

2023/2024 SEMESTER 2 – PROJECT

Course : Diploma in AI and Data Engineering Module : EGT216 - Natural Language Processing

**Instructions to students**:

* **Plagiarism**

Nanyang Polytechnic (NYP) takes plagiarism seriously and regards it as a form of academic dishonesty. Students who violate the NYP Academic Integrity Policy of NYP by committing plagiarism will be subject to disciplinary action.

Learners will face disciplinary action if they provide or receive unauthorized assistance, such as:

* Copying another learner’s answer
* Providing answers for other learners
* Falsifying data, information or quotations
* Asking another person to do the work you are supposed to do
* Taking and using ideas, words or work of others in whole or in part and passing it off as your own work without citing the original source (including AI resource)

Please refer to the NYP Academic Integrity Policy on the NYP website

Plagiarism is taking the work or ideas of another person, whether in whole or in part, and presenting it as one's own work without acknowledging the original source.

Sharing your work with other learners is considered plagiarism. ALL those involved, including those who share their work, will be penalised in accordance with the NYP Academic Integrity Policy.

Cite and reference ALL sources you have used in your report in APA style.

Page 1

* **Copyright Infringement**

It is equally important that you avoid copyright infringements and name the authors of the images you use. When searching for images, make sure that you have obtained the necessary permissions or licenses to use the images in your work. Using copyrighted images without proper permission is a direct violation of intellectual property rights. You must use copyright-free or Creative Commons licensed image databases, as this will allow you to obtain imagery in an ethical manner while adhering to the principles of academic integrity.

* **Late Submission**

For each working day of late submission, a penalty of 5% of the basic grade will be deducted from the grade. Late submission after 5 working days will be awarded 0 points.

1. **Process Flow Diagram - Resume Classification**

A black grid with white squares

Description automatically generated

1. **Process Flow Diagram - Resume Search and Summarization**

**A diagram of a computer

Description automatically generated**

1. **Pre-processing**

* **Clean Duplicates**

**A screen shot of a computer code

Description automatically generated**

From the data review, I observed that there were 9404 duplicate rows/resumes in the dataset. This accounts for 98.01% of the dataset. These duplicate rows do not provide additional information about the different categories and instead skews the representation of different categories in the dataset, particularly Java Developer category which has 825 duplicate rows. Hence, these duplicate rows need to be dropped.

* Clean Encoding

A screenshot of a computer program

Description automatically generated

From the data review, I observed that there were instances whereby non-english symbols were located in the resumes. After research, these were likely due to certain documents converting from a legacy encoding method to utf8. The conversion must have been incorrect or improperly established, causing these noise to appear in resumes. They then act as potential noise that might affect the performance of the BERT model, RoBerta or GPT2. Hence, we should recover them back to their original characters in order to recover any potential semantic meaning that may have been lost when their encoding was mixed up(mojibake).

* Clean Spaces

A computer screen with green text

Description automatically generated

This was to remove the spacing and linebreaks in the resume. This was needed because certain resumes have long lengths of spaces between words. This may be due to pdf file reading errors. The long space between the words causes the length of the resumes to be unnecessarily longer, which may contribute to longer computation time. Hence, removing the spaces may help reduce computation time. However, this operation is already conducted by various other processing such as removing stop words and stemming. It instead acts as a fallback to give an option to remove these spacings if the other processing methods are not used.

* Clean Spelling

A computer screen with text

Description automatically generated

A screenshot of a computer program

Description automatically generated

From the data review, I observed that there were minor spelling errors in certain resumes. Hence, spelling needs to be corrected in the resumes to normalize the text and ensure that they are represented correctly by the BERT model, as well as reduce vocab size. A small max dictionary edit distance was used to ensure that named entities such as Sci-kit does not get changed to science and only minor spelling errors will get corrected.

* Clean Contractions

A screenshot of a computer program

Description automatically generated

There may also exist contractions in the resumes. Contractions are just words that have been combined for spoken language. Hence, we may expand these contractions into individual words to normalize the contraction, reducing the vocab size, and ensuring that the models can represent them more accurately, making the model more stable.

