ML Final Project

Leo Kampen, Stan Danaev

December 2024

Abstract

In this paper, we describe how we implemented a job recommendation model. By leveraging the BERT model architecture and different tools for classification tasks, we were able to create a successful model with over 90 percent accuracy. The inclusion of Gustavus alumni data in our training process provides a more specified result for Gustavus students using our model. This research shows how modern solutions like machine learning and natural language processing can be implemented to enhance different fields such as education and the job market.

1 Introduction

The successful post-graduation transition for graduating students is critical to have a meaningful and purposeful career. Institutions can play an important role by providing tools that can help improve student decision making about their post-graduation plans. However, predicting post-graduation outcomes accurately remains challenging due to the complicated relationship between academic coursework and real-world experience. Students often start careers aligned with their studies, only to later transition to a field unrelated to their studies.

This project aims to develop a machine learning model that predicts potential jobs for graduating students based on their resume. The model utilizes key features such as academic major and previous work experience. To achieve this, we leveraged BERT, a state-of-the-art language model, which is a highly effective and versatile language model compared to older methods such as LSTMs and GRUs. By choosing BERT, we hope to achieve superior performance.

Our results demonstrate that the model achieves a predictive accuracy of over 90 percent. These findings indicate the feasibility of using data-driven approaches to improve career guidance systems, ultimately supporting student success.

2 Background and Related Work

We identified a study that focuses on using machine learning for career guidance to support students in career planning [4]. We also discovered a paper that implements the BERT model, helping us build the BERT model specifically designed for resume prediction.

2.1 Artificial Intelligence for career guidance - current requirements and prospects for the future

This study explores the application of machine learning in career guidance, highlighting its potential to assist students in career planning. Students showed interest in using AI for skill comparison, job suggestions, and personalized guidance. During the trials, AI applications for recommending jobs and courses received positive feedback, though students emphasized the need for greater accuracy and personalization. Students appreciated tools for skills profiling and career exploration but they raised concerns regarding privacy and data quality.

Despite their contributions, the study faced significant limitations. First, the AI applications used for this study were well-established but not state-of-the-art technology. To address this, our study implements the BERT model. Second, students rated the model's usefulness as median due to inconsistent predictions. Our study seeks to enhance the model's usefulness by improving prediction accuracy.

2.2 BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

The study introduces BERT(Bidirectional Encoder Representations from Transformers) [3], a model that pre-trains representations by leveraging context from both directions. Unlike traditional models that use a unidirectional approach, BERT employs masked language models and next-sentence prediction tasks during pre-training. These tasks enable BERT to outperform other models, including OpenAI GPT, in various benchmarks. This paper demonstrates BERT's effectiveness by advancing state-of-the-art NLP tasks such as question answering and sentiment analysis.

Additionally, the paper adds value to our project by detailing BERT's components and required training steps. While BERT can be implemented using PyTorch, understanding its functionalities remains crucial for recalibrating it to suit specific downstream tasks.

2.3 Bert Model Base

We based our model off of "ahmedheakl/bert-resume-classification" [1] on hugging face. It was a great resource to help us get started on working with BERT and the resume classification model that we built. It was a great starting point for the dataset preprocessing and the actual model. The model was already trained and processed so we had to go in and modify his code to be able to change the input dataset and change the hyperparameters but kept the overall architecture of the model.

3 Dataset and Evaluation

3.1 Midterm Dataset

Originally we had a person job fit model layout that we were basing our research on and the design that we were looking at had a dataset that went with it [2]. It looked good with 2 features of the resume and the title that went along with it but each resume was in a very specific format. It had to be in this specific format because the raw CSV file data went through data augmentation and then the tokenization for the BERT model so we had to make it the same style. We tried to adapt the LinkedIn data to that but it was time-consuming and difficult so we adjusted.

3.2 Final Dataset

We did some more looking around and found not only a better model that fits our outcome more closely but also a better dataset that had the resumes in plaintext and the corresponding label. The only downside to this switch was the decreased sample size from 30,000 to about 13,000.

