Phase 2 VDCNN Multilabel Model Training Guideline

ST1506: DSDA Final Year Project

Company Profiling Text Mining

August 2021

SINGAPORE POLYTECHNIC SP

Synopsis

The purpose of this documentation is to demonstrate a detailed breakdown on the content of the ipython file for model training, so that users will understand the rationale behind the content.

This documentation is for data scientists who want to see the usage of VDCNN-Multilabel in this specific context and possibly improve from the current work.

Table of Contents

Breakdown Of Notebook Contents

Hardware Rendering

1. When the user chooses to run the code using GPU in Google Colab, the user can run the notebook as per usual so long as the runtime has been set to GPU in Google Colab.

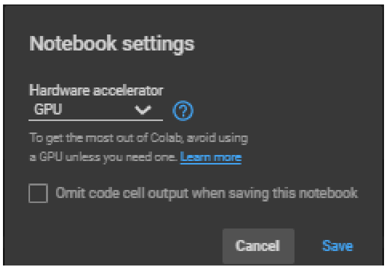


Figure 1: An example of runtime being set to GPU in Google Colab

Loading necessary libraries

1. In Section 1.2. of the notebook, we will install the libraries needed for the project. The libraries installed are shown below.

```
# install necessary libraries that might not be found
!pip install -U spacy
!python -m spacy validate
!pip install -U pip setuptools wheel
!pip install -U spacy[cuda110,transformers,lookups]
!python -m spacy download en_core_web_lg
!pip install tensorflow_addons

# check versions of libraries we are going to use
%tensorflow_version 2.x
import os
import tensorflow
import tensorflow_addons as tfa
import sklearn
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib
import spacy
import platform

message="        Versions        "
print("*"*len(message))
print(message)
print("*"*len(message))
print("Tensorflow version={}".format(tensorflow.__version__))
print("Tensorflow_addons version={}".format(tfa.__version__))
print("Keras version={}".format(tensorflow.keras.__version__)
print("Sklearn version={}".format(sklearn.__version__))
print("Numpy version={}".format(np.__version__))
print("Pandas version={}".format(pd.__version__))
print("Seaborn version={}".format(sns.__version__))
print("Matplotlib version={}".format(matplotlib.__version__))
print("SpaCy version={}".format(spacy.__version__))
print("Python version={}".format(platform.python_version()))
```

```
************************
        Versions
************************
Tensorflow version=2.5.0
Tensorflow_addons version=0.13.0
Keras version=2.5.0
Sklearn version=0.22.2.post1
Numpy version=1.19.5
Pandas version=1.1.5
Seaborn version=0.11.1
Matplotlib version=3.2.2
SpaCy version=3.1.1
Python version=3.7.11
```

Figure 2: Python Libraries Installed and the version output of imported libraries

2. In Section 1.3, we redefine all the libraries to be used for this notebook, and also add in the *'prefer_gpu' parameter for spacy to tell spacy to use gpu.*

1.3. Load the modules

Load the necessary modules for usage in the entire project.

```python
# importing necessary modules for this project
import tensorflow as tf

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import string
import spacy

# activate the GPU to run spaCy with GPU
spacy.prefer_gpu()

%matplotlib inline
```

Figure 3: Loading the needed libraries once again and set the 'prefer_gpu' parameter

3. In Section 1.4., we will load the training dataset as Pandas Dataframes. The training dataset will be used as the training dataset for the model.

```python
# use pandas to read the excel file and populate it in a pandas dataframe
companies = pd.read_excel('./clean_dataset.xlsx')

# see the top 10 companies that are populated in the dataframe
companies.head(10)
```

	Company_ID	Company	Country	PIC	Sector	Subsector	Archetype	Valuechain	Websites
0	4137190062363536	AKSORN CHAROEN TAT ACT. CO.,LTD.	THAILAND	NaN	TMT	media	media_aggregator/distributor	Midstream	https://getlinks.co/6831
1	23248790229909752	DONGGUAN SHENGYA CLEANING APPLIAME CO.LTD	CHINA	NaN	TMT	consumer electronics	consumer electronics_distributor	Downstream	https://baike.baidu.com/item/%E4
2	28486505934571008	EXIS TECH SDN. BHD.	MALAYSIA	NaN	oos	others	others	NaN	http://www.exis-tech.com/
3	38251695094669872	BEI JING ESTRABA IMPORT AND EXPORT	CHINA	NaN	NaN	NaN	NaN	NaN	NaN
4	39910921263510784	Aztech Electronics Pte Ltd	SINGAPORE	NaN	TMT	consumer electronics	consumer electronics_distributor	Downstream	https://www.aztech.com/business/

Figure 4: Loading the excel file and preview of the training dataset

Exploratory Data Analysis

1. First, we will identify the various columns in the training dataset to get a better understanding of the dataset. As shown below are the columns identified in the Dataframe.

```
[ ]   # see the row headers of the entire pandas dataframe first
      list(companies.columns)
```

```
['Company_ID',
 'Company',
 'Country',
 'PIC',
 'Sector',
 'Subsector',
 'Archetype',
 'Valuechain',
 'Websites',
 'Company Profile Information',
 'Remarks']
```

Figure 5: List of columns in the training dataset

2. Here, we identify the number of records, as well as the number of unique labels.

```
# get the total number of records in the dataframe
df_count = df_train['Company_ID'].count()

# get count of unique contries where companies are based in
df_countCountry = df_train['Country'].nunique()

# get count of total unique sectors where companies are from
df_countSector = df_train['Sector'].nunique()

# get count of total unique subseector where companies are from
df_countsubSector = df_train['Subsector'].nunique()

# get count of total unique valuechain where companies are from
df_countValuechain = df_train['Valuechain'].nunique()

# get count of total unique archetypes
df_countArchetype = df_train['Archetype'].nunique()

print('Total number of records:', df_count)
print('Total number of countries:', df_countCountry)
print('Total number of sectors:', df_countSector)
print('Total number of subsectors:', df_countsubSector)
print('Total number of valuechain:', df_countValuechain)
print('Total number of archetypes:', df_countArchetype)
```

```
Total number of records: 9600
Total number of countries: 14
Total number of sectors: 16
Total number of subsectors: 37
Total number of valuechain: 18
Total number of archetypes: 94
```

Figure 6: Number of records, and labels in interest

3. We proceed to identify the unique Archetype labels present in the dataset.

```python
print('List of unique archetype:\n{}'.format(df_archetype))
```

```
List of unique archetype:
others                                  2161
building_material_manufacturer           573
buildings & industrial_contractor        496
consumer discretionary distributor       358
cni_service providers                    279
                                        ...
MIDSTREAM                                  2
tisp - tower                               2
tisp - fiber cable                         2
building material_manufacturer             1
metals and mining                          1
Name: Archetype, Length: 94, dtype: int64
```

Figure 7: Unique archetype breakdown

4. Here, we drop the unnecessary columns from the dataset, so that they do not interfere with the EDA process.

```python
# declare the list of the row names that are redundant
rows_to_drop = ['Company_ID', 'PIC', 'Websites', 'Remarks']

# use a conditional expression to filter out those rows
df_filteredCompanies = df_train.drop(labels=rows_to_drop, axis=1)

df_filteredCompanies
```