* Clean Camel Case

A computer screen with text

Description automatically generated

From the data observation, I observed that sometimes the line breaks are lost when the resumes were being read. This results in the a capatalized word to directly follow the word from the previous line resulting in a camelCase word form. This causes the 2 words to form a completely different non-existent word, making the semantic meaning of the 2 words to be lost. Hence, we need to expand them out.

* Clean Punctuations

A computer screen shot of a computer code

Description automatically generated

Punctuations may add semantic meaning that can improve the model, but it can lead to increased complexity that the model will need to learn which may reduce the performance of the model. Larger models may be able to capture semantics of punctuations. Hence, the option to remove or keep punctuation will be mostly decided by how well the BERT model will perform in fine tuning.

* Clean Non-ASCII

A screen shot of a computer

Description automatically generated

Non-ASCII words may add to the complexity of the resumes as well or may contain information that is valuable for the Bert model to learn, such as positional information using point forms. Since the non-ASCII words may cause noise or poor representations of other words, these non-ASCII words may need to be removed. Furthermore, removing non-ascii words can also improve the normalization of the dataset. However, the choice will still mostly be decided by whether the BERT model performs well with it or not.

* Drop stop words

A screen shot of a computer

Description automatically generated

Stop words are words that are useful for language understanding, but do not carry much useful contextual information. Furthermore, stop words may cause the dataset vocab size to be much larger as there is a wide variety of stop words. Furthermore, stop words exist in text in large quantities, which may also negatively affect the performance of the resume search as it may fill skew the resume vector representations. Hence, stop words will likely need to be dropped, which will likely improve performance.

* Perform lower casing

A screenshot of a computer program

Description automatically generated

Lower casing can help normalize the text to a non capitalized form. This can help the normalize the resumes and reduce vocab size. Furthermore, with a lower case resume, BERT will be more stable during training as there will be fewer variants of the same word that it will need to train to learn the similarities between capitalized and non-capitalized words. Hence, lower casing may be needed to improve the performance.

* Perform stemming

A screen shot of a computer program

Description automatically generated

Stemming can normalize the words by cutting off letters at the end of words. However, this may lead to words that are not actual words and may lead to poorer text normalization as some words have more complex spellings. This may be able to perform well if the vocabulary of the resumes are simple, however that may not be the case with the long resumes which can be up to 1900 words long. Hence, stemming may not be useful. However, it will be tested in fine tuning first before it gets decided.

* Perform lemmatization

Lemmatization converts words to their base forms (lemmas). This normalization method is more valuable than stemming as the words formed are actual English words. Hence, lemmatization will improve text normalization, resulting in smaller vocab sizes, stabler training and improved representation of words as they are more prevalent in the dataset, giving the bert model more opportunities to learn the semantic meaning of these words.

1. **BERT Training and Validation**

A screen shot of a computer code

Description automatically generated

A screen shot of a computer screen

Description automatically generated

A screen shot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

I made the Bert model into a class for training and testing. I have set the default parameters to be:

Max\_len = 128

Validation\_size=0.2

Batch size = 16

Epochs = 50

Num labels = 25 (the number of unique classes)

Learning rate = 1.5e-5

Model\_type = bert-base-uncased

Bert config will be default

Weight\_decay will be 0.01 and 0.0 respectively for the 2 weight decays

Max\_len will be default 128. This was arbitrarily defined as the starting value for what the length of the length of the input ids should be as it gives room to increase or decrease in fine tuning.

Validation size will be 0.2 as it ensures that the validation dataset will have at least 1 example in each category while also ensuring that the training dataset has more data to train the Bert model.

Batch size was set to be 16 as it is large enough for the model to train on accurately and produce good reduction in training loss while also ensuring that the memory of the computer does not overload.

Epochs was set to 50 as I aim to observe the validation accuracy stabilise and observe the model overfit first, then by observing the stabilised accuracy of the model, we can get a better understanding of what parameters increase the maximum accuracy of the model, and which ones decrease it. This assumes that parameters that improve the performance of the model, increases the maximum stabilised accuracy of the BERT model.

Learning rate was set to 1.5e-5 as I determined that to be fast enough for convergence, while also not being too large to cause the model to converge too early

The initial model used will be bert-base-uncased as I believe bert-large will overfit to the data very easily due to the small size of the dataset after dropping duplicates.

The bert configuration will be default and will be adjusted during fine tuning.

The weight decay will be used to reduce overfitting when needed. The default values are defined arbitrarily.

