# A. MODEL SUMMARY

## A1. Team

- Competition Name: Coleridge Initiative - Show US the Data
- Team Name: Caminito’s Team
- Private Leaderboard Score: 0.478
- Private Leaderboard Place: 6th
- Name: Luis Federico Matorra
- Location: La Rioja
- Kaggle Account: https://www.kaggle.com/federicomatorra
- Name: Diego Passadore
- Location: Ciudad Autónoma de Buenos Aires
- Kaggle Account: https://www.kaggle.com/diegopassadore

## A2. Background on your team

Federico Matorra (FM) is an MD PhD, with a master's degree in clinical pharmacology and currently a candidate for a masters degree in medical informatics. Diego Passadore (DP) is a Nuclear Engineer (degree equivalent to a MSc) and MBA. FM gave his first steps in programming using Python and machine learning a couple of years ago when there was a need at word to extract structured information from radiology reports and the results of this development were presented at a conference in medical informatics in Argentina, together with DP. DP accumulated some experience in programming when at the university and later on during many years. Subsequently DP became a CIO of a company with focus in medical imaging until he accepted a higher position as CEO in Buenos Aires. During the last years and mainly as a hobby, DP completed a course in machine learning and developed some tools that were useful at that time for his colleagues. The results were also presented at a conference in the US and obtained recognition as the best poster presentation from Argentina.

We decided to enter the competition considering it was a place for continuing learning and for applying gained experience with the additional incentive of comparing the results with experts in the field.

We don't have a record of time spent in the competition but we would say that a good approximation is 7 to 10 hours per week each of us.

We decided to team up for the kaggle competition because we were working together on other projects.

We were basically working on different approaches but we dedicated some time together to discuss the alternatives and trying to understand the fundamentals of the competition and choosing the best strategies since we had some limitations related to the tools to be used.

### A3. Summary

After analyzing the objective of the competition, the problems faced with the nature of the available data,we were quite sure that the private test set was a completely different issue compared to the public data set, and so it was important to think of a good statistical approach, but in our case, with the additional restrictions of scarce resources and ignorance of the latest developments in NLP. So, in summary, although we evaluated many ideas (alone or combined, such as sentence2vec, identify abbreviations and acronyms, NER, text classification), we decided to use spaCy 2.3.5 (v2.spacy.io) for text classification in 2 steps, first for separating sentences (ie., sentences that mention a dataset vs no dataset at all) and second for identifying datasets versus ORGs, LOCs, etc., and many other acronyms and names that were only (at least in our view) false positives. To train the text classification models, we used the datasets names provided for training and managed to create a list of additional dataset names together with corresponding acronyms (positive and negatives) using an abbreviation detector (https://allenai.github.io/scispacy/) and matching for relevant words such as dataset, databank, survey, etc.

We think it was simple enough to be trained and tested fast and flexible enough for adapting to unknown publications. Getting the data for training took the longest time, compared to the training of the models, that was quite fast. The notebook that was scored in 6th place was a hybrid of text matching and the approach mentioned above.

## A4. Features Selection / Engineering

### A4.1 Choose how to clean the strings

One of the first challenges was defining how to clean the strings, after a lot of analysis, it was decided on the following approach, which preserves parentheses, brackets and periods, but eliminates the numbers in brackets (eg: [3]).

```
def text_cleaning(text):
    text = re.sub(r'[^A-Za-z0-9.!?'"'"'()\[\]]+', ' ', text)
    text = re.sub("'", '', text)
    text = re.sub(r'\[\d+\]', '', text)
    text = re.sub(' +', ' ', text)
    text = re.sub('[.]{2,}', '.', text)
    text = re.sub(r'\. \.', '.', text)
    text = re.sub(r' \.', '.', text)
    return text
```

```
text = text_cleaning("There's a woman waiting outside who wants to talk to you [3] [WM]. (The real subject is the woman-she is waiting outside.). \n Bye. \t")
text
```

Out: 'Theres a woman waiting outside who wants to talk to you [WM]. (The real subject is the woman she is waiting outside.). Bye.'

## A4.2 Working with all texts

All different texts from json files were concatenated and cleaned.

