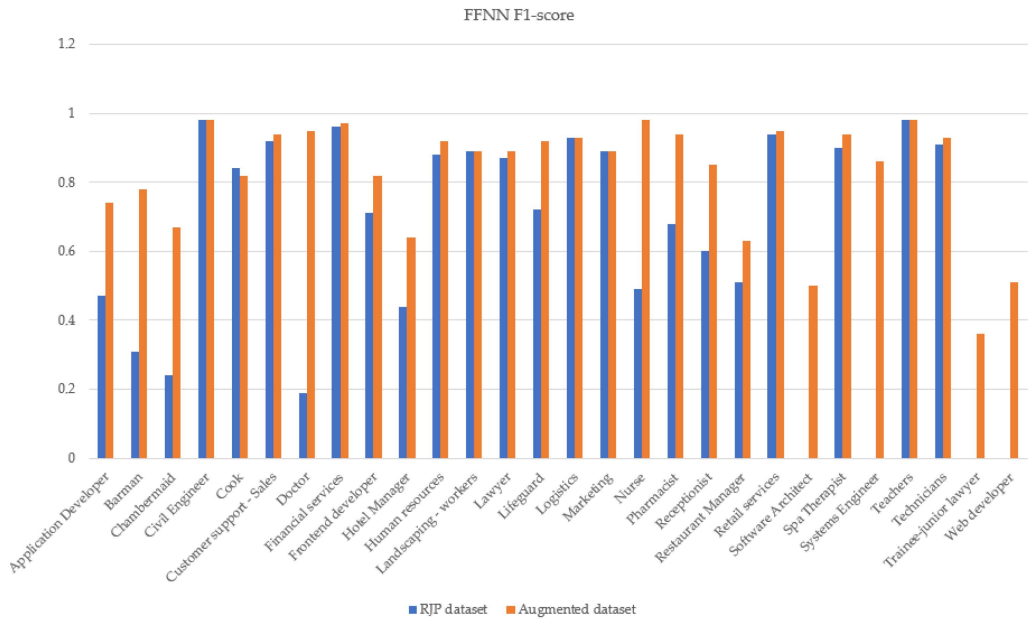

Figure 8. FFNN overall F1—score results for use Case 2.

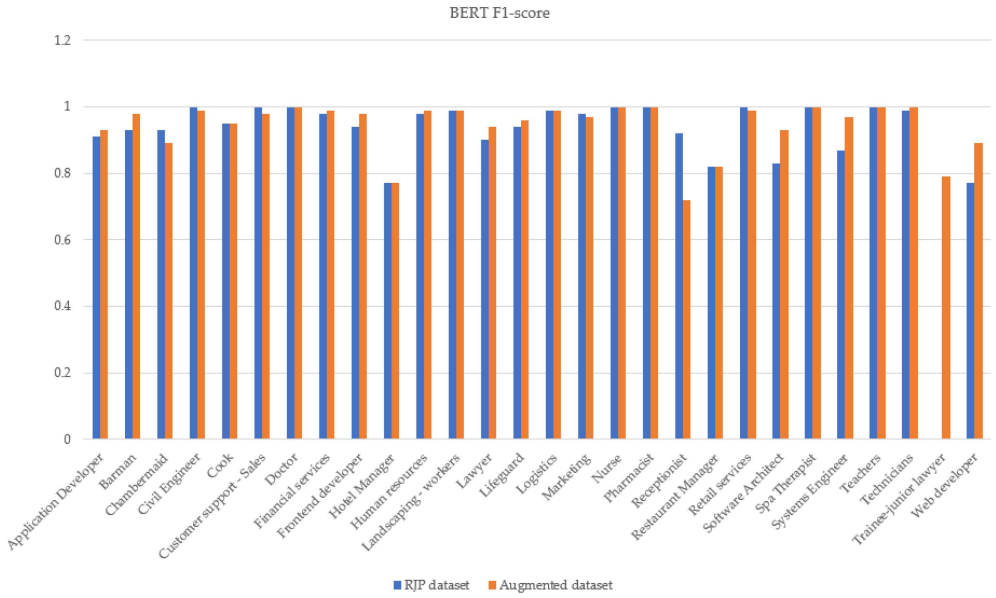

Figure 9. BERT overall F1—score results for use Case 2.

