Empowering Job Seekers with Advanced NLP: A Revolutionary Approach to Career Navigation in the Modern Job Market

DATS 6202– Natural Language Processing

Professor: Amir Jafari

By: Nammin Woo

Table of Contents

1. Introduction

This project addresses the gap in current job search tools, such as the ONET® Database shown in Fig.1, which primarily focuses on manual navigation and job title-based searches, often not adequately aligning with a job seeker's unique skills and career level. Our advanced Natural Language Processing (NLP) program is designed to overcome these limitations by offering a more dynamic, user-centric approach. It uses modern NLP techniques to streamline job searching, helping individuals find positions that precisely match their skills and career aspirations. The project comprises two main components: a Text Classification Model for accurately predicting job levels in resumes, and a Semantic Similarity Search using Siamese-BERT Networks. By deploying these innovative tools, we aim to transform the job-hunting experience into one that is more efficient, targeted, and aligned with the individual career development goals of job seekers.

Fig.1

Existing Tool: O-NET Online

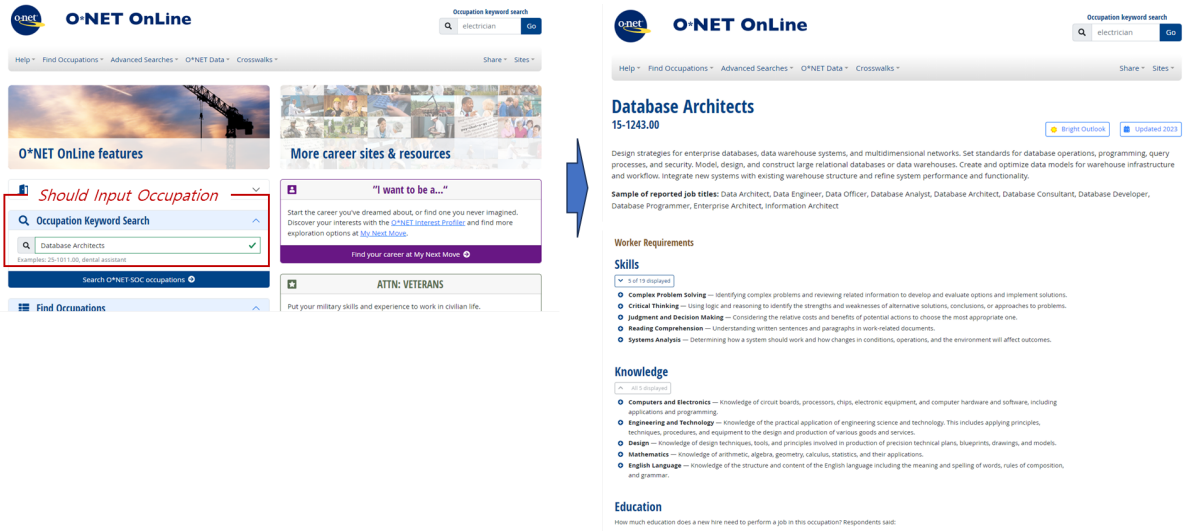


figure 1:ONET database page where job seekers type the job title they are looking for and the database shows results of the skills, knowledge, and education needed for this occupation.

2. Personal contribution to the project

My contribution to this project focused on **developing a Job Corpus** using ONET Data, developing a model based on **RoBERTa Transformer Fine tuning**, applying the **Semantic Similarity Search** methodology, and constructing **a job recommendation process** for job seekers to implement Streamlit application. Because ONET Data contains extensive content, a significant amount of time was spent defining the data needed for this project and defining preprocessing requirements. In addition, in the case of Transformer Fin tuning, the form of input data was converted to suit the form required by the model at the beginning of development and the initial training results were calculated through head modification to enable subsequent development. One interesting thing is that compared to simply modifying and tuning the final layer according to the number of label classes, a relative performance improvement was observed **when adding a dropout layer.** Lastly, organizing the job seeker job recommendation process was a good opportunity to supplement the weaknesses of the existing service and think about how to provide meaningful value from the perspective of actual job seekers. We reviewed various hypotheses, but after intense discussion with our team members, we were able to adopt a method of first inferring a broad level through a job level prediction model and additionally recommending job groups with high similarity at that level.