## A4.3 Strings matching and abbreviations (Data for NER)

To select the data to train the models we use the entity_ruler, a Spacy pipeline component, to find the known datasets (those available in the file ‘train.csv’). We also use the AbbreviationDetector script from "https://github.com/allenai/scispacy" to find unknown acronyms and datasets. Finally, we chose sentences with the words: ['dataset', 'datasets',' data-set ',' data-sets', 'data sets', 'data set', 'datum', 'databases', 'database', 'data bank', 'data banks', 'databank', 'databanks', 'metadata', 'raw data', 'time series', 'time-series'] and created patterns accordingly.

```
#patterns to find dataset
syn_dataset = ['dataset','datasets', 'data-set', 'data-sets', 'data
sets', 'data set', 'datum', 'databases', 'database', 'data bank',
'data banks', 'databank', 'databanks', 'metadata', 'raw data', 'time
series', 'time-series']
patterns = []

for dataset in syn_dataset:
    phrase = []
    for word in nlp(dataset):
        pattern = {}
        pattern["LOWER"] = str(word)
        phrase.append(pattern)
    #patterns.append({"label": dataset, "pattern": phrase})
    patterns.append({"label": "DATASET", "pattern": phrase})

from spacy.pipeline import EntityRuler
ruler = EntityRuler(nlp, overwrite_ents=True)
nlp.add_pipe(ruler)
```

```
ruler.add_patterns(patterns)
print(nlp.pipe_names)
```

Out: ['parser', 'AbbreviationDetector', 'entity_ruler']

With the previous approach we analyzed ~7000 papers and got a lot of sentences with known and some unknown datasets. On the other hand, we also found a large number of acronyms that we did not know if they were datasets or not. We Invested approximately 20 days analysing ~5000 new candidate datasets to get reliable data to train our named entities recognition model. This is a small example of the sentences that we analyzed (Fig. 1). Finally we chose 1093 sentences, 1013 positive for datasets and 80 negative (Fig. 2).

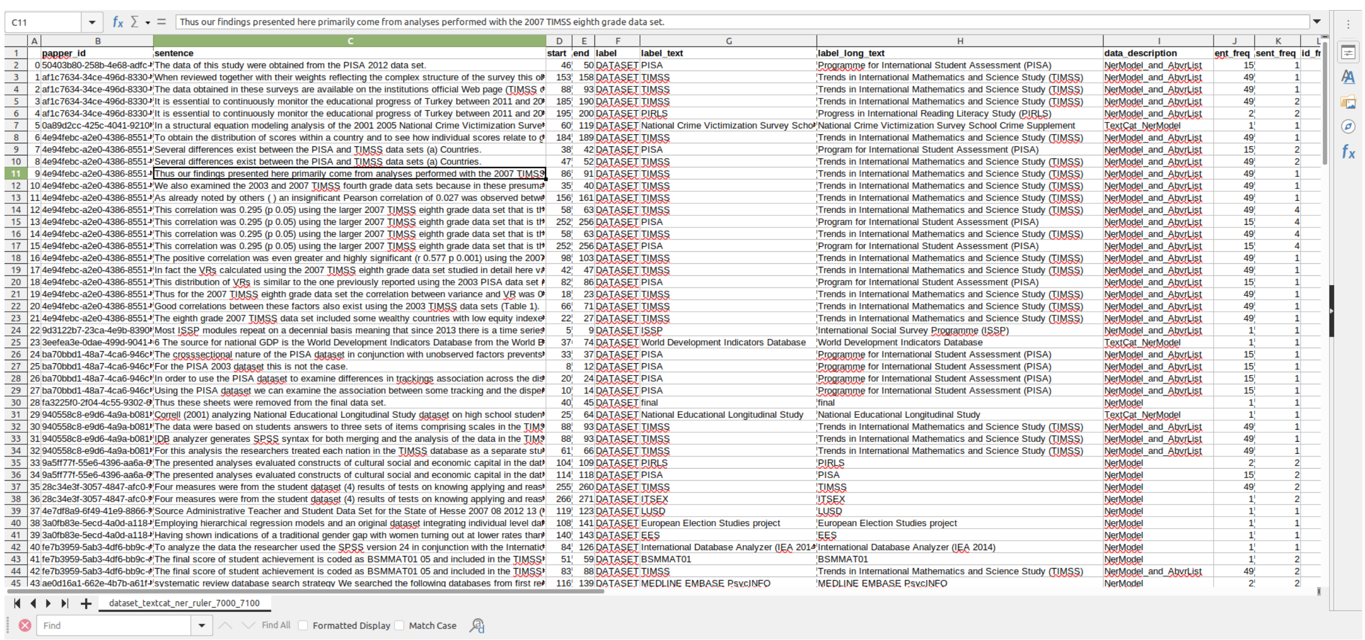

Fig. 1. Datasets were obtained manually using LibreOffice.

