Week 1: Build Industry Classifier

Kaiyun (Kiki) Kang | kk3567@nyu.edu

Objectives: build a classifier that can distinguish between enterprise saas companies and others based on the company's website

1. Preprocessing data Findings

a) Delete vacant column "content_nav"

b) Inner merge input and target data, since they have different websites

c) **Preprocessing** by removing punctuations, stop words, and implementing tokenizing, lemmatization and lower casing

- I also removing numerical numbers, since later in *tokenizer.word_index* step, some numbers have higher rank and frequencies. Numerical numbers are not much useful in this case.
- After cleaning, len(content_txt words) decreased a lot. More informative words are reserved.

d) One column information is good enough to make classification.

- I chose "content_txt", since there is no nan and have long enough information. Although after preprocessing the maximum word length for "content_txt" is 127512, the 0.95 quantile for word length of each website is only 1059. So, *maxlen* of *pad_sequences* set to 1000.
- I selected "website_summary" because it is summaries of website and it contains brief but very effective information, which is different from "content_txt". Drop 5619 nan rows.

Two new preprocessed Dfs	columns	index	Max length	0.95 quantile
	"content_txt" + "target"	"domain_name"	127512 words	1056 words
	"website_summary" + "target"	"domain_name"	365 words	33 words

2. LSTM Model Establishing Findings

a) *train_test_split* using with stratify=labels, so as to reduce effects of imbalanced data

b) *tokenizer.texts_to_sequences*: set unique vocabulary library and encode each observation from sentence to vector. *tensorflow.keras.layers.Embedding* will embed each word to multi-dimensions

c) F-1 score is suitable for evaluate the performance of imbalanced data.

d) **Performing baseline:**

- accuracy is 0.71, since the majority 0 labels occupy 71% of total data.
- f1 baseline is around 0.59 with predicting all labels to 0
- Both LSTM models of two new data frames outperform 0.59 f1 and 0.71 accuracy baseline.

e) **Tradeoff:**

- "website_summary" data has highest 0.8007 F1 score. But it deletes 5000 nan observations so that "website_summary" may take more information when we want our model to be more general for predicting new data.
- "website_summary" data is much time saving than "content_txt"

f) **Hyperparameter tuning:** by tuning, "content_txt" data prediction increases from 0.74 to 0.80.

	Train F1 Score	Test F1 Score	F1 Baseline	Training time
"content_txt" df	0.8778	0.8007	0.596	65 mins
"website_summary" df	0.8473	0.8199	0.592	5 mins

```
In [50]: model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=5, batch_size=32)
Epoch 1/5
503/503 [==============================] - 773s 2s/step - loss: 0.5002 - acc: 0.7632 - f1_score: 0.7446 - val_loss: 0.4848 - val_acc: 0.7682 - val_f1_score: 0.7399
Epoch 2/5
503/503 [==============================] - 777s 2s/step - loss: 0.3771 - acc: 0.8373 - f1_score: 0.8348 - val_loss: 0.4412 - val_acc: 0.7860 - val_f1_score: 0.7871
Epoch 3/5
503/503 [==============================] - 765s 2s/step - loss: 0.2891 - acc: 0.8794 - f1_score: 0.8778 - val_loss: 0.4736 - val_acc: 0.8028 - val_f1_score: 0.8007
Epoch 4/5
503/503 [==============================] - 748s 1s/step - loss: 0.2268 - acc: 0.9084 - f1_score: 0.9073 - val_loss: 0.5051 - val_acc: 0.7876 - val_f1_score: 0.7839
Epoch 5/5
503/503 [==============================] - 755s 2s/step - loss: 0.1853 - acc: 0.9259 - f1_score: 0.9249 - val_loss: 0.6287 - val_acc: 0.7990 - val_f1_score: 0.7935
Out[50]: <keras.callbacks.History at 0x1df7ce01a60>
```