Policy Analysis with NLP

Group WRI

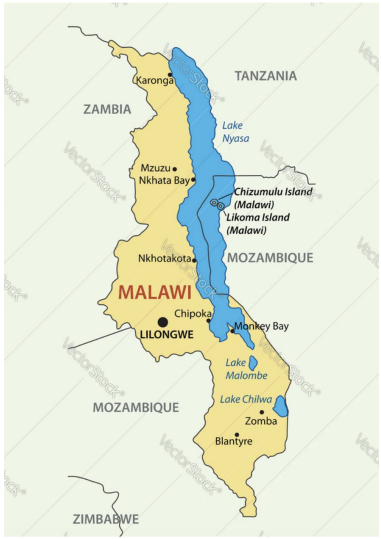
Yi Wang, Mianchun Lu, Echo Liu, Marshall Zhao, Yuefu Wu, Yirou Ge, Diana Dai, Rachel Li

Overview

- Executive Summary
- Research Questions
- Methodologies and Techniques
- Application
  - How to Better Understand Financing Availability in Forest Landscape Restoration Policy

Executive Summary

- **Topic:**
  - Identify financing incentive mechanisms from forest landscape restoration policy documents in **India**, **Kenya** and **Malawi**



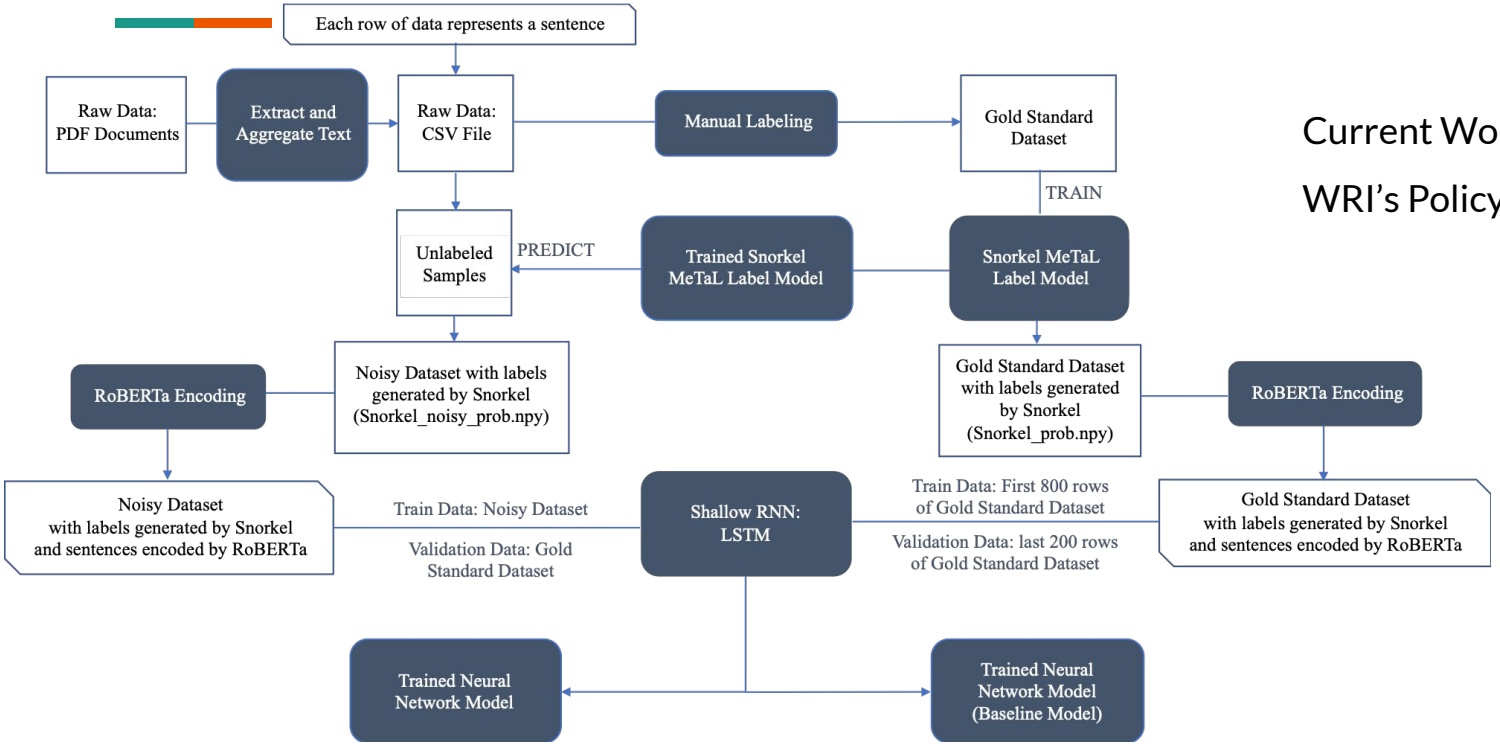
Research Questions

- **Challenge:**
  - How do we analyze policy?
  - Can Natural Language Processing help individuals understand what financing is available for sustainable development?
  - How can Natural Language Processing help standardize policy research while improving transparency and reproducibility?

- **Research Questions:**
  - How to better classify the financing incentive of relevant policies in India, Kenya and Malawi (measured by the predictive accuracy of labeling models)?
  - How to maximize the value of current models and apply them to identify other characteristics of each financing mechanism?

