USGS Notes for Extracting Entities for the Model Catalog:

Goal:

- Create NER Model to identify which methods are used in each model using text from the abstract or website description if an article doesn't exist.

- NEW Idea(s):
  - ~~Instead of trying to classify the language into NERs (difficult for methods) – predict title and use similarities of predicted titles~~
  - *It seems like it would be nice to know the datasets as well as the methods used for each model*
  - Combine pattern matcher with ner model

```
@recipe('ner.teach')
def teach(dataset, source, patterns, label):
    stream = get_stream(source)
    model = EntityRecognizer(spacy.load('en_core_web_lg'))
    matcher = PatternMatcher(model.nlp).from_disk(patterns)
    predict, update = combine_models(model, matcher)
    return {
        'view_id': 'ner',
        'dataset': dataset,
        'stream': prefer_uncertain(predict(stream)),
        'update': update
    }
```

Entities:

- **First Attempt**: 3 Entities
  - METHOD: Machine learning or statistical model (NN, SVM, etc.)
  - TASK: e.g. Image classification, segmentation, etc.
  - APPLICATION: e.g. tracking rainfall thresholds for landslides, etc.
- Observations of 3 entities: Methods were a mixture of mathematical, ML, statistical, or geological models. Applications often had long (multi-word) entities, but were often preceded by the word, "provides", the model was a failure – however it could have been due to improper training. Could come back and try this again.

- **Second Attempt**: 1 Methods (or 2 methods GEO and ML)
  - Methods = GEO model or ML/Statistical Model (try separate and together)
- Observations: There are so few ML entities in the text, there are too few to train on. GEO methods becomes complex and became geological entities instead of geological model entities

- **Third Attempt:**
  - GEO Words (see if 'make-gold' is better)
  - MODEL - Words after 'model'

- Observations: I create the GEO model with nouns, but it may be better to have adjectives (watershed model or water balance algorithm – watershed/water balance is an adjective) – searching for parts of speech, not just keywords. On first try, I did not include any of the adjectives, only nouns.
- If I want to train for words before 'model', first need more abstracts, second try without patterns. Patterns probably makes it worse.
- GEO model (6) produced weird results where it recognized some GEO terms, missed some, and others highlighted all words!!?? Accuracy was worse than earlier models but Make Gold was better. Need to redo this test. Labeled NER incorrectly and couldn't fix later.
- MODEL model (7) There are not enough words before 'model' to train on.

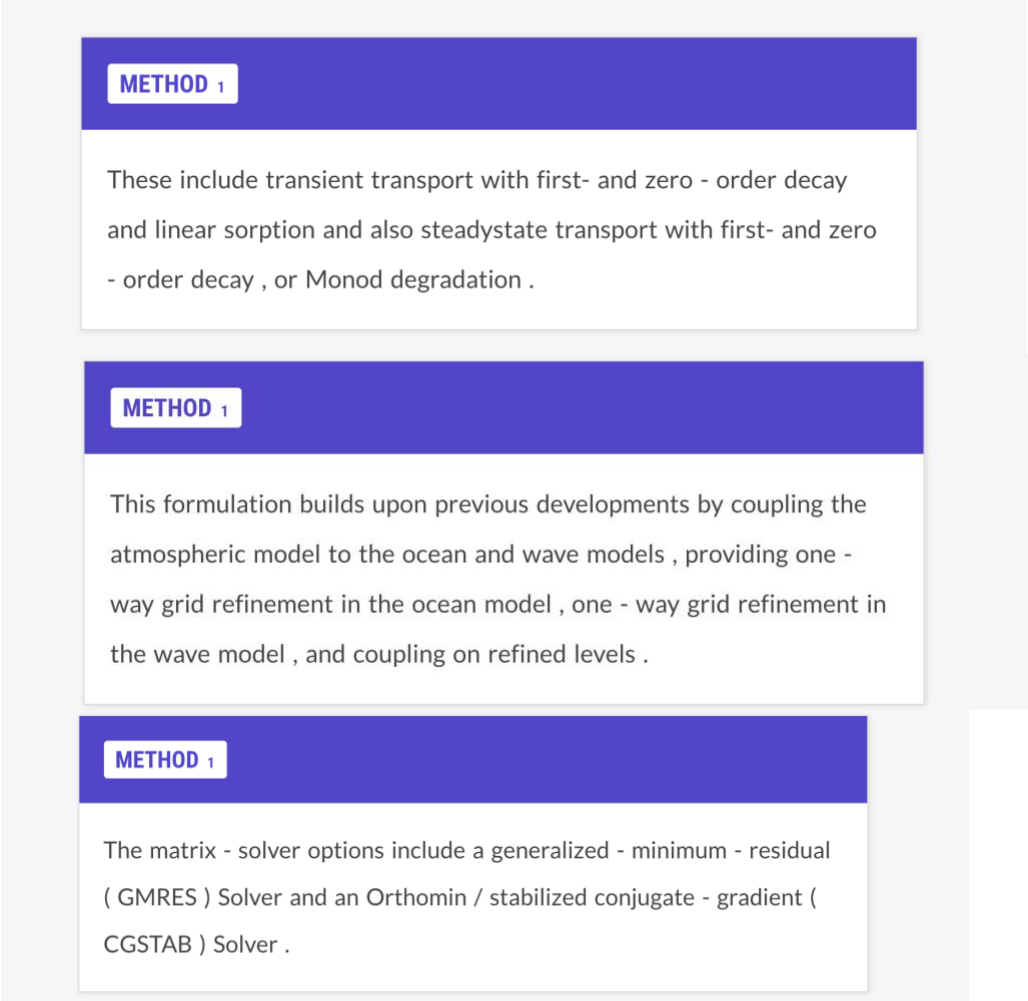
- **Fourth Attempt:**
- Downloaded huge json file for arxiv abstracts. Spent a bit of time filtering it and putting it into jsonl format
- Develop a set of rules to follow for the training. For example, decide on:
    - Using acronyms, which ones (k-NN, MCMC, SVM, etc.)
    - If model has large new acronym – include all words, include new names for algorithm or method
    - Including 'model', 'algorithm', or 'method'
- Take notes on seeds and while doing manual train so make-gold/other training is consistent
- Something seems to be wrong with text, batch-train producing error
- Retrained algo model, new jsonl (stripped text again just in case), 500 train, 500 test in algo_model3 folder,

