

# Section 1: Scraping 2018

## 1.1. Progress

Đã scrape từ data points từ 0 -> 6600.

Có tổng cộng 64643 (từ 0 -> 64642) URL để scrape, còn lại 58042 data points.

Scrape 1.2K URLs mất tầm ~8 tiếng chạy Kaggle, có thể giảm delay để chạy nhanh hơn, nhưng dễ gặp lỗi hơn.

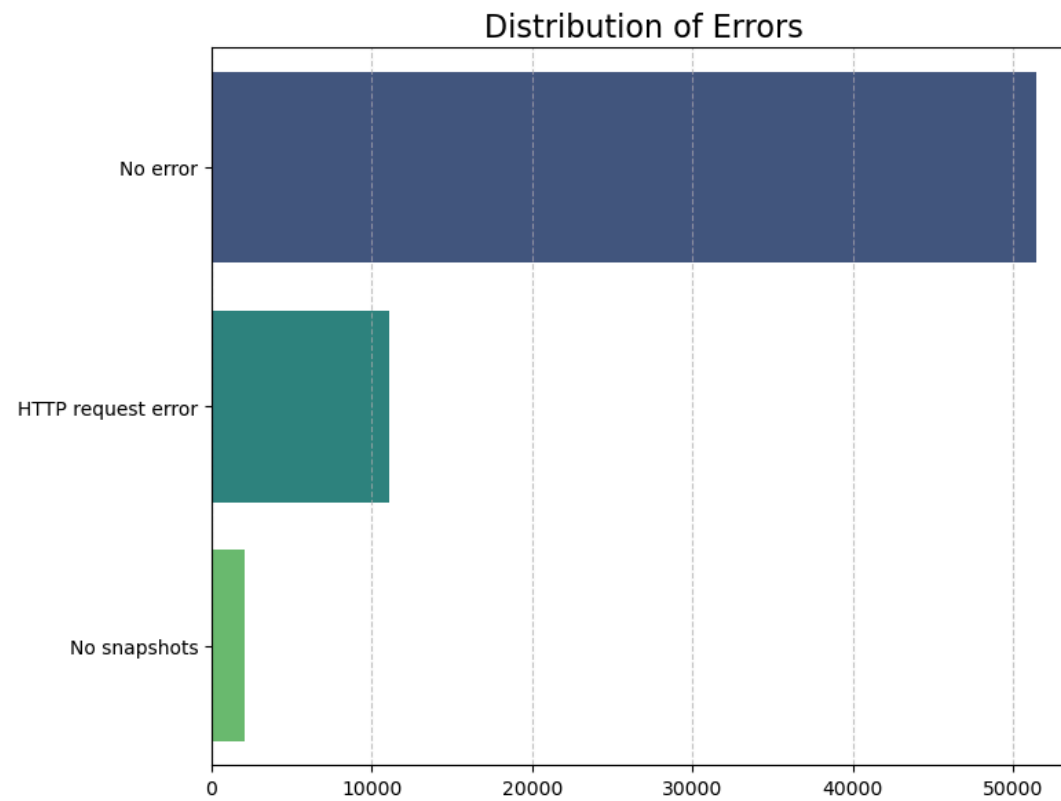
--> 1k2 URLs just 7 hours (nice)

Start	End	State	URL		Start	End	State	URL	
6601	8100	✓			32321	33521	✓	<a href="#">quang/notebook3696579db0?scriptId=247478525</a>	
8101	9600	✓	<a href="#">e/bngtiutr/us-web?scriptId=247684153</a>		33522	34722	✓	<a href="#">u/notebook5d1c80e71d?scriptId=247480480</a>	
9601	11100	✓	<a href="#">e/bngtiutr/us-web?scriptId=247761380</a>		34723	35923	✓	<a href="#">ngng/notebook9572c1e088?scriptId=247481203</a>	
11101	12600	✓			35924	37124	✓	<a href="#">zz/notebook749bf4d36a?scriptId=247481827</a>	
12601	14100	✓	<a href="#">arynbangtri/us-web?scriptId=247681819</a>		37125	38325	✓	<a href="#">quang/notebook3696579db0?scriptId=247568120</a>	
14101	15600	✓			38326	39526	✓	<a href="#">u/notebook5d1c80e71d?scriptId=247568348</a>	
15601	17100	✓	<a href="#">e/bngtiutr/us-web?scriptId=247822250</a>		39527	40727	✓	<a href="#">ngng/notebook9572c1e088?scriptId=247568464</a>	
17101	18600	✓	<a href="#">arynbangtri/us-web?scriptId=247849140</a>		40728	41928	✓	<a href="#">zz/notebook749bf4d36a?scriptId=247568564</a>	
18601	20100	✓	<a href="#">arynbangtri/us-web?scriptId=247903935</a>		41929	43129	✓	<a href="#">quang/notebook3696579db0?scriptId=247615629</a>	
20101	21600	✓	<a href="#">e/bngtiutr/us-web?scriptId=247904305</a>		43130	44330	✓	<a href="#">u/notebook5d1c80e71d?scriptId=247616316</a>	
21601	23100	✓	<a href="#">arynbangtri/us-web?scriptId=247992354</a>		44331	45531	✓	<a href="#">ngng/notebook9572c1e088?scriptId=247708155</a>	
23101	24600	✓	<a href="#">e/bngtiutr/us-web?scriptId=247992797</a>		45532	46732	✓	<a href="#">zz/notebook749bf4d36a?scriptId=247616758</a>	
24601	26100	✓	<a href="#">u/notebook5d1c80e71d?scriptId=248003596</a>		46733	47933	✓	<a href="#">quang/notebook3696579db0?scriptId=247707910</a>	
26101	27600	✓	<a href="#">quang/notebook3696579db0?scriptId=247956371</a>		47934	49134	✓	<a href="#">u/notebook5d1c80e71d?scriptId=247708051</a>	
27601	29100	✓	<a href="#">zz/notebook749bf4d36a?scriptId=247912974</a>		49135	50335	✓	<a href="#">u/notebook5d1c80e71d?scriptId=247766534</a>	
29101	30700	✓	<a href="#">ngng/notebook9572c1e088?scriptId=247912778</a>		50336	51536	✓	<a href="#">zz/notebook749bf4d36a?scriptId=247708323</a>	
30701	32320	✓	<a href="#">u/notebook5d1c80e71d?scriptId=247912698</a>		51537	52737	✓	<a href="#">quang/notebook3696579db0?scriptId=247766316</a>	
		□			52738	53938	✓	<a href="#">ngng/notebook9572c1e088?scriptId=247766636</a>	
		□			53939	55139	✓	<a href="#">quang/notebook3696579db0?scriptId=247815925</a>	
		□			55140	56340	✓	<a href="#">u/notebook5d1c80e71d?scriptId=247816090</a>	
		□			56341	57541	✓	<a href="#">ngng/notebook9572c1e088?scriptId=247816266</a>	
		□			57542	58742	✓	<a href="#">zz/notebook749bf4d36a?scriptId=247816529</a>	
		□			58743	59943	✓	<a href="#">quang/notebook3696579db0?scriptId=247878675</a>	
		□			59944	61144	✓	<a href="#">u/notebook5d1c80e71d?scriptId=247878800</a>	
		□			61145	62345	✓	<a href="#">ngng/notebook9572c1e088?scriptId=247878924</a>	
		□			62346	63546	✓	<a href="#">zz/notebook749bf4d36a?scriptId=247879050</a>	
		□			63547	64642	✓	<a href="#">quang/notebook3696579db0?scriptId=247912497</a>	