	Company	Country	Sector	Subsector	Archetype	Valuechain	Company Profile Information
0	AKSORN CHAROEN TAT ACT. CO.,LTD.	THAILAND	TMT	media	media_aggregator/distributor	Midstream	For over 80 years of experience in creating an...
1	DONGGUAN SHENGYA CLEANING APPLIAME CO.LTD	CHINA	TMT	consumer electronics	consumer electronics_distributor	Downstream	Yatai's main products cover various cleaning m...
2	EXIS TECH SDN. BHD.	MALAYSIA	oos	others	others	NaN	In the beginning, it started off by providing ...
3	BEI JING ESTRABA IMPORT AND EXPORT	CHINA	NaN	NaN	NaN	NaN	NaN
4	Aztech Electronics Pte Ltd	SINGAPORE	TMT	consumer electronics	consumer electronics_distributor	Downstream	Being a turnkey, one-stop integrated solutions...
...	...	...	...	...	...	...	...
9595	SOONBEE INVESTMENT HOLDINGS PTE. LT D	SINGAPORE	OOS	others	others	NaN	SOONBEE INVESTMENT HOLDINGS is an ACRA-registe...
9596	SICHUAN CHINA RAILWAY WENRUI REAL E	CHINA	NaN	NaN	NaN	NaN	NaN
9597	GROUP INDUSTRIES PTE LTD	SINGAPORE	CNI	building_material	building_material_manufacturer	MIDSTREAM	Group Industries Pte Ltd was founded in 1986. ...
9598	SIAM NISSAN EASTERN CO.,LTD.	THAILAND	IND	auto & mec	auto dealer	Trader	Auto Dealership - Retail
9599	FAT INVESTMENT HONG KONG LIMITED	HONG KONG	NaN	NaN	NaN	NaN	NaN

Figure 8: New training dataset after dropping columns

5. Records with all NaN values are extracted, shown, and discarded, as they cannot be used in the training process.

```python
# find all the rows with nan data in sector, subsector, archetype and valuechain
cols_to_check = ['Sector', 'Subsector', 'Archetype', 'Valuechain', 'Company Profile Information']
empty = df_filteredCompanies[df_filteredCompanies[cols_to_check].isnull().all(1)]

empty
```

	Company	Country	Sector	Subsector	Archetype	Valuechain	Company Profile Information
3	BEI JING ESTRABA IMPORT AND EXPORT	CHINA	NaN	NaN	NaN	NaN	NaN
12	CHANGTU COUNTRY LONGXING FERTILIZER CO.,LTD	CHINA	NaN	NaN	NaN	NaN	NaN
28	ZIBOBOSHANHONGLIWEI MOTOR CO.,LTD	CHINA	NaN	NaN	NaN	NaN	NaN
48	BEIJING DUO MEIDUO SHIYOU PRODUCTS SALES CO., ...	CHINA	NaN	NaN	NaN	NaN	NaN
63	TRUSVEST SDN. BHD.	MALAYSIA	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...
9576	REAL CHARM INVESTMENT LIMITED	HONG KONG	NaN	NaN	NaN	NaN	NaN
9585	JIJIN INTERNATIONAL TRADE (SHANGHAI)	CHINA	NaN	NaN	NaN	NaN	NaN
9592	QITAI TIANSHAN CEMENT CO., LTD	CHINA	NaN	NaN	NaN	NaN	NaN
9596	SICHUAN CHINA RAILWAY WENRUI REAL E	CHINA	NaN	NaN	NaN	NaN	NaN
9599	FAT INVESTMENT HONG KONG LIMITED	HONG KONG	NaN	NaN	NaN	NaN	NaN

1149 rows × 7 columns

Figure 9: Records with NaN values

6. The records with valid data are then kept.

```python
# now we get the dataset that are valid
df_valid = pd.concat([df_filteredCompanies, empty, empty]).drop_duplicates(keep=False)

df_valid
```

	Company	Country	Sector	Subsector	Archetype	Valuechain	Company Profile Information
0	AKSORN CHAROEN TAT ACT. CO.,LTD.	THAILAND	TMT	media	media_aggregator/distributor	Midstream	For over 80 years of experience in creating an...
1	DONGGUAN SHENGYA CLEANING APPLIAME CO.LTD	CHINA	TMT	consumer electronics	consumer electronics_distributor	Downstream	Yatai's main products cover various cleaning m...
2	EXIS TECH SDN. BHD.	MALAYSIA	oos	others	others	NaN	In the beginning, it started off by providing ...
4	Aztech Electronics Pte Ltd	SINGAPORE	TMT	consumer electronics	consumer electronics_distributor	Downstream	Being a turnkey, one-stop integrated solutions...
5	TONGDUN INTERNATIONAL PTE LTD	SINGAPORE	tmt	it_services	it_services	midstream	Tongdun Technology is a professional third-par...
...	...	...	...	...	...	...	...
9593	BANGKOK PATANA SCHOOL	THAILAND	CG	retail n distribution	consumer discretionary distributor	Downstream	Bangkok Patana School is Thailand's original B...
9594	INDRATI LINES PTE LTD	SINGAPORE	OOS	others	others	NaN	Indra is one of the leading global technology ...
9595	SOONBEE INVESTMENT HOLDINGS PTE. LT D	SINGAPORE	OOS	others	others	NaN	SOONBEE INVESTMENT HOLDINGS is an ACRA-registe...
9597	GROUP INDUSTRIES PTE LTD	SINGAPORE	CNI	building_material	building_material_manufacturer	MIDSTREAM	Group Industries Pte Ltd was founded in 1986. ...
9598	SIAM NISSAN EASTERN CO.,LTD.	THAILAND	IND	auto & mec	auto dealer	Trader	Auto Dealership - Retail

8447 rows × 7 columns

Figure 10: Dataframe with the valid results

7. Here, we will turn the numbers seen in the EDA into graphs, so that it is easier to spot the trend. Shown below is a bar chart of the breakdown of labels in Sector. It seems that OOS, CNI and REH make up the majority of the dataset.

   We also plotted a pie chart of the countries found in the dataset. We can see that most of the portions comes from Thai companies, followed by China and Malaysia.

```python
sns.set_style('darkgrid')
plt_dims = (30, 13)
fig, ax = plt.subplots(figsize=plt_dims)

# plot a barplot to see number of companies that belongs to specific sectors
sns.countplot(x="Sector", data=df_valid, ax=ax)
plt.show()
```