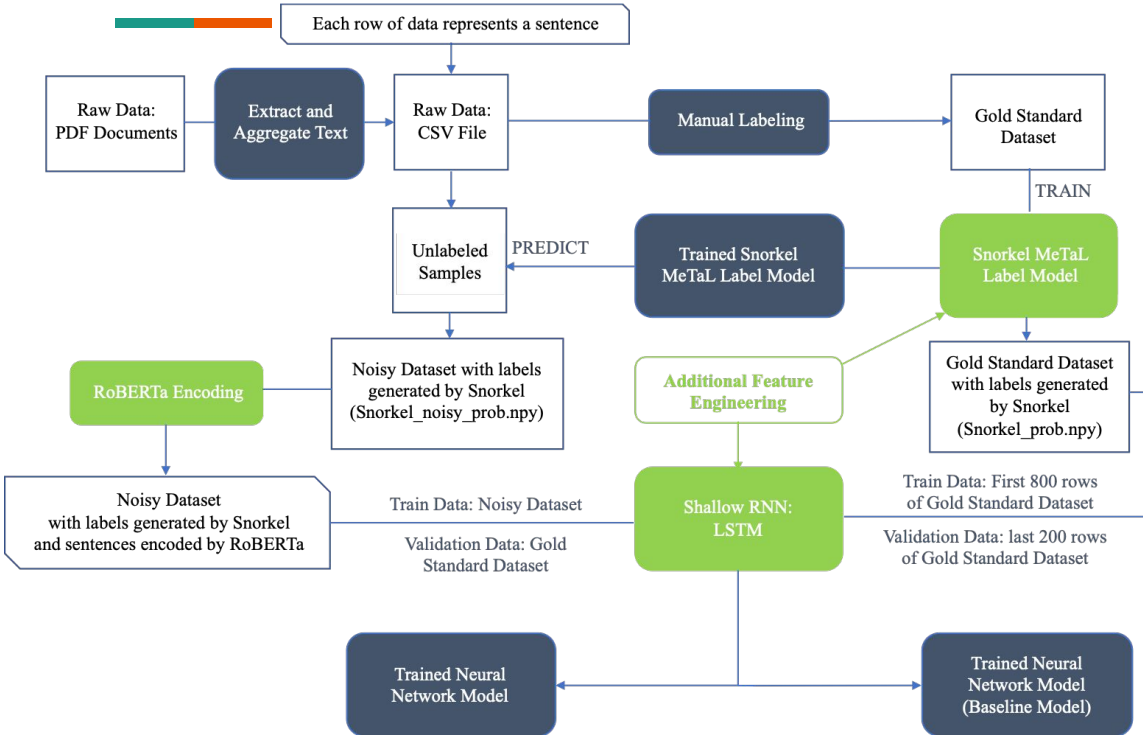
**In the future: How does the financing availability impact different socioeconomic groups, and how does it match up with agricultural subsidies for intensification?*

Methodologies - Current Work



Current Workflow of WRI's Policy Toolkits

Methodologies - Future Plan



Our Focus:

➔ Label Model Optimization:

BabbleLabble v.s Snorkel Comparison

Additional Feature Engineering

➔ Neural Networks Optimization:

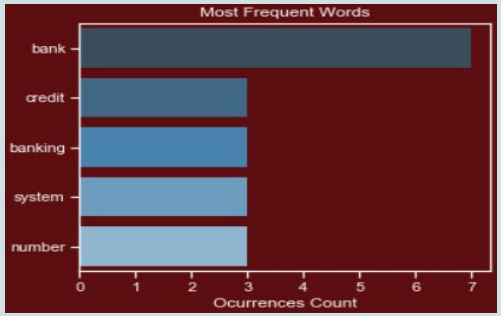
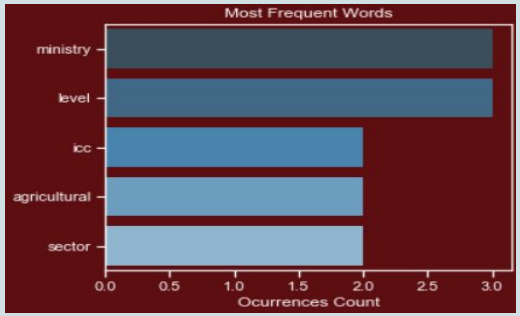
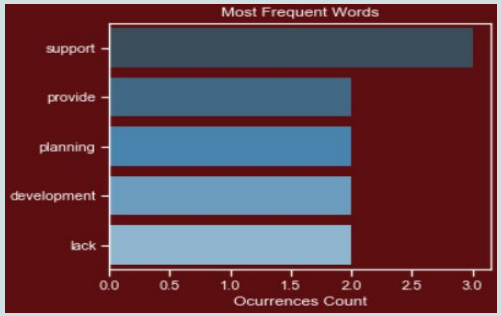
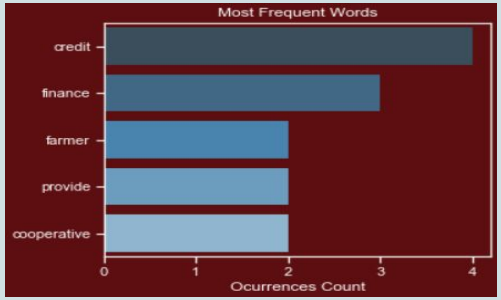
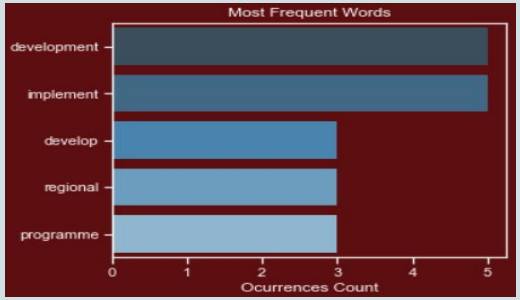
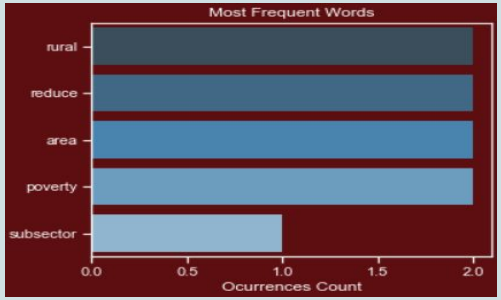
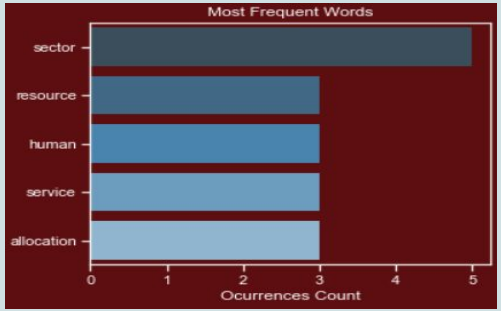
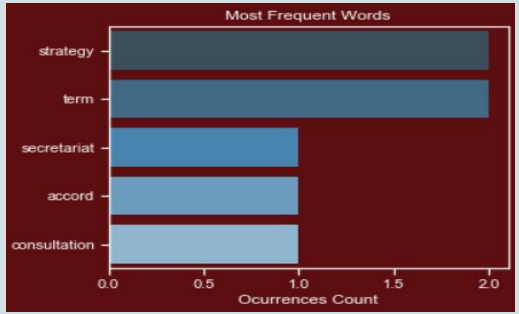
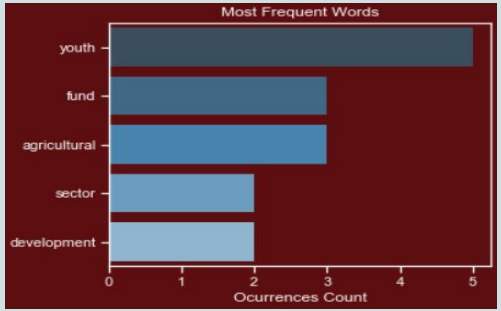
Build Deeper and More Complicated RNN