## 1.2. Analysis

### Error Distribution:

```
Error
No error          51424
HTTP request error 11123
No snapshots      2096
Name: count, dtype: int64
```



```
In [10]: def calculate_snapshot_years_percentage(df, year):
df['snapshot_year'] = pd.to_numeric(df['snapshot_year'], errors='coerce')
df_valid_years = df.dropna(subset=['snapshot_year'])

total_data_points = len(df_valid_years)
count_greater = df_valid_years[df_valid_years['snapshot_year'] > year].shape[0]
count_less_equal = df_valid_years[df_valid_years['snapshot_year'] <= year].shape[0]

percentage_greater = (count_greater / total_data_points) * 100
percentage_less_equal = (count_less_equal / total_data_points) * 100

print(f>Data points with snapshot_year > {year}: {count_greater} ({percentage_greater:.2f}%)")
print(f>Data points with snapshot_year <= {year}: {count_less_equal} ({percentage_less_equal:.2f}%)")

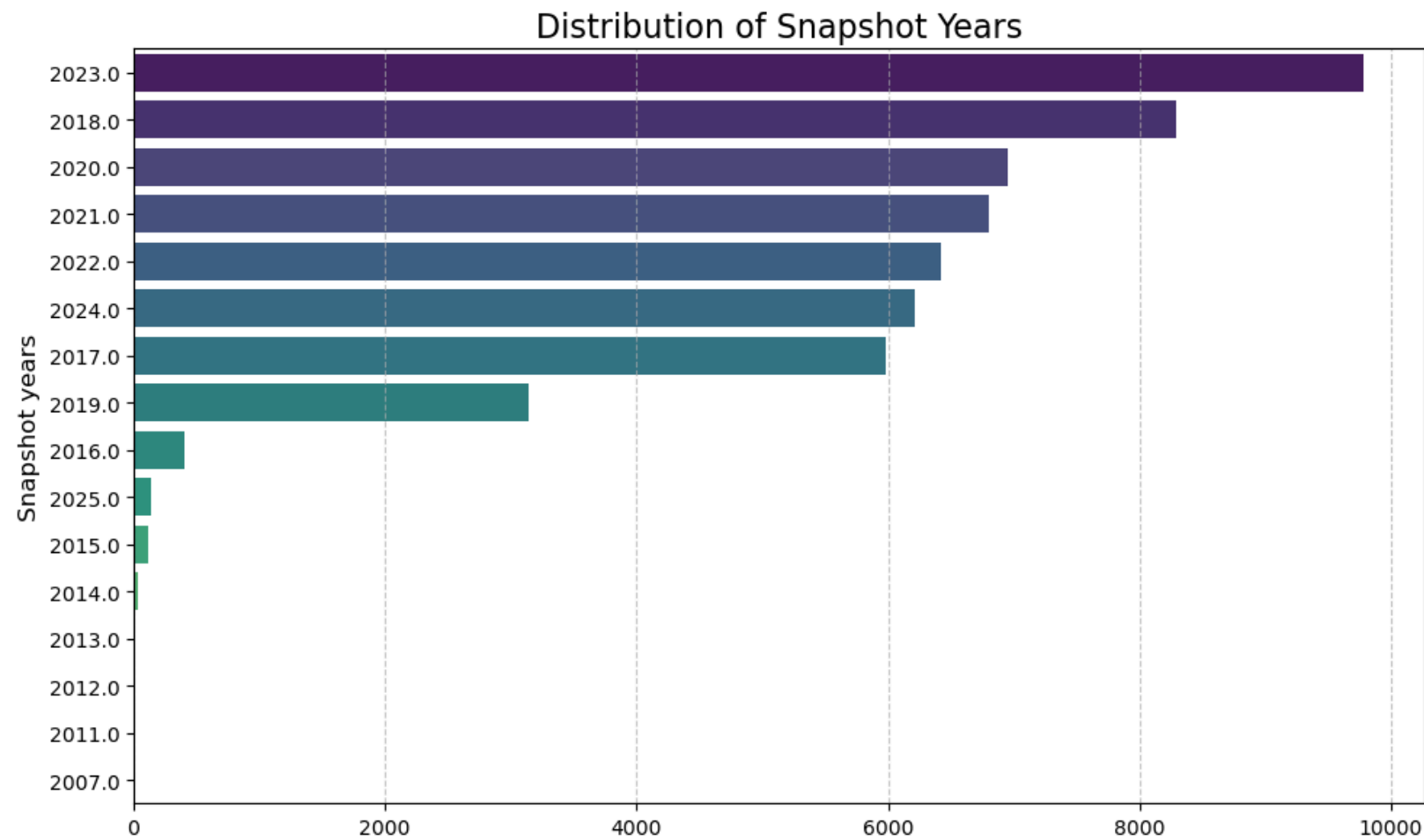
calculate_snapshot_years_percentage(scraped_df, 2019)

Data points with snapshot_year > 2019: 36300 (66.88%)
Data points with snapshot_year <= 2019: 17976 (33.12%)
```

Out of 54276 valid scraped data points:

- Data points with snapshot\_year > 2019: 36300 (**66.88%**).
- Data points with snapshot\_year <= 2019: 17976 (**33.12%**).

Distribution of Snapshot years is shown below:



## Section 2: Dual Attention Model

### 2.1. Preprocessing and Data insights

In this section, we talk about the process of **reducing the size of the Fasttext dictionary** and **cleaning the Us\_Web dataset**.

The first thing we noticed upon downloading the Fasttext dictionary was that the size was too large.

#### Load original FastText dictionary

```
In [13]: vec_file = r'C:\Users\Daryn Bang\Desktop\Internship\cc.en.300.vec.gz'
```

```
en_dictionary = FastVector(vector_file=vec_file)
```

```
words = list(en_dictionary.word2id.keys())
```

```
vectors = np.array([en_dictionary[word] for word in words])
```

```
wv_dict = dict(zip(words, vectors))
```

reading word vectors from C:\Users\Daryn Bang\Desktop\Internship\cc.en.300.vec.gz

```
In [14]: print(words[:200])
```

```
print(len(words))
```

```
[',', 'the', '.', 'and', 'to', 'of', 'a', '</s>', 'in', 'is', ':', 'I', 'for', 'that', ')', '"', '(', 'on', 'with', 'it', 'yo  
u', 'The', 'was', 'as', 'are', 'at', '/', '!', 'be', 'by', "'s", 'this', 'have', 'from', 'or', '!', 'not', 'your', 'an', '"',  
'but', '?', 'can', '-', 'will', 's', 'my', 'has', 'all', 'we', 'they', 'he', 'his', 'more', 'one', 'about', 'their', "t", 's  
o', 'which', 'It', 'out', 'up', '...', 'were', 'had', 'who', 'like', ';', '"', 'our', 'would', '"', 'time', 'been', 'if', 'als  
o', 'just', 'when', 'her', 'This', 'me', 'there', 'do', 'what', 'some', 'other', 'In', 'them', '-', '1', 'get', 'new', 'into',  
'&', 'We', 'than', 'A', 'no', 'only', 'first', 'any', 'its', 'people', '2', '$', 'very', 't', 'over', 'she', '%', 'how', 'mak  
e', 'You', 'said', 'He', 'two', 'may', 'know', 'then', 'see', 'after', 'most', 'good', 'years', 'If', 'these', 'now', '3', 'us  
e', 'because', 'well', 'work', 'could', 'us', 'don', 'way', 'much', 'back', 'many', 'think', 'where', 'even', 'him', 'through',  
'am', '10', '|', 'here', '#', 'made', 'year', 'should', '*', 'really', 'being', 'such', 'need', 'great', 'And', ']', '4', '[',  
'5', 'day', 'before', 'want', 'used', 'go', 'those', '...', 'But', 'right', "m", 'take', '-', 'May', 'still', 'last', 'off', 'to  
o', 'New', 'going', 'best', 'find', 'love', 'did', 'while', 'home', 'There', 'They', 'same', 'around', 'help', 'down', 'informa  
tion', 'UTC', 'place', 'i', '2017']  
2000000
```

As we can see, the dictionary from Fasttext has 2 million words and each word has a 300d vector. Loading this dictionary whenever we have to restart our sessions takes a huge toll on our personal laptop's RAMs, and they cannot even be loaded in cloud-based platforms like Google Colab or Kaggle. Further analysis shows that the "text" column in Us\_Web only contains 297761 unique words, with only 115k words overlapping with the Fasttext Dictionary, and the remaining 180k words missing from the dictionary.

## Filtering words

```
In [24]: all_words = [word for content_string in df['cleaned_content'] for word in content_string.split(' ')]
         unique_words = set(all_words)
         print(len(unique_words))

297761
```

Number of Tokens retained in fasttext dictionary after **ONLY** keeping the words that appear in Us\_web or Us\_patent datasets:

```
In [31]: all_words_web = [w for text in df['cleaned'] for w in text.split(' ')]
         unique_web = set(all_words_web)

         all_words_patent = [w for text in patent_df['cleaned_patent'] for w in text.split()]
         unique_patent = set(all_words_patent)

         # Union unique web and patent words
         union_tokens = unique_web.union(unique_patent)
         print(f"Total unique tokens in web U patent: {len(union_tokens)}")

Total unique tokens in web U patent: 252198
```

```
In [32]: # Intersect with FastText vocab
         filtered_words = [w for w in union_tokens if w in en_dictionary]
         print(f"Tokens retained (in FastText): {len(filtered_words)}")

         # Build the filtered word->vector dict
         filtered_word_vecs = { word: en_dictionary[word] for word in filtered_words }

Tokens retained (in FastText): 108184
```

```
In [34]: missing = [word for word in unique_words if word not in en_dictionary]
print(f"Missing words: {len(missing)}")
print("Examples:", missing[:200])
```

