Classification of product manager Types to understand the job market

Lena Corredor[1]* **Saliia Asanova**[1]* **Zikai Liu**[1]*
[1]University of California, Berkeley
{lena_corredor, saliia.asanova, zikailiu}@berkeley.edu

1 Introduction

Planning for one's career and navigating the labor market is a time consuming and ambiguous task with an uncertain timeline. This is not only the case for students and recent grads, but also the case for experienced and talented individuals. Recently, this is complicated by factors such as the tech recession and a negative macrotrend. McKinsey & Company(2023) talked about this macrotrend: The average lifespan of a company listed on the S&P 500 was 61 years in 1958 and today is now less than 18 years. This creates a lot of volatility in the market and thus impacts companies and individuals. On top of this, current technology is not working. It is making it harder for applicants to land a career move and companies to recruit talents. Natural Language Processing can play a significant role in helping candidates understand the market. One task that helps professionals is the classification of jobs into different categories. Since we have knowledge of the product manager (PM) role, our goal is to use NLP techniques to classify PM job postings into PM technology categories. This helps students and professionals plan for their careers. For example: with this understanding, some may choose to upskill into specific technologies such as AI product manager while others may want to focus on moving into a people manager role.

2 Literature Review

Through the applied NLP course, we learnt of important considerations when using NLP models for a specific task. Elements that affect the results are: 1) the data the model is trained on (the type of data, the size of the data), 2) the model selection, and the 3) evaluation process. The task here is to classify a product manager job posting into different categories. We conducted literature review in areas of job classification, taxonomy, prior studies on product management and NLP methods relating to classification and methods we used. Our contribution is to increase accuracy with annotation of data and evaluate the models selected for this task.

In terms of the NLP method, we tested it with sentenceBERT. The sentence BERT Reimers and Gurevych (2019) is a masked method that uses the Stanford Natural Language Inference Dataset (Bowman et al., 2015) and the Multi-Genre NLI dataset (Williams et al., 2018). The dataset was manually labeled 'entailment', 'contradiction', or 'neutral'. The researchers Bowman et al. also used GloVe vectors and common crawl. The sentenceBERT derives the semantic meanings using sentence embedding that can be compared using cosine-similarity. To create the labels for product management roles, we leverage our domain knowledge in this area with web search results.

Here are literature reviews relating to the method and domain we are interested in. In the article: "Deep Learning Approaches for Big Data-Driven Metadata Extraction in Online Job Postings Skondras et al. (2023)", the authors used TF-IDF on job descriptions. Then used cosine similarity to measure the differences between job postings. Their model accurately captures both the overarching and detailed nuances of the professional landscape. This is between roles like cook and receptionist whereas we are focusing on categories of product roles. In another study - "An Improved BERT model for precise Job Title Classification Using Job Descriptions Maglyas et al. (2013)", they used BERT for multi-label classification of job titles and achieved a notable performance with an accuracy of 0.842. This paper - Employing Natural Language Processing Techniques for Online Job Vacancies Classification Varelas et al. (2022)" indicated that it's promising. In the article - Machine Learning and Job Posting Classification: A Comparative Study Nasser and Alzaanin (2020), the authors used TF-IDF to extract important words for their model,

* Equal contribution.

this contributed to high evaluation metrics 'Accuracy', 'Recall', 'Precision' and F-measure. This prompted us to use TF-IDF as one of our methods. We used logistic regression to weight and do a bias matrix map from the existing embedding to the 9 classes we labeled. This is similar to the methods "Classification and regression tree analysis in public health: Methodological review and comparison with logistic regression Lemon et al. (2003)". We learned from paper "A Comparative analysis of logistic regression, random forest and KNN models for the text classification Shah et al. (2020)" that logistic regression obtained precision of 0.94 compared to K-Nearest Neighbor at 0.96 in the business section, indicating it can be a suitable candidate for classification.

We also learnt about the role in the 2013 paper - What are the roles of software product managers Maglyas et al. (2013)? Although this paper is a while back, it does outline the core responsibilities of a PM: 1) identifying features that prove significant value for the customer; 2) work with engineering and other teams to build the product; 3) gain authority and influence for the product. This is considered as a general product manager and it's categorized as "Other Type of Product Management" within the technology dimension. One paper used ISCO taxonomy for job classification. Because of annotating resource limitations, we have to consider other approaches to get a bigger training dataset. In this article "Deep Learning Approaches for Big Data-Driven Metadata Extraction in Online Job Postings Skondras et al. (2023)", the author substantiated the benefits of using synthetic data to enhance job posting classification. This research confirms that AI-generated datasets can enhance the efficacy of NLP algorithms, especially in the domain of multi-class classification job postings. We also need balanced data to increase model effectiveness (Dealing with Imbalanced Data for Fraudulent Job Postings - An Application of BERT and Sampling Techniques".