Fine tune RoBERTA Encoding

Additional Feature Engineering

EDA

- Calculate the similarity matrix with some samples from gold_standard data
  - Purpose: getting some insights by comparing the sentences in different classes & supporting following steps and models.
  - Samples:
    - Three sentences in “positive” class
      - *E.g. “lack of a database and information to support regional development is another major constraint…… support development planning and provide investment opportunities.”*
    - Three sentences in “negative” class
      - E.g. “access to bank credit by farmers is still a major challenge …… limited number of banks in rural areas are some of the factors that make it difficult for farmers to access bank credit.”
    - Three sentences in “neutral” class
      - E.g. “the purpose of the middle level institutions is to provide a link between national and local level implementation…... coordinating the planning of the strategy at the agricultural sector level and monitoring its implementation to ensure that its goals are achieved.”
  - Hypothesis: The similarity coefficient in the same class should be larger than the similarity coefficient between different classes

	Negative	Neutral	Positive
First One			
Second One			
Third One			

EDA

```
dataset=list(gold_standard['sentences'])

def cos_sim(a,b):
    dot_product=np.dot(a,b)
    norm_a=np.linalg.norm(a)
    norm_b=np.linalg.norm(b)
    return dot_product/(norm_a * norm_b)

def getTF(sentence):
    pip=Nlp_sample_pipeline(sentence)
    tokens=pip.text_token()
    tokens = tokens[(tokens["is_alphabet"] == True) & (tokens["is_stopword"] == False)
                    & (tokens["lemma_lower"].str.len() > 1)]["lemma_lower"].value_counts()
    return tokens

def getIDF(term,dataset):
    totle=len(dataset)
    numDocumentsWithThisTerm = 0
    for item in dataset:
        if term.lower() in item.lower().split():
            numDocumentsWithThisTerm = numDocumentsWithThisTerm+1
    if numDocumentsWithThisTerm > 0:
        return 1.0 + log(float(len(dataset)) / numDocumentsWithThisTerm)
    else:
        return 1.0

def getSimilarity(sentence1,sentence2):
    tf1=getTF(sentence1)
    tf2=getTF(sentence2)
    all_words_list=[]
    for item in tf1.index:
        all_words_list.append(item)
    for item in tf2.index:
        if item not in all_words_list:
            all_words_list.append(item)
    all_words_list_size = len(all_words_list)

    v1=np.zeros(all_words_list_size, dtype=np.float)
    v2=np.zeros(all_words_list_size, dtype=np.float)
    i=0
    for item in all_words_list:
       # idf=getIDF(item,dataset)
        if item in tf1.index:
            v1[i]=float(tf1[tf1.index==item])/float(len(tf1))
        else:
            v1[i]=0
        if item in tf2.index:
            v2[i]=float(tf2[tf2.index==item])/float(len(tf2))
        else:
            v2[i]=0
        i+=1
    return cos_sim(v1,v2)
```

- Two methods were used to calculate the similarity matrix:
  - TFIDF (term frequency–inverse document frequency) with cosine similarity
  - Google Word2Vec model with cosine similarity
- Results:
  - Both of these two matrices show that the average similarity coefficient between two classes is smaller than the average similarity coefficient in each one class.

TFIDF	Positive 0	Positive 1	Positive 2	Negative 0	Negative 1	Negative 2
Positive 0		0.1487	0.1165	0.0329	0.1227	0.0127
Positive 1			0.1634	0.0598	0.0382	0.0690
Positive 2				0.0562	0.1027	0.1575
Negative 0					0.1945	0.0183
Negative 1						0.0390
Negative 2						

average similarity coefficient	
positive class	0.1429
negative class	0.0839
between two classes	0.0724

word2vec	Positive 0	Positive 1	Positive 2	Negative 0	Negative 1	Negative 2
Positive 0		0.7614	0.7190	0.7448	0.7206	0.7467
Positive 1			0.8236	0.7213	0.7464	0.7032
Positive 2				0.7910	0.8155	0.7449
Negative 0					0.8788	0.8313
Negative 1						0.7940
Negative 2						

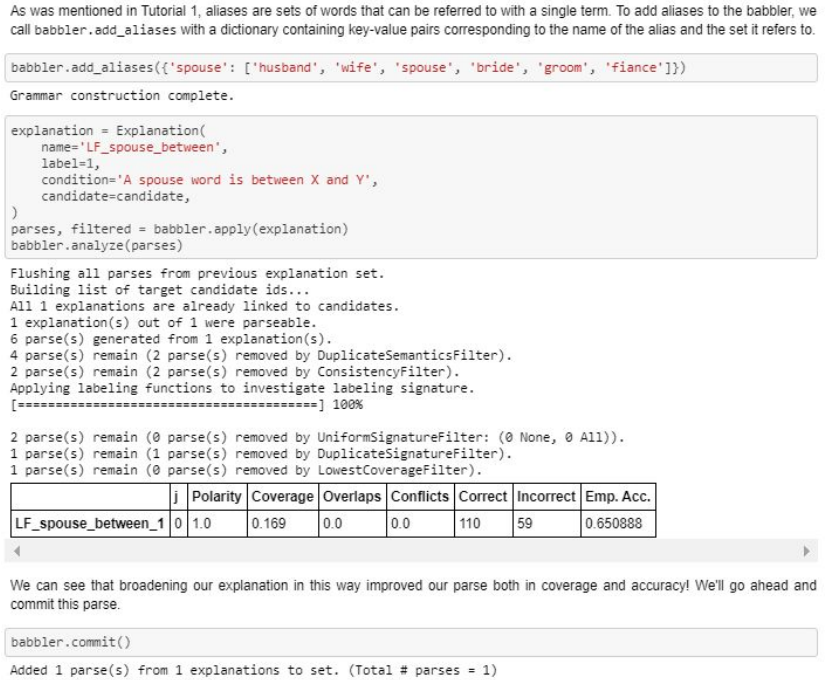
average similarity coefficient	
positive class	0.7680
negative class	0.8347
between two classes	0.7483

Label Model Optimization

BabbleLabble is one of many projects exploring how weak supervision sources can be used to train machine learning systems

Process:

1. Collect those reasons as ***natural language explanations***
2. Convert natural language explanations via ***semantic parser(e.g. SippyCup)*** into labeling functions
3. Analyze the accuracy of the function
4. Use executable functions to automatically label additional data.