Missing words: 182466

Examples: ['参与', '甘酸っぱく', 'リフト', 'portfolios3', 'wwwdrummondgroupcom', 'wwwgovernorcagov', '84433pimco', 'ambius', 'mssc', '9r046', 'couchbaselite', '強制', 'trzupek', 'tagr', 'ed5hvaxvl00', 'rapix', 'camanche', 'aps12212', 'wwwnanomixdxcom', 'mikealcottmyfwcom', 'nenovabui', 'parissa', '000149315223024544', '年中', '61574', 'cpjsf', '535bn', 'scheckenbach', 'egglands', 'messerschmitt', 'otcqbbrxmd', '入り込み', 'sharyland', 'aleutian', 'securityeventbritecom', 'ircollegiumpharmacom', 'tqwsntzxdurfpjj', 'slc5a1', '72431', 'dccucs', 'speedpay', 'platformerydex', 'mavacamten', 'uicc', 'wwwalarmcom', 'lunasins', 'brohman', 'tradiers', 'harada', 'v6017820', 'chromatide', 'oxyfresh', 'usmca', '当てはまら', 'paulas', '約束', 'calpainopathy', 'simkins', 'sendsei', 'biontech', 'ヘッダー', 'rellasercom', 'ovps', 'partnersread', 'bedminster', '37964m107', 'createsession', '領収書と払い戻し', 'vitess', 'ransomjonesgwtechinccom', 'bigdaddysdinercloudcroftcom', 'tecnam', '20261', 'intelgenx', 'constanze', 'talbott', 'ドーナ', '341kb', 'qulifies', 'frameborderquot', 'つくり上げ', 'ffvs', 'slavik', 'ココ', 'quimica', 'searchengineoptimization', '31354', 'infowwccpacom', 'tissuetek', 'newstem', 'productsconductor', 'dectin', 'bthornhillff', '党史', 'cosimos', 'buenao', 'bra', 'gelysate', 'uhplc', 'rizzuti', 'landmens', 'ブエルト', 'ostdiek', 'sitesmapsmarket', 'nucors', '創立', 'wwwguideautowebcom', 'wwweurop', 'wwwnewsnet', 'hewb', 'ormisrepresentations', '30x1g', 'engineeringcom', '利用率', '45415', '012624', 'vlmtg2332abca', '进一步', 'kimpton', 'bwbatlas', 'ajrccm', 'tcbk', '就是', 'haccp', 'glyko', '还原', 'sanmersen', 'derusso', 'irusfoodscom', '71949', 'corinda', '1472774', 'ijss', '刘宗超', '64400', 'damageinducible', 'mozes', '69374h428', 'wurls', 'usd100', '詰まっ', 'フッター', 'wwwsearchvitalcom', 'bloombergnefs', 'allkem', 'easyfillnet', 'twinmotion', 'cryofat', 'techblick', 'spanishbroadcasting', 'gibraltars', 'dynatronics', 'pinkbiel', 'friedbergdirectfxca', 'ahearn', '48422', 'bsgfeedbackberkleysuretycom', 'liveweb', 'dalgliesh', 'skretting', 'mashgin', 'trmondelezinternationalcom', 'clubcorp', 'freepikcom', 'brexhaust', 'elitebridalcom', 'マオリ語', 'e20162976', 'marous', '30335h758', 'irmaa', 'allwell', 'saltzgiver', 'iedifactsheet', 'overdiscounting', 'セナ', '41125', 'sagehr', '税別', 'wwwheliotechnologiescom', 'キャニオン', 'hanssar', '開通', 'ibic', 'casq', '下押し', 'usphq', 'ageles', '04700', 'amznto', 'molchadsky', 'jaybees', 'wwwhumbllyhempcom', 'serhant', '論争', 'flexscan', 'preauthorization', 'nejdek', 'aspinettadegreesprcom', 'farmercore', 'kaynebdccom']

A lot of the missing words from the Fasttext dictionary showcase that the **majority of the missing tokens are either non-ASCII, words that don't exist or numbers that don't mean anything.**

To this end, we first decided to **filter the Fasttext dictionary** so that it only includes words that appear either in the “text” column of the Us\_Web dataset or “patent\_abstract” column of the Us\_patent dataset. This significantly reduces the number of tokens in the dictionary, making it more feasible to load into either local computers or cloud platforms. Additionally, we also **performed data cleaning on the Us\_Web and the Us\_patent datasets.** We do experiments below to show that **extracting keywords using Dual Attention from a cleaned dataset will harbor more fruitful results than the original uncleaned version.**

## 2.2. Progress

- We explore the **current implementation** for the Dual Attention Model
- Try to **modify some values** of the Dual Attention Model to analyze the performance changes
- Try to **change the structure** of the Dual Attention Model to see if there is some **potential improvements** for the model

**Previously the structure of the Dual Attention Model** was defined like this

- **2 LayerNorm layers**: one for the Word-level and one for the Page-level
- **Sigmoid Layer**: to provide binary probability
- **Softmax Layer + Sparsemax Layer**  $\Rightarrow$  Like in the definition of model
- **Embeddings Layer**: fasttext vectors would be loaded and saved here  $\rightarrow$  Fasttext vectors have dimension of 300
- **Dropout**  $\Rightarrow$  for making the model more robust
- **Affine Layer**: Linear Layer for transform from input\_dim  $\rightarrow$  hidden\_dim  
 $\rightarrow$  Need this layer **if the hidden dimension you want is different from the embedding dimension**
- **Decoder Layer**: Linear Layer for transform from hidden\_dim  $\rightarrow$  label\_dim

### Flow:

Embeddings  $\rightarrow$  (Affine Layer)  $\rightarrow$  Word-level Attention  $\rightarrow$  Webpage Vectors + Word Attention  $\rightarrow$  (Layer Norm 1)  $\rightarrow$  Firm Vector + Page Attention  $\rightarrow$  (Layer Norm 2)  $\rightarrow$  Classifier

Based on the architecture, we **analyze several components** like below:

- Change hidden dim 300  $\rightarrow$  512
- Add Layer Norms to the 2 level Attentions (previously in the code they were commented and not used)
- Add Affine Layer (previously it was commented)
- Change those 2 simple Attentions  $\rightarrow$  Scaled Dot Product attention  
(each word or page has their own Q K V values, then calculate softmax of (masked Q\*K) \* V / dimension d)



## 2.3. Analysis

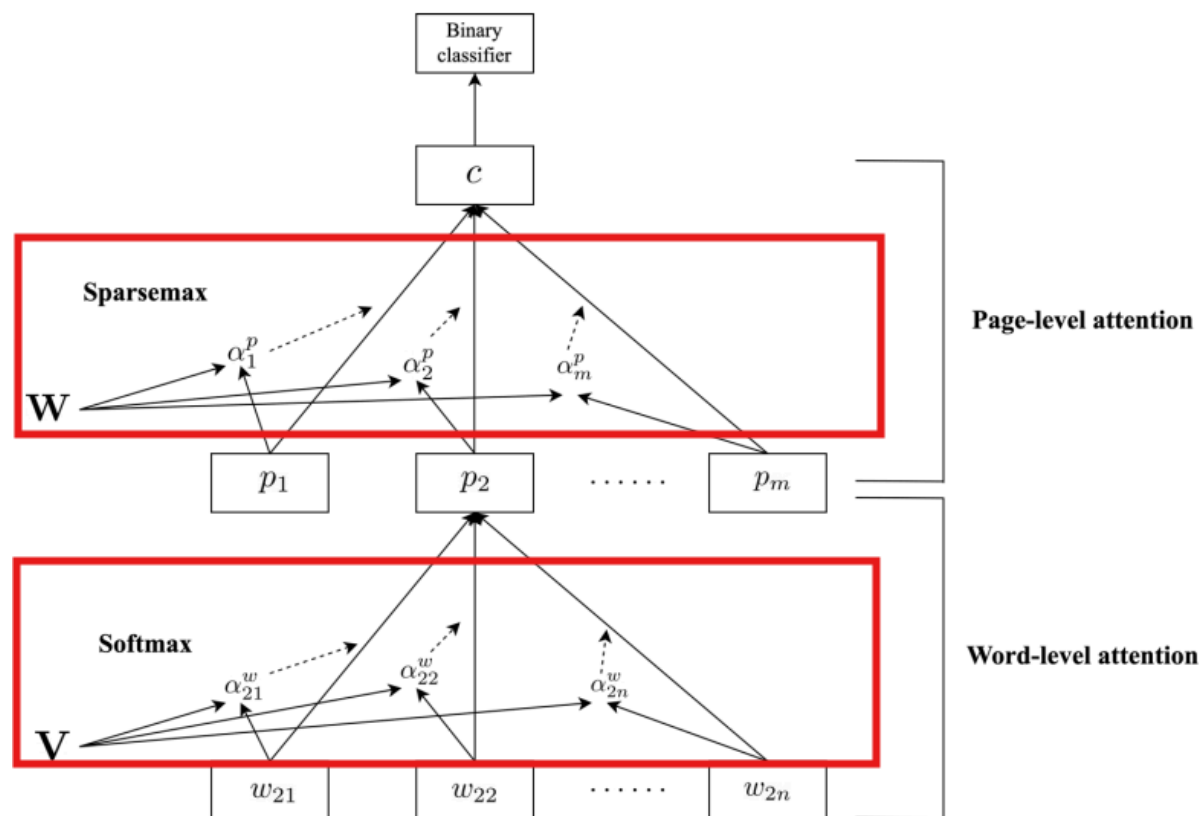


Figure 1. The structure of the **dual-attention model**

optimizer = torch.optim.**Adagrad**(model.parameters(), lr=0.02, weight\_decay=0.0000, lr\_decay=0.01)