- **Fifth Attempt:**
- RULES: do not include method, algorithm, or model in any labels, including common acronyms (kNN, SVM, etc.), do not include NN – could be neural network or nearest neighbor, include all 'new' algorithm names
- This model is a kind of combo of known ML algorithms + anything that comes before method, algo, etc. included 'regression' and 'classification', not 'model', 'algorithm', 'method' or 'technique'
- Manually labeled 200 texts
- Trained a model with 65% accuracy and was doing quite well. Went back into make gold again and the model now has 33% ☹ Took at least 5 hours to train this model, so I took a break from training another one to learn more about Prodigy Spacy.
- Turns out make-gold really doesn't use any rejected items, and assumes that if something is not labeled, it is not an entity and vice versa. I used a lot of REJECT by unselecting the entity if everything was labeled.
- Does IGNORE samples in make-gold make for a better model (see ALGO model 2 where I thought I was using the IGNORE button wrong but it made for a better model)
- Running the ALGO on the USGS texts returns geological terms

Other Observations while manually scraping text:

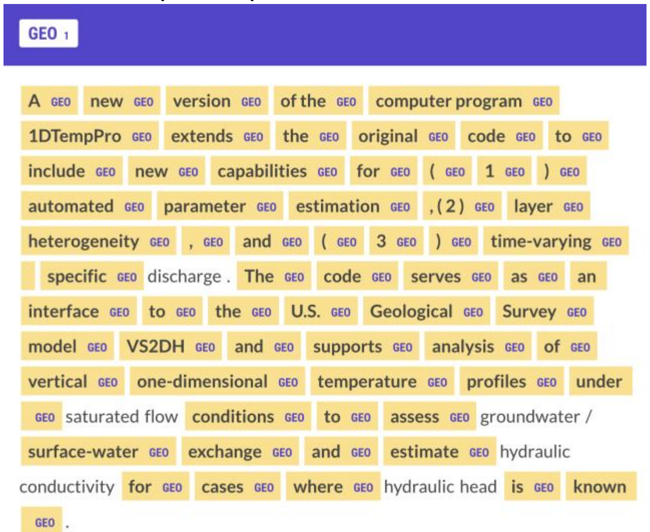
- Tasks and applications may overlap. Difficulty distinguishing between the two for some articles.
- Some texts may not have a method.
- One article had a one-sentence section called "Introduction" as well as a short "Model Overview and System Requirements", but no Abstract section.
- Other sections to consider (though not consistent): "purpose and scope", "summary", "Model Overview and System Requirements"
- May reference models outside of text – Future models, other models …
- Mostly specific mathematical models for waterflow, watershed, water balance, landslides, etc. (not machine learning methods)