3.3 Gustavus LinkedIn Alumni Data

Once we figured out the base dataset to use it took some trial and error to get the LinkedIn data to work. Inside LinkedIn there is a feature where you can download a PDF and it pulls all the relevant information from a given page like experience, education, etc.. We also tried to automate this process but couldn't in the time we had so we had to do it manually. Because of this, we chose to do two LinkedIn resumes per label in the large dataset. There were 43 labels so in total we added just under 86 labels because some labels don't have a corresponding Gustavus alumni in that field. Once we had all the PDFs we made simple Python code to read them in and transform them into a CSV file with the label and plaintext resume. Once we had that csv file we then simply added them to the large 13,000 dataset to make our final dataset that we used to train our model.

3.4 Preprocessing

To preprocess our data we did the basic preprocessing steps that are involved with getting data ready to use for a BERT model. We did tokenization which included adding [CLS], [SEP], and [PAD] to each of our resumes. We included padding of 512 and then preprocessed the labels. We did this by encoding the categories and mapping each string label to an integer for the BERT model to handle. After we did the tokenization and label processing we split the dataset into 20 percent test and 80 percent train. After we did these prepreprocessing steps we now had data that can be successfully used with BERT.

4 Methods

4.1 Changes we made

We made some big changes to our model from the midterm report that we were originally planning on using. Our midterm report model was based on a person job fit model which we were basing off a model that used distilBERT base uncased and moved to a model that used the regular BERT base uncased. The original model we were looking at was quite complicated and was hard to adapt to our needs. Notably, the person-job fit model works by taking in one job title and returning the best resume for that job and we wanted to switch those inputs and outputs so that it would take in a resume and return the best job for that resume. This posed a much more difficult task than expected and after we had got our model to successfully run, our accuracy was very low so we decided to go in a different direction.

4.2 Final Model

We did more research and found a model better fit for our outcome. The model we landed on was similar but used a BERT base uncased like I mentioned before and was much easier to work with. It took some work to get this new model working but once we did we could change anything we wanted to make it geared towards the outcome we wanted. For the layout of our model, we have the dataset that is tokenized and processed for our BERT model and used to train it. We initialized the model and made the training arguments where we set our hyperparameters. Then we made our evaluation metric which included accuracy, loss, precision, recall, and an f1 score. Once the model is trained then you can simply input a resume to the sent variable and it will run through the model and output the label that best fits that resume.

4.3 Features and Hyper parameters

The features that we ended up using were.'Category', 'Text', 'input ids', 'token type ids', and 'attention mask'. The original features were 'Category' and 'Text', which are the job title and resume respectively. Through these two features our BERT model could relate each category to each resume and with just those two main features can successfully output a favorable outcome for what we were looking for. For our hyperparameters, most of them are optimized in the BERT base uncased model but we did end up changing a couple. We kept the original optimizer which was Adam because, through homework and other research, it seemed to be the best option for our task. We changed the initial learning rate from 1e-4 to 2e-5 so we could start at a more accurate place while the Atom optimizer variably changes it. The other notable change we made was the epochs which was originally set at 3 but we changed to 5 because we noticed that the loss and accuracy kept getting better.

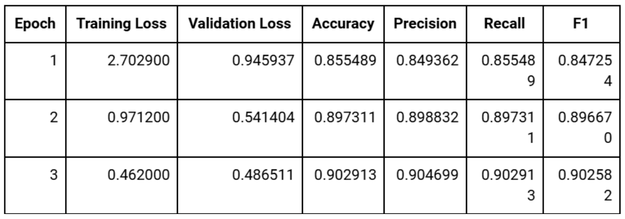
4.4 Overcoming Limitations

The final model that we created overcomes a lot of issues we had in our midterm proposal model. The current model we have is much more user-friendly and easier to read when looking at the code. The accuracy from our previous model to this one has also greatly increased with our current model having over 90 percent accuracy.

5 Experiments

In this section, we present the results of selecting and running different experiments to achieve the best performance. It is worth noting that experiment selection was constrained by the computational power of the machine, which limited our ability to run the model on multiple tests.