3 Data Collection and Preparation

We obtained the job posting dataset from Indeed using a web scraper tool with the search term 'product manager'. This search also picks up non-PM related roles, but has the terms product and manager in the title. For the midterm, our annotation size is 100 jobs. However, we were able to obtain 5000 job postings that can be used for the final. Instead of preprocessing steps, we dedicated one classification category called: Non-PM, which is for any role that is not computer product manager type. For the finals, we can also use heuristics to preprocess the set. For example: remove any posting less than 200 words.

We first created the labeling guidelines for 9 categories of product manager by technology. The 9 categories are listed below. Their labeling guidelines are in this spreadsheet due to their length.

- **AI/ML Product Manager**
- **Platform Product Manager**
- **Cloud Product Manager**
- **Other Types of Product Manager**
- **Fintech Product Manager**
- **Data Product Manager**
- **API/Integrations Product Manager**
- **Trust & Safety Product Manager**
- **Non-PM**

Next, we did an exercise for the Inter Annotator Agreement. There were 3 annotators. The exercise was for the three annotators to independently categorize 15 job postings into the 9 categories mentioned above. With this, we calculated the inter annotator agreement values using Cohen's Kappa. The Cohen's Kappa values we obtained are:

Annotator Pair	Cohen's Kappa
Annotator 1 vs 2	0.4599
Annotator 1 vs 3	0.6847
Annotator 2 vs 3	0.4529

Table 1: Cohen's Kappa for Annotator Agreement

Common Cohen's Kappa Interpretations:

Cohen's Kappa Range	Agreement
0.41 - 0.60	Moderate
0.61 - 0.80	Substantial
0.81 - 1.00	Almost Perfect

Table 2: Common Interpretations of Cohen's Kappa

Since the values are moderate and substantial, we aimed to move annotator understanding to substantial. The annotators discussed the differences

in understanding and made changes to the labeling guidelines so that we agree on the new classification.

Next, the three annotators classified 100 postings in total into the 9 categories. With this, we discovered the majority class baseline is 32% with Non-PM being the biggest class.

Our preprocessing pipeline is designed to enhance the textual representation of job titles and descriptions, enabling effective classification. First, we tokenize using BERT's tokenizer, where each job title and description is combined and converted to a maximum length of 512 tokens. For the TF-IDF vector, We remove English stop words and limit the vocabulary size, focusing on high-relevance feature selection. Additionally, a label encoding process maps categorical job labels to numeric classes.

For feature extraction, the BERT embeddings are generated using mean pooling across tokens, ensuring context-aware representation Devlin et al. (2019). For TF-IDF, we extract the top 1000 features, incorporating both unigrams and bigrams. Embeddings are then scaled before classification using Logistics Regression.

4 Feature Engineering & Model Selection

For this study, we focused on two primary textual feature engineering techniques: TF-IDF and BERT embeddings.

TF-IDF (Term Frequency-Inverse Document Frequency) Sammut and Webb (2010) is a common text representation technique that assigns weights to words in each document (in this case, job descriptions) based on the term's frequency within the document and its rarity across the entire dataset. This method highlights unique words that distinguish different PM roles, allowing us to identify specialized terms, such as "AI" for AI/ML PMs or "Payments" for Payments PMs. For this model, the TF-IDF vectorizer was configured to extract the top 768 features to maintain dimensional consistency with BERT embeddings, n-grams ranging from unigrams to bigrams, and removed common English stop words. This feature extraction was then paired with a Logistic Regression classifier.

To capture deeper contextual nuances, we utilized BERT (Bidirectional Encoder Representations from Transformers) Embeddings (BERT-base-uncased) Devlin et al. (2019), a pre-trained transformer model that generates context-aware embeddings for the input text. We used BERT to encode both job titles and descriptions into a single text input for each role. The resulting embeddings were then fed into a Logistic Regression model for classification. Our BERT configuration included a maximum token length of 512 and a batch size of 1, ensuring that longer descriptions fit within the model's context window. However, given that many job descriptions exceed the 512-token limit, some information was truncated, potentially affecting model performance, as discussed in Section 5.

