National College of Ireland

Configuration Manual

MSc Research Project
Master of Science in AI

Alex Power
X24100421

School of Computing
National College of Ireland

Supervisor: SM Raza Abidi

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: Alex Power

Student ID: X24100421

Programme: MSc in Artificial Intelligence

Module: Practicum

Supervisor: SM Raza Abidi

Submission Due Date: December 12 2025

Project Title: Comparative Evaluation of Resume–Job Matching Using Transformer-Based Semantic Similarity and TF-IDF

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Alex Power

Date: 24/11/2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	✓
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Comparative Evaluation of Resume–Job Matching Using Transformer-Based Semantic Similarity and TF-IDF

Alex Power
x24100421@student.ncirl.ie

1. Introduction

This configuration manual provides a detailed explanation of how to correctly use the system created for this research. This research aims to evaluate the effectiveness, accuracy, computational efficiency, and human-decision alignment of two models at the task of matching and ranking job postings to a given resume.

This research uses two models, SBERT MiniLM-L6-v2, a transformer-based semantic similarity model, and a traditional TF-IDF model as the baseline. SBERT uses contextual embeddings for resume-job matching, allowing for a deeper understanding of each job description allowing for semantic matching. TF-IDF is a traditional sparse-vector model which primarily uses text-matching to match and rank jobs, with little ability to understand the broader context of jobs and resumes.

The purpose of this research is to determine the extent of each models performance, accuracy, and how likely the ranked jobs would align with human decision. Furthermore, the computational requirements will be noted, including the specifications of the computer ran on.

This manual outlines the system specifications, software used, dataset source and preparation, execution of code with graphs, and instructions on how to run the system, including the Streamlit dashboard used for human evaluation testing.

2. System Specification

To ensure the successful execution of the code, your system should meet the following minimum specifications:

- **Operating System:** Windows 10 or higher, macOS 10.15 or higher, or a Linux distribution (e.g., Ubuntu 20.04).
- **Processor:** Intel Core i5 or equivalent AMD processor (Quad-Core or higher).
- **Memory:** 8 GB RAM, (16 GB recommended)
- **Storage:** At least 10 GB free SSD space.
- **GPU**: Not required.
- **Python Version:** Python 3.9 or higher.

3. Software Used

The following software and libraries are used in this project:

3.1 Programming Language

- Python: The primary programming language used for data preprocessing, model implementation, similarity computation, running the Streamlit dashboard, and evaluation metrics.

3.2 Python Libraries

- NumPy: For numerical operations.
- Pandas: For loading, cleaning, preprocessing resume/job postings and analysis.
- Scikit-learn: For TF-IDF vectorization, cosine similarity scoring, and evaluation metrics.
- Sentence-Transformers (SBERT MiniLM-L6-v2): For generating semantic-embeddings used for ranking resume-job matches.
- Matplotlib: For data visualization.
- Plotly: For interactive visualizations.
- Streamlit: Running the interactive dashboard used for human participant evaluation.
- Joblib: For saving and loading precomputed results to ensure participants were not left waiting in between evaluations.

3.3 Development Environment

- Jupyter Notebook and Visual Studio Code: For running and editing the Python scripts.

3.4 Optional Tools

- Anaconda: For managing temporary environments and dependencies.
- Git & Github: Used for version control and project backup.

4. Dataset Source

The primary dataset used for this research was sourced from Kaggle, titled “LinkedIn Job Postings (2023 - 2024)” (Koneru_2024). A secondary dataset from Kaggle, titled “Resume Dataset”, which was used for creating artificial resumes for matching to ensure anonymity (Bhawal_2021).

- Job Description Dataset:
  The primary evaluation dataset is a collection of job postings which was collected from LinkedIn from 2023-2024. This dataset is publicly available on Kaggle and lists a diverse range of job descriptions, perfect for this research.
- Resume Dataset:
  A resume dataset was used to guide the structure, writing style and creation of synthetic resumes. These resumes were created for the testing to ensure anonymity, while

ensuring the resumes reflected real-world education, job titles and skills. These were used by participants in resume-job matching ranking evaluation.

The Job Description dataset can be accessed at:
https://www.kaggle.com/datasets/arshkon/linkedin-job-postings
The Resume dataset can be accessed at:
https://www.kaggle.com/datasets/snehaanbhawal/resume-dataset

The Job Description file is located in the input folder in the root and is named "job_postings.csv". The synthetic resumes are located in the input folder and named respectively.

5. Execution of the Code Implementation

Follow these steps to correctly install and execute the resume-job matching system using both TF-IDF and SBERT models.

Step 1: Setting Up the Environment

A. Create a New Environment:

In the terminal, cd into the correct folder and run the following commands one after the other. This will create a virtual environment.

```
python3 -m venv venv
venv\Scripts\Activate
```

B. Install Required Libraries:

To install the required libraries, run the following command.
Pip install -r requirements/requirements.txt
This will install sentence-transformers, scikit-learn, pandas, numpy, streamlit, joblib, matplotlib and plotly.

