**Model Documentation**

Competition Name: Coleridge Initiative - Show US the Data

Team Name: OsciiArt resistance0108 Naoto\_Usuyama

Public Leaderboard Score: 0.648

Private Leaderboard Place: 0.513

**Background**

Account Name: OsciiArt

Name: Osamu Akiyama

Location: Japan

Email: [oakiyama1986@gmail.com](mailto:oakiyama1986@gmail.com) or [spnxxxx@gmail.com](mailto:spnxxxx@gmail.com)

Occupation: medical doctor, 2nd year resident

Affiliation: Osaka University Hospital

Education:

2015-2020 M.D. Faculty of Medicine, Osaka University

2009-2012 M.S. Graduate School of Biostudies, Kyoto University

I have experienced research of epidemiology using Japanese government data that may have helped me in this competition. I decided to enter this competition because I’m interested in the automation of research. I spent about 1-2 hours/day on this competition. I did the leaderboard proving, the acronym detection, and the string matching with external data.

Account Name: resistance0108

Location: Japan

Email: [resistance0108@gmail.com](mailto:resistance0108@gmail.com)

Name: Jun Saitou

Occupation: Data Scientist

Affiliation: Accenture Japan

Education:

2015-2017 M.S. Faculty of Math, Tokyo Institute of Technology

2011-2015 B.S. Graduate School of Math, Tokyo Institute of Technology

I earned a gold medal at 8th in IEEE-CIS Fraud Detection competition. During that competition, I learned how to succeed at Kaggle. I have been a data scientist for 3 months and almost no ML career before. I entered this competition because this was the only competition that isn’t image-related and is more than 2 months to go. I joined this team because OsciiArt was seeking teammates and I wanted to team with him. I spent about 1-2 hours on weekdays and 1-7 hours on holidays. In this competition, OsciiArt was seeking the best way of string-matching. Naoto Usuyama helped OsciiArt incorporating NER with Spacy in string-matching model. I was building models independently from others and sought the possibility of sentence-based models, but none of my models were completed before the deadline due to lack of time.

Account Name: naotous

Name: Naoto Usuyama

Email: [usuyama.naoto@gmail.com](mailto:usuyama.naoto@gmail.com)

Occupation: Senior Applied Researcher

Affiliation: Microsoft Research, Healthcare

Education: MSc/BSc Computer Science, University of Tokyo

**Solution Overview**

Our solution is composed of 6 parts below.

* Leaderboard (LB) probing
* Acronym detection
* Acronym detection version 2
* String-matching with dataset-names from external data
* Dataset-name variation detection using NER
* String-matching with dataset-names from the train data

1. **Leaderboard (LB) probing**

The metric of this competition is F-score. Under this metric, assuming a current score is ***F***, a newly detected label can improve the score when the expected value that the label is true positive is greater than ***0.8F***. Therefore, it is important to estimate the private test score in order to determine the detection threshold. For example, if ***F*** is 0.6, the best threshold is 0.48 and if ***F*** is 0.4, the best threshold is 0.32. For this reason, it is very important to know the number of the training-data labels in the private test data, because this affects the private test score strongly.

To tackle this problem, we did LB probing. In this competition, the public test data contains duplicates of the train data. Therefore, we can create a submission only with true positive labels and with no false positive labels by applying true positive labels of the train data to their duplicates. Thus, by setting the number of true positive labels to a value related to the hidden test data, we can get information about the hidden test data from the submission score. Using this strategy, we got rough estimates of values below (actual codes are [this](https://www.kaggle.com/osciiart/lb-probing-3) and [this](https://www.kaggle.com/osciiart/lb-probing-4)).

* The number of the public test data: 923
* The number of the private test data: 7,695
* The number of labels in the public test data: 8,546
* The number of labels in the private test data: 62,671
* The number of detected labels in the test data by string-matching of train-data labels: 1,717

From these results, we found that the public test score of train-data label string-matching is very high (0.530), but there are very few train-data labels in the test data (1,717). Therefore, at least 1,600 of the train-data labels in the test data might be in the public test data and very few might be in the private test data. Therefore, the public score of submission which discards the train-data labels from the prediction will correlate well with the private test score. The best submission can be obtained by finding the submission with the max score without train-data labels and adding train-data label string-matching to it.

By this approach, we succussed to select the best-private-score submission from our 201 submissions.

1. **Acronym detection**

Most datasets have acronyms (e.g., National Education Longitudinal Study → NELS). So, we did acronym detection to detect dataset-names that are not included in the train-data labels. The following procedure was used to extract them.

1. Make a list of words by splitting a text by space.
2. If a word in the list is surrounded by () and has uppercase characters and no lowercase characters, it is detected as an acronym candidate.
3. If the number of characters in the acronym candidate is less than the threshold, remove it.
4. Extract a few words before the acronym candidate from the text as a dataset-name candidate.
5. If the initial characters of each word in the dataset candidate can form the acronym candidate, detect them as a dataset-name/acronym pair. (The dataset candidate is allowed to have initial characters unrelated to the acronym candidate.)
6. Extract only those dataset-names that contain keywords (study, studies, data, survey, panel, census, cohort, longitudinal, or registry).
7. Exclude dataset-names that contain ban words (system, center, committee, etc.).
8. Apply the clean\_text function.
9. Exclude dataset-name if Jaccard scores between the dataset-name and any train-data labels or acronym-detection labels are greater than or equal to 0.5.
10. Perform string-matching to the train and test data with the detected dataset-names and count the number of occurrences among the texts of each dataset-name. Extract only those dataset-names whose count is above the threshold, because It is more likely to be a dataset-name if it appears in a lot of texts.
11. Finally, perform string-matching using the extracted dataset-names. Only when a dataset name appears more than a threshold number of times in the text, it is detected as a label.