As was mentioned in Tutorial 1, aliases are sets of words that can be referred to with a single term. To add aliases to the babbler, we call `babbler.add_aliases` with a dictionary containing key-value pairs corresponding to the name of the alias and the set it refers to.

```
babbler.add_aliases({'spouse': ['husband', 'wife', 'spouse', 'bride', 'groom', 'fiance']})
```

```
Grammar construction complete.
```

```
explanation = Explanation(
    name='LF_spouse_between',
    label=1,
    condition='A spouse word is between X and Y',
    candidate=candidate,
)
parses, filtered = babbler.apply(explanation)
babbler.analyze(parses)
```

```
Flushing all parses from previous explanation set.
Building list of target candidate ids...
All 1 explanations are already linked to candidates.
1 explanation(s) out of 1 were parseable.
6 parse(s) generated from 1 explanation(s).
4 parse(s) remain (2 parse(s) removed by DuplicateSemanticsFilter).
2 parse(s) remain (2 parse(s) removed by ConsistencyFilter).
Applying labeling functions to investigate labeling signature.
[======================================] 100%

2 parse(s) remain (0 parse(s) removed by UniformSignatureFilter: (0 None, 0 All)).
1 parse(s) remain (1 parse(s) removed by DuplicateSignatureFilter).
1 parse(s) remain (0 parse(s) removed by LowestCoverageFilter).
```

	j	Polarity	Coverage	Overlaps	Conflicts	Correct	Incorrect	Emp. Acc.
LF_spouse_between_1	0	1.0	0.169	0.0	0.0	110	59	0.650888

We can see that broadening our explanation in this way improved our parse both in coverage and accuracy! We'll go ahead and commit this parse.

```
babbler.commit()
```

```
Added 1 parse(s) from 1 explanations to set. (Total # parses = 1)
```

A Babble Labble Pipeline on github

Label Model Optimization

BabbleLabble

1. Collect those reasons as ***natural language explanations***
2. Convert natural language explanations via ***semantic parser(e.g. SippyCup)*** into labeling functions
3. Analyze the accuracy of the function
4. Use executable functions to automatically label additional data.

vs.

Snorkel

1. ***Write Labeling Functions (LFs)***
2. Use Snorkel's LabelModel to automatically learn the accuracies of LFs
3. Get generative model with re-weighted labeling functions
4. Training discriminative models to develop large labeled dataset.

Label Model Optimization

Additional Feature Engineering: more labeling function

In our following steps, we will try BabbleLabble to generate noisy label, and combing BabbleLabble with Snorkel to classify whether the policy documents provide information of financial incentives, discentives, or neutral information.

Besides, we will try to provide explanations for the golden standard.

Then, we will try to add additional label functions with BabbleLabble/Snorkel.

- The words that are correlated with the sentiment of financial incentives
- Frequency, distance between specific words
- ...

To test the accuracy of this model, we will keep using gold standard data and measure it with the F1 score.

Neural Networks Optimization

➔ **Develop Deeper and More Complicated Neural Networks**

The current neural networks that WRI uses is a shallow RNN which includes only a LSTM layer and a fully-connected layer.

In order to improve the performance, we are planning to build deeper and more complicated neural networks which may include:

- Additional embedding layer if it's needed
- Dense layers/ Fully-connected layers
- plain RNN layers
- LSTM layers
- GRU layers

*The performance of the neural networks will be evaluated based on the F1 score.

```
class RecurrentNetwork(nn.Module):
    def __init__(self, embeddings, input_size, hidden_dim, n_layers):
        super(RecurrentNetwork, self).__init__()

        # create embedding layer
        embedding, num_embeddings, embedding_dim = create_emb_layer(embeddings, True)
        # define embedding layer and parameters
        self.embedding = embedding
        self.num_embeddings = num_embeddings
        self.embedding_dim = embedding_dim
        self.hidden_dim = hidden_dim
        self.n_layers = n_layers
        # define rnn layers
        self.rnn = nn.GRU(input_size, hidden_dim, n_layers, batch_first=True)
        # define fully connected layer
        self.fc = nn.Linear(hidden_dim, 4)
        self.sigmoid = nn.Sigmoid()

    # x is a PaddedSequence for an RNN
    def forward(self, x):
        # put the words through an embedding layer
        emb = self.embedding(x)
        # initialize hidden state
        batch_size = x.size(0)
        hidden = torch.zeros(self.n_layers, batch_size, self.hidden_dim)
        # feed the sequence of embeddings and hidden state into the RNN and obtaining output
        out, hidden = self.rnn(emb, hidden)
        # feed the final hidden state into a dense layer
        out = out.transpose(0, 1)[-1]
        output = self.fc(out)
        return output
```

Neural Networks Optimization

➔ **Fine tune RoBERTa Encoding Method**

Current pre-trained roBERTa model: roberta.large (RoBERTa using the BERT-large architecture)

Possible alternatives:

roberta.large.mnli (roberta.large fine-tuned on The Multi-Genre NLI Corpus)

roberta.large.wsc (roberta.large fine-tuned on Winograd Schema Challenge data)

other roberta models we ourselves fine-tuned on different corpus or datasets

Sample Sentence:

part ii farm and 6. maintenance of 1 0 percent tree cover. 7. farm forestry development notices. 8. protection of land prone to degradation. 9. seed production plans.

Sample Encoding Results: Dimension: (1, 50, 1024)

```
[[[-0.6025578   0.09012693  0.11786644 ... -0.33344632 -0.3759515
    0.20418033]
  [-0.40039277 -0.09533648  0.13030334 ...  0.24041015  0.1664356
   -0.47346532]
  [-0.53849345  0.22787295  0.05920596 ... -0.2844693  -1.2695744
   -0.94067895]
  ...
  [ 0.          0.          0.         ...  0.          0.
    0.        ]
  [ 0.          0.          0.         ...  0.          0.
    0.        ]
  [ 0.          0.          0.         ...  0.          0.
    0.        ]]]
```

Neural Networks Optimization

➔ **Additional Feature Engineering**

The current shallow RNN takes the roBERTa encoding of sentences as the input and outputs the predicted probabilities of three labels: positive, negative, and neutral.