Given our limited dataset (105 samples across 9 classes), logistic regression was chosen to preserves the contextual information from pre-trained BERT and mitigate overfitting risks due to its lower model complexity compared to methods like BertForSequenceClassification Sun et al. (2020), which require more extensive parameter estimation. Additionally, by keeping the embedding dimensions consistent across BERT and TF-IDF, we enable a direct comparative analysis of these techniques using a unified classification head. This study thus evaluates TF-IDF's term-based representation against BERT's contextual embeddings in a constrained-data classification setting.

Model Choices and Data Splitting: We split the data into 80% for training and 20% for validation, ensuring a balanced distribution across all classes using stratified sampling. Three models were chosen for classification:

- Majority Class Baseline: This simple baseline predicts the most common class in the dataset. Here, the "Other type of PM" category, making up 32.38% of the data, was always predicted.
- TF-IDF with Logistic Regression: This model achieved a training accuracy of 66.67% but dropped to 42.86% on the validation set, indicating potential overfitting.
- BERT-base Embedding Classifier with Logistic Regression: This model achieved a perfect 100% training accuracy, though validation accuracy fell to 52.38%, suggesting overfitting.

Though regularized (C=0.1 in logistic regression) Brightwood (2024), BERT embeddings with 768 dense dimensions still overfit the small training set, achieving 100% accuracy but only 52.38% on the validation set, which underscores the need for additional labeled data (Figures 3 and 4). BERT

embeddings are dense, meaning nearly all values are non-zero, whereas TF-IDF embeddings are highly sparse, with most values at zero. Due to this sparsity, we applied no regularization (C=1.0) for the TF-IDF model to prevent over-constraining its limited representation.

5 Classification Results Insights

Categories with fewer samples, like "Cloud," "Data," and "Platform," showed low recall and precision, which may stem from limited representation in the training set. "Non-PM" and "Payments" roles, which had relatively larger sample sizes, exhibited better performance. Further refinement of class labels and expanding the dataset would improve classification accuracy and address overlaps between categories. TF-IDF's feature weights highlighted top words for each role (Table 3), providing insights into how keywords contribute to the classifier's decision-making process.

For evaluation, we applied both quantitative and qualitative methods to assess the model's classification accuracy. Quantitatively, we measured accuracy, precision, recall, F1-score to evaluate the model's ability to assign job postings accurately to each PM specialization. The confusion matrices (Figures 1 and 2) highlight distinct patterns: the BERT model shows higher accuracy in "Other type of PM" and "Non-PM" classes, whereas TF-IDF captures fewer true positives but shows consistency across multiple categories. For instance, BERT predicts the Google APM 2025 job —which all three annotators agreed to label as "Other type of PM"—as "Other type of PM" with a confidence score of 84.35%, compared to TF-IDF's 30.34% for the same category (Table 4). This outcome aligns with our expectations, as BERT's embeddings capture nuanced contextual information, while TF-IDF is more effective at identifying domain-specific lexical cues.

In our qualitative evaluation, a person with a PM background reviewed a sample of 10 job postings classified by the model, which achieved an observed accuracy of 50%. The model was particularly inclined to classify postings under the "Other type of PM" category, suggesting it captures general role-specific language but struggles with more specialized distinctions. This limitation may stem partly from BERT's tokenization length constraint (512 tokens), as seen with the "Senior Manager Product Operations" role from ID.me, where critical keywords such as "Gen AI" and "QA" in the responsibilities section might be truncated. Additionally, human annotators leverage implicit knowledge—such as associating "Databricks" with data platforms—which the model's embedding and keyword extraction approach lacks. To improve classification accuracy, we propose refining definitions between "Non-PM" and "Other type of PM" to address ambiguities that contribute to misclassification.

Class	Top Features
AI/ML	website, seamless, workflows, users, experiences, platform, learning, app, ml, ai
API	business requirements, principles, develop, acceptance, military, benefits, proficient, agile, methodologies, familiarity
Cloud	security, saas, offering, public, infrastructure, cloud storage, azure, deployment, storage, cloud
Data	owner, business requirements, highest, experience managing, maximize, actionable, product owner, requirements, daily, data
Non-PM	design, underwriting, insurance, brand, mitsubishi, regulatory, id, hvac, marketing, sales
Other type of PM	looking, digital, guarantee, year, associates, portfolio, mobile, product manager, magic, healthcare
Payments	offer, financial, services, small, chase, clients, banking, payment, payments, client
Platform	management, engineering, teams, government, technical, platform, contract, cloud, silk, enterprise

Table 3: TFIDF Feature Contribution

Category	Google APM 2025 - BERT (%)	Google APM 2025 - TF-IDF (%)
AI / ML	0.67	8.78
API	5.40	2.50
Cloud	0.03	2.33
Data	0.35	3.88
Non-PM	0.41	24.70
Other type of PM	84.35	30.34
Payments	0.46	8.73
Platform	8.23	16.36
Trust & Safety	0.10	2.38