These parameters will be used in my control experiment when running my first BERT model before additional fine tuning.

In the testing stage, I added an accuracy metric to each batch to observe both MCC and accuracy during the testing of the BERT model

1. **Resume Classification**

A screen shot of a computer

Description automatically generatedA screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

I observed that the most optimal method of training bert was using the parameters shown in the picture (please also refer to the code I used).

Clean Encoding: I observed that clean encoding helped improve the model because the semantic meaning of the words with broken encodings were recovered. This may have helped provide BERT with additional formatting and positional information between words as points forms are fixed for example. The clean encoding function also fixed broken encoding involving certain letters such as ï. This may help normalize words as now the broken encoding is fixed, resulting in proper words being formed. As a result, the model was trained more stable, words are represented more effectively, and the model was able to capture more dependencies between words and characters or symbols. Hence, I believe that was why clean encoding improved the accuracy of Bert

Clean spaces: I believe that clean spaces reduced the performance as the spacing in resumes provided semantic meaning, which now the BERT model was unable to capture. Hence, the performance was reduced

Clean Spelling; I observed this to improve the accuracy of BERT. This was likely since the spell checker was only allowed to correct words if only 2 modifications were needed. This ensured that the resumes had corrected spelling, which normalized the words, providing a greater representation of words in the resume. But, it also ensured that larger corrections were not made, ensuring that named entities such as sci-kit do not get lost as much as compared to other spelling models used such as peter nordvigs method and pyspellchecker. This allows for the normalization of text without the lost of named entities.

Clean Contractions: I observed this to have greatly improved the accuracy of BERT as compared to other preprocessing methods. I believe this method improved the model because it was able to expand contractions such as “it’s” to “it is”, which helped normalize the text. But in addition, it helped remove stop words as now the expanded word can now be identified removed as a stop word by the stop word process, allowing them to synergise well.

Clean Camel Case: I observed this to have decreased the accuracy of BERT substantially. This is likely because it also cleans out the spaces and the line breaks, resulting in the loss of the formatting information that may be crucial for the BERT model to learn. It may also be because the regular expression had unintended effects that caused the formatting or the words in the resumes to be wrongly formatted, such as making words that were supposed to be in camel case style separated. This may have caused the vocab size to increase and the text to diversify as the resulting words formed may not be actual words. Hence, it likely caused the performance of the model to decrease.

Clean Punctuation: I observed this to have increased the accuracy of BERT. This was likely because the punctuation added complexity to the resumes which bert-base was unable to capture effectively. By removing it, BERT was likely able to focus more on the words in the resumes, allowing BERT to perform better as it was able to capture the contextual information of the resumes more, instead of the punctuations.

Clean Non-ASCII: This reduced the accuracy of the BERT Model. This was likely because the non ascii words or characters held important information whether semantically or in terms of formatting and positioning. By removing them, this information was lost, which resulted in poorer performing BERT.

Lower Casing: This improved the model as the bert-base-uncased was already trained on lower case words. Hence, by normalizing the words to lower case, the model was able to capture the semantic meaning of these words more easily. In addition, the text will be normalized, resulting in smaller vocab size, stable training and better representation of words

Dropping stop words: This improved the model as it removed words which did not add semantic meaning to the resumes. This allowed the model to focus on words which matter more in the resumes, such as topics, skills etc. This allowed the BERT model to capture more semantic meaning and contextual dependencies in the resumes, which improves the performance of the bert model

Stemming: This reduced the accuracy of the model as it likely diversified the vocab size instead of normalizing it. By cutting of the ends of words, the resulting words do not actually exist , resulting in diversity in the vocabulary. This is especially the case because the resumes in the dataset are verbose, with resumes that have lengths of about 1900 words. Hence, stemming likely resulted in poorer text normalization and poorer representation of words. Hence, stemming reduced the accuracy of the model

Lemmatization. This normalized the text in the resumes by turning words into their base forms (lemmas) resulting in normalized text and allows the model to perform better on a smaller vocab size and with greater representations of words.

Bert config vocab size: Setting this value to be 4000 helped bert perform better on the dataset. This was likely because it helped reduce the vocab size of bert to only the top 4000 most frequent words in the resumes dataset. This likely resulted in the gibberish in the resumes, due to errors in the decryption and sending of data, to be OOV. This is useful as the BERT will then be able to focus on the semantic meaning and the contextual dependencies of the most important and prevalent words in the resumes dataset. Hence, reducing the vocab size improved the performance.