In the RJP dataset, BERT generally exhibits better performance than FNN based on precision, recall, and F1-score metrics. Both models achieve high scores for classes like "civil engineer", "customer support—sales", and "teachers". However, both models underperform in the "trainee-junior lawyer" class. At the same time, many classes show strong results; a few, such as "hotel manager" and "software architect", exhibit varied performance. Overall, BERT seems more effective for this dataset, but there are areas of potential enhancement.

BERT consistently achieves higher performance than FNN across the majority of the classes, particularly in precision. Several classes have BERT achieving perfect or near-perfect scores in precision, recall, and F1-score. FNN, while generally lagging behind BERT, also showcases strong results in many classes. However, there are noticeable weak points: for certain classes, FNN's precision dips as low as 0.57, and its recall even further to 0.36. On the other hand, BERT's lowest precision and recall scores are 0.60 and 0.75, respectively. One significant observation is a class where FNN's recall is notably lower at 0.36, yet BERT achieves a perfect score of 1. Overall, while both models display commendable performance in various categories, BERT emerges as the more robust choice for the augmented dataset, though there remain specific areas where both models could see improvement.

Upon comparing the results from the RJP dataset and the augmented dataset, we observe several prominent trends. Firstly, BERT consistently outperforms FNN in both datasets, yet the margin of superiority is more pronounced in the augmented dataset. This suggests that the complexity and diversity introduced by data augmentation further accentuate the strengths of the BERT model.

In the RJP dataset, certain classes that had mediocre performance with FNN witnessed a marked improvement in the augmented dataset. This indicates that the augmentation process could potentially introduce more varied representations of those classes, aiding FNN in its generalization capability. BERT, being inherently more capable of understanding context due to its architecture, also benefits from this, but to a lesser extent, as it already achieved near-optimal results in the RJP dataset.

However, not all classes benefited uniformly. There were instances where performance metrics for both FNN and BERT either remained unchanged or slightly decreased in the augmented dataset. This could be due to the introduction of noisy or mislabeled data during augmentation. Over-augmentation can sometimes introduce artifacts or irrelevant variations that can confuse the models, especially the simpler ones like FNN.

Furthermore, we noted that in the augmented dataset, the gap between the best-performing and worst-performing classes in terms of F1-score has narrowed down for FNN, suggesting a more uniform performance. BERT, on the other hand, maintains its superiority across the board, but some classes did not witness a substantial boost, potentially indicating that BERT had already learned a near-optimal representation for those classes in the original RJP dataset.

In conclusion, data augmentation, as reflected in the results on the augmented dataset, generally aids in model generalization and performance, especially for models with simpler architectures like FNN (Table 3). However, augmentation is not a guaranteed solution; careful consideration must be given to the type and amount of augmentation applied to ensure that it genuinely aids the learning process. BERT's consistent performance across both datasets highlights its suitability for such classification tasks.

5. Discussion

In our study on the effectiveness of augmented data for job postings with multi-class classification, we found that blending real job postings with AI-generated data enhanced the performance of both the FFNN and BERT models, especially in the case of underrepresented classes. This augmentation not only expanded the training sample size but also enriched the model's domain understanding with diverse linguistic patterns. When paired with the davinci-003 dataset, BERT consistently displayed superior accuracy, while FFNN also performed well, though with some limitations in certain occupation classes.

Our results underscore the potential of integrating augmented data for broader labor market analyses, such as predicting skill requirements, estimating salaries, and analyzing employment trends. Augmented datasets, with their richness and diversity, provide a robust foundation for addressing these complex tasks. Furthermore, this methodology's potential could extend to other sectors, such as healthcare, social media text classification, or spam detection. Leveraging augmented data can enhance models' understanding of varied linguistic patterns, potentially improving classification accuracy, especially with limited data.