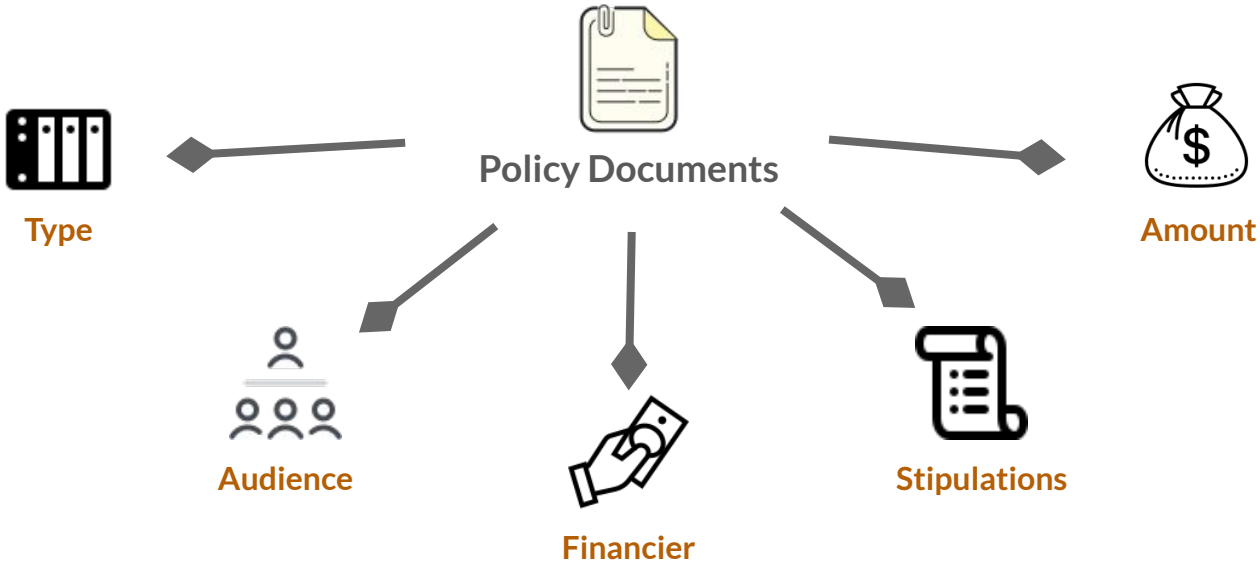
In order to optimize the performance of the RNN, we are planning to add additional features as inputs which may include:

- N-Grams: Unigrams, bigrams, and trigrams
- Part-of-Speech tags generated by Hidden Markov Model
- LIWC Features: number of words per sentence/ number of positive words/ number of negative words
- Named Entity Features
- Word Frequency & Similarity Coefficient

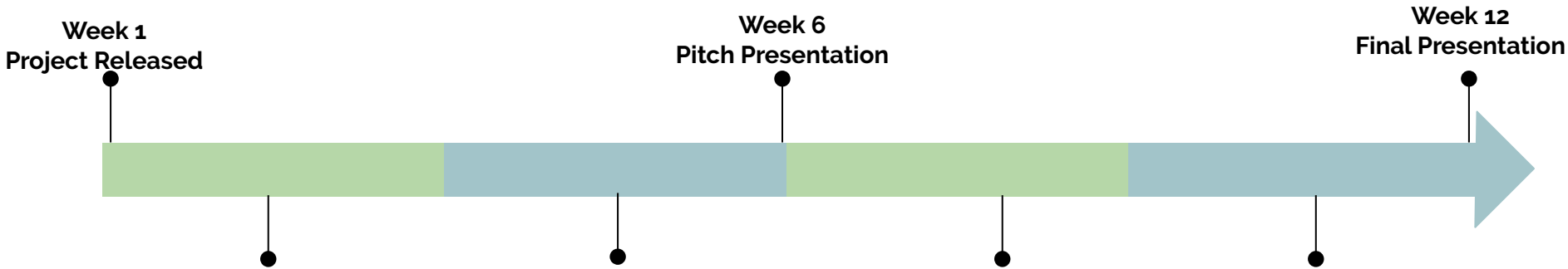
We also plan to train the model and run the codes on Google Cloud for faster and better performance considering the amount of computation it requires

Application

- ➔ To apply the classification models in a more global scale;
- ➔ To identify more aspects of each financing mechanism utilizing previous labelling models., additional characteristics of the financing policy including:



Schedule



Week 2-3
Background and Knowledge

Learning the basic idea of the projects, clearly understanding the objectives and outputs, and establishing the framework for the project.

Week 4-6
Framework and Sample Analysis

Setting up entire analysis framework for the project after exploratory data analysis and simple analysis with some samples.

Week 7-9
Feature Engineering and labeling

Comparing the results of Snorkel and BabbleLabble, and fine-tuning one of them by adding more strategies after feature engineering

Week 10-11
Finish the project and Conclude

Fine-tuning RNN with RoBERTa embedding, concluding the outputs and results, and putting forward suggestions and solutions

Discussion and confirmation of success criteria

- **Business:**
  - Reconfirm if the business insight generated from the project aligns with the objectives
  - Discuss in detail how our model will help with practical application
  - Determine the groups that will be benefited by our conclusion

- **Technical:**
  - Review the performance of model
  - Compare the initial result of Github Repository to finished work
  - Propose potential future work and challenges

Methodologies--EDA

- Calculate the similarity between documents
  - Make use of the tokens created by spaCy in the last step, and stopwords, numbers, space, and punctuations have been removed
  - Convert tokens of each document into vectors using bag of words with TF (term frequency)
  - calculate the cosine similarity
- These results are consistent with the assumption.