- GEO either has a lot of material to highlight or to try to weed out what are actual GEO models. ML has maybe 1-2 recognizable ML/Statistical Model in 20 texts (most are geological modeling). few examples for ML-method, so couldn't train well.
- Models built on prior models (are those models the method?)
- Accuracy correct? models this far label just about everything as an entity
- Need to tease out the geologic models.
- When trying GEO alone, I realized that I started to label any clear geology noun (volcano, stream, ocean, reservoir, etc.). These are not necessarily geologic models, but could be useful… the models are much harder to separate see GeoModel Notes and examples below
- GeoModel Notes:
    - digital elevation model, diffuse layer model,
    - 3D method of columns (landslides)
    - Vertical fluid flow (seepage flux)
    - "building a 3D domain"
    - Limit-equilibrium analysis, slope-stability analysis, slope stability
    - Aqueous geochemical calculations
    - (1D) transport calculations
    - Ion aqueous model, Pitzer aqueous model
    - Peng-Robinson equation (for gas solubility)
    - Transport simulations, geochemical simulations, heat transport equations,
    - Flow and transport: aquifer, pollutants, sediment, solute, heat, thermal energy,
    - Population (density, estimation, prediction, etc.)
    - Water-balance model, water balance modeling,
    - Migration/Migration Cycle
- See examples below for help labeling:



Volcano (volcanic ash)
Earthquakes (subduction zone, ground motion prediction
Flooding/Waves
Migration
Reservoir water balance
Aqueous geochemical model
Precipitation/Slope Stability Analysis
Hydrodynamics, Vertical Fluid Flow
data estimation from isopleth map
solute transport, thermal energy transport, sediment transport
watersheds and redevelopment
water quality
evolution of coastal zone
climate change

- Used ‘bad examples’ with few GEO-like examples and one with just about everything except for questionable GEO terms as below:



Error when training arxiv algo_model2

Prodigy ner.batch-train algo_method2 en_vectors_web_lg --init-tok2vec /Users/sashaqanderson/Dropbox/USGS/NER_Work/algo_model/pretrain_algo_model/model999.bin --output algo_model2 --eval-split 0.2 --label ALGO

BEFORE 0.000

Correct 0

Incorrect 21

Entities 0

Unknown 0

LOSS RIGHT WRONG ENTS SKIP ACCURACY

0%| | 0/895 [00:00<?, ?it/s]['B-ALGO', 'I-ALGO', 'I-ALGO', 'L-ALGO', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O']

['B-ALGO', 'L-ALGO', 'B-ALGO', 'L-ALGO', 'B-ALGO', 'L-ALGO', 'B-ALGO', 'L-ALGO', 'B-ALGO', 'L-ALGO', 'B-ALGO', 'L-ALGO', 'B-ALGO', 'L-ALGO', 'B-ALGO', 'L-ALGO', 'B-ALGO', 'L-ALGO', 'B-ALGO', 'L-ALGO', 'B-ALGO', 'L-ALGO', 'U-ALGO', 'U-ALGO', 'B-ALGO', 'L-ALGO']

['O', 'O', 'B-ALGO', 'I-ALGO', 'L-ALGO', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O']

['B-ALGO', 'L-ALGO', 'B-ALGO', 'L-ALGO', 'B-ALGO', 'L-ALGO', 'B-ALGO', 'L-ALGO', 'B-ALGO', 'L-ALGO', 'B-ALGO', 'L-ALGO', 'U-ALGO', 'U-ALGO', 'B-ALGO', 'L-ALGO']

['B-ALGO', 'L-ALGO', 'B-ALGO', 'L-ALGO', 'B-ALGO', 'L-ALGO', 'B-ALGO', 'L-ALGO', 'B-ALGO', 'L-ALGO', 'B-ALGO', 'L-ALGO', 'B-ALGO', 'L-ALGO', 'B-ALGO', 'L-ALGO', 'B-ALGO', 'L-ALGO', 'B-ALGO', 'L-ALGO', 'B-ALGO', 'L-ALGO', 'B-ALGO', 'L-ALGO', 'U-ALGO', 'U-ALGO', 'B-ALGO', 'L-ALGO']

['U-ALGO']

['B-ALGO', 'L-ALGO', 'B-ALGO', 'L-ALGO', 'B-ALGO', 'L-ALGO', 'B-ALGO', 'L-ALGO', 'B-ALGO', 'L-ALGO', 'B-ALGO', 'L-ALGO', 'B-ALGO', 'L-ALGO', 'B-ALGO', 'L-ALGO', 'B-ALGO', 'L-ALGO', 'U-ALGO', 'U-ALGO', 'U-ALGO']