⇒ same as original

⇒ We **split the data 90% train and 10% test/val for all the following tests**

## Note for Time and Performance of Training

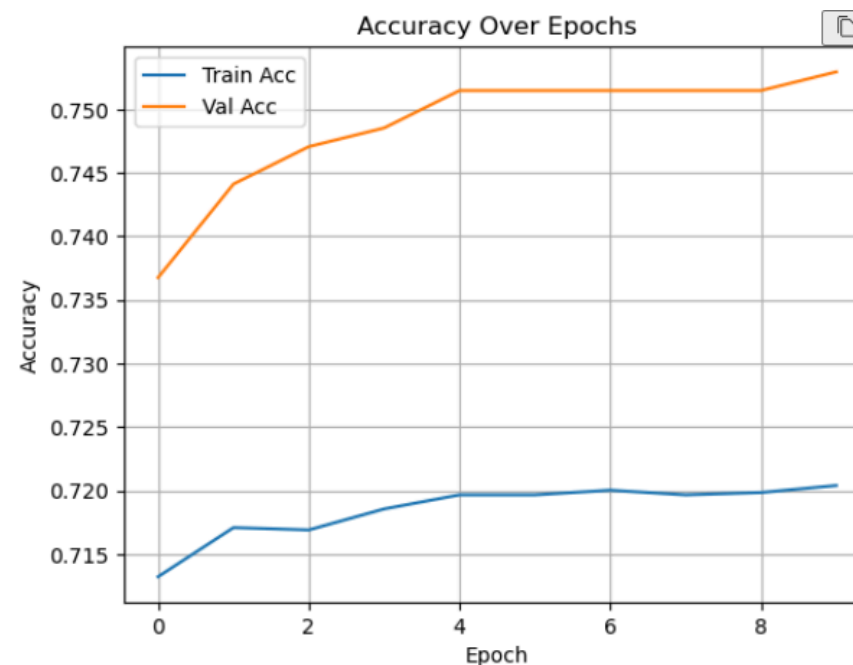
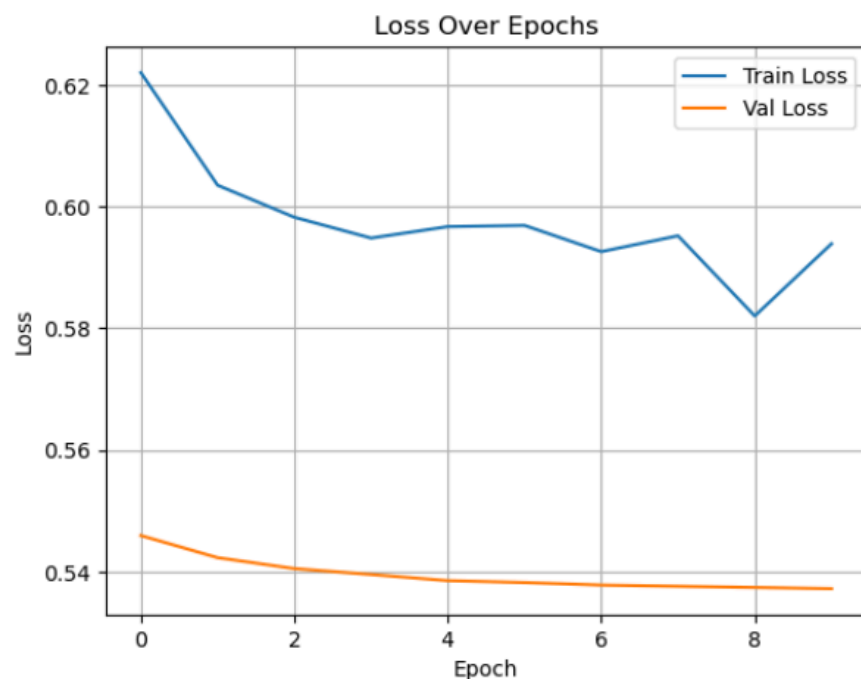
- In the previous implementation, we loaded the **fasttext model** into the Embedding Layers, so it takes such a **long time to load when we first initialize the model and during training** also.
- Then, we decided to **recreate the vector model** that just contains only the embedding of those tokens that is in our data and in overall, it helps us reduce the time needed for training **4 or 5 times**.
- To further demonstrate, now we show the **performance during training, time needed** for training and our conclusion

## Architecture Analysis

### 2 layer norm + No affine + hidden 300 (Best)

⇒ ~ 8/9 min / epoch ⇒ ~2 min / epoch

⇒ **accuracy ~ 0.75 val** ⇒ peak at epoch 4



Epoch 8/10

Train Loss: 0.5952 Val Loss : 0.5376

Train Metrics: {'accuracy': 0.7196691176470589, 'precision': 0.7268268759195684, 'recall': 0.6048979591836735, 'f1\_score': 0.6602806861216306, 'roc\_auc': 0.7862025117739403}

Val Metrics : {'accuracy': 0.7514705882352941, 'precision': 0.7896678966789668, 'recall': 0.656441717791411, 'f1\_score': 0.7169179229480737, 'roc\_auc': 0.8263318429170566}

Epoch 9/10

Train Loss: 0.5820 Val Loss : 0.5374

Train Metrics: {'accuracy': 0.7198529411764706, 'precision': 0.7274066797642437, 'recall': 0.6044897959183674, 'f1\_score': 0.6602764155149353, 'roc\_auc': 0.7863868677905945}

Val Metrics : {'accuracy': 0.7514705882352941, 'precision': 0.7896678966789668, 'recall': 0.656441717791411, 'f1\_score': 0.7169179229480737, 'roc\_auc': 0.8265051471352812}

Epoch 10/10

Train Loss: 0.5939 Val Loss : 0.5372

Train Metrics: {'accuracy': 0.7204044117647059, 'precision': 0.7280314187530682, 'recall': 0.6053061224489796, 'f1\_score': 0.6610207265433474, 'roc\_auc': 0.7864978499761108}

Val Metrics : {'accuracy': 0.7529411764705882, 'precision': 0.7925925925925926, 'recall': 0.656441717791411, 'f1\_score': 0.7181208053691275, 'roc\_auc': 0.8266654535371389}

### 0 layer norm + No affine + hidden 300

⇒ 10 min / epoch ⇒ 2-3 min / epoch

⇒ accuracy bad ⇒ precision high, recall near 0 ==> almost predict all as non-tech

Epoch 8/10

Train Loss: 0.6746 Val Loss : 0.6762

Train Metrics: {'accuracy': 0.5516544117647059, 'precision': 0.8235294117647058, 'recall': 0.005714285714285714, 'f1\_score': 0.011349817592217268, 'roc\_auc': 0.7581749368643779}

Val Metrics : {'accuracy': 0.5235294117647059, 'precision': 1.0, 'recall': 0.006134969325153374, 'f1\_score': 0.012195121951219513, 'roc\_auc': 0.7929534504869848}

Epoch 9/10

Train Loss: 0.6722 Val Loss : 0.6761

Train Metrics: {'accuracy': 0.5518382352941177, 'precision': 0.8333333333333334, 'recall': 0.006122448979591836, 'f1\_score': 0.012155591572123177, 'roc\_auc': 0.7582252405979113}

Val Metrics : {'accuracy': 0.5235294117647059, 'precision': 1.0, 'recall': 0.006134969325153374, 'f1\_score': 0.012195121951219513, 'roc\_auc': 0.7929621156978961}

Epoch 10/10

Train Loss: 0.6745 Val Loss : 0.6760

Train Metrics: {'accuracy': 0.5518382352941177, 'precision': 0.8333333333333334, 'recall': 0.006122448979591836, 'f1\_score': 0.012155591572123177, 'roc\_auc': 0.7582743840010921}

Val Metrics : {'accuracy': 0.5235294117647059, 'precision': 1.0, 'recall': 0.006134969325153374, 'f1\_score': 0.012195121951219513, 'roc\_auc': 0.7930054417524522}

## 2 layer norm + affine + hidden 512

- 60 min / epoch  $\Rightarrow$  20 min / epoch
- Accuracy ok at 0.68 but lower than "2 layer norm + No affine + full data + hidden 300 "

Epoch 1/2

-----

Train Loss: 0.6890 Val Loss : 0.6274

Train Metrics: {'accuracy': 0.6691176470588235, 'precision': 0.6398450946643718, 'recall': 0.606938775510204, 'f1\_score': 0.6229576874738165, 'roc\_auc': 0.6979282642823016}

Val Metrics : {'accuracy': 0.6691176470588235, 'precision': 0.6836363636363636, 'recall': 0.5766871165644172, 'f1\_score': 0.6256239600665557, 'roc\_auc': 0.7012191951752105}

Epoch 2/2

-----

Train Loss: 0.6539 Val Loss : 0.6340

Train Metrics: {'accuracy': 0.6665441176470588, 'precision': 0.6492957746478873, 'recall': 0.5644897959183673, 'f1\_score': 0.6039301310043668, 'roc\_auc': 0.7003975155279503}

Val Metrics : {'accuracy': 0.6764705882352942, 'precision': 0.7054263565891473, 'recall': 0.558282208588957, 'f1\_score': 0.6232876712328768, 'roc\_auc': 0.7056601157672178}