Our exploration of large language models (LLMs), including text-davinci-003, Falcon, Vicuna, and Wizard, provided significant insights. By adjusting model parameters like temperature and top_p, we were able to obtain contextually appropriate outputs. Integral to this process was prompt engineering, which played a crucial role in ensuring the generation of relevant, high-quality synthetic data. The importance of this approach becomes even more evident when considering the need for AI models to adhere to ethical and operational parameters, especially when maintaining unbiased data.

However, blending real-world and synthetic data introduces challenges, chiefly in maintaining data quality and consistency. Balancing synthetic diversity without losing domain-specific nuances is crucial. Moreover, training models on combined datasets emphasizes the need for these models to generalize across diverse sources. As AI's role expands in areas like content recommendation and hiring, it is vital to address potential biases for both operational (e.g., an algorithm might overlook qualified candidates due to inaccurate data blending) and ethical reasons (e.g., biased data might lead a recommendation system to favor certain demographic groups).

In conclusion, our research offers insights that have implications beyond the realm of job postings. The combination of prompt engineering and strict guardrails [45] on LLMs response can provide us with augmented data appropriate for training that is less computationally expensive (compared to LLMs capability of extracting knowledge from text) yet powerful methods, such as transformer-based classifiers or deep learning models. Future work could delve deeper into refining synthetic data generation processes and further exploring applications in varied domains.