Bert config attention heads: reducing the attention heads to 8 improved the accuracy of the model. This was likely due to the fact that the fewer attention heads helped prevent the bert model from overfitting too much. It may also be because the intricate details in the resume are not as important in identifying the category of the resume, and capturing the general details of the resume is more important instead. Hence, fewer attention head (8) is more optimal.

Bert model type: bert-base-uncased seems to perform the best as compared to bert-base-cased, bert—large-uncased and bert-large-cased. This was likely because the capitalizations of the words in the resumes are not as important in distinguishing the category of different resumes in the resumes dataset. The bert-large, with its larger number of parameters, is likely overfitting to the small dataset of the training dataset. Hence, bert-large was unable to train and perform well as compared to bert-base.

1. **Vectorization**

* TF-IDF

A screen shot of a computer code

Description automatically generated

A screenshot of a computer

Description automatically generated

TF-IDF Document vectorization was used as there could be a relationship between the frequencies of the words as well as the use of rare words in the resume. By using tf-idf document vectors, I will extract the top 10 most similar resumes based on this relationship.

* Word2Vec (GoogleNews-vectors-negative300.bin.gz)

A screen shot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Using the word2vec model I downloaded, I will make embedding vectors for each word in each resume, then aggregate them to form a document/resume vector for each resume. This is based on the idea that similar aggregated word vectors are similar documents/resumes. Hence, I will explore this method using word2vec.

* BERT document vectors

A computer screen shot of a program

Description automatically generated

Similar to word2vec, I will make embedding vectors for each word in each resume, then aggregate them to form a document/resume vector for each resume. This is based on the idea that similar aggregated word vectors are similar documents/resumes. Hence, I will explore this method using bert. This should theoretically perform better than word2vec due to the larger dimension of the document/resume vectors produced by bert (768) which is higher than word2vec’s (300).

1. **Resume Search**

Please refer to my resume to know what are the contents of my\_resume\_text

* TF-IDF

A screen shot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

A graph of a number of blue squares

Description automatically generated with medium confidence

My resume is for the job role of Product Manager (AI and Data). The job focuses on using AI and Data in the job to improve business opertaions and improve customer relationships and upsell opportunities, which was emphasised in the resume. Hence, it makes sense that the majority(5) of the top 10 most similar resumes is in data science category. ETL developer makes sense as the resume does talk about knowing how to sift through and use data advantageously for insights. However, hadoop and web designing may not be as relevant. Furthermore, the similarity scores of my resume to other resumes are very low, at about 0.25.

TF-IDF may not be as useful. The resume similarity between the resumes and my resume are so low that it becomes difficult to differentiate similar documents and dissimilar documents/resumes. Hence TF-IDF may not be very useful

* Word2Vec

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A graph of a number of people

Description automatically generated

Data science is the main category(3) as my resume emphasised a lot on my experience in getting advantageous data insights.

Business Analyst is also valid because I did emphasis my ability to collect insightful data for businesses

DevOps Engineer and database is also valid because the resume did emphasise on abilities such as cloud, databases and managed service providers which devops and database engineers usually have experience in.

Web Designing is anomalous as the resume did not emphasise on my ability to design website. This is not valid

Hadoop and blockchain is not very accurate as I have not had any experience and did not talk about any of those in the resume.

ETL Developer is valid as i did emphasise on extracting information and insights from data in the resume.

Word2Vec was able to extract documents which have high ranges of similarity score, allowing it to more easily differentiate between dissimilar and similar documents. This is more useful than TF-IDF. However, the top 10 resumes come from diverse categories, some of which are not valid to my resume type. This suggests that the word2vec document vectors may not be as reliable at finding similar documents of similar classes.

* Bert Document/Resume Vectors

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A graph of a number of blue columns

Description automatically generated

Web Designing, Blockchain and Java Developer are not valid as my resume does not emphasise on these areas.

Data science is valid as my resume emphasised programming capabilities

Automation testing is valid as my resume did include IP configuration abilities and world skills mechatronics experience.