Step 2: Preparing the Dataset

A. The dataset is already placed in the input folder, under jobs.xlsx.
B. The models are able to access this for the running and evaluation of the system.
C. To use another dataset, simply place dataset in the input folder and rename to jobs.xlsx.
D. The resumes are also already placed in the input folder with names relating to the domain they are related to. To use ulterior resumes, when on the Streamlit dashboard, drag and drop the .txt file into the box to run evaluations.

Step 3: Running the Code

A. Open Jupyter Notebook or VS Code in the project directory.
B. The preferred method of running the system is utilizing the Streamlit Dashboard.
C. Run the following command to open a localhost of the Streamlit Dashboard:

```
streamlit run ui_app.py
```

D. The dashboard will prompt the user for a name, for this research, I have been inputting a number to ensure real names are not recorded.

Figure 1: System Dashboard for Qualitative Evaluation

E. As seen in figure one, this dashboard allows the user to upload a resume, run both models and view the results. It allows for an anonymized view of both model outputs (to ensure results aren't skewed from knowing the model).

Figure 2: System Dashboard wit Job Titles loaded

F. As seen in figure two, the user can select from both models, and allows the user to rate the ranked jobs on a Likert Scale of 1-5, where 1 is irrelevant and 5 is highly relevant (Perfect Match). The results can be accessed from the ratings folder in the folder root.

Additional System Running (Optional)

A. The system can also run a batch of resumes against the job description dataset. This was used to ensure participants were not waiting for a model to run in between each evaluation. These can be accessed on the sidebar by selecting a precomputed results and clicking the button below.
B. To create a batch run, refer to the batch_run.py file which has hardcoded resume file names, and run the following command:

```
python batch_run.py
```

C. This script processes multiple resume outputs, with runtime comparisons, and similarity scores.

Step 4: Model Evaluations

A. The dashboard contains some preliminary evaluations including similarity score and runtime. In addition to this, batch_run.py file was used to compute additional evaluations used for this research. Thes include MRR, nDCG, Precision@K, Recall@K, Spearman Correlation, Runtime, and Memory Usage. These evaluations and supplementary graphs can be found in the evaluation.ipynb and human_evaluation.ipynb.
B. Both of these notebooks were used for further evaluation and testing consistency of outputs. The MRR and nDCG are based on preliminary human feedback as a baseline to ensure results are not skewed. Prior to this method, all MRR, and nDCG results were returning a 1.0.
C. Results from the human-alignment analysis can be found in the results folder in the root of the project directory as ratings.csv and are anonymized
D. Graphs of results have been provided in the above mentioned evaluation notebooks.
E. Additional Graphs and Tables used for this research will be provided in the appendix of this configuration manual.

Step 5: Running the System

A. Using the Streamlit Dashboard, the user is prompted to give their name. I suggest inputting a number to bypass this as it is primarily to ensure all participants completed their ranking of all outputs.
B. The user is then shown an upload box where they can upload a resume. Refer to the synthetic resumes in the input folder for this. Once one is chosen, the user can click the run matching button.
C. The system will run for a few minutes, updating the user on which model is currently being ran.
D. Once the models have been ran, the output will show the top five results for each model, evaluations at the bottom and similarity score for each matched job. The user is able to click on a matched job title and it shows the full job description, with the ability to rank it at the bottom of this popup. There is also space for a comment, but this is optional.
E. In the sidebar, there is an option to show pre-loaded results from both models. By selecting and clicking the button below, the user will be shown the results for each

model. Visually, it looks the same as if the user had ran them. These pre-loaded results are saved as json files in the batch_outputs folder.

Further explanations will follow in the System Architecture section.

6. System Architecture

The system architecture for this research is designed for the models to both run through one combined pipeline, allowing for a majority of aspects of this research to run from one command. The architecture is modular, allowing each component to operate independently, allowing for changes to be made easily to the pipeline.py without needing to touch code that's completely fine.

A. System Components

The system is comprised of six core elements. The data input layer is managed in the data_loader.py file and manages loading the resumes and job descriptions. The preprocessing layer cleans the files, which is also part of the data_loader.py file. The model layer consists of both models in the tfidf_model.py file and the bert_model.py file. These files manage the models, loading them, and applying them to the datasets, including the ranking. The similarity and ranking module consists of the evaluation.py file where initial evaluations were collected. The user interaction module consists of the streamlit interface which is comprised of ui_app.py and app.py. These files are for the dashboard and allowing the system to run from it. The final module is the evaluation layer which consists of final evaluations collected from users and additional metrics not calculated in the similarity ranking module.

B. Folder Explanation

The ratings folder consists of the base_ratings folder which was used for setting a baseline for the evaluations Precision@K, Recall@K and Spearman Correlation. The ratings.csv consists of all the human-centered feedback from the qualitative evaluation.