5.1 Model 1: Original Dataset and 3 Epochs



Epoch	Training Loss	Validation Loss	Accuracy	Precision	Recall	F1
1	2.702900	0.945937	0.855489	0.849362	0.855489	0.847254
2	0.971200	0.541404	0.897311	0.898832	0.897311	0.896670
3	0.462000	0.486511	0.902913	0.904699	0.902913	0.902582

Figure 1: Original Dataset+ 3 Epochs

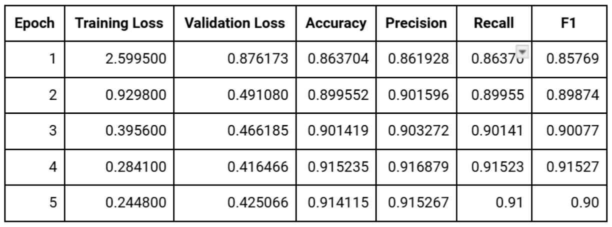
This table shows the training progress over 3 epochs. Key metrics include:

- Training Loss decreases significantly from 2.70 in epoch 1 to 0.46 in epoch 3
- Validation Loss decreases from 0.94 to 0.48, indicating improved generalization
- Accuracy, Precision, Recall, and F1 Score all improve steadily, stabilizing around 0.90 by epoch 3

Overall, the model shows consistent improvement in performance. However, we need additional tests to further improve performance.

5.2 Model 2: Original Dataset and 5 epochs

- Training Loss decreases significantly from 2.60 in epoch 1 to 0.24 in epoch 5
- Validation Loss decreases from 0.88 to 0.43, indicating improved generalization



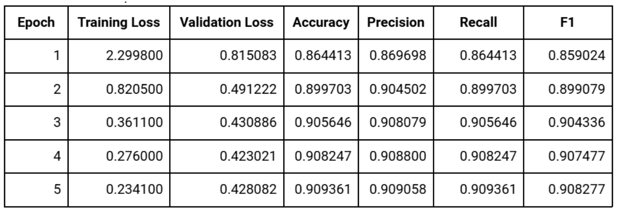
Epoch	Training Loss	Validation Loss	Accuracy	Precision	Recall	F1
1	2.599500	0.876173	0.863704	0.861928	0.86370	0.85769
2	0.929800	0.491080	0.899552	0.901596	0.89955	0.89874
3	0.395600	0.466185	0.901419	0.903272	0.90141	0.90077
4	0.284100	0.416466	0.915235	0.916879	0.91523	0.91527
5	0.244800	0.425066	0.914115	0.915267	0.91	0.90

Figure 2: Original Dataset+ 5 Epochs

- Accuracy, Precision, Recall, and F1 Score all improve steadily, stabilizing around 0.91 by epoch 5

The results indicate that the model performs better with 5 epochs compared to 3 epochs. We did not present results for testing our model beyond 5 epochs because the model failed to improve.

5.3 Model 3: Additional Gustavus Dataset + 5 epochs



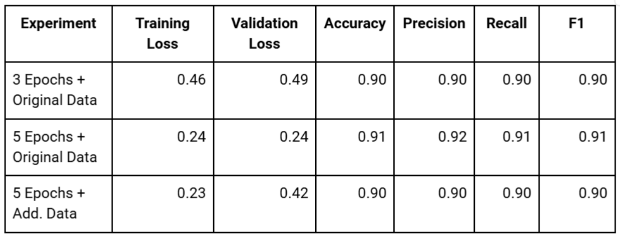
Epoch	Training Loss	Validation Loss	Accuracy	Precision	Recall	F1
1	2.299800	0.815083	0.864413	0.869698	0.864413	0.859024
2	0.820500	0.491222	0.899703	0.904502	0.899703	0.899079
3	0.361100	0.430886	0.905646	0.908079	0.905646	0.904336
4	0.276000	0.423021	0.908247	0.908800	0.908247	0.907477
5	0.234100	0.428082	0.909361	0.909058	0.909361	0.908277

Figure 3: Original Dataset+ 5 Epochs

- Training Loss decreases from 2.30 in epoch 1 to 0.23 in epoch 5
- Validation Loss decreases from 0.81 to 0.42, indicating improved generalization
- Accuracy, Precision, Recall, and F1 Score all improve steadily, stabilizing around 0.90 by epoch 5

We can observe that adding 100 data units only slightly improved model performance.