SAP Developer is valid as my resume did emphasise on creating applications with my programming abilities

Business analyst is valid because my resume emphasised on my ability to create business value using data analysis.

The BERT resume vectors can extract documents which have high ranges of similarity score, allowing it to more easily differentiate between dissimilar and similar documents, similar to word2vec. However, the majority (5) of the top 10 most similar resumes should not be similar to my resume, meaning that the bert document vectors are not as reliable at finding similar documents of similar classes.

1. **Resume Summarization**

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

To ensure that cohere is able to recognise its purpose at the start, I started the prompt with:

“You are tasked to summarize a given resume in order to identify key attributes of the potential employee to give the company a concise and unbiased overview of the key attributes of a potential employee”

This was aimed to ensure that cohere understands its purpose

I then informed cohere of the inputs to expect, the processing it needs to conduct, as well as the output desired

Input:

“You will be provided a resume text, as well as named entities and phrases as input.”

Processing:

“You are to analyze the resume text to identify information such as education, skills, langauges, achievements, work experience and objectives. You are to then condense the information into a clear and concise summary for the company to read and have an overview of the potential of the person as an employee of the company”

Output:

“The output will be the summarized version of the resume”

I also gave cohere an example of a small portion of a resume that it might need to summarise. I also gave a target output by providing it information of what I expect from the summary.

One-shot learning:

“For example, a small portion of the resume may contain information such as: Contributed to the development of Large Language Models in AI Singapore, which increased performance by 50%.”

“The sample output should include information such as: He has substantially improved performance by 50% in AI Singapore with his contributions in enterprise-level AI initiatives in AI Singapore.”

However, cohere believed my experiences with modules in NYP were actual companies. Hence, I appended additional information to ensure it does not make the same mistake.

“Take note that experiences may include modules or courses, and not necessarily positions at companies"

To improve coheres ability to fully capture the important information from the resume, I gave it additional information regarding the phrase chunks and named entities in my resume, in order for it to fully capture those important information.

important\_details:

"These are some key named entities, you should consider in your text summarisation to identify unique educational institutes, companies or locations: {ner\_categories}\nThese are some key phrases you should consider in order to understand what the person did, and what were his key abilities: {phrases}"

1. Optimization

* BERT Fine-Tuning

**A screenshot of a computer

Description automatically generated**

This is the optimized code for the BERT-Fine-Tuning (but not the best one).

The clean contractions function was disabled in order to reduce the processing time of the code by about 90s. This was because clean contractions was the longest function to run, taking about 1.5minutes to complete. Hence, it was disabled.

The epochs used to train BERT was changed from 50 to 35. This was to reduce the overfitting that may occur, as observed in the previous iterations of BERT fine tuning.

The weight decay 1 parameter for one of the weight decay parameters for adamW was increased from 0.01 to 0.02 as a measure to reduce the chances of overfitting.

* Roberta

**A screen shot of a computer program

Description automatically generated**

A screen shot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

I will try to optimize the resume classification solution using Roberta. Roberta was trained on a larger, more diverse corpus of data, so it should be able capture a larger spectrum of language nuance. It also trains with dynamic masking, so it will be more adaptable to different contexts because of varied word mask positions. Roberta also was trained with longer sequences, making it more capable of getting contextual dependencies in the data. Lastly, it has a streamlined training process due to it not having Next Sentence Prediction, which makes it more optimized for learning contextual information from individual sentences. Because of these reasons, I tried to optimize the resume classification solution by using and fine tuning Roberta.

* GPT2

A screen shot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

GPT2 has 117M parameters whereas BERT Base has 110M parameters. GPT2 may be able to perform better than bert base, while also not overfit like BERT Large. I will use it to try and observe if it can perform better than, or faster than bert base. From the observed accuracy and time, GPT2 did not surpass bert base despite having more parameters. GPT2 had a lower testing MCC and accuracy of 0.246 and 0.268 respectively. Bert base had a higher testing MCC and accuracy of 0.652 and 0.675 respectively. However, GPT2 did finish training at a faster time(340s) than bert base(652s) with similar configurations.

* Resume Search Preprocessing

A screenshot of a computer

Description automatically generated

Disabled clean contractions to reduce processing time by about 90s

Included the use of clean encoding and clean spelling to normalize the text further before actually vectorizing them into document vectors.