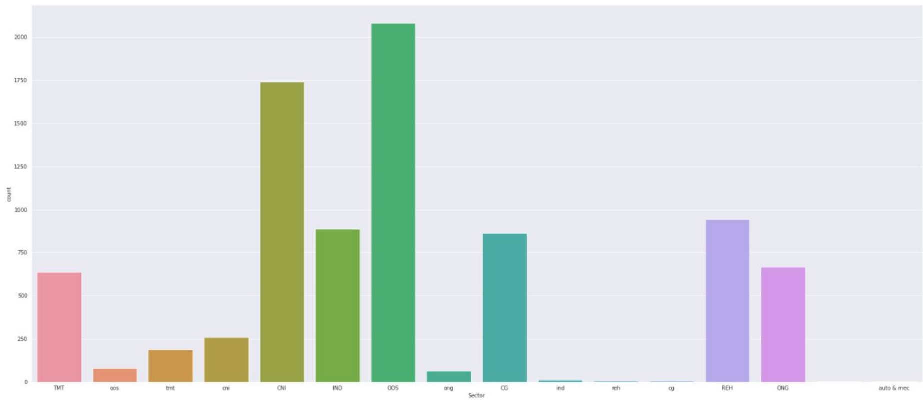


Figure 11: Bar Chart of the various Sector counts

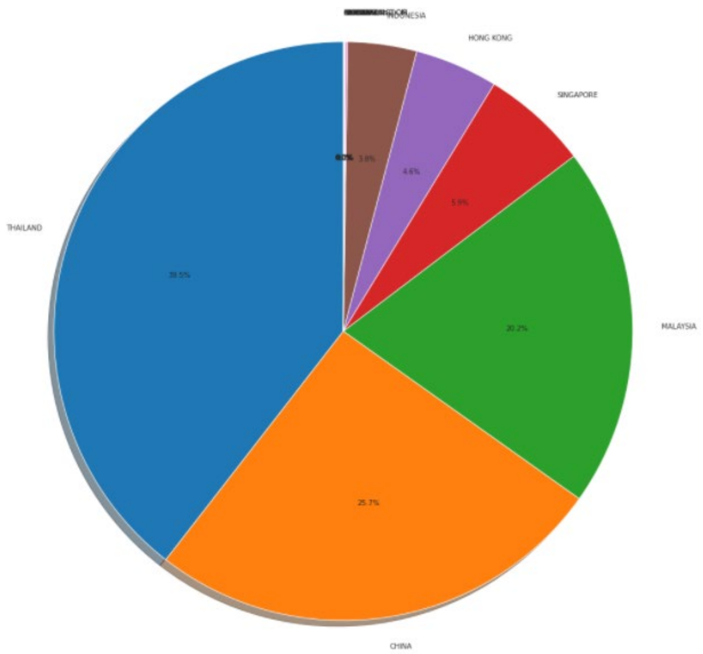


Figure 12: Pie chart breakdown of the countries the companies originate from

8. We would like to see examples of how the worded company profile description looks like. Therefore, we printed the 1st 10 companies text description for analysis.

```python
# configurate pandas dataframe to let us see the entire company description IN FULL
pd.set_option('display.max_colwidth', None)

# get the 1st 10 results and observe
df_valid.loc[0:10,'Company Profile Information']
```

```
0
For over 80 years of experience in creating and developing high-quality learning materials has enabled us to provide world-class educatio
nal innovation to meet the needs of all teachers, students, institutions and educational authorities.
1
Yatai's main products cover various cleaning machinery, cleaning agents, cleaning tools, stone maintenance and other cleaning products; c
leaning solutions and services include product technical consultation, product customization, employee training, maintenance and so on.\n
2
In the beginning, it started off by providing technical support for test handlers, then moving into module design and production. Its fir
st, in-house designed, full-fledged handler was introduced in 2008.\nSince then, the company has designed and produced a wide range of tu
rret and pick-and-place solutions for its customers all over the world.
4                                                    Being a turnkey, one-stop integrated solutions provider based
in Singapore, Aztech is equipped with state-of-the-art equipment, R&D, design, manufacturing and packaging capabilities to deliver a seam
less, unified experience. Each and every time. Always striving towards the edge of technology for more than 34 years, we have been buildi
ng capabilities to serve clients' manufacturing needs, including the consumer electronics, telecommunications, healthtech, LED lighting,
automotive and technology start-up market segments.
5     Tongdun Technology is a professional third-party intelligent risk management and decision-making service provider headquartered in
Hangzhou, Zhejiang. By integrating artificial intelligence into business scenarios, Tongdun Technology offers solutions in intelligent us
er analysis, intelligent risk management, intelligent antifraud and intelligent operation to clients from various industries including fi
nancial industry, internet business, logistics, healthcare, retail, smart cities and government bodies. Over 10,000 corporate clients hav
e chosen Tongdun Technology's products and services
6
For over 30 years, EMASTIGA has been servicing the construction and the oil and gas industries with distinction.We try our hardest to cre
ate win-win situations and value-add to our clients with every transaction.
7
Sales of hydraulic equipment and parts
8
For Excelkos, it all started with a far-sighted vision to deliver quality industrial products and solutions to meet the increasing demand
s of various industries.  The vision became a reality in 1988 when the Excelkos Group was established to distribute and provide quality r
ubber industrial chemicals.
9
Kum Eng Huat is the authorised dealer for Osram and Philips, as well as a trusted supplier of lighting solutions for major projects with
the local government.
10
Wholesale of other building_materials, not elsewhere classified
Name: Company Profile Information, dtype: object
```

Figure 13: First 10 company text descriptions

Data Pre-processing

1. To ensure that there are no escape characters such as ('\n'), we will first replace those with a space (' ') to ensure that it does not interfere with the model learning process later on.

   We also used a distribution plot to see the most frequent word length among the entire dataset. From the frequency distribution plot, we can assume that most company descriptions have around 700 to 800 word descriptions.

```
# get rid of the \n found in the respective descriptions
df_valid = df_valid.replace('\n',' ', regex=True)
```

```
# now plot a distribution plot to see the word length distribution
sns.distplot(df_valid["length"], kde=True)
plt.title('Word Length Distribution')
plt.show()
```

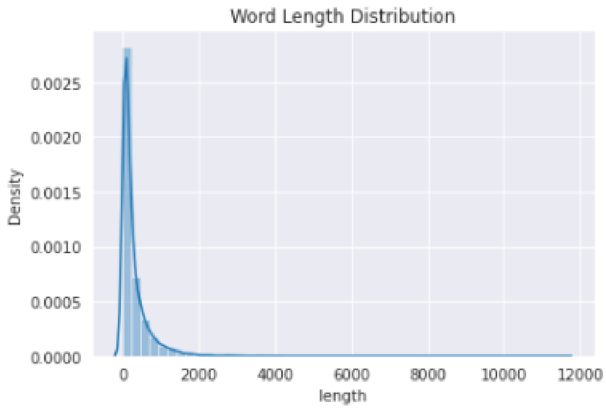


Figure 14: Removing special characters and getting the word length distribution

2. We then proceed on to populate NaN cells with spaces to proceed on with text tokenization later on.

```
# fill na with space instead of others
df_valid.fillna(" " ,inplace=True)
df_valid
```

	Company	Country	Sector	Subsector	Archetype	Valuechain	Company Profile Information	length
0	AKSORN CHAROEN TAT ACT. CO.,LTD.	THAILAND	TMT	media	media_aggregator/distributor	Midstream	For over 80 years of experience in creating and developing high-quality learning materials has enabled us to provide world-class educational innovation to meet the needs of all teachers, students, institutions and educational authorities.	238
1	DONGGUAN SHENGYA CLEANING APPLIAME CO.LTD	CHINA	TMT	consumer electronics	consumer electronics_distributor	Downstream	Yatai's main products cover various cleaning machinery, cleaning agents, cleaning tools, stone maintenance and other cleaning products; cleaning solutions and services include product technical consultation, product customization, employee training, maintenance and so on.	273
2	EXIS TECH SDN. BHD.	MALAYSIA	oos	others	others		In the beginning, it started off by providing technical support for test handlers, then moving into module design and production. Its first, in-house designed, full-fledged handler was introduced in 2008. Since then, the company has designed and produced a wide range of turret and pick-and-place solutions for its customers all over the world.	344