### 0 layer norm + affine + hidden 512

- 60 min / epoch → 20 30 min / epoch
- Accuracy stuck at 0.62 - 0.64

```
Epoch 1/2
-----
Training: 100%|████████████████████████████████████████| 472/472 [49:22<00:00, 6.28s/it]
Evaluating: 100%|████████████████████████████████████████| 472/472 [12:58<00:00, 1.65s/it]
Computing Val Loss: 100%|████████████████████████████████████████| 59/59 [01:34<00:00, 1.60s/it]
Evaluating: 100%|████████████████████████████████████████| 59/59 [01:26<00:00, 1.47s/it]

Train Loss: 0.6597
Val Loss : 0.6513
Train Metrics: {'accuracy': 0.6364406779661017, 'precision': 0.5778534923339012, 'recall': 0.7804878048780488, 'f1_score': 0.6640563821456539, 'roc_auc': 0.7464141692553669}
Val Metrics : {'accuracy': 0.6220338983050847, 'precision': 0.5694444444444444, 'recall': 0.7509157509157509, 'f1_score': 0.6477093206951027, 'roc_auc': 0.7311794409586208}

Epoch 2/2
-----
Training: 100%|████████████████████████████████████████| 472/472 [49:03<00:00, 6.24s/it]
Evaluating: 100%|████████████████████████████████████████| 472/472 [10:25<00:00, 1.33s/it]
Computing Val Loss: 100%|████████████████████████████████████████| 59/59 [01:49<00:00, 1.86s/it]
Evaluating: 100%|████████████████████████████████████████| 59/59 [01:53<00:00, 1.92s/it]

Train Loss: 0.6541
Val Loss : 0.6440
Train Metrics: {'accuracy': 0.6372881355932203, 'precision': 0.5783214407067618, 'recall': 0.7832489645651174, 'f1_score': 0.6653635652853792, 'roc_auc': 0.750274408537805}
Val Metrics : {'accuracy': 0.6220338983050847, 'precision': 0.5694444444444444, 'recall': 0.7509157509157509, 'f1_score': 0.6477093206951027, 'roc_auc': 0.7313989900740689}

Training finished!
```

**Based on all the information above we see that**

- All the above training time could be a little faster if we run on colab or kaggle
- Layer Norm helps us to get better performance when added
- Affine Layer is needed when the embedding dimension is different from the hidden layer dimension
- Increase the hidden layer dimension does not improve the performance but increase the training time
- Training with the recreated vector is much faster

### Scaled-dot Product Attention

**Now we use Scaled-dot Product Attention instead of the 2 simple Attention**

- We run this with the best choice from above 4 configs: 2 layer norm + no affine + 300 hidden dimension
- 5-10 min / epoch (run on colab)
- accuracy of 1 epoch reach 0.72
- when try to use residual (mimic the Transformer architecture):

*residual\_page\_vec = webpage\_vec.sum(dim=1) # (B, D)*

*final\_vec = self.ln\_page(final\_vec + residual\_page\_vec)*

⇒ performance improve: accuracy 0.75

⇒ All to prove that this kind of attention is nice, but makes the training time increase much

Epoch 1/3

Epoch time: 4.6253204345703125e-05 seconds

Train Loss: 0.6298 Val Loss : 0.5391

Train Metrics: {'accuracy': 0.7213235294117647, 'precision': 0.7424714434060229, 'recall': 0.5836734693877551, 'f1\_score': 0.6535648994515539, 'roc\_auc': np.float64(0.7843749914681591)}

Val Metrics : {'accuracy': 0.7235294117647059, 'precision': 0.7716535433070866, 'recall': 0.6012269938650306, 'f1\_score': 0.6758620689655173, 'roc\_auc': np.float64(0.8197636130463416)}

Epoch 2/3

Epoch time: 368.7835772037506 seconds

Train Loss: 0.5784 Val Loss : 0.5274

Train Metrics: {'accuracy': 0.725, 'precision': 0.7016060862214708, 'recall': 0.6775510204081633, 'f1\_score': 0.6893687707641196, 'roc\_auc': np.float64(0.7887755102040815)}

Val Metrics : {'accuracy': 0.7441176470588236, 'precision': 0.740506329113924, 'recall': 0.7177914110429447, 'f1\_score': 0.7289719626168224, 'roc\_auc': np.float64(0.8238145991473432)}

Epoch 3/3

Epoch time: 740.161591053009 seconds

Train Loss: 0.5715 Val Loss : 0.5242

Train Metrics: {'accuracy': 0.7270220588235294, 'precision': 0.7100565955594254, 'recall': 0.6657142857142857, 'f1\_score': 0.6871708447440489, 'roc\_auc': np.float64(0.7908879257388574)}

Val Metrics : {'accuracy': 0.75, 'precision': 0.7516129032258064, 'recall': 0.7147239263803681, 'f1\_score': 0.7327044025157232, 'roc\_auc': np.float64(0.825105715573117)}



## Additional Analysis:

When training the Dual Attention model using the **original uncleaned “text”** column, the keywords the model extracted contained numbers and a lot of irrelevant information, as shown below:

```
text \
40938 leaders|leadership|chief|president|executives|biopharmaceuticals|chairman|relations|biopharmaceu...
8713 2021|closing|2118|2023|2022|2024|solutionsservice|close|technology|technologies|2020|technologic...
54187 mortgage|mortgages|loans|2023|borrowers|loan|2024|buydown|2022|23|lending|credit|homeowners|home...
42890 ニュース|世界|global|グローバル|ヌビーン|お知らせ|沿っ|閲覧|イン|内容|ニーズ|向け|続け|サイト|居住|アクセス|在住|
続行|対象|遵守|における|あらかじめ|ウェブサイト|過...
11862 cancers|cancer|investors|contact|pipeline|rights|monocarboxylate|patients|copyright|development|...
3168 laboratories|oncocyte|92618|infooncocytecom|publications|949|2024|clinical|7600|strategic|partne...
10598 fuel|global|aviation|zoom|page|wing|fuels|sustainable|biofuels|renewables|automatic|ready|tailwi...
14776 technologies|technology|drilling|engineering|drill|oilfield|drillscan|failures|downhole|related|...
45293 cookie|cookies|conditions|terms|policies|casscassinfoom|websites|agreement|consent|website|brow...
15428 gambling|gaming|play|card|2023|entertainment|2777|policy|privacy|cookie|2021|copyright|corruptio...

sents \
40938 oh|cano|cox|peggy|jeff|vaccine|gmbh|biotech|drug|ryan|paul|filings|cpg|sec|diseases|janssen|phd|...
8713 november|volvo|uly|iii|18|financing|automotive|20|shareholder|credit|liabilities|xii|19|evs|603|...
54187 tech|investors|sale|loan|financing|rentals|45|credit|trulia|payments|loans|mortgages|sell|hotpad...
42890 llc|assets|ceo|deutschland
11862 cyt|oncology|cancer|drug|therapeutics|monocarboxylate|ovarian|cancers
3168 laboratories|clinical|cushing|lab
10598 oard|racing|forma|11|150|ow|400|plants|xcf|fuel|nevada|plant|carbon|75|york|oil|stanton|ny|reno|...
14776 investors|hps|wellbore|efficiency|hp|oil|gas|technology|pro|technologies|health|prevention|drill...
45293 st|owned|instituted|thereon|contract|barter|liabilities|fitness|systems|auto|agents|permanently|...
15428 351|mga|2019|tax|rn|2020|tc|19|b2b|card|2022|54|106|2021|burton|https|anti|107|128|44|uk|december
```

Keywords extracted **after cleaning the “text” column** and applying the Dual Attention model:

	text \
17065	california transparency chain chains trafficking foods slavery laws disclosures resources busine...
19025	sensors sensor flexpoints flexpoint innovative silicon new digital valley industry humidity tech...
6538	press relations life loan insurance reporting contact links loss privacy salt policy lake screen...
8656	cookie cookies flexshares selection pdf optional choices click choose website preferences clicki...
17479	disabilities accessibility patients text accessible blog non healthcare keyboard usability guide...
11621	laser industrial industries industry industrys shipbuilding photonics nuclear family beam relati...
8480	investors investments investor investment funds invest investing shares riskier stock risk risks...
15257	relations brochures brochure relational dfw cloud lh documentation css tn family contact better ...
6963	shareholder shareholders investors energy invest invests dividends investment dividend gas stock...
20474	toys games shopping gamestop game gaming store menu shop return returns search retro playstation...

	sents \
17065	available careers laws nongovernmental corporation product independent able foods brands labor b...
19025	include therapy toys west gloves innovative representatives bachelors bend horn products systems...
6538	city contact loan lake road insurance corporation
8656	pdf etf transcript prospectus works personalised help fund technologies
17479	careers labs logos key healthcare diagnostics structures certified laboratory professionals assi...
11621	welding oem aerospace machinery designed careers florida engraving located equipment industries ...
8480	include etf available vehicles approved built authorized pursue tuned utilized seeking diversifi...
15257	shop realty copyright ebook mc careers projects architects buildings lh facades trunk operable h...
6963	west subsidiaries properties headquartered dedicated canton owns acquired llc purchase ri suite ...
20474	clothing shop store france generations electronic playstation electronics nfl careers toys vinyl...