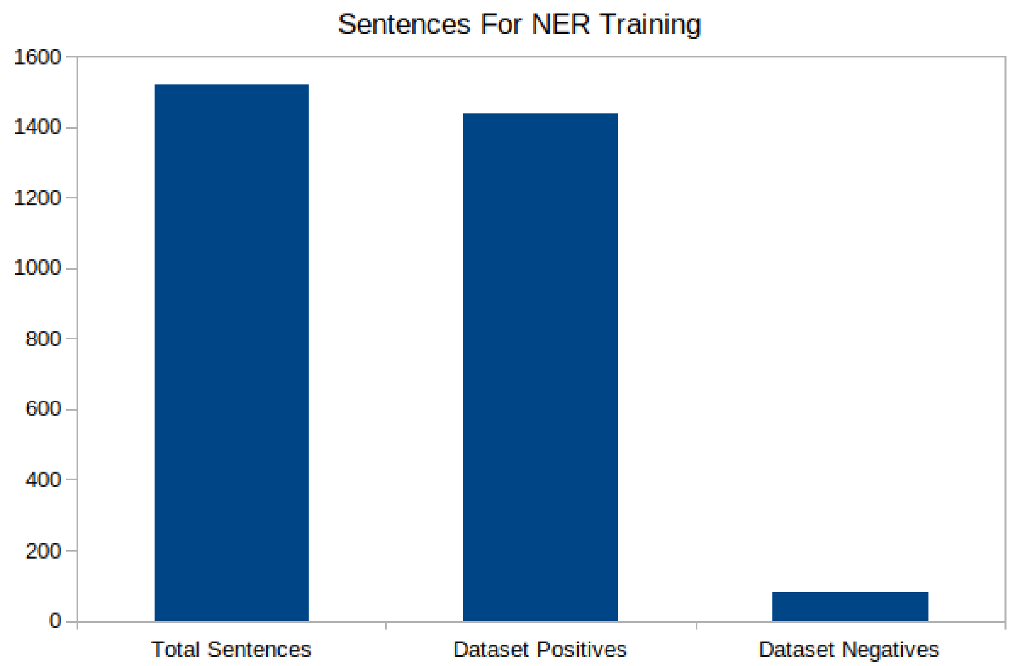


Fig. 2. Number of statements used for training named entities with spacy.

## A4.4 Strings matching and abbreviations (Sentences for TEXTCAT)

Using the same procedure as in A4.3, but with an automatic approach, we selected sentences positives por containing a mention to a dataset and negatives for not including dataset names. The result was a set of approximately 28,000 sentences, of which 13,832 were positives and 14,167 negatives (see Fig. 3). This data was subsequently used for training a text classification model (textcat) in spaCy.