Figure 15: A snippet off populating NaN cells to space

3. Section 3.4. of the notebook retrieves the 4 labels, namely sector, subsector, archetype and valuechain, and assign unique labels to the respective company description. These unique labels will be used later as the training labels when training the model later.

```
# Programmatically assign tags to each definition
sector_keywords = pd.read_excel('./sector_master_definition.xlsx')
df_keywords = sector_keywords[['Sector', 'Subsector', 'Archetype', 'Value Chain', 'Sector Keywords']]

# capitalise all tags
df_keywords['Value Chain'] = df_keywords['Value Chain'].str.upper()
df_keywords.fillna(' ', inplace=True)
df_keywords['Sector Keywords'] = df_keywords['Sector Keywords'].str.upper()
df_keywords['Sector Keywords'].replace(' ', '[]', inplace=True)

# save unique tags, sorted for consistency across runs
sector = np.sort(df_keywords['Sector'].unique())
subsector = np.sort(df_keywords['Subsector'].unique())
archetype = np.sort(df_keywords['Archetype'].unique())
valuechain = np.sort(df_keywords['Value Chain'].unique())
print(len(sector), len(subsector), len(archetype), len(valuechain))
tag_counts = [len(sector), len(subsector), len(archetype), len(valuechain)]

# assign number tag list to each row
taglist = []
for index, row in df_keywords.iterrows():
    temp = []

    temp.append(np.where(sector == row['Sector'])[0][0])
    temp.append(np.where(subsector == row['Subsector'])[0][0])
    temp.append(np.where(archetype == row['Archetype'])[0][0])
    temp.append(np.where(valuechain == row['Value Chain'])[0][0])

    taglist.append(temp)

# assign completed taglist to column in dataframe
df_keywords['list_tag'] = taglist
```

Figure 16: Assigning tags to each company description

4. We redid data pre-processing for the training dataset for homogeneity. We also combined all keywords from all Sectors, removed extraneous keywords and sort them in a list. This will later be used for spaCy tokenization pipeline.

```
# process data for homogenity
df_valid['Valuechain'] = df_valid['Valuechain'].str.split().str.join(' ')
df_valid['Valuechain'] = df_valid['Valuechain'].str.upper()
df_valid['Sector'] = df_valid['Sector'].str.upper()
df_valid['Valuechain'].replace('', ' ', inplace=True)

taglist_df = []
# process tags for records
for index, row in df_valid.iterrows():
    temp = []

    try: # for error handling
        temp.append(np.where(sector == row['Sector'])[0][0])
        temp.append(np.where(subsector == row['Subsector'])[0][0])
        temp.append(np.where(archetype == row['Archetype'])[0][0])
        temp.append(np.where(valuechain == row['Valuechain'])[0][0])

        taglist_df.append(temp)
    except:
        # drop data if not valid
        df_valid.drop(index, inplace=True)


df_valid['list_tag'] = taglist_df

df_valid.shape
```

```
(8423, 9)
```

```
# combine all keywords from all sectors
keywords_masterlist = []
for index, row in df_keywords.iterrows():
    keywords_masterlist += eval(row['Sector Keywords'])

# remove extraenous keywords, then sort
keywords_masterlist = sorted(list(set(keywords_masterlist)))
print(len(keywords_masterlist))
```

```
1481
```

Figure 17: Pre-Processing df_valid

5. Similar to point 4 and the previous data pre-processing methods, we also did a similar data pre-processing to the test dataset under Section 3.6. in the notebook.

```
# import validation dataset
df_test = pd.read_excel('./val_dataset.xlsx')
df_test.replace('NAN', np.NaN, inplace=True)

# drop unnecessary columns
df_test.drop(rows_to_drop, axis=1, inplace=True)

# replace newline characters in validation data
df_test = df_test = df_test.replace('\n', ' ', regex=True)

# fill in NAN values in validation data
df_test.fillna(' ', inplace=True)

# change dtype of validation data columns
for i in columns_to_convert:
    df_valid[i] = df_valid[i].astype(str)

# process validation dataset
df_test['Valuechain'] = df_test['Valuechain'].str.split().str.join('
df_test['Valuechain'] = df_test['Valuechain'].str.upper()
df_test['Sector'] = df_test['Sector'].str.upper()
df_test['Valuechain'].replace('', ' ', inplace=True)

# add tags to validation data
taglist_df = []
# process tags for records
for index, row in df_test.iterrows():
    temp = []

    try: # for error handling
        temp.append(np.where(sector == row['Sector'])[0][0])
        temp.append(np.where(subsector == row['Subsector'])[0][0])
        temp.append(np.where(archetype == row['Archetype'])[0][0])
        temp.append(np.where(valuechain == row['Valuechain'])[0][0])

        taglist_df.append(temp)
    except Exception as e:
        # drop data if not valid
        print(row.name, e, '\n')
        df_test.drop(index, inplace=True)

df_test['list_tag'] = taglist_df
```

Figure 18: Data pre-processing for the test dataset

6. We went on to ensure that the dtypes for the respective columns are in string and not other dtypes. Afterwards, we went on to run the custom-defined spaCy pipeline.

   We first create a custom retokenizer pipeline such that we can combine special company names together. In the process, we also make a caps become small caps. We remove stop words also and puctuations.

```python
# import required libraries
from spacy.language import Language
from spacy.tokens import Doc
from spacy.lang.char_classes import ALPHA, ALPHA_LOWER, ALPHA_UPPER, CONCAT_QUOTES, LIST_ELLIPSES, LIST_ICONS
from spacy.util import compile_infix_regex

# initialise nlp engine
nlp = spacy.load("en_core_web_lg")

# declare custom properties
Doc.set_extension('processed', default=True, force=True)

# Modify tokenizer infix patterns
infixes = (
    LIST_ELLIPSES
    + LIST_ICONS
    + [
        r"(?<=[0-9])[+\-\*^](?=[0-9-])",
        r"(?<=[{al}{q}])\.(?=[{au}{q}])".format(
            al=ALPHA_LOWER, au=ALPHA_UPPER, q=CONCAT_QUOTES
        ),
        r"(?<=[{a}]),(?=[{a}])".format(a=ALPHA),
        r"(?<=[{a}0-9])[:<>=/](?=[{a}])".format(a=ALPHA),
    ]
)

infix_re = compile_infix_regex(infixes)
nlp.tokenizer.infix_finditer = infix_re.finditer

# custom retokenizer
@Language.component('custom_retokenizer')
def custom_retoken(doc):
    doc_text = doc.text.upper()
    doc_split = [i.text.upper() for i in doc]
    temp_kw = [i.lstrip().rstrip() for i in keywords_masterlist if len(i.lstrip().rstrip().split(' ')) > 1]
    for token in temp_kw:
        token_length = len(token.split(' '))
        token_split = token.split(' ')
        if token in doc_text and token_split[0] in doc_split:
            merge_pos = doc_split.index(token_split[0])
            with doc.retokenize() as retokenizer:
                try:
                    retokenizer.merge(doc[merge_pos:merge_pos + token_length], attrs={'LEMMA' : token.lower()})
                except:
                    print(merge_pos, merge_pos+token_length)

    return doc

# custom lemmatizer
@Language.component("custom_preprocess")
def custom_preprocess(doc):
    temp = []
    # filter through each token and add to preprocessed text if requirements met.
    for t in doc:
        if (not t.is_punct and not t.like_num and not t.is_stop and not t.is_digit and not (t.ent_type == 396 or t.ent_type == 397)):
            temp.append(t.lemma_.lower())

    temp = ' '.join(temp)
    doc._.processed = temp

    return doc

# add custom pipeline components to default pipeline
nlp.add_pipe('custom_retokenizer')
nlp.add_pipe('custom_preprocess', last=True)
```