Table 4: Classification Probabilities for Google APM 2025 using BERT and TF-IDF Models

6 Next steps

Tasks	Team Members	Deadline
Annotate additional job postings, providing more examples for each category and supporting model learning within the same dataset.	Salia & Lena	Nov 7, 2024
Implement dataset rebalancing with synthetic data.	Zikai	Nov 10, 2024
Hyperparameter tuning for BERT, experimenting with regularization strength, max token length, and batch size to reduce overfitting.	Zikai	Nov 15, 2024
5-fold cross-validation to verify model stability.	Salia	Nov 18, 2024
PM expert review to present updated classifications to a PM expert for feedback.	Lena	Nov 22, 2024
Compile findings, methodology adjustments, and evaluation outcomes into the final report and prepare presentation slides.	All	Nov 26, 2024

Table 5: Project Tasks and Deadlines

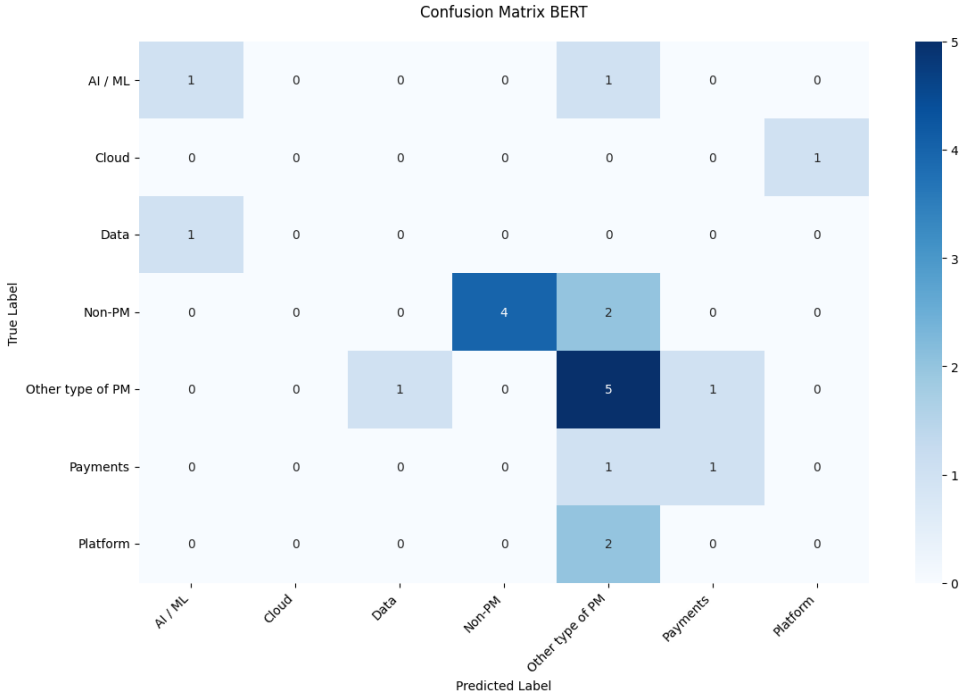

Figure 1: BERT Confusion Matrix

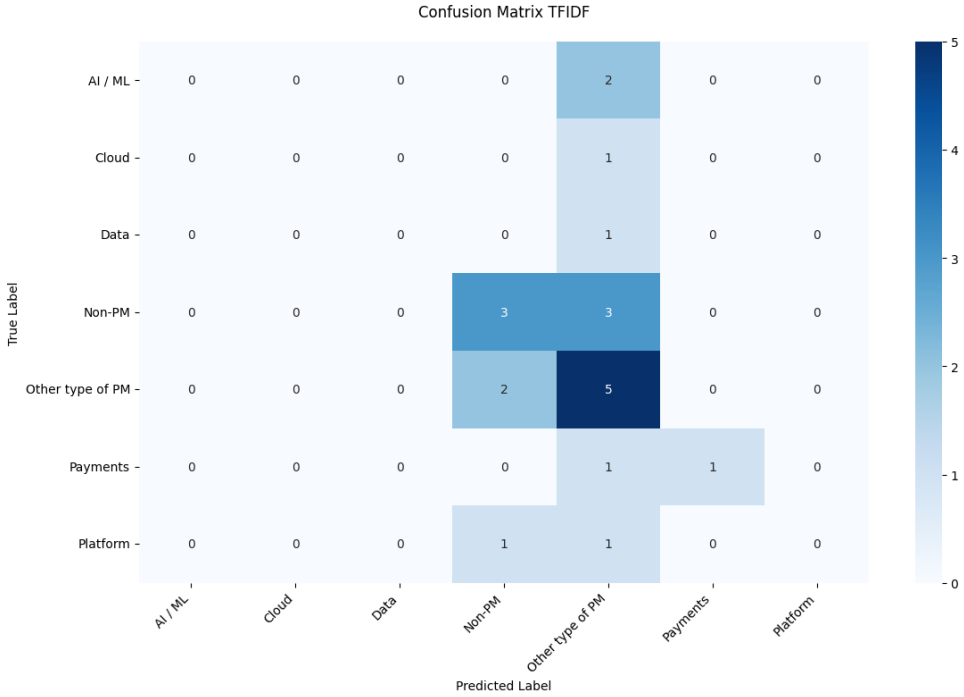

Figure 2: TF-IDF Confusion Matrix

```
Extracting BERT embeddings for training set...
100%|██████████| 1/1 [01:55<00:00, 115.75s/it]
Extracting BERT embeddings for validation set...
100%|██████████| 1/1 [00:28<00:00, 28.09s/it]  Train accuracy: 1.0000
  Val accuracy: 0.5238

Classification Report:
                  precision    recall  f1-score   support

          AI / ML      0.500     0.500     0.500         2
            Cloud      0.000     0.000     0.000         1
             Data      0.000     0.000     0.000         1
           Non-PM      1.000     0.667     0.800         6
Other type of PM      0.455     0.714     0.556         7
         Payments      0.500     0.500     0.500         2
         Platform      0.000     0.000     0.000         2

         accuracy                          0.524        21
        macro avg      0.351     0.340     0.337        21
     weighted avg      0.532     0.524     0.509        21
```

Figure 3: BERT Classification Matrix

```
Training logistic regression classifier...
Train accuracy: 0.6667
Validation accuracy: 0.4286

Classification Report:
                  precision    recall  f1-score   support

          AI / ML      0.000     0.000     0.000         2
            Cloud      0.000     0.000     0.000         1
             Data      0.000     0.000     0.000         1
           Non-PM      0.600     0.500     0.545         6