3. Result

3-1. **Develop a Job Corpus**

A unique JOB Corpus was created by the following preprocessing steps and rules:

Fig.2

```
def job_corpus(df, idx):
    df = copy.deepcopy(df)
    df_names = ['df_Abilities', 'df_Knowledge', 'df_Skills', 'df_Tech_Skills', 'df_Interests']
    df_name = df_names[idx]
    field_name = df_name.split('_')[1] # Extract category field from dataframe name
    scale_names = ['LV', 'LV', 'LV', '', 'OI'] # 'LV', df_Tech_Skills: none, Interest: 'OI'
    if df_name != 'df_Tech_Skills':
        # 1. Filter Level scale > 50%
        #filter = (df['Scale ID'] == scale_names[idx]) & (df['Value_ratio'] > 0.5)
        #df = df[filter]
        # 2. Choose Top 3 High Demand Elements in an Occupation
        job_col = ['O*NET-SOC Code', 'Title', 'Description_SOC']
        sort_rule = ['O*NET-SOC Code', 'Value_ratio', 'Element ID']
        ele_col = ['Element Name', 'Description_Ele']
        df = df.sort_values(sort_rule, ascending=[True, False, True])
        df = df.groupby(job_col).head(5) #leave the top 5 elements by each Job code
        # 3. Merge 'Description_Ele' of 5 elements into 'Description_top_ele'
        df['Description_top_ele'] = df.groupby(job_col)['Description_Ele'].transform(lambda x: ' '.join(x)
```

figure 2:Function() to make individual corpus contains sorting and aggregating rules

Step 1. Read bulk data (*.zip file) and unzip to unit files

- 40 individual text files contain either Base information like code-description mapping or specific Element Contents specified in each subcategory information

Step 2. Pick key categories and Generate Individual Corpus

- Choose Resume related 5 categories - Ability, Skill, Tech-Skill, Knowledge, Interest Category, and perform primary aggregation (Contents data + Base information)
- Create ‘Job Corpus’ of each Category

  * Each category has about 50 unique elements which describe specific contents in a category, and the corpus was constructed by extracting 5 highest demand elements in each category

Step 3. Create Aggregated Job Corpus data based on Occupation dataset

- Generate Full Corpus and Add Job Level label (classification model target)

  *Summarized job description + Combination of job corpus in each category

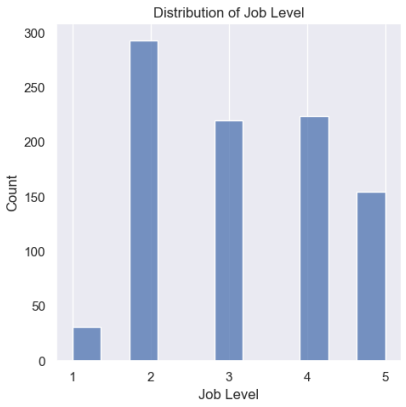
Below are a result of basic exploration of the data.

Fig 3.

Job Corpus Data

```
C.Features:
O*NET-SOC Code           object
Title                    object
Description              object
Description_Abilities    object
Description_Knowledge    object
Description_Skills       object
Description_Tech         object
Description_Interests    object
Description_Job          object
Job Zone                 float64
dtype: object
```

```
D.Null value check:
O*NET-SOC Code           0
Title                    0
Description              0
Description_Abilities    0
Description_Knowledge    0
Description_Skills       0
Description_Tech         0
Description_Interests    0
Description_Job          0
Job Zone                 93
dtype: int64
```



	Description_Job	Job Zone
Chief Executives	Determine and formulate polic...	5.0
Chief Sustainability Officers	Communicate and ...	5.0
General and Operations Managers	Plan, direct, ...	4.0

Fig 3. The Final ONET Job Corpus Data has 1,016 records with 10 features. Job levels are uniformly distributed except level 1.

3-2. **Classification model**

Implementation of a text classification model was conducted to ascertain the job level by analyzing the contextual information present in the Job Corpus data. The concept of job levels in this context refers to the categorization of various occupations based on comparable levels of experience, educational background, and requisite training as in Table 1. This approach enables a nuanced understanding of where a particular occupation fits within a broader professional hierarchy.

Table 1.

Definition of Job Level

Level	Expected Position	Description
1	Intern	Little or No Preparation Needed
2	Entry Level	Some Preparation Needed
3	Junior	Medium Preparation Needed
4	Mid Level	Considerable Preparation Needed
5	Senior	Extensive Preparation Needed

Table 1: description of job level categories

The modeling strategy included a spectrum of techniques, from the training of classical models from the ground up to the fine-tuning of pretrained transformers. Architectural modifications were also explored, specifically by augmenting the transformer model with additional layers at the head. Moreover, there was an experimentation with the recomposition of the job corpus, selectively concentrating on certain categories to sharpen the model's capacity for recognizing and emphasizing pertinent data.

