CMPE 493 Introduction to Information Retrieval
Fall 2021
Term Project
Text Classification for COVID-19 Scientific Literature

Adalet Veyis Turgut, Ufuk Arslan, Zuhal Didem Aytaç

Project Description

In the scope of this project, we performed multilabel topic classification for COVID-19 scientific literature. The developed system assigns the articles in one or more of the following topic classes: Treatment, Diagnosis, Prevention, Mechanism, Transmission, Epidemic Forecasting, and Case Report.

We used the BC7-LitCovid-Train.csv dataset (24,960 articles) to train our system and BC7-LitCovid-Dev.csv (6,239 articles) to test our system.

Results will be shared in the upcoming slides.

Previous Progress & Baseline Approach

For the mid-report, we completed the preprocessing of the data and implemented a baseline approach (KNN).

- **Preprocessing**: We performed preprocessing for both train and test datasets. We applied the following steps to columns *title, abstract, keywords*:
  - **Preparation**
    - We dropped the rows that does not contain any value for *title, abstract, keywords* columns.
    - We split the keywords and label columns with respect to semicolons.
  - **Case-fold, punctuation removal**
    - We applied casefold on article strings and removed the punctuation using *string.punctuation*
  - **Stemming**
    - We applied stemming using *nltk.PorterStemmer()* to reduce the words to root forms.
  - **Stop-word removal, tokenization**
    - We removed the stopwords using *nltk.corpus* library. We tokenized the strings.
  - **Frequent word removal**
    - We detected the frequent words within each column using *nltk.FreqDist()*. We removed the 10 most frequent words.
- **Count Vectorizing**
- **Tdif Transforming**
- **Fitting to The Model & Prediction**
- **Evaluation of Results**

Train dataset description before preprocessing

	journal	title	abstract	keywords	pub_type	authors	doi	label
count	24960	24960	24960	18968	24960	24859	24406	24960
unique	3800	24913	24893	18824	339	24545	24388	183
top	J Med Virol	Crucial laboratory parameters in COVID-19 diag...	Null.	covid-19;sars-cov-2	Journal Article	Suwanwongse, Kulachanya;Shabarek, Nehad	10.3785/j.issn.1008-9292.2020.02.03	Prevention
freq	300	2	12	7	12182	5	2	9046

Train dataset description after preprocessing

	journal	title	abstract	keywords	pub_type	authors	doi	label
count	18968	18968	18968	18968	18968	18912	18759	18968
unique	3014	18895	18932	18595	270	18704	18749	165
top	J Med Virol	pregnanc	2019 respiratori tract caus newli emerg first ...		Journal Article	Suwanwongse, Kulachanya;Shabarek, Nehad	10.1016/j.ijantimicag.2020.106020	[Prevention]
freq	298	4	3	38	9372	5	2	6756

Test dataset description before preprocessing

	journal	title	abstract	keywords	pub_type	authors	doi	label
count	6239	6239	6239	4754	6239	6212	6100	6239
unique	2132	6239	6234	4742	170	6194	6100	103
top	J Med Virol	Daring discourse: are we ready to recommend ne...	Abstract:	2019 novel coronavirus;antiviral therapy;infec...	Journal Article	Siddiqui, Ruqaiyyah;Khan, Naveed Ahmed	10.2214/AJR.20.23415	Prevention
freq	103	1	3	2	3056	3	1	2256

Test dataset description after preprocessing

	journal	title	abstract	keywords	pub_type	authors	doi	label
count	4754	4754	4754	4754	4754	4737	4707	4754
unique	1644	4751	4751	4714	141	4723	4707	88
top	J Med Virol	triag consider refer structur heart intervent ...	everi month dtb scan sourc inform treatment ma...		Journal Article	Siddiqui, Ruqaiyyah;Khan, Naveed Ahmed	10.2214/AJR.20.23415	[Prevention]
freq	103	2	2	7	2378	3	1	1713

Baseline Approach (KNN)

- We built count vectors for the *title, abstract, keywords* columns of the train and test datasets using sklearn *CountVectorizer()*. The train count vector has shape (18968, 64807) and the test count vector has shape (4754, 64807).
- We transformed the count vectors to normalized tf-idf vectors using sklearn TfidfTransformer().
- We interpreted the *label* columns of the train and test datasets using sklearn MultiLabelBinarizer(). The binarized train label column is of shape (18968, 7) and the binarized test label column is of shape (4754, 7), as expected.
- Our model is built as MultiOutputClassifier(KNeighborsClassifier()). We chose to use KNN and wrapped it with MultiOutputClassifier as we intend to fit each article to multiple classes.
- We fit the train dataset to our model.
- We made the model predict the labels for the dev dataset.