['U-ALGO']

['U-ALGO']

['O', 'B-ALGO', 'L-ALGO', 'B-ALGO', 'L-ALGO', 'B-ALGO', 'L-ALGO', 'B-ALGO', 'L-ALGO', 'B-ALGO', 'L-ALGO', 'B-ALGO', 'L-ALGO', 'B-ALGO', 'L-ALGO', 'B-ALGO', 'L-ALGO', 'B-ALGO', 'L-ALGO', 'B-ALGO', 'L-ALGO', 'B-ALGO', 'L-ALGO', 'B-ALGO', 'L-ALGO', 'U-ALGO', 'U-ALGO', 'B-ALGO', 'L-ALGO']

['U-ALGO']

['B-ALGO', 'L-ALGO', 'B-ALGO', 'L-ALGO', 'B-ALGO', 'L-ALGO', 'B-ALGO', 'L-ALGO', 'B-ALGO', 'L-ALGO', 'B-ALGO', 'L-ALGO', 'B-ALGO', 'L-ALGO', 'B-ALGO', 'L-ALGO', 'U-ALGO', 'U-ALGO', 'U-ALGO']

['O', 'U-ALGO', 'O', 'O', 'O', 'O', 'U-ALGO', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O']

Traceback (most recent call last):

File "/opt/anaconda3/envs/USGS_1/lib/python3.7/runpy.py", line 193, in _run_module_as_main

"__main__", mod_spec)

File "/opt/anaconda3/envs/USGS_1/lib/python3.7/runpy.py", line 85, in _run_code

exec(code, run_globals)

File "/opt/anaconda3/envs/USGS_1/lib/python3.7/site-packages/prodigy/__main__.py", line 380, in <module>

controller = recipe(*args, use_plac=True)

File "cython_src/prodigy/core.pyx", line 212, in prodigy.core.recipe.recipe_decorator.recipe_proxy

File "/opt/anaconda3/envs/USGS_1/lib/python3.7/site-packages/plac_core.py", line 328, in call

cmd, result = parser.consume(arglist)

File "/opt/anaconda3/envs/USGS_1/lib/python3.7/site-packages/plac_core.py", line 207, in consume

return cmd, self.func(*(args + varargs + extraopts), **kwargs)

File "/opt/anaconda3/envs/USGS_1/lib/python3.7/site-packages/prodigy/recipes/ner.py", line 621, in batch_train

examples, batch_size=batch_size, drop=dropout, beam_width=beam_width

File "cython_src/prodigy/models/ner.pyx", line 362, in prodigy.models.ner.EntityRecognizer.batch_train

File "cython_src/prodigy/models/ner.pyx", line 453, in prodigy.models.ner.EntityRecognizer._update

File "cython_src/prodigy/models/ner.pyx", line 446, in prodigy.models.ner.EntityRecognizer._update

File "cython_src/prodigy/models/ner.pyx", line 447, in prodigy.models.ner.EntityRecognizer._update

File "/opt/anaconda3/envs/USGS_1/lib/python3.7/site-packages/spacy/language.py", line 529, in update

proc.update(docs, golds, sgd=get_grads, losses=losses, **kwargs)

File "nn_parser.pyx", line 444, in spacy.syntax.nn_parser.Parser.update

File "nn_parser.pyx", line 546, in spacy.syntax.nn_parser.Parser._init_gold_batch

File "transition_system.pyx", line 102, in spacy.syntax.transition_system.TransitionSystem.get_oracle_sequence

File "transition_system.pyx", line 163, in spacy.syntax.transition_system.TransitionSystem.set_costs

ValueError: [E024] Could not find an optimal move to supervise the parser. Usually, this means that the model can't be updated in a way that's valid and satisfies the correct annotations specified in the GoldParse. For example, are all labels added to the model? If you're training a named entity recognizer, also make sure that none of your annotated entity spans have leading or trailing whitespace or punctuation. You can also use the experimental `debug-data` command to validate your JSON-formatted training data. For details, run:

python -m spacy debug-data –help

- Algo model 3 does well on some labels and poorly on others. See print-stream. Try words before ‘model’ again, this time with arxiv data, there should be plenty to train on.

Prodigy ner.manual algo_terms blank:en /Users/sashaqanderson/Dropbox/USGS/NER_Work/arxiv_model/algo_data_model2.jsonl --label ALGO