The batch_outputs folder consists of .json files of pre-computed runs from the system. This was essential, as in the first test of running the system, it took too long for each model to run one after the other and it would have kept the participant waiting around ten minutes in-between runs. The input folder consists of the job descriptions, saved as jobs.xlsx, and the resumes which are named according to their domain.

The requirements folder consists of requirements.txt which is necessary for efficiently installing all required libraries when running the system.

7. Conclusion

This configuration manual outlines the steps required for setup and execution, while providing an explanation of the structure, system requirements and software used. This system was developed for evaluating the SBERT model and TF-IDF at the task of resume-job matching. The purpose of this research was to highlight the performance of each model and the limitations, providing situations where each model might best perform. The qualitative evaluations help

provide added context to how the models perform in relation to how a human might decide on the selection of jobs/resumes.

The manual detailed the system architecture, explaining each file and folder, to ensure that the reader fully understands the methods and reasoning behind the choices made. The reader has been guided through how to set up, run and evaluate the model themselves to apply this system to conduct further research. The use of the Streamlit dashboard was documented to ensure that future testing is clear and easy to perform.

The system supports both quantitative and qualitative evaluation. The quantitative evaluations include MRR, nDCG, Precision@K, Recall@K, Spearman Correlation, runtime and memory usage. Qualitative evaluation consists of peer-centered feedback ranking the outputs of the models on a Likert Scale of 1-5.

This manual should provide all the information required to understand, set up and run the system, and evaluate. This allows for future research to replicate the results to explore the strengths and limitations of the models tested.

References

1. Bhawal, S. (2021). *Resume Dataset*. [online] www.kaggle.com. Available at: https://www.kaggle.com/datasets/snehaanbhawal/resume-dataset [Accessed 24 Nov. 2025].
2. Koneru, A. (2024). *LinkedIn Job Postings (2023 - 2024)*. [online] www.kaggle.com. Available at: https://www.kaggle.com/datasets/arshkon/linkedin-job-postings?select=postings.csv [Accessed 24 Nov. 2025].
3. Python Software Foundation (2019). *Python.org*. [online] Python.org. Available at: https://www.python.org.

Appendix

As part of the evaluations of the research, supplementary graphs were created to support the analysis. Not all of these visualizations were included in the report, as they were not essential to the core of the discussion, with some repeating information another graph displays. Metrics such as Precision@1/3/5 had similar results for all, and while they were noted and discussed, each Precision@K did graph did not warrant additional figures. For a complete overview of graphs used in this research, they have been presented below. They can also be found in the artefact folder in the human_evaluations.ipynb notebook.