Baseline Approach - Results

LABEL BASED MEASURES

	precision	recall	f1-score	support	# of words	# of docs
Treatment	0.8024	0.8043	0.8033	1681	927216	6710
Diagnosis	0.7487	0.7210	0.7346	1190	672208	4695
Prevention	0.8831	0.8590	0.8709	2085	1108409	8300
Mechanism	0.8235	0.7271	0.7723	828	456426	3518
Transmission	0.5448	0.3946	0.4577	185	108847	841
Epidemic Forecasting	0.7266	0.6433	0.6824	157	67910	519
Case Report	0.6831	0.3434	0.4570	364	145382	1571
micro avg	0.8108	0.7553	0.7821	6490		
macro avg	0.7446	0.6418	0.6826	6490		
weighted avg	0.8053	0.7553	0.7763	6490		
samples avg	0.8014	0.7819	0.7754	6490		

INSTANCE BASED MEASURED

mean precision	0.8014
mean recall	0.7819
F1	0.7915

Result evaluation and differences with biocreative

We have ~0.8 scores overall which we found great considering we just used a simple KNN model.

When we look at to the classes separately, we see that classes with less documents have lower scores. Reason is clear: algorithm couldn't develop a good estimator with relatively smaller data.

We ran the KNN with k=3,5,7. Best result was when k=5. Others were really close though, differences were ~0.02.

We tried different combinations of title, keyword and abstract columns. In the end, we trained the model with the concatenation of these three columns.

Our own evaluation script has relatively high scores compared to the script prepared by biocreative.

Main reason is that they compare class probabilities with actual results, whereas we compare class predictions with actual results. In short, we did not look at fractional class probabilities between 0 and 1, rather we looked at final predictions as 0 or 1.

We tried to obtain fractional probabilities from the model, but predict_proba() method of sklearn was not returning correct results.

BERT

- We used **BERT** (Bidirectional Encoder Representations from Transformers) to train our dataset
- **BERT** is a transformer-based machine learning technique used in natural language processing (NLP)
- **Preparation**
  - We encoded the labels as different columns
  - We tokenized the texts with **Bert-Tokenizer**.
  - We set MAX_TOKEN_COUNT = 512, N_EPOCHS = 3, BATCH_SIZE = 10
  - We created class *AbstractDataset* that inherits from PyTorch Dataset; with attributes data, tokenizer and max token count. We created an instance with train dataset.
  - We used *“bert-base-cased”* model
  - We created class *AbstractDataModule* that inherits from PyTorch Lightning Data Module; with attributes batch size, train dataframe, test dataframe, tokenizer and max token count. We created an instance with train and validation datasets.
  - We created the class *AbstractTagger* that inherits from PyTorch Lightning Module; with fields BERT model, classifier, training and warmup steps and criterion (nn.BCELoss())

BERT

- We calculated
  - steps_per_epoch=len(train_df) // BATCH_SIZE
  - total_training_steps = steps_per_epoch * N_EPOCHS
  - warmup_steps = total_training_steps // 5
- We created an instance of *AbstractTagger* with calculated warm-up steps and training steps.
- We created a trainer instance and called *trainer.fit(model, data_module)*, where model is the AbstractTagger instance and data_module is the *AbstractDataModule* instance defined above.
- We created another *AbstractDataset* instance with the validation dataset and made the model predict the classes.
- We saved the results to a csv file and ran the evaluation script.

BERT - Evaluation

LABEL BASED MEASURES

	precision	recall	f1-score	support
Case Report	0.8969	0.8846	0.8907	364
Diagnosis	0.8782	0.8908	0.8844	1190
Epidemic Forecasting	0.7895	0.6688	0.7241	157
Mechanism	0.9075	0.8527	0.8792	828
Prevention	0.9477	0.9300	0.9388	2085
Transmission	0.6515	0.6973	0.6736	185
Treatment	0.9031	0.8977	0.9004	1681
micro avg	0.9027	0.8891	0.8958	6490
macro avg	0.8535	0.8317	0.8416	6490
weighted avg	0.9031	0.8891	0.8958	6490
samples avg	0.9202	0.9146	0.9040	6490