Figure 19: Using spAcy custom-made tokenizer pipeline and the standard pipeline

7. We then proceed to run the pipelines and append it to the dataframe.

```python
# run the pipeline on data
processed_doc = list(nlp.pipe(df_valid['Company Profile Information']))
```

```
3 5
95 97
3 5
4 6
6 8
```

```python
# add lemmatised words to dataframe
df_valid['processed'] = [doc._.processed for doc in processed_doc]

df_valid
```

	Company	Country	Sector	Subsector	Archetype	Valuechain	Company Profile Information	length	list_tag	processed
0	AKSORN CHAROEN TAT ACT. CO.,LTD.	THAILAND	TMT	media	media_aggregator/distributor	MIDSTREAM	For over 80 years of experience in creating and developing high-quality learning materials has enabled us to provide world-class educational innovation to meet the needs of all teachers, students, institutions and educational authorities.	238	[6, 18, 49, 5]	year experience create develop high-quality learning material enable provide world-class educational innovation meet need teacher student institution educational authority
1	DONGGUAN SHENGYA CLEANING APPLIAME CO.LTD	CHINA	TMT	consumer electronics	consumer electronics_distributor	DOWNSTREAM	Yatai's main products cover various cleaning machinery, cleaning agents, cleaning tools, stone maintenance and other cleaning products; cleaning solutions and services include product technical consultation, product customization, employee training, maintenance and so on.	273	[6, 8, 20, 3]	yatai main product cover cleaning machinery clean agent clean tool stone maintenance cleaning product clean solution service include product technical consultation product customization employee training maintenance
2	EXIS TECH SDN. BHD.	MALAYSIA	OOS	others	others		In the beginning, it started off by providing technical support for test handlers, then moving into module design and production. Its first, in-house designed, full-fledged handler was introduced in 2008. Since then, the company has designed and produced a wide range of turret and pick-and-place solutions for its customers all over the world.	344	[4, 24, 64, 0]	beginning start provide technical support test handler move module design production in-house design full-fledged handler introduce company design produce wide range turret pick-and-place solution customer world
4	Aztech Electronics Pte Ltd	SINGAPORE	TMT	consumer electronics	consumer electronics_distributor	DOWNSTREAM	Being a turnkey, one-stop integrated solutions provider based in Singapore, Aztech is equipped with state-of-the-art equipment, R&D, design, manufacturing and packaging capabilities to deliver a seamless, unified experience. Each and every time. Always striving towards the edge of technology for more than 34 years, we have been building capabilities to serve clients' manufacturing needs, including the consumer electronics, telecommunications, healthtech, LED lighting, automotive and technology start-up market segments.	524	[6, 8, 20, 3]	turnkey one-stop integrate solution provider base singapore aztech equip state-of-the-art equipment r&d design manufacturing packaging capability deliver seamless unified experience time strive edge technology year build capability serve client manufacturing need include consumer electronics telecommunication healthtech lead lighting automotive technology start-up market segment
5	TONGDUN INTERNATIONAL PTE LTD	SINGAPORE	TMT	it_services	it_services	MIDSTREAM	Tongdun Technology is a professional third-party intelligent risk management and decision-making service provider headquartered in Hangzhou, Zhejiang. By integrating artificial intelligence into business scenarios, Tongdun Technology offers solutions in intelligent user analysis, intelligent risk management, intelligent antifraud and intelligent operation to clients from various industries including financial industry, internet business, logistics, healthcare, retail, smart cities and government bodies. Over 10,000 corporate clients have chosen Tongdun Technology's products and services	593	[6, 17, 46, 5]	tongdun technology professional third-party intelligent risk management decision-making service provider headquarter hangzhou zhejiang integrate artificial intelligence business scenario tongdun technology offer solution intelligent user analysis intelligent risk management intelligent antifraud intelligent operation client industry include financial industry internet business logistic healthcare retail smart city government body corporate client choose tongdun technology product service
...	...	...	...	...	...	...	...	...	...	...
593	BANGKOK PATANA SCHOOL	THAILAND	CG	retail n distribution	consumer discretionary distributor	DOWNSTREAM	Bangkok Patana School is Thailand's original British international school for students from 2 to 18 years. Our not-for-profit school has remained at the forefront of education in Thailand and is today one of the most respected educational establishments in South East Asia. We provide learning from Foundation Stage right through to Senior Studies, for over 2,200 students from around 60 countries. The School follows a modified English National curriculum up to Year 11 and the International Baccalaureate in Years 12 and 13. Bangkok Patana School is committed to nurturing lifelong learners who can adapt to an ever-changing world. Our goal is to give your children the best intellectual, physical and social preparation for their futures.	741	[0, 28, 18, 3]	bangkok patana school thailand original british international school student year not-for-profit school remain forefront education thailand today respected educational establishment south east asia provide learn foundation stage right senior studies student country school follow modify english national curriculum year international baccalaureate years bangkok patana school commit nurture lifelong learner adapt ever-changing world goal child good intellectual physical social preparation future

Figure 20: Running the custom-made tokenizer and the standard pipeline and populating it into the dataframe

8. In the VDCNN research papers, the researches made use of character embedding before feeding it into their model. As such, we have done char2vec word vectorisation.

   We first randomise our training dataset in order to achieve a more balanced and fair experiment. We then went on to define the alphabet lookup and our char2vec function for char2vec word embedding.

```
0]: # randomise dataset first here
    df_rand = df_valid.sample(frac=1)
```

```
]: ALPHABET = 'abcdefghijklmnopqrstuvwxyz0123456789-,;.!?:’"/|_#$%ˆ&*˜‘+=<>()[]{} '
   FEATURE_LEN = 1024 #maxlen as depicted in the papers
```

```
]: def get_char_dict():
       char_dict={}
       for i,c in enumerate(ALPHABET):
           char_dict[c]=i+1
       return char_dict

   def char2vec(text, max_length=FEATURE_LEN):
       char_dict = get_char_dict()
       data=np.zeros(max_length)

       for i in range(0, len(text)):
           if i >= max_length:
               return data

           elif text[i] in char_dict:
               data[i] = char_dict[text[i]]

           else:
               data[i]=len(ALPHABET)

       return data
```

```
]: char2vec_vectors = []

   for text in df_rand["processed"].fillna("NA").values:
     char2vec_vectors.append(char2vec(text))
   data = np.array(char2vec_vectors)
```

```
]: data.shape
]: (8423, 1024)
```

```
]: # char2vec in np array form
   data
]: array([[14.,  1., 14., ...,  0.,  0.,  0.],
          [19., 21., 26., ...,  0.,  0.,  0.],
          [15., 18.,  7., ...,  0.,  0.,  0.],
          ...,
          [ 3., 15., 13., ...,  0.,  0.,  0.],
          [16., 20., 67., ...,  0.,  0.,  0.],
          [ 6.,  9., 18., ...,  0.,  0.,  0.]])
```

Figure 21: Randomising training dataset and making char2vec word vectorization.