Fig 4.

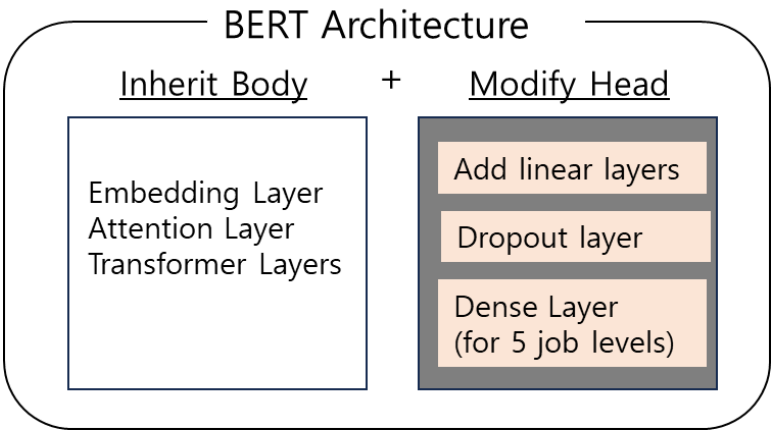


Fig 4. Modifying trials in BERT Architecture

Table 2 stands for specific training history in building text classification model.

When it comes to comparing predict power, **four points were derived as followings**.

1. Classical models like logistic regression also show relatively good performance
2. In overfitting situation, simply modifying final layer of transformer does not guarantee success.
3. Adding a dropout layer could help mitigate overfitting (RoBERTa Tuning cases)
4. When shorten input data length by recomposing Input categories (pick three among full 5 categories), performance was slightly improved (Best model)

Table 2. *Specific History of Training*

#	Base model	Accuracy	Description
NB	Scratch	63.90%	
logistic	Scratch	**69.68**%	
MLP	Scratch	69.00%	
BERT	Tuning	46.21%	Modify only the final layer
XLNet	Tuning	57.04%	Modify only the final layer
RoBERTa	Tuning	61.01%	Modify only the final layer
RoBERTa	Tuning	**61.37%**	Add drop out Layer
ELECTRA	Tuning	65.34%	
BERT(Grid Search)	Tuning	68.95%	Hyper Parameter Turning
BERT(Grid Search)	**Tuning**	**72.00%**	**Recompose Corpus**

3-3. **Siamese-BERT Networks**

A key feature of the approach is the use of Siamese-BERT Networks (Sentence-BERT) to perform semantic similarity searches. This advanced model, an evolution of BERT, employs cross-encoder networks and is fine-tuned using Siamese and triplet network structures. It generates fixed-sized sentence embedding vectors for effective semantic analysis (Reimers et al., 2019).

2.3.2 Embedding Vectors

We produce embedding vectors for both job description corpus and resumes, allowing us to infer semantic similarities between the two. This approach is crucial in identifying occupations that match the context of a resume, thereby enabling more accurate and relevant job recommendations for job seekers.

Fig 5.

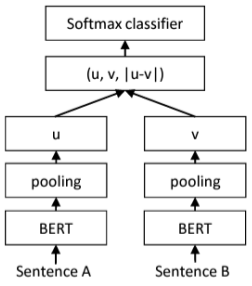


Figure 1: SBERT architecture with classification objective function, e.g., for fine-tuning on SNLI dataset. The two BERT networks have tied weights (siamese network structure).

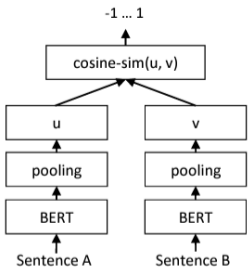


Figure 2: SBERT architecture at inference, for example, to compute similarity scores. This architecture is also used with the regression objective function.

Fig 5. Structure of Sentence-BERT (Reimers et al., 2019)

3-4. **Job recommendation process**

The Streamlit application serves as the interactive front-end of our system, designed to seamlessly integrate the sophisticated backend models with a user-friendly interface. This application is essential for processing resumes uploaded in PDF format as in Figure 9, where it extracts and structures text data for analysis, transforming unstructured resume content into a format amenable to computational processing.

Key to the application's functionality is the employment of a BERT-based classifier for predicting job levels. This classifier analyzes resumes against the backdrop of job descriptions, determining the most suitable job level for each user. The results are presented in a clear, interpretable format, aiding users in understanding their career prospects.