Other type of PM      0.333     0.714     0.455         7
         Payments      1.000     0.500     0.667         2
         Platform      0.000     0.000     0.000         2

         accuracy                          0.429        21
        macro avg      0.276     0.245     0.238        21
     weighted avg      0.378     0.429     0.371        21
```

Figure 4: TF-IDF Classification Matrix

References

Seraphina Brightwood. 2024. Regularization techniques in logistic regression.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. *Preprint*, arXiv:1810.04805.

Stephen C. Lemon, Jason Roy, Melissa A. Clark, Peter D. Friedmann, and William Rakowski. 2003. Classification and regression tree analysis in public health: Methodological review and comparison with logistic regression. *Annals of Behavioral Medicine*, 26(3):172–181.

Andrey Maglyas, Uolevi Nikula, and Kari Smolander. 2013. What are the roles of software product managers? an empirical investigation. *Journal of Systems and Software*, 86(12):3071–3090.

Ibrahim Nasser and Amjad H. Alzaanin. 2020. Machine learning and job posting classification: A comparative study. *International Journal of Engineering and Information Systems (IJEAIS)*, 4(9):6–14. Available at SSRN: `https://ssrn.com/abstract=3723037`.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *Preprint*, arXiv:1908.10084.

Claude Sammut and Geoffrey I. Webb, editors. 2010. *TF–IDF*, pages 986–987. Springer US, Boston, MA.

Krunal Shah, Harshil Patel, Dhaval Sanghvi, and Maitri Shah. 2020. A comparative analysis of logistic regression, random forest and knn models for the text classification. *Augmented Human Research*, 5:12.

Panagiotis Skondras, Nikos Zotos, Dimitris Lagios, Panagiotis Zervas, Konstantinos C. Giotopoulos, and Giannis Tzimas. 2023. Deep learning approaches for big data-driven metadata extraction in online job postings. *Information*, 14(11).

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2020. How to fine-tune bert for text classification? *Preprint*, arXiv:1905.05583.

George Varelas, Dimitris Lagios, Spyros Ntouroukis, Panagiotis Zervas, Kenia Parsons, and Giannis Tzimas. 2022. Employing natural language processing techniques for online job vacancies classification. In *Artificial Intelligence Applications and Innovations. AIAI 2022 IFIP WG 12.5 International Workshops*, pages 333–344, Cham. Springer International Publishing.