## Section 3: Transformation Matrix

### 3.1. Progress

- We explore **how to run and create** the transformation matrix based on the sample code provided
- Explore some **minor errors and missing** in the code
- Our analysis about the **current implementation**
- **Our guess** about the **reason for not getting good performance** when we entering a query to get similar patents and companies that tends to have those patents using those trained Linear Transformation Matrices A and B

### 3.2. Analysis

#### *Minor Errors*

#### Step 5: Prepare training data

- Creates training input (X\_all) and output (Y\_all) matrices for companies with matched data.
- Then splits them into training and validation sets, and converts them into PyTorch tensors.

```
for country, info in data_info.items():
    X_all = np.stack([patent_rep_dict[i] for i in shared_ids])
    Y_all = np.stack([product_rep_dict[i] for i in shared_ids])

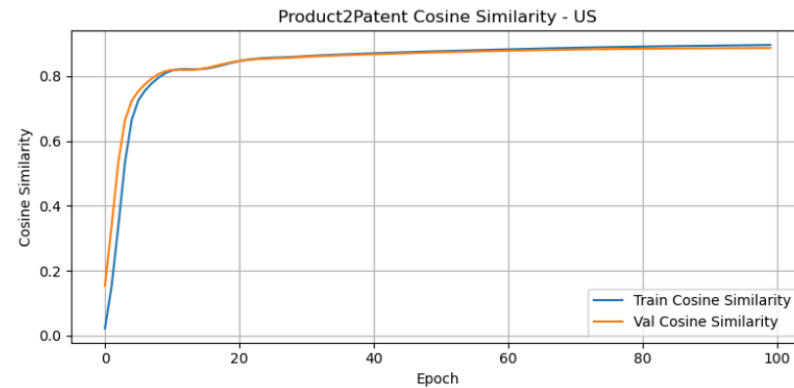
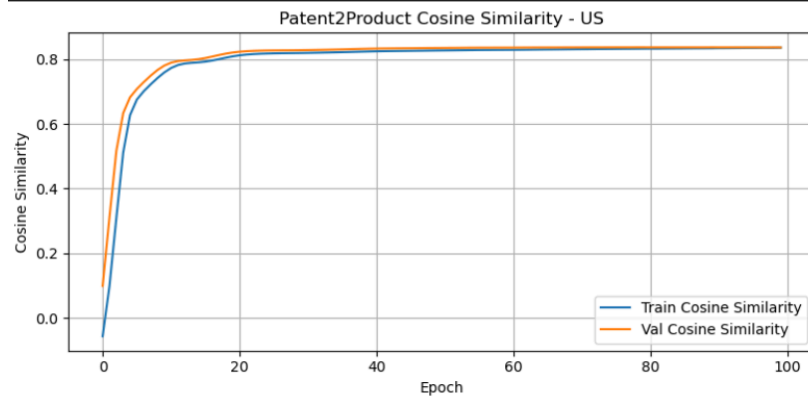
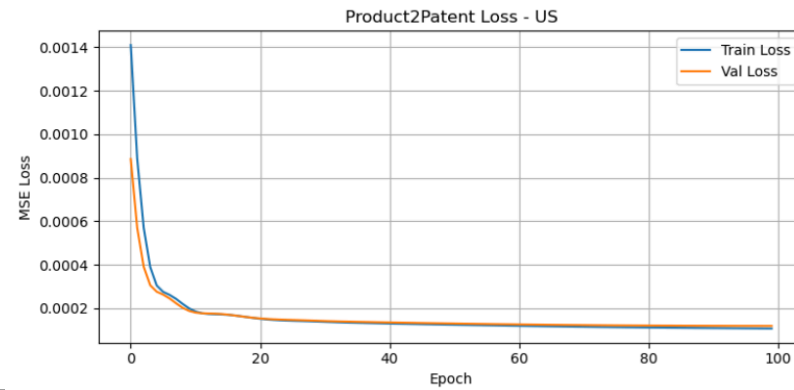
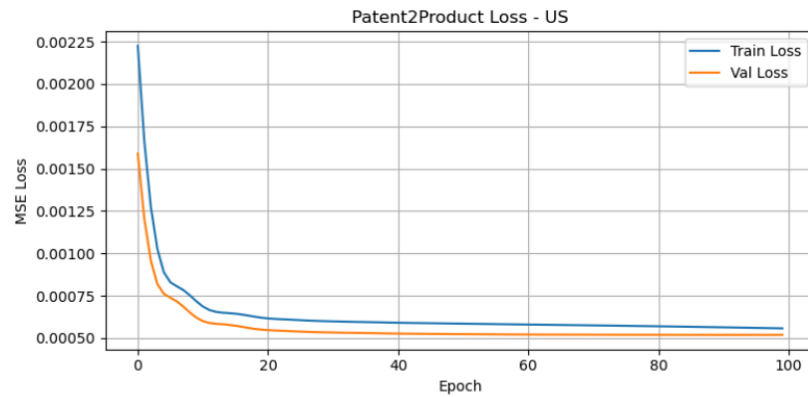
    X_train, X_val, Y_train, Y_val = train_test_split(X_all, Y_all, test_size=0.2, random_state=42)
    X_train = torch.tensor(X_train, dtype=torch.float32)
    X_val = torch.tensor(X_val, dtype=torch.float32)
    Y_train = torch.tensor(Y_train, dtype=torch.float32)
    Y_val = torch.tensor(Y_val, dtype=torch.float32)
```

- Although the code seems to be separate between each country, they are not.
- Currently all those loops just loop over each country (US JP CN) but assign the same thing / value to all of them making the result the same (we can see this when looking at the training loss, loss plot, and cosine similarity after training)
- And more, sample code just provides the way to compute matrix A from Patent to Product and missing the other

## Current Implementation

- Use the Dual Attention Model previously to provide the keywords that attends to make the company embedding
- Conduct the company embedding by taking the average of all the keyword embeddings  
 ⇒ each company has one company vector
- Group the patents dataframe by the hojin\_id (company id)
  - + So each company would have only one vector patent
  - + Collect all the tokens (keywords) from all the patents of that company
  - + Then like above, we just take the embeddings of each token, then averaging them all to provide the embedding of patent of that company (each company has one patent vector)
- Then, we create a simple network with Linear Layer 1 → ReLU → Linear Layer 2
- Train the model with X is the patent vector + Y is the company vector + Loss is the MSE
- Then the matrix A is calculated by taking Weights of Linear Layer 2 x Linear Layer 1 (no consider about the Bias)
- The same thing works for matrix B

## Our Analysis



⇒ We can see that the Loss and the Cosine Similarity generalize well

**By using the method of multiplying the 2 weights of the 2 linear layers to make the matrices A and B and get something like this**


### Analysis A

We conduct a **sample function to allow entering the query**, then we get **similar patents** to the query and **similar companies**

- **query** → tokens → average → **market vec**  
→ multiply with B → **predicted\_tech\_field\_vec** → calculate cosine similarity → **top-k patents**
- **predicted\_tech\_field\_vec**  
→ multiply with A → **predicted\_market\_field\_vec** → calculate cosine similarity → **top-k companies**

When we try with the query ***“methods for directing waste gases”***

We get some output like this

 Top-k similar field-level patents (firm-level):

1. Firm ID: 701882, Name: TECHNOL FUEL CONDITIONERS INC, Cosine Sim: 0.2306

Showing 1 patent(s):

- appln\_id: 54353267, preview: **heavy oil emulsion** comprising decant oil oil water stabilized adding certain ester preferably ester saccharide ester **gallotannins** saponin red gum

2. Firm ID: 704444, Name: Renovare Environmental, Cosine Sim: 0.1755

Showing 3 patent(s):

- appln\_id: 420880808, preview: method manufacturing **reforming fuel adding water fuel oil** manufacturing apparatus provided method **apparatus comprising preprocessing water water tank** aerating **water**

- appln\_id: 422737268, preview: provided **apparatus** manufacturing reformed fuel **apparatus** manufacturing reformed fuel includes **water** tank unit configured supply **water** aerated ultrasonic **wave** electric

- appln\_id: 443390442, preview: provided reformed **fuel** manufacturing **apparatus** including one **water** tank configured pretreat introduced **water** using **water** tank catalyst one oil tank

3. Firm ID: 702189, Name: Greenbelt Resources Corporation, Cosine Sim: 0.1638

Showing 1 patent(s):

- appln\_id: 423199576, preview: **system using membrane separating desired component liquid gas mixture** obtain organic **solvent** plant fermenting **biomass** feedstock corn sugarcane **produce ethanol**

4. Firm ID: 705580, Name: Medical Marijuana Inc., Cosine Sim: 0.1561


Showing 3 patent(s):

- appln\_id: 448179853, preview: method producing hemp oil comprising decarboxylation cbda hemp oil evaporation cbd decarboxylated hemp oil produce cbd oil selective thc cbn
- appln\_id: 474399202, preview: method producing hemp oil comprising decarboxylation cbda hemp oil evaporation cbd decarboxylated hemp oil produce cbd oil selective thc cbn
- appln\_id: 486252916, preview: one aspect method treating smokeless tobacco addiction comprising administering individual need thereof pharmaceutical composition comprising nicotine therapeutically effective amount cannabidiol

5. Firm ID: 706329, Name: ALTERNATIVE ENERGY DEVELOPMENT CORPORATION, Cosine Sim: 0.1452

Showing 1 patent(s):

- appln\_id: 49548142, preview: system production fuel gas solid biomass fuel combustion said fuel gas disclosed comprises gasification zone producing fuel gas solid biomass

 Top-k firms likely aligned with your product:

1. Firm ID: 703619, Name: MUCINNO HOLDING INC, Cosine Sim: 0.2795
2. Firm ID: 701731, Name: VISION SENSING ACQUISITION CORP, Cosine Sim: 0.2579
3. Firm ID: 700368, Name: YUM CHINA HOLDINGS, INC., Cosine Sim: 0.2286
4. Firm ID: 704054, Name: Cactus Wellhead, Cosine Sim: 0.2129
5. Firm ID: 700261, Name: Powell Industries, Cosine Sim: 0.2127