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

```
def cos_sim(a,b):
    dot_product=np.dot(a,b)
    norm_a=np.linalg.norm(a)
    norm_b=np.linalg.norm(b)
    return dot_product/(norm_a * norm_b)

def getSimilarity(tf1,tf2):
    all_words_list=[]
    for item in tf1.index:
        all_words_list.append(item)
    for item in tf2.index:
        all_words_list.append(item)
    all_words_list_size = len(all_words_list)

    v1=np.zeros(all_words_list_size, dtype=np.int)
    v2=np.zeros(all_words_list_size, dtype=np.int)
    i=0
    for item in all_words_list:
        if item in tf1.index:
            v1[i]=int(tf1[tf1.index==item]["lemma_lower"])
        else:
            v1[i]=0
        if item in tf2.index:
            v2[i]=int(tf2[tf2.index==item]["lemma_lower"])
        else:
            v2[i]=0
        i+=1
    return cos_sim(v1,v2)
```

	'1.PESA'	'2.FRA'	'10.NMSA'
'1.PESA'	1.00000	0.42115	0.31466
'2.FRA'		1.00000	0.21417
'10.NMSA'			1.00000

Methodologies--Information Extraction

- Build NLP pipeline with some policy documents
  - Purpose: extracting important information from the documents
  - Randomly choose three policy documents from India and extract text from the PDF
    - '1.PESA.pdf': 17 sentences; 1172 tokens
    - '2.FRA.pdf': 440 sentences; 17326 tokens
    - '10.NMSA_Guildlines_English.pdf': 1184 sentences; 24556 tokens
  - Build a NLP pipeline to have a basic understanding of each document: (spaCy)
    - Tokenizing: we iterate over the object created by spaCy, and add the token text, lemma, part-of-speech and other properties to the data list and save it to csv
    - Entity recognition: the model in the spaCy was trained from wikipedia
    - Sentiment analysis: we count the number of positive and negative words per sentence based on the english opinion lexicon shared by UIC.

text	text_lower	lemma	lemma_lower	part_of_s	is_alphabet	is_stopword
ribal	ribal	ribal	ribal	PROPN	TRUE	FALSE
Affairs	affairs	Affairs	affairs	PROPN	TRUE	FALSE
nment	nment	nment	nment	NOUN	TRUE	FALSE
of	of	of	of	ADP	TRUE	TRUE
India	india	India	india	PROPN	TRUE	FALSE
Photo	photo	Photo	photo	PROPN	TRUE	FALSE

text	text_lower	label
India Photo Credit	india photo credit	ORG
Ministry of Tribal Affairs	ministry of tribal affairs	ORG
Government of India	government of india	ORG
Ministry of Tribal Affairs	ministry of tribal affairs	ORG
Government of India	government of india	ORG
United Nations	united nations	ORG

Label Model Optimization

BabbleLabble vs. Snorkel

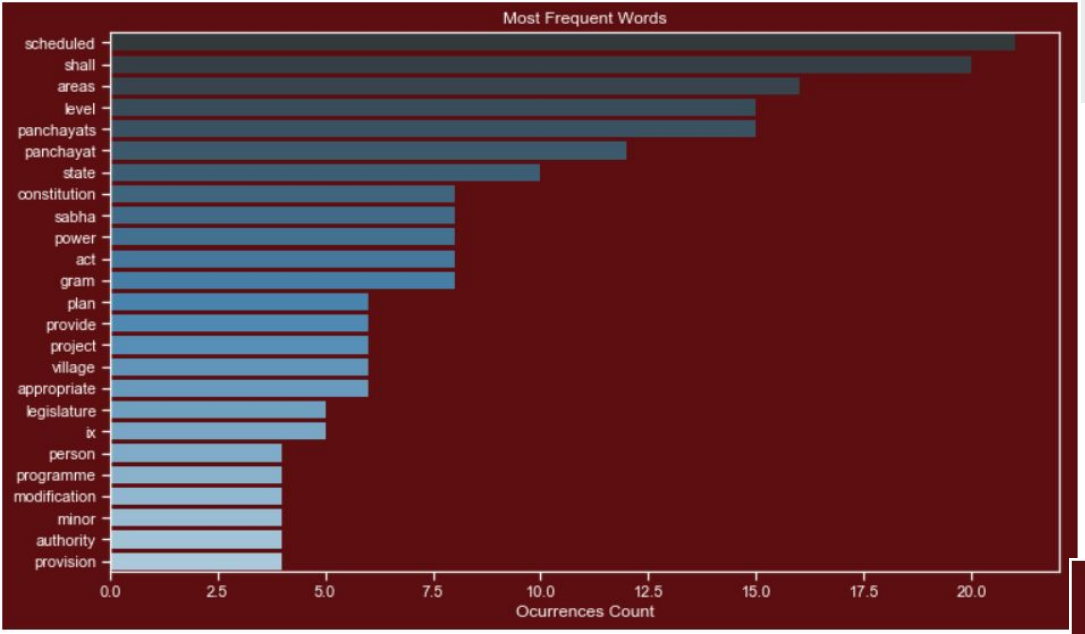
BabbleLabble is one of many projects exploring how weak supervision sources can be used to train machine learning systems

By researching on the data and expected outputs to classify, we could generate a set of labeled data and correlated explanations, which will help further develop our labeling functions.

In our following steps, we will try BabbleLabble to generate noisy label, and combing BabbleLabble with Snorkel to classify whether the policy documents provide information of financial incentives, discentives, or neutral information.

Snorkel learns generative model and later produces a set of probabilistic labels. There are 3 steps when implementing Snorkel:

1. Writing labeling functions
2. Modeling accuracies and correlations (generative model with re-weighted labeling functions)
3. Training discriminative models to develop large labeled dataset.