9. We will now prepare a one-hot encoding function to one hot encode our labels.

```
: # define one hot encode function
  def one_hot(arr, n_cat):
      output = []
      for n in arr:
          result = np.zeros(n_cat)
          result[n] = 1

          output.append(result)

      return np.array(output, dtype=int)
```

Figure 22: One-hot encoding function

10. We then proceed to assign the training dataset (data) and test dataset (data_test) into X_train and y_train, X_test and y_test respectively. The y labels are the label for the respective company descriptions.

```
from keras.preprocessing import sequence
# distribution = int(df_rand.shape[0] * 0.9)

# # split datasets to train and test and do a 90%, 10% split
# X_train, X_test = data[:distribution], data[distribution:]
# y_train, y_test = np.array(list(df_rand.iloc[:distribution]['list_tag'])), np.array(list(df_rand_test.iloc[distribution:]['list_tag']))

X_train = sequence.pad_sequences(np.array(list(data)))
y_train = np.array(list(df_rand['list_tag']))

X_test = sequence.pad_sequences(np.array(list(data_test)))
y_test = np.array(list(df_rand_test['list_tag']))

print(f'Train data shape: {X_train.shape}\nTest data shape: {X_test.shape}')
```

Figure 23: Defining train and test dataset to be fed into the model

11. As we are going to make use of branching, we have to prepare our labels for the respective branches. Here, we apply the one-hot encoding function to the respective labels for 'Sector', 'Sub-sector', 'Archetype' and 'Value chain'.

```
# since we are going to make use of multilabel classification, we need to split our data into the respective 4 classes

# names for the respective classes
label_names = ['sector', 'subsector', 'archetype', 'valuechain']

# First output
y1_train = one_hot(y_train[:,0], tag_counts[0])
y1_test =  one_hot(y_test[:,0], tag_counts[0])

# Second output
y2_train = one_hot(y_train[:,1], tag_counts[1])
y2_test =  one_hot(y_test[:,1], tag_counts[1])

# Third output
y3_train = one_hot(y_train[:,2], tag_counts[2])
y3_test =  one_hot(y_test[:,2], tag_counts[2])

# Fourth output
y4_train = one_hot(y_train[:,3], tag_counts[3])
y4_test =  one_hot(y_test[:,3], tag_counts[3])

y_train_multi = [y1_train, y2_train, y3_train, y4_train]
y_test_multi = [y1_test, y2_test, y3_test, y4_test]

# declare how we gonna print the loss for the respective classes
losses = {i : 'binary_crossentropy' for i in label_names}
```

Figure 24: Splitting into the respective labels

Model Training

1. As defined in the paper, the researchers used KMaxPooling in the main model. However, Keras does not provide a KMaxPooling layer and we have to self define it ourselves. Here, we wrote a function to define it.

```python
# self-defined k max pooling layer (since keras does not offer)
from tensorflow.keras.layers import Flatten, Layer, InputSpec

class KMaxPooling(Layer):
    """
    K-max pooling layer that extracts the k-highest activations from a sequence (2nd dimension).
    TensorFlow backend.
    """
    def __init__(self, k=3, **kwargs):
        super().__init__(**kwargs)
        self.input_spec = InputSpec(ndim=3)
        self.k = k

    def compute_output_shape(self, input_shape):
        return (input_shape[0], (input_shape[2] * self.k))

    def call(self, inputs):

        # swap last two dimensions since top_k will be applied along the last dimension
        shifted_input = tf.transpose(inputs, [0, 2, 1])

        # extract top_k, returns two tensors [values, indices]
        top_k = tf.nn.top_k(shifted_input, k=self.k, sorted=True, name=None)[0]
        # top_k = tf.nn.top_k(shifted_input, k=self.k)[0]

        # return flattened output
        return Flatten()(top_k)
```

Figure 25: Define KMaxPooling layer function

2. We then proceed to define the convolution blocks for 64, 128, 256 and 512 neurons. We adopted a depth-9 architecture as depth-9 has already a lot of parameters. It is noted in the research paper that lower parameter models for VDCNN still perform better than that with higher depth.

```python
# convolutional block layer for 64 and 128 neurons
from tensorflow.keras.regularizers import l2
from tensorflow.keras.layers import Conv1D, Dense, Dropout, MaxPooling1D, GlobalAveragePooling1D, Input, Lambda, Embedding
from tensorflow.keras.layers import ReLU, BatchNormalization, Activation
from tensorflow.keras import Sequential
from tensorflow.keras.models import Model

def convolutional_block(input_shape, num_filters):
  model = Sequential()

  # 1st conv layer
  model.add(Conv1D(filters=num_filters, kernel_size=3, strides=1, padding='same', input_shape=input_shape))
  model.add(BatchNormalization())
  model.add(Activation("relu"))

  # 2nd conv layer
  model.add(Conv1D(filters=num_filters, kernel_size=3, strides=1, padding='same'))
  model.add(BatchNormalization())
  model.add(Activation("relu"))

  return model
```

Figure 26: Defining a ConvBlock

3. We have to reshape the data first before feeding it into the ConvBlock. Here, we create a function to transform the shape of the data before feeding into the ConvBlocks.

```python
# we need to define this function so that we can get the input shape to be fed to the conv blocks
def conv_shape(conv):
  return conv.get_shape().as_list()[1:]
```

Figure 27: Defining conv_shape to reshape the data

4. Here, we will then build the main VDCNN model. This function will represent the VDCNN model in one branch, except for the embedding and the input layer as that will be defined later on.

```
from tensorflow.keras.layers import ReLU

def multi_branch(x, name, output_dim):

  # 1st layer temp conv(64)
  x = Conv1D(filters=64, kernel_size=3, strides=2, padding='same')(x)

  # 2nd + 3rd layer convblock(64) * 2
  x = convolutional_block(conv_shape(x), num_filters[0])(x)
  x = convolutional_block(conv_shape(x), num_filters[0])(x)

  # 4th layer pool/2
  x = MaxPooling1D(pool_size=3, strides=2, padding='same')(x)

  # 5th + 6th layer convblock(128) * 2
  x = convolutional_block(conv_shape(x), num_filters[1])(x)
  x = convolutional_block(conv_shape(x), num_filters[1])(x)

  # 7th layer pool/21
  x = MaxPooling1D(pool_size=3, strides=2, padding='same')(x)

  # 8th + 9th layer convblock(256) * 2
  x = convolutional_block(conv_shape(x), num_filters[2])(x)
  x = convolutional_block(conv_shape(x), num_filters[2])(x)

  # 10th layer pool/2
  x = MaxPooling1D(pool_size=3, strides=2, padding='same')(x)

  # 11th + 12th layer convblock(512) * 2
  x = convolutional_block(conv_shape(x), num_filters[3])(x)
  x = convolutional_block(conv_shape(x), num_filters[3])(x)

  # k max pooling (k=8)
  k_max = KMaxPooling(k=8)(x)

  # fully connected layers * 2(prev activation relu, softmax is a straight no, sofar best is tanh)
  fc1 = Dense(2048, kernel_initializer='he_normal', activation=ReLU(6))(k_max)
  fc2 = Dense(2048, kernel_initializer='he_normal', activation=ReLU(6))(fc1)

  # output layer(changed softmax to sigmoid)
  output = Dense(output_dim, activation='sigmoid', name=name)(fc2)

  return output
```