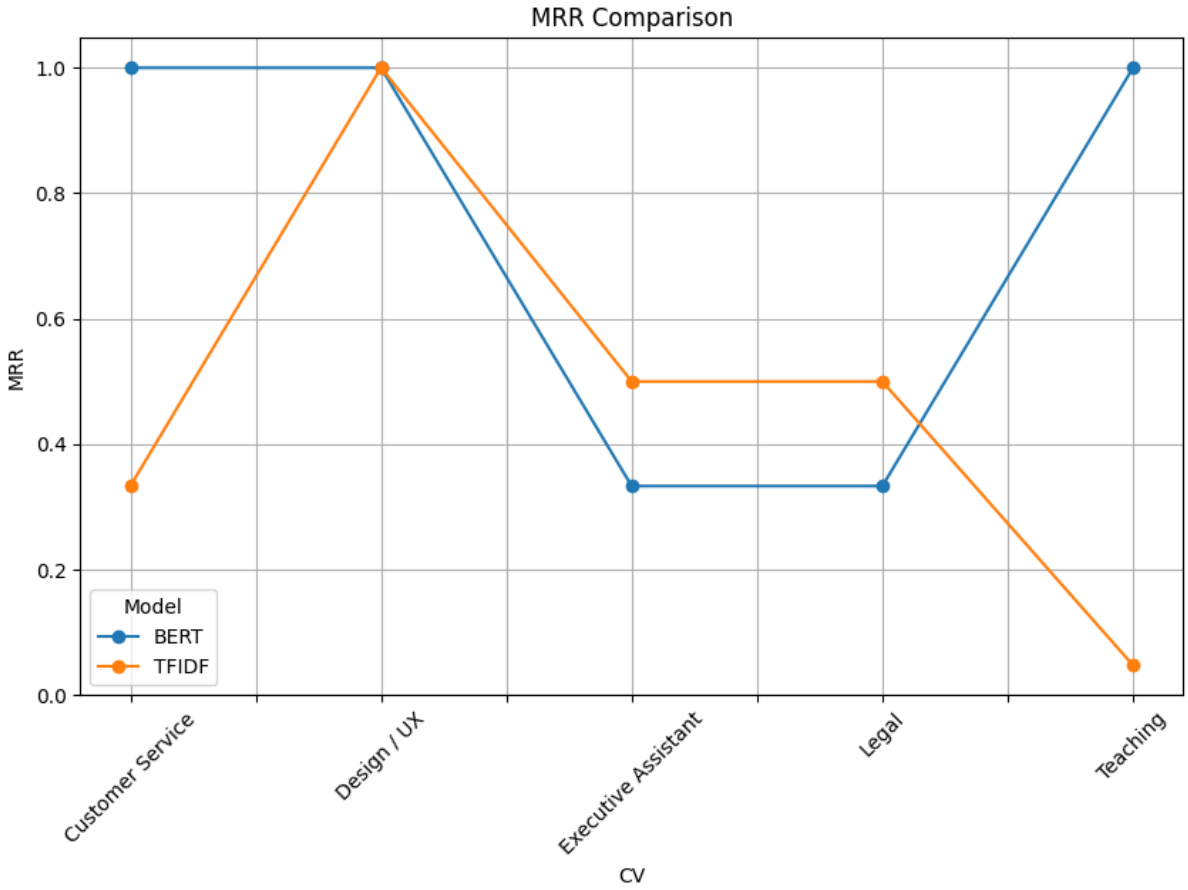

Figure 3: Graph of MRR Comparison Between Models

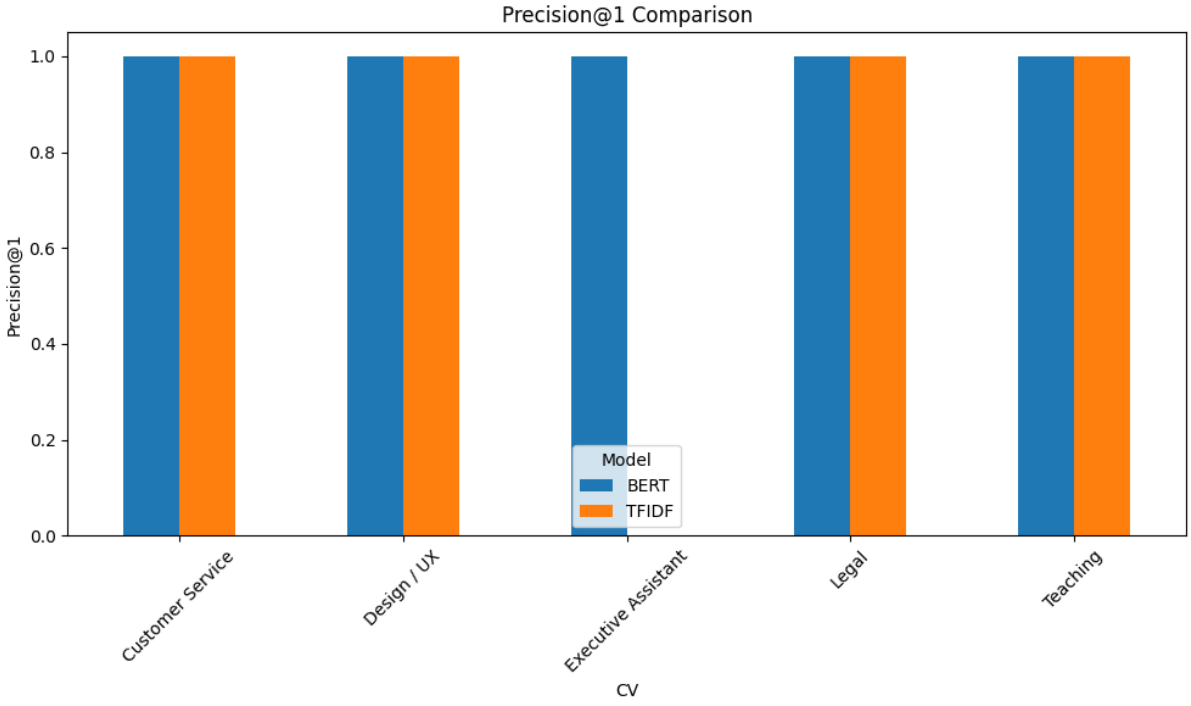