The acronym itself is also detected as a dataset-name. String matching is performed on the dataset-name and the acronym. The acronym is detected as a label only when it and its long name appear more than a threshold number of times in the text.

By this acronym detection, we get a score of 0.418 on the public leaderboard and 0.436 on the private leaderboard. Each threshold was chosen based on the public leaderboard score.

1. **Acronym detection version 2**

To obtain more dataset-names, we performed a more aggressive acronym detection. We extract words that contain uppercase characters and no lowercase characters from the texts as acronym candidates. We search a chunk of words that is valid as the full name of the acronym candidate among the entire text (the actual code is [this](https://www.kaggle.com/osciiart/210615-det-acronym-ver2/notebook?scriptVersionId=66042722)). This acronym detection does not improve the leaderboard score because it detects many false-positive labels. But we used the dataset-names detected by it for the NER model's training, which we describe in the later step.

1. **String matching with dataset-names from external data**

We used the external U.S. government’s dataset-names obtained from [this notebook](https://www.kaggle.com/mlconsult/100000-govt-datasets-api-json-to-df/). To reduce false-positive, we apply some processing below.

1. Ext Extract only those dataset-names that contain keywords (study, studies, etc.).
2. Apply the clean\_text function.
3. Exclude dataset-name if Jaccard scores between the dataset-name and any train-data labels or acronym-detection labels are greater than or equal to 0.5.
4. Exclude dataset-name if the number of words it contains is less than the threshold.
5. Perform string-matching to the train and test data with the extracted dataset-names and count the number of occurrences among the texts of each dataset-name. Extract only those dataset-names whose count is above the threshold.
6. Finally, perform string-matching using the extracted dataset-names.

This approach improved the score of public LB score from 0.418 to 0.424 and private LB score from 0.436 to 0.486.

1. **Dataset-name variation detection using named entity recognition (NER)**

We attempted to train a NER model as a solution to this competition, using BERT or RoBERTa with the train data or the Rich Context competition data as training data. However, NER models never outperformed rule-based approaches. We think this is because a large number of true-positive labels are intentionally excluded from the provided train data. Therefore, The train data is incomplete as training data for machine learning. However, we found NER can be used to cover the weakness of string-matching. That is, NER is useful for detecting dataset-name variations that cannot be detected by string-matching. For example, National Education Longitudinal Study is sometimes quoted as National Educational Longitudinal Survey.

We used spacy library to train a NER model. We used the train-data label, the acronym-detection label, the acronym-detection-version-2 labels, and the external U.S. government label for training.

We detect dataset-name candidates from the test data using the trained NER model. We calculated Jaccard scores between dataset-name candidates and any train-data labels or acronym-detection labels. We selected candidates with Jaccard scores greater than or equal to 0.5 as variations of the existing labels.

This approach improved the score of public LB score from 0.424 to 0.614 and private LB score from 0.486 to 0.513. The details of the NER training are described in the next section.

1. **String matching with dataset-names from the train data**

Finally, we applied basic string-matching using train-data labels. We also used acronyms of train-data labels for string-matching. This approach improved the score of public LB score from 418 to 0.614 and private LB score from 0.436 to 0.513.

**NER Training Details**

A typical approach for this competition is to train a named entity recognition (NER) model to detect dataset mentions. However, one of the biggest challenges in this competition is the scarcity of training labels. Instead of adding more annotations manually, weakly supervised learning generates training labels programmatically by combining heuristics, ontology-based string matching, pretrained-models, etc.

In this competition, we combined the following labeling functions to generate training labels:

1. acronym detection
2. the government database
3. spacy's pretrained NER model

We simply combine the generated labels and use the union as training labels. (Other ways of combining labeling sources might be worth exploring, e.g. soft labels, majority vote, etc.) In addition, we apply keyword/rule-based filters to clean up generated labels and reduce false-positive labels.

We take the spacy's pretrained NER model en\_core\_web\_lg-3.0.0 and fine-tune it using our generated labels.

Things didn’t work

* Data augmentation by randomly swapping dataset-names (probably it was too noisy)
* Spacy RoBERTa NER model (just much slower and didn’t see accuracy improvement)

**Model Execution Time**

Acronym detection (on the train data): 5 minutes

Acronym detection version 2 (on the train data): 4 hours

External data preprocessing: 5 minutes

NER train data preprocessing: 4 hours (with NVIDIA P100)

NER training: 15 minutes (with NVIDIA P100)

NER prediction (on the train data): 3 hours (with NVIDIA P100)

**Submission Model**

A trained model and all codes are attached. The contents of code files are equal to [GitHub repository](https://github.com/OsciiArt/Kaggle_Coleridge_4th_Solution) Kaggle\_Coleridge\_4th\_Solution (main) and [GitHub repository coleridge\_ner](https://github.com/usuyama/coleridge_ner) (sub). Please refer to README.md of the main repository to reproduce the training and the prediction. The trained model is equal to the output of the sub repository.

**Final Submission Notebooks**

1. [Submission 1](https://www.kaggle.com/osciiart/210622-det1-neru-train-govt?scriptVersionId=66367000), Public: 0.614, Private: 0.513

Acronym detection, external-data string matching, dataset-name variation detection with NER, and train-data string matching

1. [Submission 2](https://www.kaggle.com/osciiart/210621-train-acronym-ver1?scriptVersionId=66241779): Public: 0.622, Private: 0.493

Acronym detection, external-data string matching, and train-data string matching