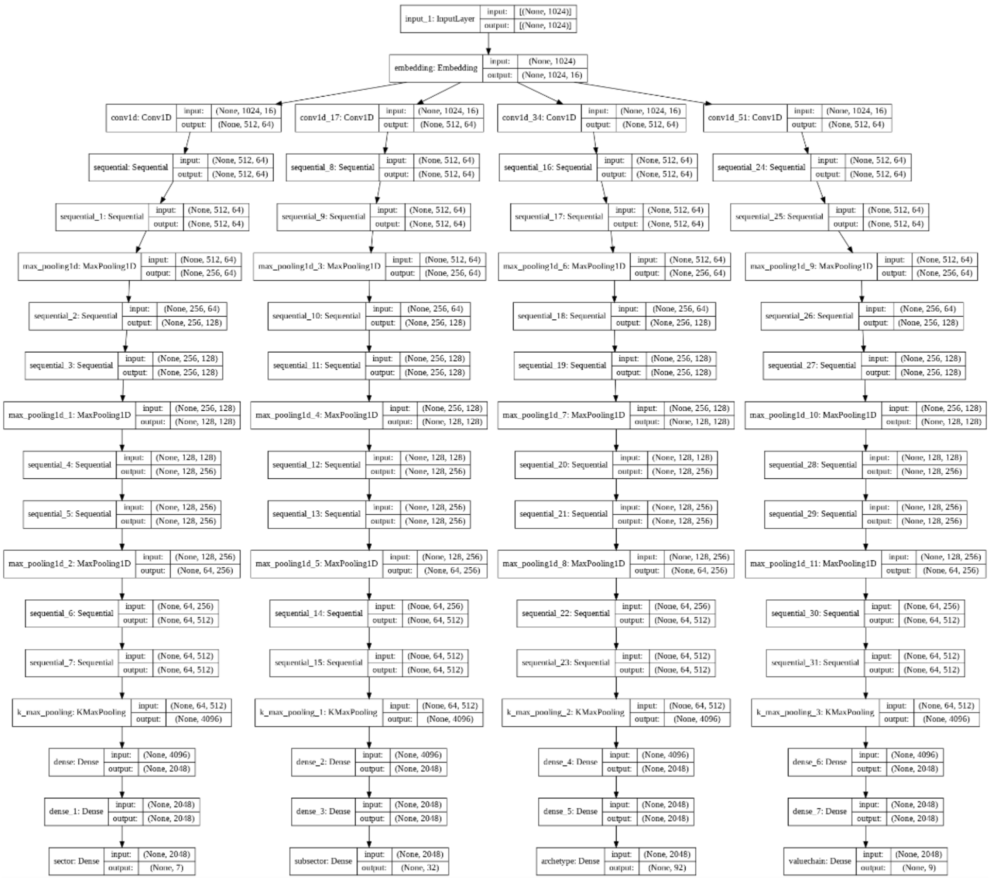
Figure 28: VDCNN model defined

5. We then created the main model which will be responsible for the number of branches to create, defining of the Input layer and Embedding layer, and fitting the entire model into Keras Model once done.

```
def create_model(labels, dist_words, input_dim, output_dim):

  # 0th layer lookup table
  inputs = Input(shape=(input_dim, ))
  embedded_seq = Embedding(dist_words, 16, input_length=input_dim)(inputs)

  # attempt to break the labels from here onwards first
  branches = []

  for i in range(len(labels)):
    branches.append(multi_branch(embedded_seq, labels[i], output_dim[i]))

  # fit the entire nn using Keras Model class so that we can print out the model summary
  model = Model(inputs=inputs, outputs=branches, name='company_classification_model')

  return model
```

Figure 29: Main branch creator

6. Once done, we will call the *create_model* function to create the entire multi-label model.



```
Model: "company_classification_model"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            [(None, 1024)]       0
__________________________________________________________________________________________________
embedding (Embedding)           (None, 1024, 16)     1104        input_1[0][0]
__________________________________________________________________________________________________
conv1d (Conv1D)                 (None, 512, 64)      3136        embedding[0][0]
__________________________________________________________________________________________________
conv1d_17 (Conv1D)              (None, 512, 64)      3136        embedding[0][0]
__________________________________________________________________________________________________
conv1d_34 (Conv1D)              (None, 512, 64)      3136        embedding[0][0]
__________________________________________________________________________________________________
conv1d_51 (Conv1D)              (None, 512, 64)      3136        embedding[0][0]
__________________________________________________________________________________________________
sequential (Sequential)         (None, 512, 64)      25216       conv1d[0][0]
__________________________________________________________________________________________________
sequential_8 (Sequential)       (None, 512, 64)      25216       conv1d_17[0][0]
__________________________________________________________________________________________________
sequential_16 (Sequential)      (None, 512, 64)      25216       conv1d_34[0][0]
__________________________________________________________________________________________________
sequential_24 (Sequential)      (None, 512, 64)      25216       conv1d_51[0][0]
__________________________________________________________________________________________________
sequential_1 (Sequential)       (None, 512, 64)      25216       sequential[0][0]
__________________________________________________________________________________________________
sequential_9 (Sequential)       (None, 512, 64)      25216       sequential_8[0][0]
__________________________________________________________________________________________________
sequential_17 (Sequential)      (None, 512, 64)      25216       sequential_16[0][0]
__________________________________________________________________________________________________
sequential_25 (Sequential)      (None, 512, 64)      25216       sequential_24[0][0]
```

Figure 30: Model diagram and snippet of model summary

7. We then proceed to start training the model, adding Accuracy, Loss, Precision, Recall, Area Under Curve (AUC) as metrics. We set epochs to 100 and batch size to 64. We also added a validation_split of 0.2 to analyse overfitting later on. SGD optimizer was used here instead.

```
from tensorflow.keras.metrics import Precision, Recall, AUC
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow_addons.metrics import HammingLoss
from timeit import default_timer as timer

# as stated in the paper, they use SGD with lr=0.01, momentum=0.9, weight decay=0.001
opt = SGD(lr=0.01, momentum=0.9, decay=0.001)

# # use earlystopping to prevent model overfitting
# es  = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=10)

# model training
model.compile(loss=losses, optimizer=opt, metrics=['accuracy', Precision(), Recall(), AUC(name='auc_precision_recall', num_thresholds=100
00), HammingLoss(mode='multilabel', threshold=0.6)])
# history = model.fit(X_train, y_train_multi, validation_split=0.2, epochs=200, batch_size=64, callbacks=[es])
start = timer()
history = model.fit(X_train, y_train_multi, validation_split=0.2, epochs=100, batch_size=64)
end = timer()
print("Total time taken for execution: ", end-start, "s")
```