Figure 4: Graph of Precision@1 Comparison

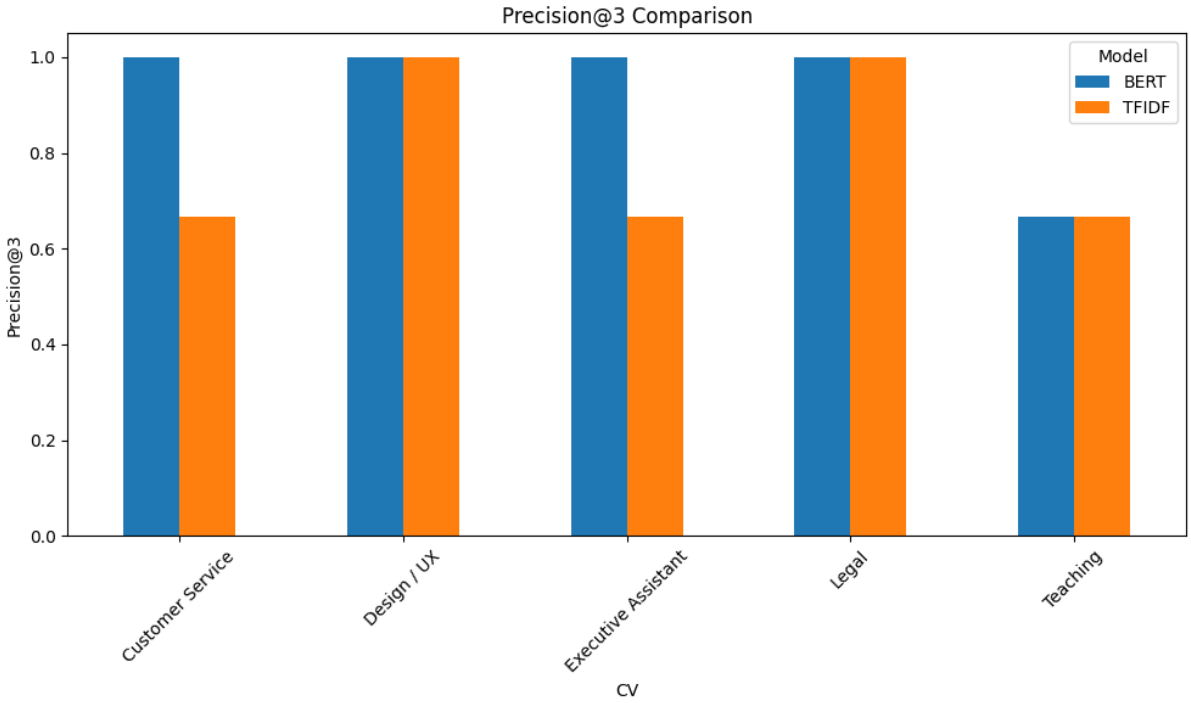

Figure 5: Graph of Precision@3 Comparison

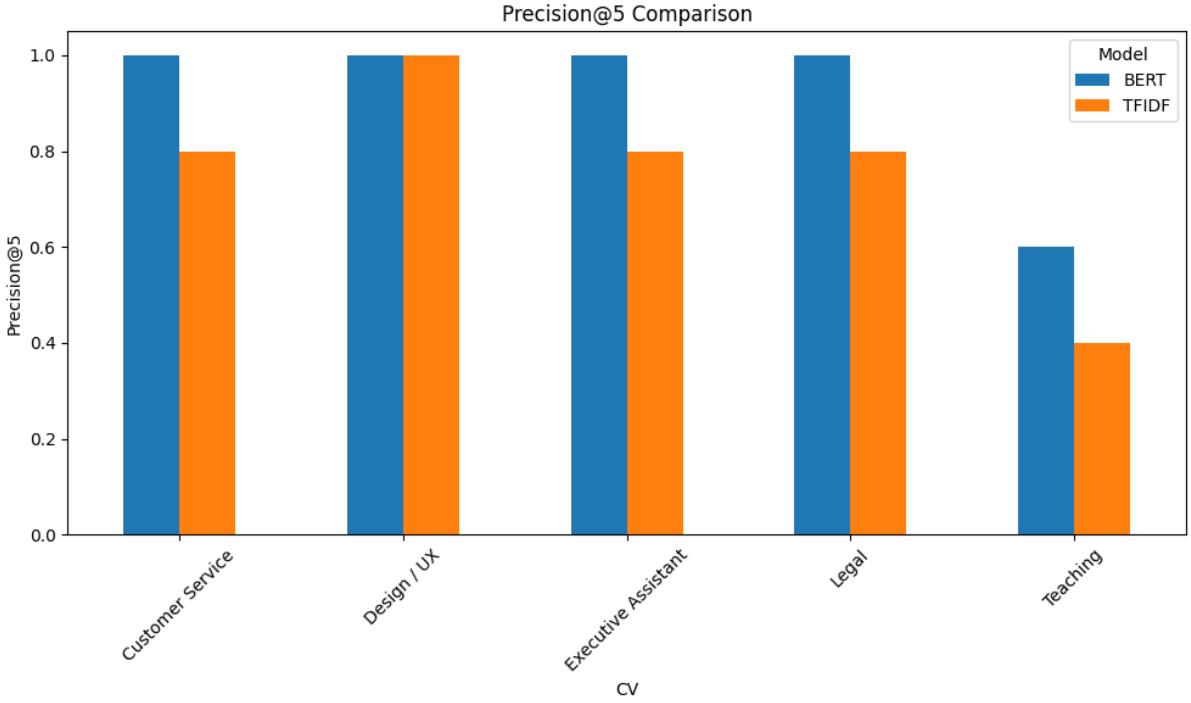