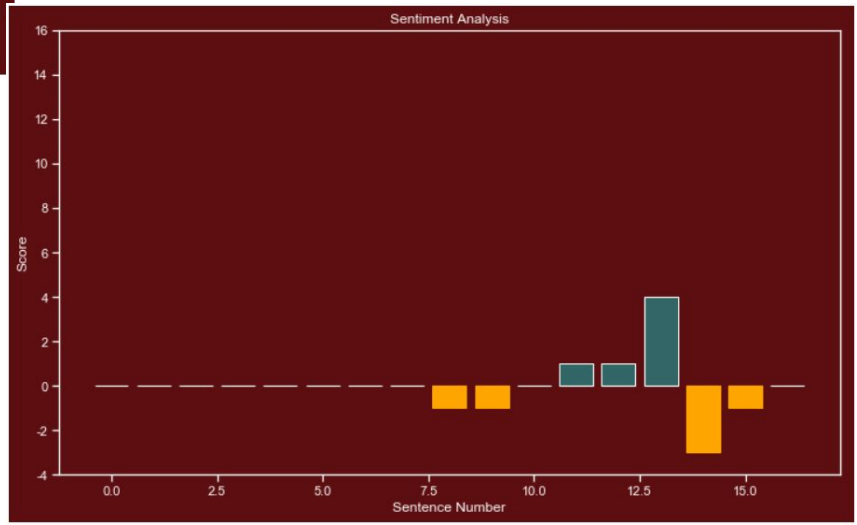


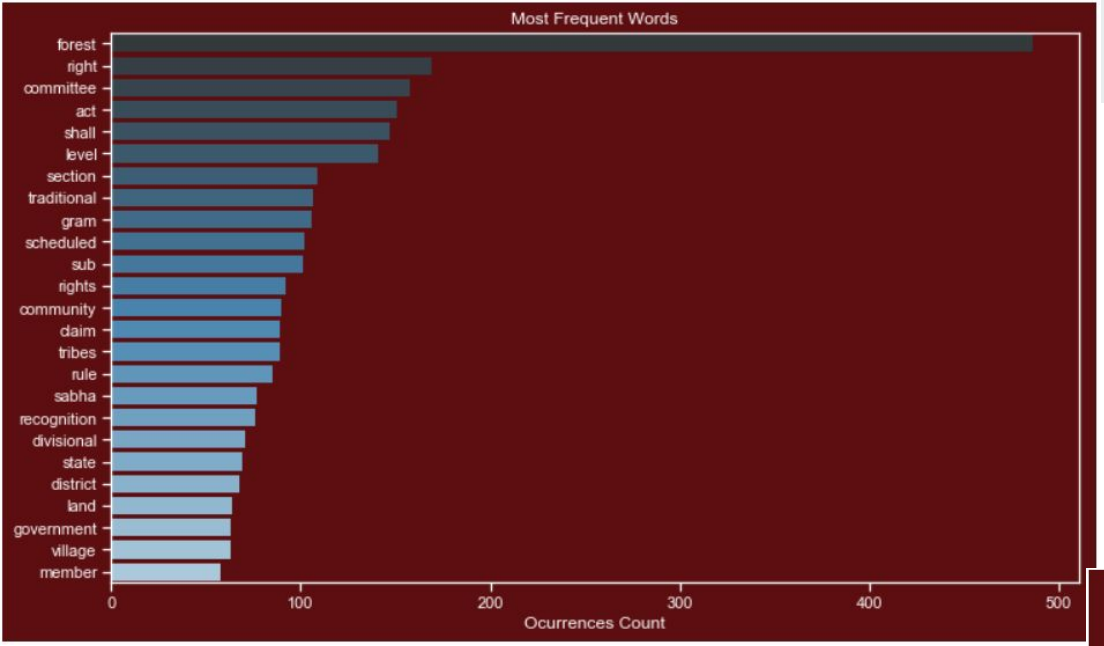
‘1.PESA.pdf’

“THE PROVISIONS OF THE PANCHAYATS (EXTENSION TO THE SCHEDULED AREAS) ACT”

For the comparison of words frequency, we only took into account the top 25 alphabet tokens that are longer than 1 character (not number/punctuation/space) and are not stop words.

For the sentiment analysis, we calculated the score of each sentence, and the score was decided by the number of positive words and negative words in the sentence.

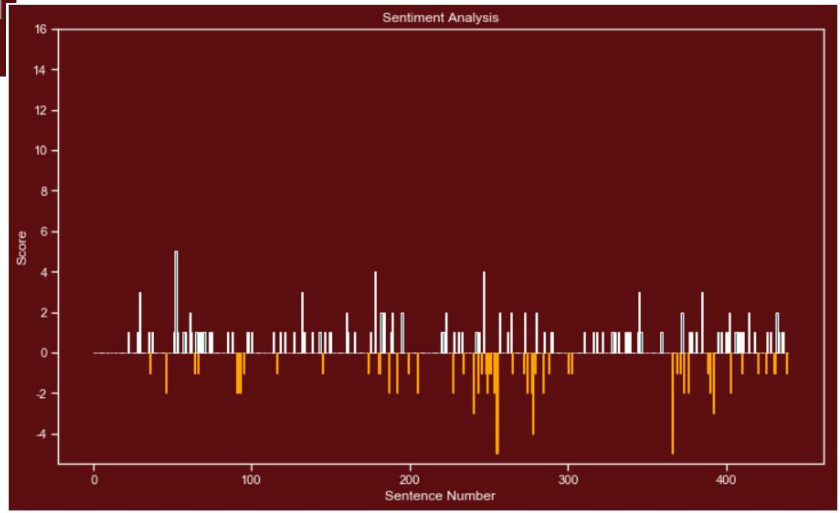


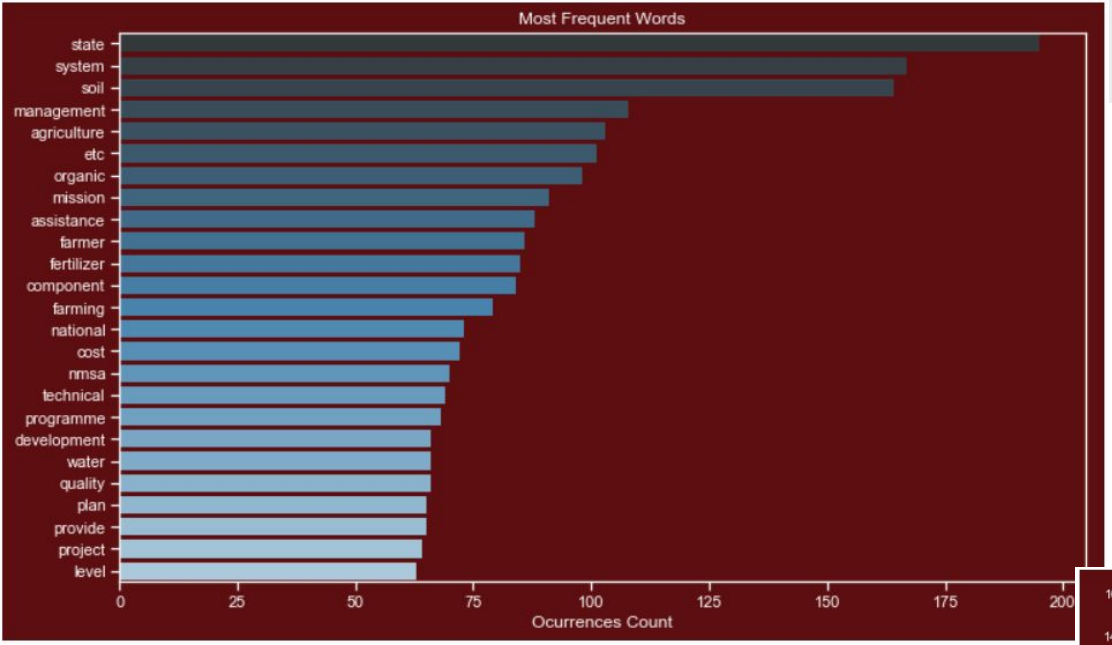


‘2.FRA.pdf’