5.4 Comparison of Results



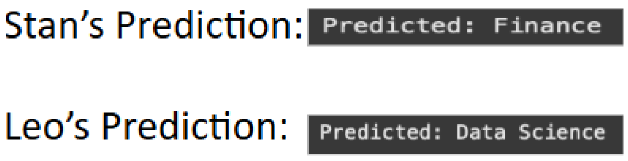
Experiment	Training Loss	Validation Loss	Accuracy	Precision	Recall	F1
3 Epochs + Original Data	0.46	0.49	0.90	0.90	0.90	0.90
5 Epochs + Original Data	0.24	0.24	0.91	0.92	0.91	0.91
5 Epochs + Add. Data	0.23	0.42	0.90	0.90	0.90	0.90

Figure 4: Comparison of Results

- 3 Epochs with Original Data: Achieves an accuracy, precision, recall, and F1 of 0.90 with training loss of 0.46
- Improves performance slightly, achieving 0.91 accuracy and precision with a lower training/validation loss (0.24).
- Accuracy remains at 0.90, but training loss is the lowest at 0.23

This comparison highlights the effects of increasing epochs and incorporating additional data on model performance.

5.5 Real World Predictions



Stan's Prediction: Predicted: Finance

Leo's Prediction: Predicted: Data Science

Figure 5: Stan's and Leo's predictions

As we have shown earlier, the additional data provide only marginal performance improvements. However, achieving better performance for an already high-performing model will require substantially more computational resources. In our opinion, the most important factor for evaluating results is to test the model's ability to predict accurately for a specific institution. The image above indicates that our model successfully classified the Gustavus student's job results.

6 Discussion

6.1 Tests

We did a couple of tests with our model to test the accuracy of the outputs. As far as the scope of the model, it performs very well. The only issue that we've run into is the lack of labels for specified job recommendations. For example, Stan inputted his resume and got an output of "Finance". This seems correct for his resume but that isn't necessarily what he wants to go into. Finance has many subsets and ideally, those would be reflected in our model but with limited time and resources we weren't able to implement specific labels. However with the labels that we had it performed very well to sort where a resume should go. With Leo's resume, he wants to go into Machine Learning but since that wasn't a label, he got "Data Science" which is the next closest label for what he wants to do. When testing other resumes we noticed the same results by looking through it, understanding what job the resume currently had, and seeing what the model predicts, it almost always gets it correct. In all, we weren't surprised at the results because we knew that the lack of label specification would be a limiting factor but inside of the label limitations it does very well in its classification task.

6.2 Future Work

We have a good start to what we want our final model to look like but with the limited time we have we have some more things we would like to accomplish.

We were limited with the dataset we had and ideally, we would want to make a dataset with all Gustavus alumni data that could be specified to a specific field. For example, we could make a specified Gustavus computer science dataset and model for computer science majors to see what field they should go into based on their resume. With a more specialized dataset, we could then control the labels that are imputed so it's not as general as our current model is.

We would also like to implement a UI for our model where you could input your resume as a PDF and it would convert it to plaintext and run through our model in the back-end and produce the output to the user.

7 Conclusion

In this study, we developed a BERT model aimed at predicting post-graduation careers for graduating students. By leveraging a BERT model, we successfully built a model capable of accurately assigning job labels based on a given resume. These findings demonstrate BERT's effectiveness in performing specific downstream tasks. Additionally, our paper presents compelling arguments for institutions to adopt machine-learning tools to guide their students in career planning.

Despite its contributions, this investigation is subject to certain limitations. First, while including a diverse range of job categories for prediction, expanding labels will be required in the future to enable more precise predictions. Second, institutions implementing this model will need to supplement it with additional data customized to their specific contexts.

In conclusion, the results of this study highlight the potential of machine learning adaptation within academic institutions. This research provides a foundation for further exploration of practical applications of machine learning.

References

[1] Ahmed Heakl, Youssef Mohamed, Noran Mohamed, Ali Sharkaway, and Ahmed Zaky. Resumeatlas: Revisiting resume classification with large-scale datasets and large language models. *arXiv preprint arXiv:2406.18125*, 2024.

[2] Wahib Mzali. Multilabel resume dataset. https://www.kaggle.com/datasets/wahib04/multilabel-resume-dataset, 2021.

[3] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108, 2019.

[4] Stina Westman, Janne Kauttonen, Aarne Klemetti, Niilo Korhonen, Milja Manninen, Asko Mononen, Salla Niittymäki, and Henry Paananen. Artificial intelligence for career guidance – current requirements and prospects for the future. *IAFOR Journal of Education: Technology in Education*, 9(4):43–62, 2021.