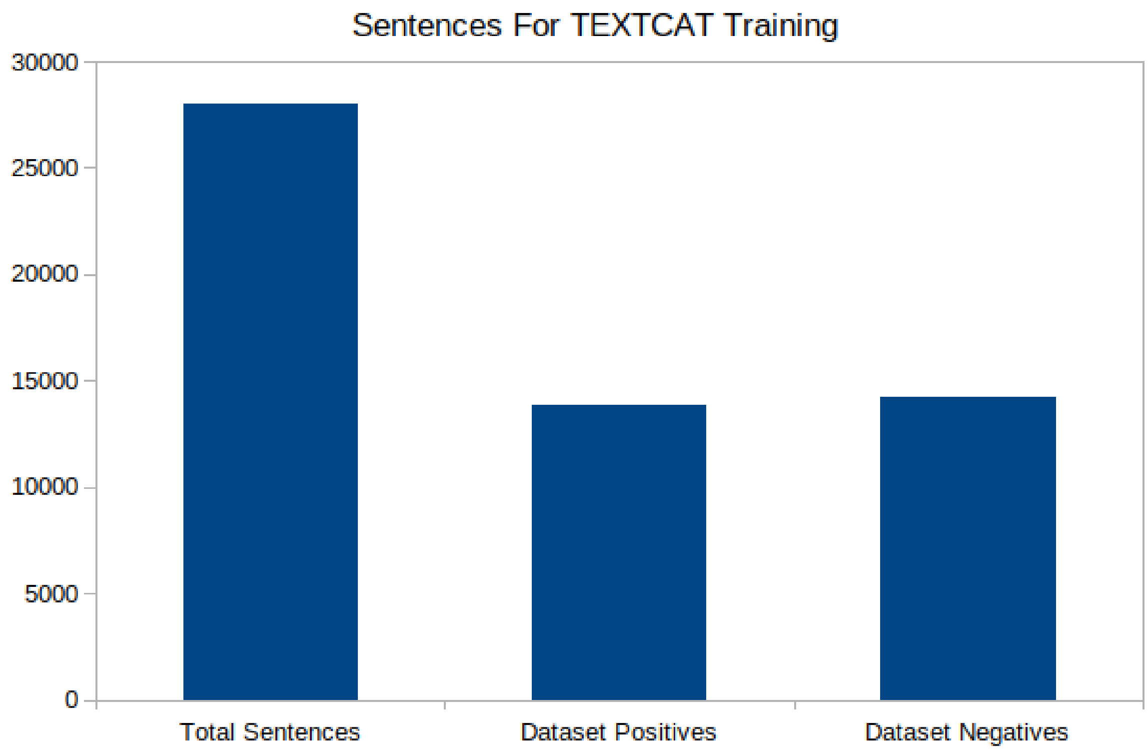


Fig. 3. Number of sentences used for training Text Categorization (TEXTCAT) with spacy.

## A4.5 Strings matching and abbreviations (Dataset for TEXTCAT)

Following the procedure described in A4.3, a total of 1,191 dataset names were found. On the other hand, with the abbreviation detector we obtained around 2,500 names of organizations, programs and localizations that were considered not to be dataset and were used as negatives for training a text categorization model.

### A4.6 Develop Word2Vec Embeddings (Gensim)

Word embeddings were trained from the whole corpus of texts (but considering only those sentences with more than 5 words) using Word2Vec in Gensim and subsequently converted to the format required by spaCy. Tokenization was previously done using spaCy and those tokens with a frequency of 6 or lower were discarded. The number of dimensions was set to 50, the context window for words during training was set to 11 and the number of epochs for training was 5.

## A5. Training Method(s)

To train our models spaCy 2.3.5 was used. spaCy is a free, open-source library for advanced Natural Language Processing (NLP) in Python and is designed specifically for process and "understand" large volumes of text. It can be used to build information extraction or natural language understanding systems, or to pre-process text for deep learning. In this sense, in the present work we trained 3 different models.