Figure 6: Graph of Precision@5 Comparison

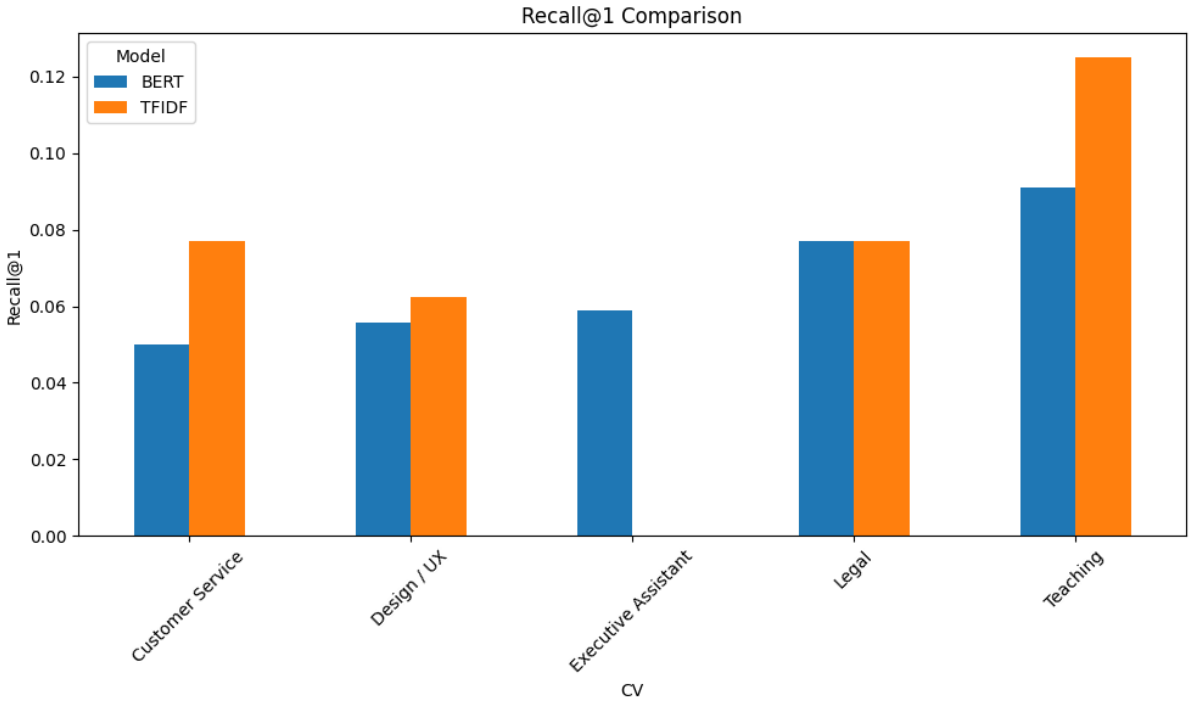

Figure 7: Graph of Recall@1 Comparison

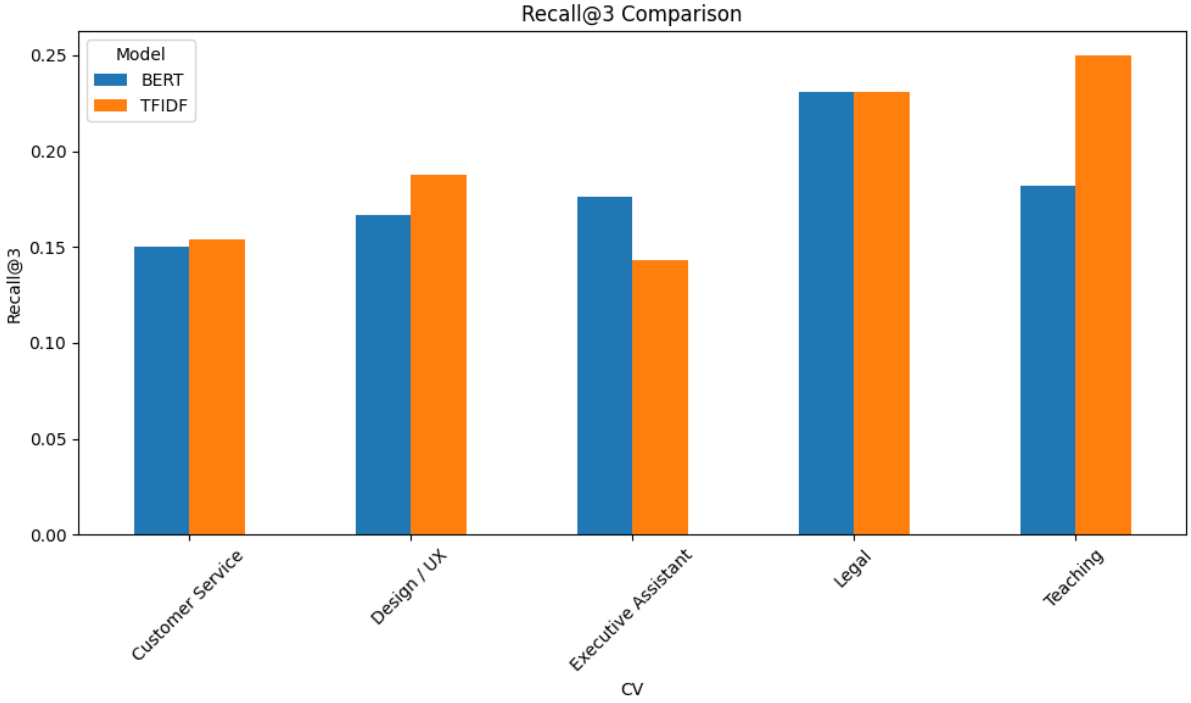