But when we try with another query “**computer vision and machine learning**”

The output looks like

🔍 Top-k similar field-level patents (firm-level):

1. Firm ID: 706434, Name: Voip-Pal.com, Cosine Sim: -0.0878

Showing 1 patent(s):

- appln\_id: 502209909, preview: facilitating uninterrupted transmission internet protocol ip transmission endpoint change disclosed ip transmission received first relay port second relay port call

2. Firm ID: 700415, Name: Wialan Technologies Inc, Cosine Sim: -0.1149

Showing 1 patent(s):

- appln\_id: 46865588, preview: network based method enhances handshake client virtual private network vpn server internet protocol ip address assignment client tunnel done existing

3. Firm ID: 704270, Name: CBD OF DENVER, INC., Cosine Sim: -0.1200

Showing 3 patent(s):

- appln\_id: 46526537, preview: compound formula wherein q co ch cr oh cr oc alkyl r h x hydroxyalkyl c p w sum p

- appln\_id: 48553929, preview: barricade light retrofitted receive light emitting diode circuit circuit includes pulse activated switch light sensitive switch operation

- appln\_id: 53740788, preview: potent tumor inhibitor prepared compound formula wherein q co ch oh c oh halogen halogen hydrogen c one two p

4. Firm ID: 700679, Name: PaxMedica, Cosine Sim: -0.1258

Showing 1 patent(s):

- appln\_id: 576286576, preview: present invention provides composition method treating cognitive social behavioral disability neurodevelopmental disorder autism spectrum disorder asd central nervous system disorder

5. Firm ID: 705299, Name: NeoVolta, Cosine Sim: -0.1280


Showing 3 patent(s):

- appln\_id: 550030272, preview: adaptive solar power battery storage system disclosed capture alternative energy use desired regardless power generating circuit topology ac dc adaptive



- appln\_id: 577873874, preview: dc supply circuitry disclosed supply dc power ac source dc pv panel input associated pv energy storage system es inverter

- appln\_id: 588838656, preview: adaptive solar power battery storage system disclosed capture alternative energy use desired regardless power generating circuit topology ac dc adaptive

 Top-k firms likely aligned with your product:

1. Firm ID: 703023, Name: CARPARTS.COM INC, Cosine Sim: 0.4250
2. Firm ID: 705371, Name: Littelfuse, Cosine Sim: 0.4221
3. Firm ID: 704134, Name: Advance Auto Parts, Cosine Sim: 0.4159
4. Firm ID: 703924, Name: Qorvo, Cosine Sim: 0.4121
5. Firm ID: 705724, Name: Luna Innovations, Cosine Sim: 0.4111

⇒ Negative values occurs ⇒ Maybe because there are no vectors in our training that aligns with the query

⇒ And our thoughts is that maybe because we only use the weights and ignore the bias to estimate the linear transformation between patents vectors with product vectors, so maybe we will “miss” some information provided by the bias or maybe the relationship is not linear

## Analysis B

- From our above thoughts, we now instead of using the 2 matrices to produce the estimated vectors, we now use the 2 models that are trained to produce those vectors
- And now the cosine similarity is much higher. BUT the result seems not good at all, just the score is higher and seems better

### Query: “*methods for directing waste gases*”

🔍 Top-k similar field-level patents (firm-level):

1. Firm ID: 700464, Name: NL Industries, Cosine Sim: 0.9312

Showing 3 patent(s):

- appln\_id: 45415606, preview: shock absorbing subassembly use oil well drilling string drilling bit absorb reduce bit induced vibration impact load wherein shock absorber
- appln\_id: 45454997, preview: improved apparatus releasing perforation gun tubing string well casing disclosed explosion detonator fire perforation gun open previously sealed passageway wall
- appln\_id: 45465326, preview: system disclosed effecting movement one operating device capper chuck predetermined endless path straight line reach parallel conveyor container like may

2. Firm ID: 703191, Name: Marathon Oil Corporation, Cosine Sim: 0.9289

Showing 3 patent(s):

- appln\_id: 45548132, preview: sealing mean coke oven chuck door sealing mean includes metal sealing member springiness flexibility supported cantilever fashion inside door frame
- appln\_id: 45768694, preview: group viii metal salt iridium chloride rhodium chloride used catalyst condensation reaction aldehyde produce acetal alpha aldehyde catalyst also useful
- appln\_id: 45825680, preview: installation jacket substructure component offshore platform sea floor underwater fixture containing one wellhead jacket first ballasted rest vertical orientation sea

3. Firm ID: 702078, Name: DEVON ENERGY CORP, Cosine Sim: 0.9286

Showing 3 patent(s):

- appln\_id: 45415482, preview: method apparatus discharging freshly reduced material shaft furnace controlling release spent reducing gas hazard apparatus includes pair surge bin mean

- appln\_id: 45768381, preview: apparatus provided singly discharging laterally aligned cylindrical workpiece lower end inclined rollway rocker pivot intermediate end disposed beneath rollway stop

- appln\_id: 45965412, preview: metal tying wire product alkali coating weighing least wire surface coating baked temperature range essentially physically adsorbed water driven achieve

4. Firm ID: 702096, Name: WESTLAKE CORP, Cosine Sim: 0.9283

Showing 3 patent(s):

- appln\_id: 46136739, preview: invention provides process making subjecting ethylene oxychlorination reaction hydrogen chloride gas containing molecular oxygen gas phase elevated temperature contact fluidized

- appln\_id: 46330725, preview: converter catalytic conversion exhaust gas internal combustion engine comprising housing outer shell spaced apart inner shell two shell slidable respect

- appln\_id: 46340782, preview: coupler used coupling plastic conduit separation building plastic end region overlapped accurately positioned sealed coupling plastic conduit end coupler plastic


5. Firm ID: 700995, Name: Unknown, Cosine Sim: 0.9279

Showing 3 patent(s):

- appln\_id: 45803454, preview: interconnector filtration module comprises inner outer integrally bonded coaxial rubber sleeve inner sleeve comprises rubber lower indentation hardness outer sleeve

- appln\_id: 46478024, preview: interconnector coaxially adjacent permeate tube filtration apparatus comprises molded elastomeric spheroidal body opposed coaxial bore receiving tube end extend filter

- appln\_id: 48280658, preview: two stage blender capable handling high viscosity mixture mixture discharged first stage directed second stage discharged relatively high discharge pressure

 Top-k firms likely aligned with your product:

1. Firm ID: 706585, Name: ADS TACTICAL INC, Cosine Sim: 0.9666

2. Firm ID: 703430, Name: Proto Labs, Cosine Sim: 0.9624

3. Firm ID: 701258, Name: Smith-Midland, Cosine Sim: 0.9613

4. Firm ID: 705108, Name: Global Digital Solutions, Cosine Sim: 0.9599

5. Firm ID: 701980, Name: WENTWORTH ENERGY, INC., Cosine Sim: 0.9597

## Query: “**computer vision and machine learning**”

### Get output

 Top-k similar field-level patents (firm-level):

1. Firm ID: 703790, Name: Horizon Technology Finance, Cosine Sim: 0.9803

Showing 3 patent(s):

- appln\_id: 462877, preview: impact mill including base portion disposed **rotor** rotatably mounted bearing housing rotor upwardly aligned cylindrical surface portion coaxial rotational axis

- appln\_id: 45531807, preview: present invention generally concern use bragg optical fiber chirped pulse amplification **system** production ultrashort optical pulse bragg optical fiber waveguide

- appln\_id: 45856168, preview: method apparatus dynamically transparently renewing license associated downloaded content licensing associated content allow provider retain control content downloaded **user computer**

2. Firm ID: 700524, Name: Ares Capital Corporation, Cosine Sim: 0.9780

Showing 3 patent(s):

- appln\_id: 461950, preview: flow restrictor provided reduce pressure flow fluid fuel flow restrictor take form capillary void abrupt flow disruption flow restrictor may

- appln\_id: 466824, preview: **digital** right management **audience** measurement system method disclosed example method includes receiving request **upload** medium **content** content distributor attempting obtain

- appln\_id: 467363, preview: **electronic device** prior entering **distribution channel** equipped loss prevention **client** permit limited use **device** correct authentication provided legitimate purchaser permitting

3. Firm ID: 705575, Name: Hercules Capital, Cosine Sim: 0.9778

Showing 3 patent(s):

- appln\_id: 467615, preview: set source point represent stroke input user identified set source point may refined modified set source point may stored **memory**

- appln\_id: 467646, preview: disclosed method system detecting boundary area different **language body text**

- appln\_id: 45457949, preview: resonant tunneling diode one dimensional **electronic** photonic structure electromechanical mem device formed heterostructure nanowhisker forming length segment whisker different material


4. Firm ID: 700678, Name: Wells Fargo, Cosine Sim: 0.9763

Showing 3 patent(s):

- appln\_id: 463630, preview: antenna adapted **wireless network** variably controlled stagger antenna **array architecture** disclosed antenna array contains plurality driven radiating element spatially arranged
  - appln\_id: 466769, preview: **tool technique** provided allow partner analytics provider others use submit executable plug in **analytics** provider others may also write certified
  - appln\_id: 467661, preview: method system providing cost estimation **connection** water distribution **network** provided method includes estimating energy cost within water distribution network associated
5. Firm ID: 701749, Name: Bank of America, Cosine Sim: 0.9759