Author Contributions: G.T., P.S. and P.Z. conceptualized the research. The methodology was framed by P.Z. and P.S. Data curation and formal analysis were led by P.S., D.L., N.Z. and P.Z., with software development by P.S. and P.Z.; P.S. drafted the original manuscript, with subsequent reviews by P.Z., K.C.G., N.Z. and G.T. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data and code presented in this study are available on GitHub: https://github.com/panagiotis-skondras/informatics (accessed on 15 October 2023).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. OpenAI API. Available online: https://bit.ly/3UOELSX (accessed on 15 October 2023).
2. GPT4All API. Available online: https://docs.gpt4all.io/index.html (accessed on 15 October 2023).
3. Ye, J.; Chen, X.; Xu, N.; Zu, C.; Shao, Z.; Liu, S.; Cui, Y.; Zhou, Z.; Gong, C.; Shen, Y.; et al. A Comprehensive Capability Analysis of GPT-3 and GPT-3.5 Series Models. *arXiv* **2023**, arXiv:2303.10420.
4. Anand, Y.; Nussbaum, Z.; Duderstadt, B.; Schmidt, B.; Mulyar, A. GPT4All: Training an Assistant-style Chatbot with Large Scale Data Distillation from GPT-3.5-Turbo. 2023. Available online: https://github.com/nomic-ai/gpt4all (accessed on 16 September 2023).
5. The Rise of Open-Source LLMs in 2023: A Game Changer in AI. Available online: https://www.ankursnewsletter.com/p/the-rise-of-open-source-llms-in-2023 (accessed on 15 October 2023).
6. 12 Best Large Language Models (LLMs) in 2023. Available online: https://beebom.com/best-large-language-models-llms/ (accessed on 15 October 2023).
7. Penedo, G.; Malartic, Q.; Hesslow, D.; Cojocaru, R.; Cappelli, A.; Alobeidli, H.; Pannier, B.; Almazrouei, E.; Launay, J. The RefinedWeb Dataset for Falcon LLM: Outperforming Curated Corpora with Web Data, and Web Data Only. *arXiv* **2023**, arXiv:2306.01116.
8. Chiang, W.; Li, Z.; Lin, Z.; Sheng, Y.; Wu, Z.; Zhang, H.; Zheng, L.; Zhuang, S.; Zhuang, Y.; Gonzalez, J.E.; et al. Vicuna: An Open-Source Chatbot Impressing GPT-4 with 90%* ChatGPT Quality. Available online: https://lmsys.org/blog/2023-03-30-vicuna/ (accessed on 15 October 2023).
9. Xu, C.; Sun, Q.; Zheng, K.; Geng, X.; Zhao, P.; Feng, J.; Tao, C.; Jiang, D. WizardLM: Empowering Large Language Models to Follow Complex Instructions. *arXiv* **2023**, arXiv:2304.12244.
10. White, J.; Fu, Q.; Hays, S.; Sandborn, M.; Olea, C.; Gilbert, H.; Elnashar, A.; Spencer-Smith, J.; Schmidt, D. A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT. *arXiv* **2023**, arXiv:2302.11382.
11. Strobelt, H.; Webson, A.; Sanh, V.; Hoover, B.; Beyer, J.; Pfister, H.; Rush, A.M. Interactive and Visual Prompt Engineering for Ad-hoc Task Adaptation with Large Language Models. *IEEE Trans. Vis. Comput. Graph.* **2023**, *29*, 1146–1156. [CrossRef]
12. Liu, Y.; Deng, G.; Xu, Z.; Li, Y.; Zheng, Y.; Zhang, Y.; Zhao, L.; Zhang, T.; Liu, Y. Jailbreaking ChatGPT via Prompt Engineering: An Empirical Study. *arXiv* **2023**, arXiv:2305.13860.
13. Gao, A. Prompt Engineering for Large Language Models. 2023. Available online: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4504303 (accessed on 24 October 2023).
14. Liu, V.; Chilton, L.B. Design Guidelines for Prompt Engineering Text-to-Image Generative Models. In Proceedings of the CHI Conference on Human Factors in Computing Systems, New Orleans, LA, USA, 29 April–5 May 2022; pp. 1–23. [CrossRef]
15. Sabit, E. Prompt Engineering for ChatGPT: A Quick Guide to Techniques, Tips, And Best Practices. *TechRxiv* **2023**. techrXiv:22683919.v2.
16. Bayer, M.; Kaufhold, M.A.; Reuter, C. 2022. A Survey on Data Augmentation for Text Classification. *ACM Comput. Surv.* **2023**, *55*, 146. [CrossRef]
17. Shi, Z.; Lipani, A. Rethink the Effectiveness of Text Data Augmentation: An Empirical Analysis. *arXiv* **2023**, arXiv:2306.07664.
18. Kumar, V.; Choudhary, A.; Cho, E. Data Augmentation using Pre-trained Transformer Models. *arXiv* **2021**, arXiv:2003.02245.
19. Li, Y.; Wang, X.; Miao, Z.; Tan, W.C. Data augmentation for ML-driven data preparation and integration. *Proc. VLDB Endow.* **2021**, *14*, 3182–3185. [CrossRef]
20. Whitehouse, C.; Choudhury, M.; Aji, A.F. LLM-powered Data Augmentation for Enhanced Crosslingual Performance. *arXiv* **2023**, arXiv:2305.14288v1.
21. Cer, D.; Yang, Y.; Kong, S.; Hua, N.; Limtiaco, N.; St. John, R.; Constant, N.; Guajardo-Cespedes, M.; Yuan, S.; Tar, C.; et al. Universal Sentence Encoder for English. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, Brussels, Belgium, 31 October–4 November 2018; pp. 169–174. [CrossRef]
22. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv* **2019**, arXiv:1810.04805.
23. Nasser, I.; Alzaanin, A.H. Machine Learning and Job Posting Classification: A Comparative Study. *Int. J. Eng. Inf. Syst.* **2020**, *4*, 6–14.