Figure 31: Start of model training

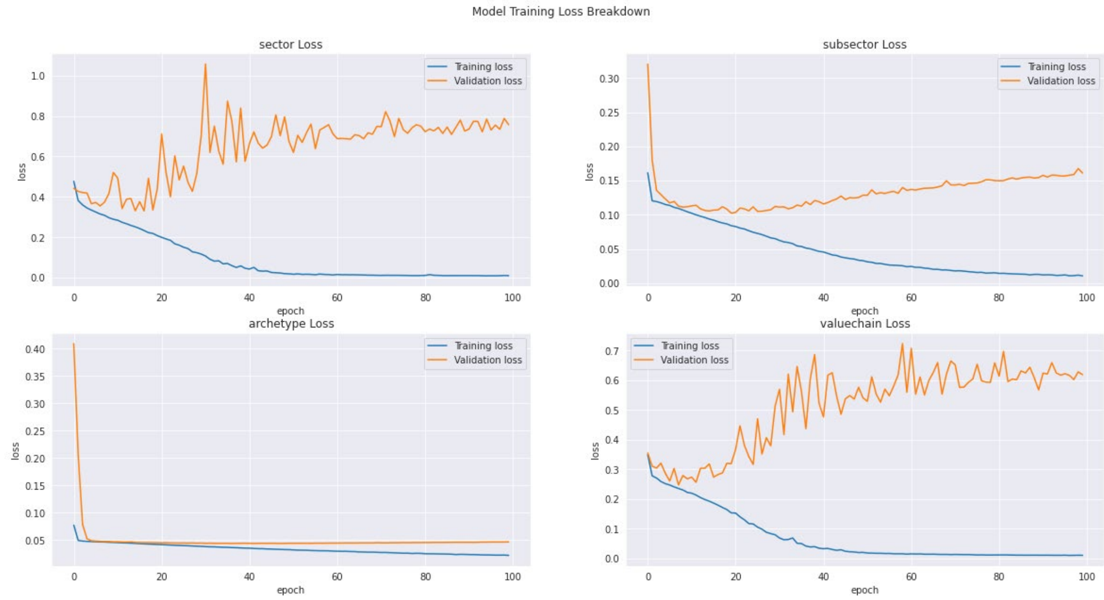
Model Evaluation

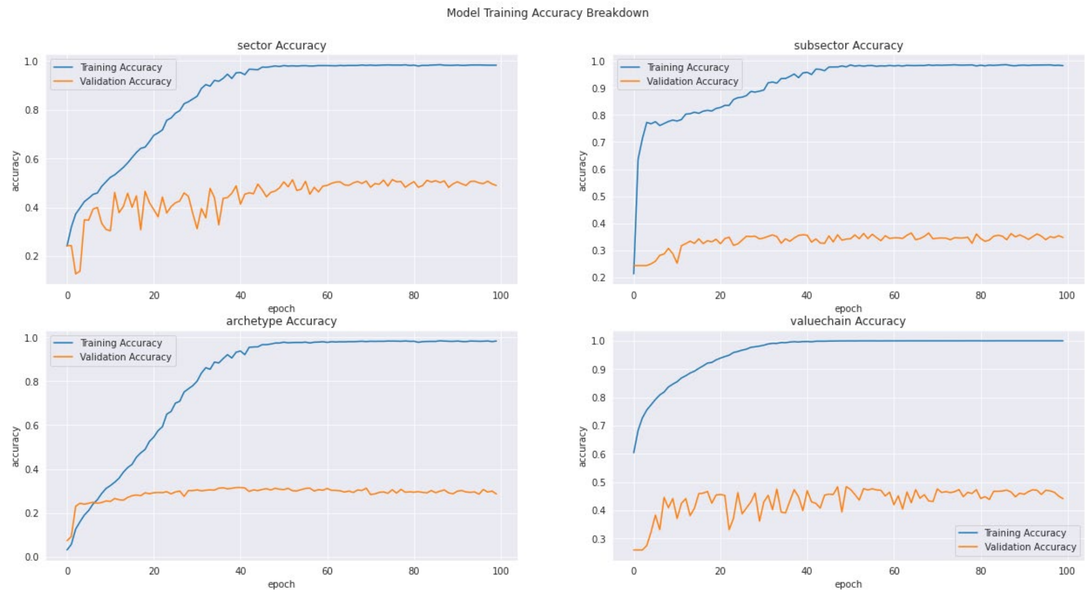
1. When our model training has been completed, we will use our X_test dataset to evaluate the model. This is a breakdown of the loss, accuracy, precision, recall, PR AUC and F1 Score of the model for the 4 categories.

Label	Loss	Accuracy	Precision	Recall	PR AUC
sector	0.329	78.786%	0.056	0.914	0.265
subsector	0.073	82.791%	0.704	0.015	0.921
archetype	0.035	76.286%	0.847	0.510	0.009
valuechain	0.279	92.940%	0.654	0.817	0.764

Figure 32: Evaluated score of the model

2. We also plotted graphs for the sector loss, accuracy and the respective metrics as defined earlier on.





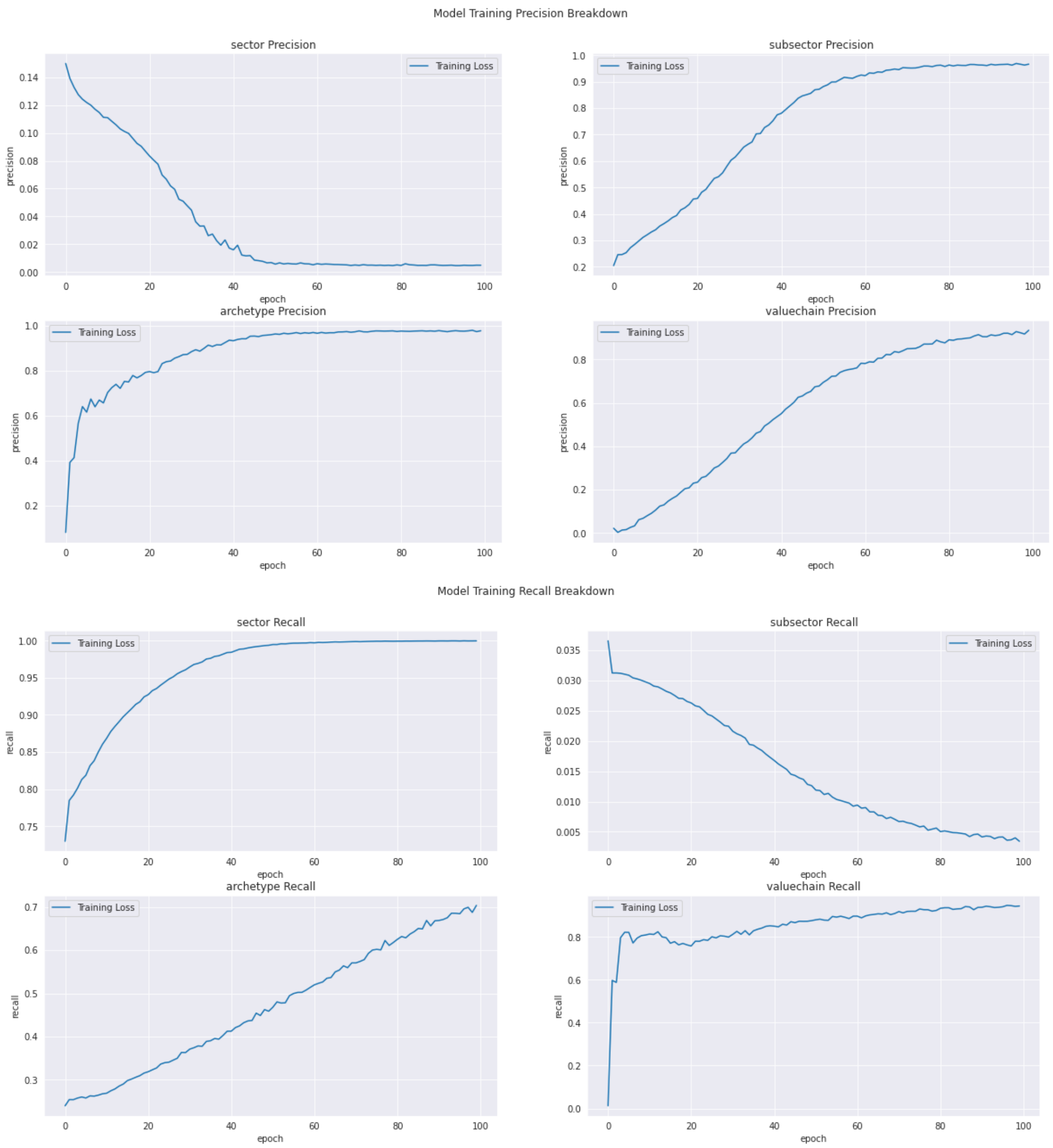


Figure 33: Graphs for the model metrics