Showing 3 patent(s):

- appln\_id: 461822, preview: mattress foundation perimeter structure frame spring mounted upon frame grid supported spring perimeter structure perimeter frame extends frame grid forming
- appln\_id: 462149, preview: cap plurality rib **structure** situated inner wall cap facilitate formation air passageway cap penetrated pipette tip air passageway aid venting
- appln\_id: 462246, preview: present invention generally relates method preparing stable colloidal dispersion nanoparticulate size fumed silica particle particularly invention relates method rapidly wetting

 Top-k firms likely aligned with your product:

1. Firm ID: 703655, Name: STARTENGINE CROWDFUNDING INC, Cosine Sim: 0.9725
2. Firm ID: 703692, Name: Information Services Group, Cosine Sim: 0.9715
3. Firm ID: 701093, Name: VIVEON HEALTH ACQUISITION CORP, Cosine Sim: 0.9693
4. Firm ID: 705108, Name: Global Digital Solutions, Cosine Sim: 0.9680
5. Firm ID: 701510, Name: ServiceNow, Cosine Sim: 0.9676

### Analysis C: Applying Transformer Embeddings

Fasttext is a **purely static, subword-aware token-level embedding**, so our assumption was that perhaps using a more advanced embedding for our training process might produce better results, so we looked into **Transformer-based embeddings** and used them to train, and below are some of our findings.

NOTE: **Transformers are contextual models**. They're trained on sentences, paragraphs—i.e., *sequences with grammar and order*. In the case of using Transformer embeddings on the patent data, it will most likely produce great results as the data are already **structured in meaningful sentences**; however, applying Transformer embeddings to web data to generate relevant keywords has to be done in a cautious manner, since the **Dual Attention Model functions purely on word-level tokens and the words themselves are isolated keywords** that hold **NO contextual value** to adjacent words. Despite this, we decided to experiment with using Transformer embeddings, specifically **Sci-BERT**, which works well for both general English and scientific/patent text, and store all the word embeddings in a .vec dictionary format (similarly to the configuration in ms. Xinyu's Dual Attention notebook).

Info about Sci-BERT embeddings:

- The **Embedding size is 768d** (compared to Fasttext's 300d). => preprocessing and storage overhead.
- We set the **hidden dim size to 300** as well; this means that during the training process, the tokens would get mapped from 768d -> 300d, which can help speed up the training process.
- However, BERT embeddings historically outperforms static-style embeddings like Fasttext. => Massively better semantic grounding.

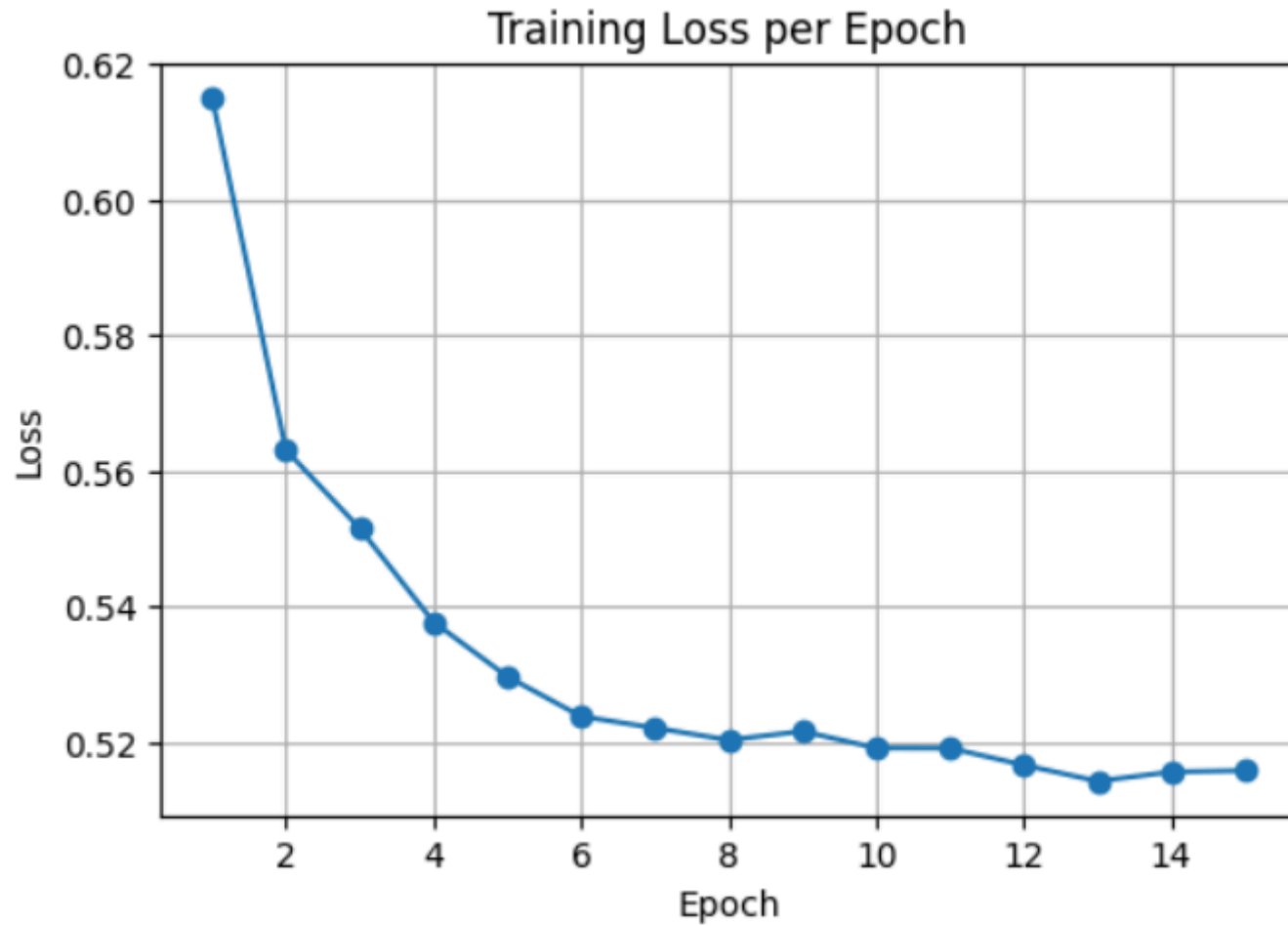
```
Epoch 1/20
  Train loss: 0.6151
  Train Acc: 0.7076, Prec: 0.7499, Rec: 0.5391, F1: 0.6272
  LR now: 0.003000
-----
Epoch 2/20
  Train loss: 0.5631
  Train Acc: 0.7312, Prec: 0.7458, Rec: 0.6237, F1: 0.6793
  Val Acc: 0.7317, Prec: 0.7350, Rec: 0.6271, F1: 0.6768
  LR now: 0.003000
```

Right at the beginning, it already looks like there is an immediate improvement, as after 2 epochs, the performance metrics seem to already be around ~0.73.

After training for **15 epochs**:

```
Epoch 15/15
  Train loss: 0.5158
  Train Acc: 0.7634, Prec: 0.7454, Rec: 0.7315, F1: 0.7383
  Val Acc: 0.7443, Prec: 0.7203, Rec: 0.7013, F1: 0.7107
  LR now: 0.000375
```

Training loss across 15 epochs:



Despite that the training loss decreases stably throughout the training process, the training accuracy as well as the loss accuracy seems to converge at  $\sim 0.75$ .



=> Our experiments suggest that BERT embeddings are, in fact, more effective than Fasttext embeddings since we can see the accuracy at the first few epochs are quite high (in comparison to the first few epochs of models trained on Fasttext embeddings). However, both embedding training seem to result in the model converging at around 0.75 validation accuracy, which indicates that we might need to inspect our data more carefully.

Further testing should involve using the Transformer embeddings for the patent data and the Transformation Matrix.

## Overall

- I think that we understand and be able to ourselves know the pros and cons of our system
- We think that the main point is the current Dual Attention Model they lack of information like the order of the words in the page, for example when we consider about the Transformer architecture, there is a component call Positional Encoding, which encodes the information about the order of the words at the input phase and plug them into the Embeddings of the input by the Residual Connection
- Another thing is that, we think that because we currently use the FastText for the embedding phase, this makes our query and also the results of the embeddings when we conduct the cosine similarity relies on TOKENS-level. Like you can see from the previous sections where we show the output analysis of our current methods, although the patent is not related to the query, but if that patent abstraction contains a lot of words that somehow related to the query, so it would still be consider “similar”

### ⇒ Our thoughts and recommendations are that

- We should use other kinds of Embedding techniques that do not depend on the TOKEN-level anymore. Maybe Transformer Embeddings for Sentence-level Embeddings or some other pretrained Embeddings.
- Also, we should consider about another type of RELATION between the Company and Patents
- MISSING CONTEXT or EXTERNAL DATA should be considered also because all the works we currently do are just based on EXTRACTING KEYWORDS, which makes us only be able to use the TOKEN-level EMBEDDING only.