Figure 8: Graph of Recall@3 Comparison

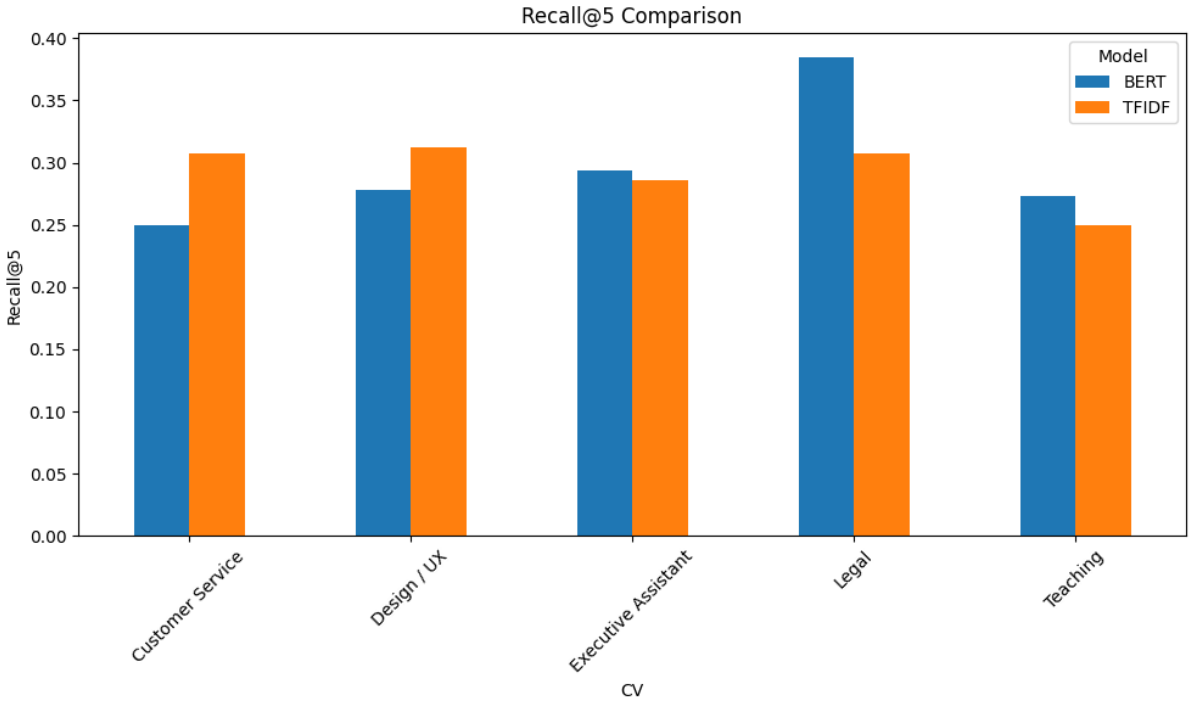

Figure 9: Graph of Recall@5 Comparison

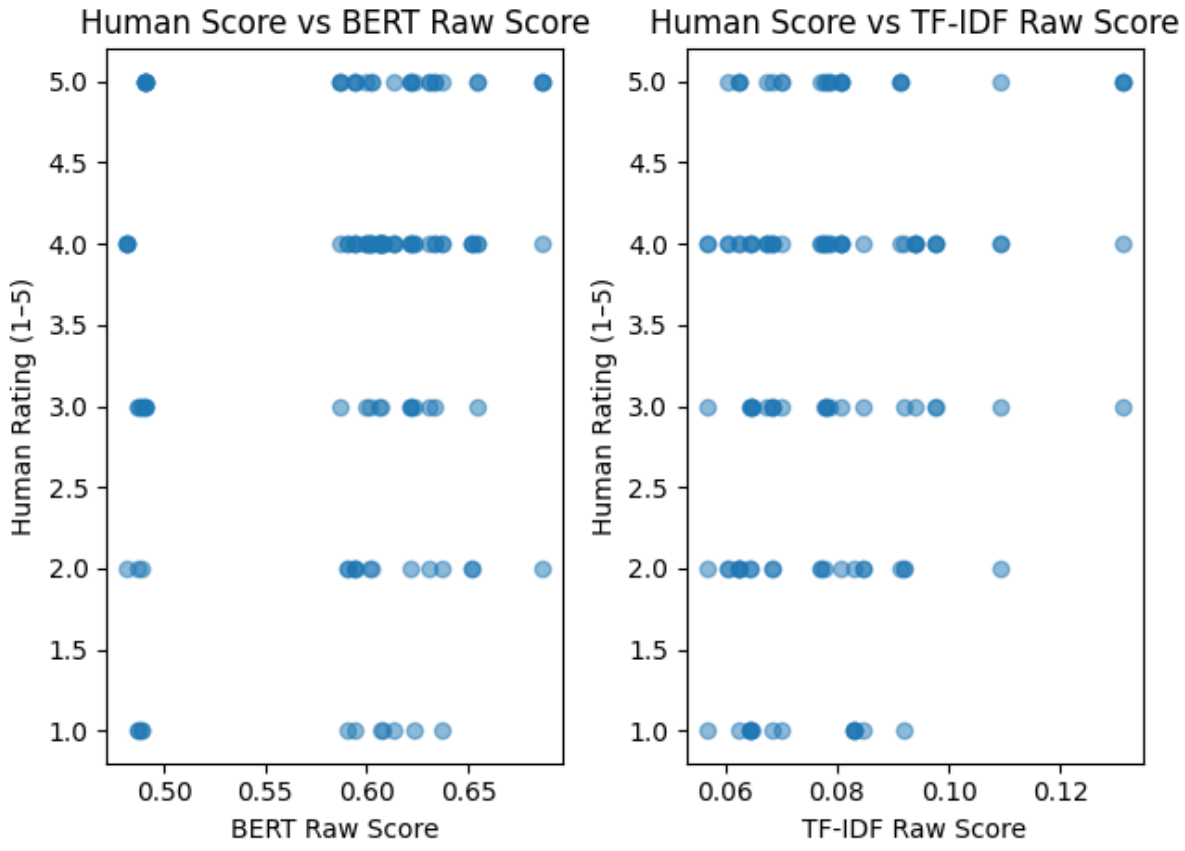