“Forest Right Act 2006
——Act, Rules and Guidelines”

- First document: panchayats, sabha, village, areas, power
- Second document: tribe, village, sabha, land, forest



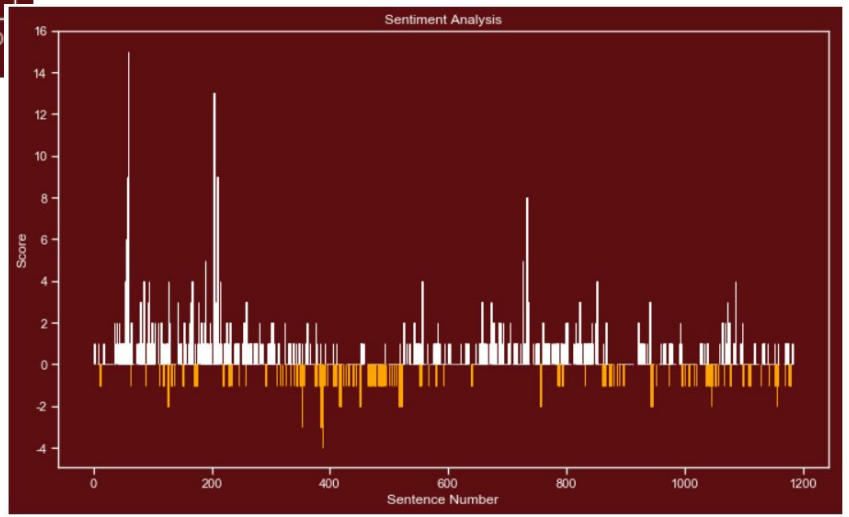


‘10.NMSA_Guildlines_English.pdf’

“National Mission for Sustainable Agriculture”

- Third document: soil, agriculture, farmer, fertilizer, farming

Third document has really high sentiment scores, which differentiated it from previous two documents.



Re-state understanding of the project

- Main Client: World Resources Institute (WRI)
  - An international research non-profit organization
  - Mission:
    - facilitate sustainable development of environment
    - Create chances for economic growth
    - Provide care for human health and future generations
- Project: Global Restoration Initiative
  - Main goal: help local and national governments implement restoration projects by statistical analysis
  - Framework benefits:
    - detect problems, conflicts, and associations across the restoration drafts
    - A versatile model that can be modified to scrutinize initiatives
    - A build-up tool to improve productivity and efficiency

Impression, Thoughts, Challenges

- Impression:
  - Raw Data
  - Transformed Raw Data
  - Gold Standard Data
  - Noisy Data with Snorkel (impression on data/project:intro of the articles)

- Thoughts:
  - Understand more about current policy strategy, forest restoration situation, and financing
  - Get familiar with the Babble Labble
  - Adopt additional feature engineering techniques
  - Conduct the data augmentation with synonym replacement

- Challenges:
  - A Better Way To Do Policy Analysis
    - Extract financial incentive mechanisms from policy documents
  - Lack of Experience for NLP

Phase 0 findings

- WRI is an international research non-profit organization
  - founded in 1982 with headquarter in D.C.
  - More than 1,000 experts in around 60 countries
- WRI collaborates with a wide range of organizations including national governments, privately owned businesses, and openly held companies, etc.
- Environmental research institutions with different focus:
  - Brookings Institution (research and education in the social sciences)
  - Urban Institute (economic and social policy research)
  - Pembina Institute (energy)

Research Questions:

- What
- What are the differences among the mechanisms of the restoration policies in these three countries?

Evaluation / impression of the data

- Review the work WRI has done as well as the results
  - We had a better understanding of the data WRI provided and purpose.
    - Raw data
    - Interim data
    - Processed data (gold_standard.csv and snorkel_noisy_proba.npy)
  - We started to get familiar with current pipeline: from noisy labeling, roBERTa encoding, to LSTM
    - Current achievements: 35 weak supervision strategies based on the position of certain words were built. A snorkel model was built based on these strategies.
    - Snorkel:
    - RoBERTa:
- Questions about the data:
  - The description and the content of the gold_standard.csv are inconsistent

Feature Engineering

- Next steps:
  - Challenges:
    - there are serial numbers(1.1, 1.2, 1.2.1) and page numbers in the catalog as well as main body content, and it's hard to throw these number away.
    - the text extracted from pdf is not clear: *"sustainable agriculture namiowat mision ron sustawanis acrcuirune orenamiona gudeunes national mission for sustainable agriculture"*
  - The methods we want to try in next steps:
    - Relationship extraction
    - Text summarization
    - Snorkel for sentiment analysis
    - Clustering
    - think about how to make good use of and give explanations to our result

Hypotheses and discussion of data analysis techniques

Feature Engineering

- Topic Modeling
- Name Entity Recognition
- SpaCy dependency parsing
- Hidden markov model
- Other possible features to consider: Ngrams/Cue Words/Part-of-speech Tags (count of noun/verb/adj) /LIWC features (number of words/positive words/negative words)

Improve Snorkel Model

- Currently using a simple LSTM to replace the DNN of Snorkel
- Possible direction: trying out different ML models including a more complex LSTM with Snorkel

Fine Tune roBERTa on noisy labels

Executive Summary

- **Challenges:**
  - How do we analyze policy?
  - Can Natural Language Processing help individuals understand what financing is available for sustainable development?
  - How can Natural Language Processing help standardize policy research while improving transparency and reproducibility?
- **Potential applications/benefits of the project outputs:**
  - Provide local organizations engaging country-level services with concentrated and accurate information on financing availability
  - Identify how restoration policy affects different socioeconomic and marginalized groups at country level in terms of financing availability
  - Identify how restoration financing mechanisms match up with agriculture subsidies for intensification