INSTANCE BASED MEASURED

mean precision	0.9202
mean recall	0.9146
F1	0.9174

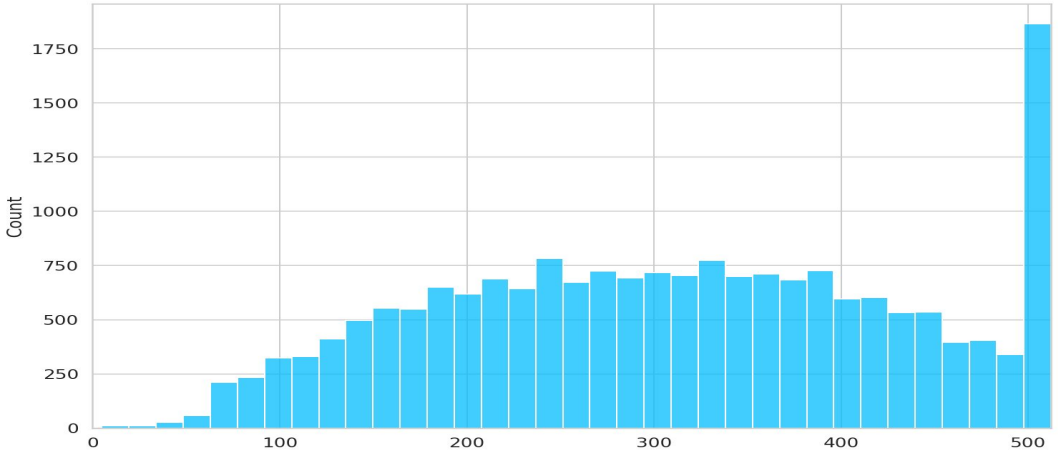
Evaluation

	BERT
mean precision	0.9202 (KNN: 0.8014)
mean recall	0.9146 (KNN: 0.7819)
F1	0.9174 (KNN:0.7915)
AUROC	
Treatment	0.9806
Diagnosis	0.9786
Prevention	0.9872
Mechanism	0.9823
Transmission	0.9733
Epidemic F.	0.9871
Case Report	0.9853

- 14.85% increase in mean precision
- 16.56% increase in mean recall
- 15.90% increase in F1
- BERT gave improved results for all parameters, however it is much slower in terms of runtime.

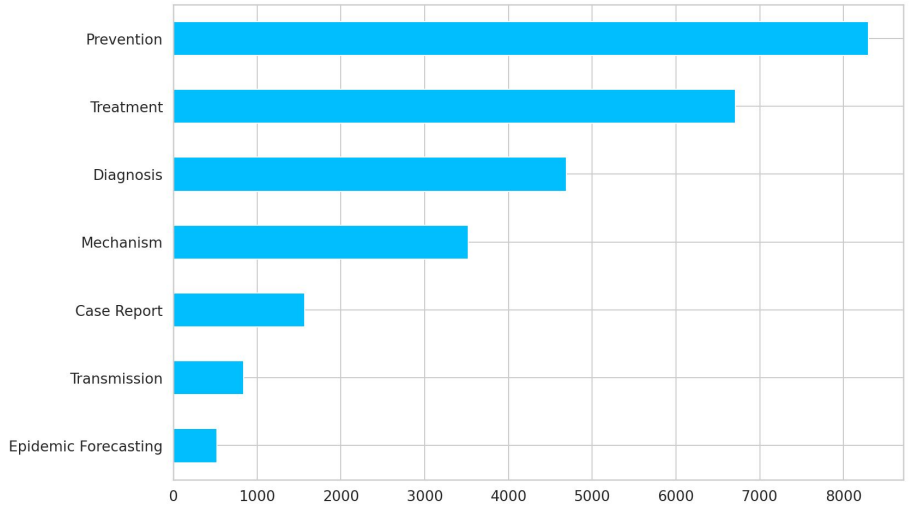
Possible Directions for Improvement

- BioBERT can used to increase the accuracy of our model, as BioBERT is a pre-trained biomedical language representation model designed for biomedical text mining that is perfect for our task.
    - You can find more information about BioBERT here.
- Since BERT has a limit of maximum token count as 512, we cut rest of the abstracts that are longer than 512 tokens. Instead, abstracts can be split into multiple subparts, classified separately and finally results can be combined back together.



Error Analysis

- Labels with relatively lower instances in the training dataset yields to poor results.
- Imbalance in the dataset biases the classifier towards the major class.



Q&A

- You can reach us for further details and questions.
  - Adalet Veyis Turgut
    - adalet.turgut@boun.edu.tr
  - Ufuk Arslan
    - ufukarslan99@gmail.com
  - Zuhal Didem Aytaç
    - zdaytac@gmail.com

References

1. Bird S, Klein E, Loper E. Natural language processing with Python: analyzing text with the natural language toolkit. " O'Reilly Media, Inc." 2009.

2. Pedregosa F, Varoquaux, Ga"el, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: Machine learning in Python. Journal of machine learning research. 2011;12(Oct):2825–30.

3. McKinney W, others. Data structures for statistical computing in python. In: Proceedings of the 9th Python in Science Conference. 2010. p. 51–6.

4. Harris CR, Millman KJ, van der Walt SJ, Gommers R, Virtanen P, Cournapeau D, et al. Array programming with NumPy. Nature. 2020;585:357–62.

5. arXiv:1910.03771

6. https://curiousily.com/posts/multi-label-text-classification-with-bert-and-pytorch-lightning/