24. Zaroor, A.; Maree, M.; Sabha, M. JRC: A Job Post and Resume Classification System for Online Recruitment. In Proceedings of the 2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI), Boston, MA, USA, 6–8 November 2017; pp. 780–787. [CrossRef]
25. De Mauro, A.; Greco, M.; Grimaldi, M.; Ritala, P. Human resources for Big Data professions: A systematic classification of job roles and required skill sets. *Inf. Process. Manag.* **2018**, *54*, 807–817. [CrossRef]
26. Zhang, M.; Jensen, K.N.; Plank, B. Kompetencer: Fine-grained Skill Classification in Danish Job Postings via Distant Supervision and Transfer Learning. *arXiv* **2022**, arXiv:2205.01381.
27. Goindani, M.; Liu, Q.; Chao, J.; Jijkoun, V. Employer Industry Classification Using Job Postings. In Proceedings of the 2017 IEEE International Conference on Data Mining Workshops (ICDMW), New Orleans, LA, USA, 18–21 November 2017; pp. 183–188. [CrossRef]
28. Varelas, G.; Lagios, D.; Ntouroukis, S.; Zervas, P.; Parsons, K.; Tzimas, G. *Employing Natural Language Processing Techniques for Online Job Vacancies Classification*; IFIP Advances in Information and Communication Technology; Springer: Cham, Switzerland, 2022; pp. 333–344. [CrossRef]
29. Hugging Face Libraries. Available online: https://huggingface.co/docs/hub/models-libraries (accessed on 15 October 2023).
30. Scrappy. Available online: https://scrapy.org/ (accessed on 13 October 2023).
31. Requests. Available online: https://python.langchain.com/docs/integrations/tools/requests (accessed on 15 October 2023).
32. Beautiful Soup. Available online: https://www.crummy.com/software/BeautifulSoup/bs4/doc/ (accessed on 15 October 2023).
33. MariaDB. Available online: https://mariadb.org (accessed on 13 October 2023).
34. ChatGPT—Python Parameters Tuning. Available online: https://platform.openai.com/docs/api-reference/completions/create (accessed on 15 October 2023).
35. GPT4All—Python Parameters Tuning. Available online: https://docs.gpt4all.io/gpt4all_python.html#the-generate-method-api (accessed on 15 October 2023).
36. Sparck, J.K. A Statistical Interpretation of Term Specificity and Its Application in Retrieval. *J. Doc.* **1972**, *28*, 11–21. [CrossRef]
37. Cosine Similarity. Available online: https://www.sciencedirect.com/topics/computer-science/cosine-similarity (accessed on 15 October 2023).
38. Josifoski, M.; Sakota, M.; Peyrard, M.; West, R. Exploiting Asymmetry for Synthetic Training Data Generation: SynthIE and the Case of Information Extraction. *arXiv* **2023**, arXiv:2303.04132v1.
39. Xu, B.; Wang, Q.; Lyu, Y.; Dai, D.; Zhang, Y.; Mao, Z. S2ynRE: Two-Stage Self-Training with Synthetic Data for Low-resource Relation Extraction. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics, Toronto, ON, Canada, 9–14 July 2023; Association for Computational Linguistics: Kerrville, TX, USA; Volume 1, pp. 8186–8207. [CrossRef]
40. Jeronymo, V.; Bonifacio, L.; Abonizio, H.; Fadaee, M.; Lotufo, R.; Zavrel, J.; Nogueira, R. InPars-v2: Large Language Models as Efficient Dataset Generators for Information Retrieval. *arXiv* **2023**, arXiv:2301.01820v4.
41. Veselovsky, V.; Ribeiro, M.H.; Arora, A.; Josifoski, M.; Anderson, A.; West, R. Generating Faithful Synthetic Data with Large Language Models: A Case Study in Computational Social Science. *arXiv* **2023**, arXiv:2305.15041v1.
42. Abonizio, H.; Bonifacio, L.; Jeronymo, V.; Lotufo, R.; Zavrel, J.; Nogueira, R. InPars Toolkit: A Unified and Reproducible Synthetic Data Generation Pipeline for Neural Information Retrieval. *arXiv* **2023**, arXiv:2307.04601v1.
43. Skondras, P.; Psaroudakis, G.; Zervas, P.; Tzimas, G. Efficient Resume Classification through Rapid Dataset Creation Using ChatGPT. In Proceedings of the 14th International Conference on Information, Intelligence, Systems and Applications (IISA 2023), Volos, Greece, 10–12 July 2023.
44. Tay, Y.; Dehghani, M.; Bahri, D.; Metzler, D. Efficient Transformers: A Survey. 2022. *ACM Comput. Surv.* **2023**, *55*, 1–34. [CrossRef]
45. Safeguarding LLMs with Guardrails. Available online: https://towardsdatascience.com/safeguarding-llms-with-guardrails-4f5d9f57cff2 (accessed on 15 October 2023).