Figure 10: Graphs of Human Score vs Model Raw Score

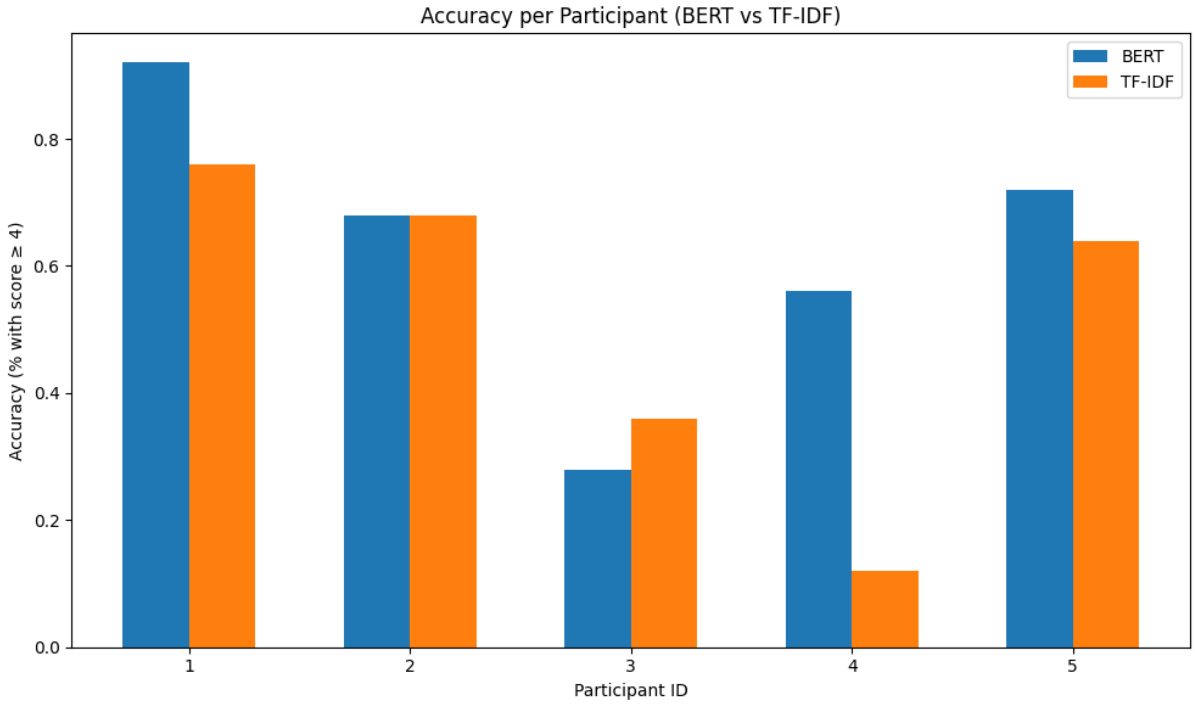

Figure 11: Participant Accuracy per Model