### A5.1 Training the named entity recognizer (NER)

To train NER, a pre-trained model of spaCy (en_core_web_sm) was used as the base model for transfer learning. Likewise, we used the vectors generated in A4.5 and the data obtained in A4.3 were prepared to be trained by spaCy and were divided into train (80%), test (20%). Finally, we trained a model using the spacy train command as can be seen below, and then for the competition we used our best model (Fig. 4).

```
!python -m spacy train en NER-LAST-VEC-1100
'/spacy_2.3.5/data_for_training/train_last_sent_929_format.json'
'/spacy_2.3.5/data_for_training/dev_last_sent_164_format.json'
--base-model 'en_core_web_sm' --vectors
'/spacy_2.3.5/Diego/w2v-w11-f7-50-spacy'  -p ner -R
```

```
✔ Created output directory: NER-LAST-VEC-1100
Training pipeline: ['ner']
Starting with base model 'en_core_web_sm'
Replacing component from base model 'ner'
Loading vector from model
'/home/fede/kaggle_competition_2/spacy_2.3.5/Diego/w2v-w11-f7-50-spacy'
Counting training words (limit=0)

Itn  NER Loss   NER P   NER R   NER F   Token %  CPU WPS
---  ---------  ------  ------  ------  -------  -------
  1   2776.414  76.256  81.068  78.588  100.000    49777
  2   1714.480  81.517  83.495  82.494  100.000    52181
  3    951.704  82.629  85.437  84.010  100.000    51633
  4    803.463  83.568  86.408  84.964  100.000    52734
  5    654.993  85.024  85.437  85.230  100.000    50206
  6    597.684  82.524  82.524  82.524  100.000    49756
  7    563.151  85.577  86.408  85.990  100.000    51791
  8    562.420  85.167  86.408  85.783  100.000    51232
  9    496.719  85.437  85.437  85.437  100.000    52604
 10    352.006  85.854  85.437  85.645  100.000    51433
 11    386.430  87.192  85.922  86.553  100.000    51690
 12    301.259  86.275  85.437  85.854  100.000    52828
 13    309.978  84.058  84.466  84.262  100.000    51166
 14    260.473  83.902  83.495  83.698  100.000    52488
 15    207.163  84.058  84.466  84.262  100.000    52573
 16    184.052  82.692  83.495  83.092  100.000    52188
 17    266.454  82.297  83.495  82.892  100.000    51824
 18    182.525  83.333  84.951  84.135  100.000    51955
 19    166.007  82.775  83.981  83.373  100.000    51702
 20    180.905  82.775  83.981  83.373  100.000    51942
 21    166.192  83.173  83.981  83.575  100.000    52412
 22    146.576  83.495  83.495  83.495  100.000    51733
 23    130.246  83.495  83.495  83.495  100.000    51538
 24     88.741  82.297  83.495  82.892  100.000    48313
 25    160.169  82.297  83.495  82.892  100.000    51915
 26    108.485  82.692  83.495  83.092  100.000    51441
 27    155.380  83.092  83.495  83.293  100.000    51386
 28    142.694  83.092  83.495  83.293  100.000    51477
 29     94.085  83.092  83.495  83.293  100.000    52607
 30     77.252  83.495  83.495  83.495  100.000    52585
✔ Saved model to output directory
NER-LAST-VEC-1100/model-final
✔ Created best model
NER-LAST-VEC-1100/model-best
```

Fig. 4.Training NER with spaCy.

A5.2-3 Training a text classification model (TEXTCAT)

Two text classification models were trained, one to recognize positive statements for a dataset (A4.4) and the other to recognize a dataset itself (A4.5). Both were trained using empty baseline models, empty vectors and the "ensemble" architecture, which spaCy defines as "*Stacked ensemble of a bag-of-words model and a neural network model. The neural network uses a CNN with mean pooling and attention. The "ngram_size" and "attr" arguments can be used to configure the feature extraction for the bag-of-words model.*"

## A6. Interesting findings

After reading different discussions from the participants, such as *What is your best score without string matching?* [4] or *Are we headed for shake-up armageddon?* [5], we thought that the solution of the named entities had the disadvantage of generating a large number of false positives. On the other hand, string matching definitely didn't seem like an answer to the challenge. So, we think that our most important trick was to have a good set of data for training, complementing those provided by the organizers, and also part of our success was the approach we designed to differentiate whether or not a named entity is a dataset (A4.4).

## A7. Simple Features and Methods

In terms of simplifying our model, we believe that the models used are straightforward. In this sense, the most complex thing was obtaining reliable and supervised data to be able to train our models. However, the challenge outweighed the different alternatives with unsupervised data that we tested. In this regard, perhaps our knowledge in analyzing and reading scientific papers especially oriented to medicine and engineering, helped us to generate an approach that was successful.

## A8. Model Execution Time

To train the NER model (A5.1), it took about 20 minutes. The TEXTCAT model that recognizes Dataset over Organizations, took approximately 10 minutes. Finally, the TEXTCAT model that recognizes sentences that include at least one dataset, took no more than 50 minutes. The notebook that was scored in 6th place took about 6 hours to make the prediction in the private data set that was recorded in the Public Score. Training word embeddings within a MacBook Pro took several hours.

## A9. References


1. https://v2.spacy.io/
2. https://github.com/allenai/scispacy
3. https://github.com/RaRe-Technologies/gensim
4. https://www.kaggle.com/c/coleridgeinitiative-show-us-the-data/discussion/232964
5. https://www.kaggle.com/c/coleridgeinitiative-show-us-the-data/discussion/238005
6. https://jupyter.org/
7. https://www.anaconda.com/products/individual
8. https://www.libreoffice.org/discover/libreoffice/