Additionally, the application offers job recommendations based on semantic similarity analysis. It compares the content of the user's resume with pre-encoded job descriptions, suggesting jobs that align with the user's skills and experience.

Fig 6.

Streamlit architecture

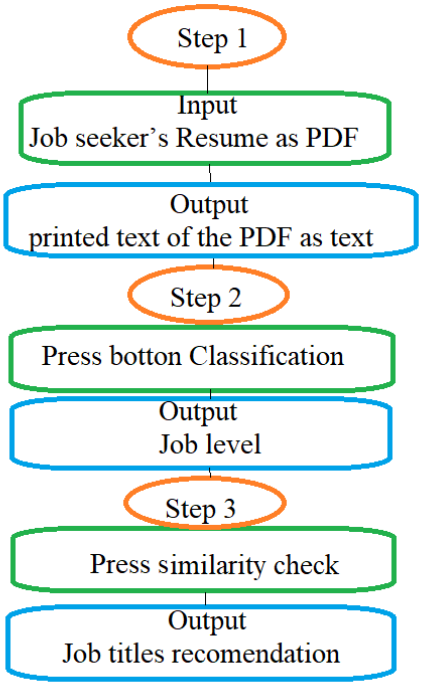


Fig 6. Final Streamlit looks like with three main functions: reading a file and tokanize, classifying job level, and suggesting job titles

This feature not only enhances job relevance but also personalizes the job-seeking experience. By helping them to gain the initiative to search specific information about the occupations that they would be interested in rather than randomly inputting keywords that would not fit them.

4. Summary and conclusions

In this project, we focused on creating a unique JOB Corpus using ONET data, also tried understanding the complex context in large Job Corpus and interpreting it as a 'Job zone'

meaning a category of occupations having similar levels of experience, education, and training.

And, this project could support current job search tools in ONET® Database, which primarily focuses on manual navigation and job title-based searches, often not adequately aligning with a job seeker's unique skills and career level. Particularly, the advanced Natural Language Processing (NLP) program is designed to overcome these limitations by offering a more dynamic, user-centric approach. Hence, modern NLP techniques that were used to streamline job searching, help individuals find positions that precisely match their skills and career aspirations. For example, this feature not only enhances job relevance but also personalizes the job-seeking experience. i.e, by helping them to gain the initiative to search specific information about the occupations that they would be interested in rather than randomly inputting keywords that would not fit them.

On the other hand, a limitation of this project is the data size was small and the context of the job corpus used for model development might be limited. Acctually, during the model development process, we recognized that overfitting occurred and tried to solve the problem through various methods. In particular, in the case of Naive Bayes, unlike other algorithms, the difference between train and test performance was relatively reduced, and the deviation (shade) of cross validation was also small, showing that overfitting was controlled.

5. Percentage of the code

***borrowing code = (140-80)/(140+360)*100 = 12%**

Work code List

Code 1: Create job_corpus Code 2: Classify Job_zone(Robera)

Code 3: Semantic Similariy Code 4: Utils

Calculation

	C1	C2	C3	C4	Sum
All lines from Internet	10	70	10	50	**140**
Modified	10	30	10	30	**80**
On my Own.	90	10	10	250	**360**

Reference

ChatGPT, personal communication, December 11, 2023

Cardiff NLP. (n.d.). Twitter RoBERTa Base Emotion Classification Model. Hugging Face. Retrieved from https://huggingface.co/cardiffnlp/twitter-roberta-base-emotion

Dixon, N., Goggins, M., Ho, E., Howison, M., Long, J., Northcott, E., Shen, K., & Yeats, C. (2023). Occupational models from 42 million unstructured job postings. Patterns, 4(7), Article 100757. https://doi.org/10.1016/j.patter.2023.100757

Hagan, M. T., Demuth, H. B., Beale, M. H., & Jesús Orlando De. (2016). *Neural network design.* s. n.

National Center for O*NET Development. *O*NET 28.0 Data Dictionary*. O*NET Resource Center. Retrieved December 11, 2023, from https://www.onetcenter.org/dictionary/28.0/text/

National Center for O*NET Development. *O*NET OnLine*. Retrieved December 11, 2023, from https://www.onetonline.org/

Reimers, N., & Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. arXiv preprint arXiv:1908.10084.

Truong, M. (2023, January 11). Quick Semantic Search using Siamese-BERT Networks Medium.https://towardsdatascience.com/quick-semantic-search-using-siamese-bert-networks-1052e7b4df1