Weekly Report - Week 1 & 2

Bailey Russo

Letters Project

January 17, 2025

Time-Log

- What did you do this week?
    - Initial research into relevant literature
    - Completed Methods Research document
    - Went through code of previous iteration of this project
    - Implemented code to preprocess new datasets
- What are you going to do next week?
    - Setup Microsoft Planner to track project history
    - Create a repository for all members of project to share code
    - Perform first run of fine-tuning pretrained LLM on new datasets
- Blockers, things you want to flag, problems, etc.

Abstract

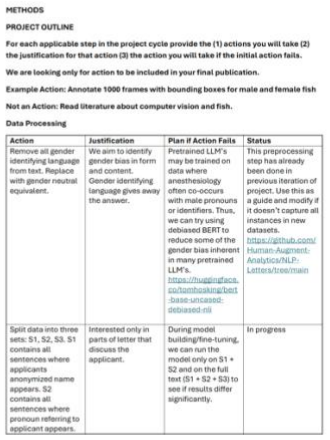
Jack Blandin and Ian A. Kash. 2024. Learning Fairness from Demonstrations via Inverse Reinforcement Learning. In Proceedings of the 2024 ACM Conference on Fairness, Accountability, and Transparency (FAccT '24). Association for Computing Machinery, New York, NY, USA, 51–61. https://doi.org/10.1145/3630106.3658539

Summary:

Defining fairness in algorithms can be difficult, especially when applying them to new situations. This research presents a new method that uses inverse reinforcement learning (IRL) to learn fairness preferences from human experts or existing algorithms, allowing these preferences to be applied in different classification tasks without needing extensive adjustments. This approach helps create fair classifiers in areas where fairness guidelines have not yet been clearly established.

What did you do and prove it

Screenshots of Methods Research document:



METHODS

PROJECT OUTLINE

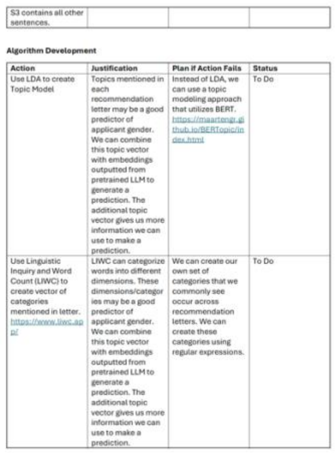
For each applicable step in the project cycle provide the (1) actions you will take (2) the justification for that action (3) the action you will take if the initial action fails.

We are looking only for action to be included in your final publication.

Example Action: Annotate 1000 frames with bounding boxes for male and female fish

Not an Action: Read literature about computer vision and fish.

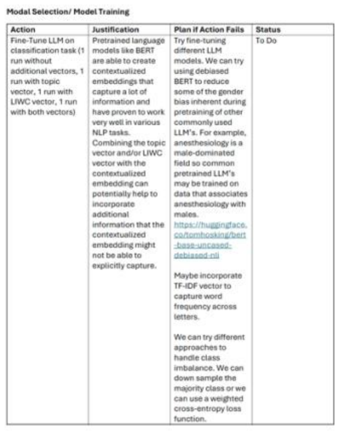
Data Processing



Action	Justification	Plan if Action Fails	Status
Remove all gender identifying language from text. Replace with gender neutral equivalent.	We aim to identify gender bias in form and content. Gender identifying language gives away the answer.	Pretrained LLM's may be trained on data where anesthesiology often co-occurs with male pronouns or identifiers. Thus, we can try using debiased BERT to reduce some of the gender bias inherent in many pretrained LLM's. https://huggingface.co/tomhosking/bert-base-uncased-debiased-nli	This preprocessing step has already been done in previous iteration of project. Use this as a guide and modify if it doesn't capture all instances in new datasets. https://github.com/Human-Augment-Analytics/NLP-Letters/tree/main
Split data into three sets: S1, S2, S3. S1 contains all sentences where applicants anonymized name appears. S2 contains all sentences where pronoun referring to applicant appears. S3 contains all other sentences.	Interested only in parts of letter that discuss the applicant.	During model building/fine-tuning, we can run the model only on S1 + S2 and on the full text (S1 + S2 + S3) to see if results differ significantly.	In progress

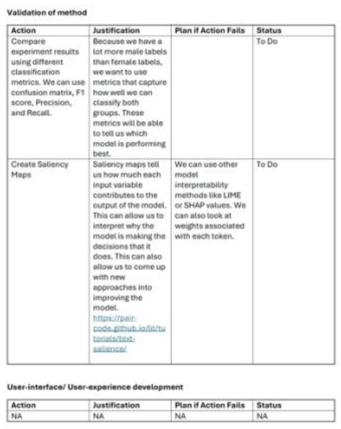
Algorithm Development

Action	Justification	Plan if Action Fails	Status
Use LDA to create Topic Model	Topics mentioned in each recommendation letter may be a good predictor of applicant gender. We can combine this topic vector with embeddings outputted from pretrained LLM to generate a prediction. The additional topic vector gives us more information we can use to make a prediction.	Instead of LDA, we can use a topic modeling approach that utilizes BERT. https://maartengr.github.io/BERTopic/index.html	To Do
Use Linguistic Inquiry and Word Count (LIWC) to create vector of categories mentioned in letter. https://www.liwc.app/	LIWC can categorize words into different dimensions. These dimensions/categories may be a good predictor of applicant gender. We can combine this topic vector with embeddings outputted from pretrained LLM to generate a prediction. The additional topic vector gives us more information we can use to make a prediction.	We can create our own set of categories that we commonly see occur across recommendation letters. We can create these categories using regular expressions.	To Do



Modal Selection/ Model Training

Action	Justification	Plan if Action Fails	Status
Fine-Tune LLM on classification task (1 run without additional vectors, 1 run with topic vector, 1 run with LIWC vector, 1 run with both vectors)	Pretrained language models like BERT are able to create contextualized embeddings that capture a lot of information and have proven to work very well in various NLP tasks. Combining the topic vector and/or LIWC vector with the contextualized embedding can potentially help to incorporate additional information that the contextualized embedding might not be able to explicitly capture.	Try fine-tuning different LLM models. We can try using debiased BERT to reduce some of the gender bias inherent during pretraining of other commonly used LLM's. For example, anesthesiology is a male-dominated field so common pretrained LLM's may be trained on data that associates anesthesiology with males. https://huggingface.co/tomhosking/bert-base-uncased-debiased-nli Maybe incorporate TF-IDF vector to capture word frequency across letters. We can try different approaches to handle class imbalance. We can down sample the majority class or we can use a weighted cross-entropy loss function.	To Do



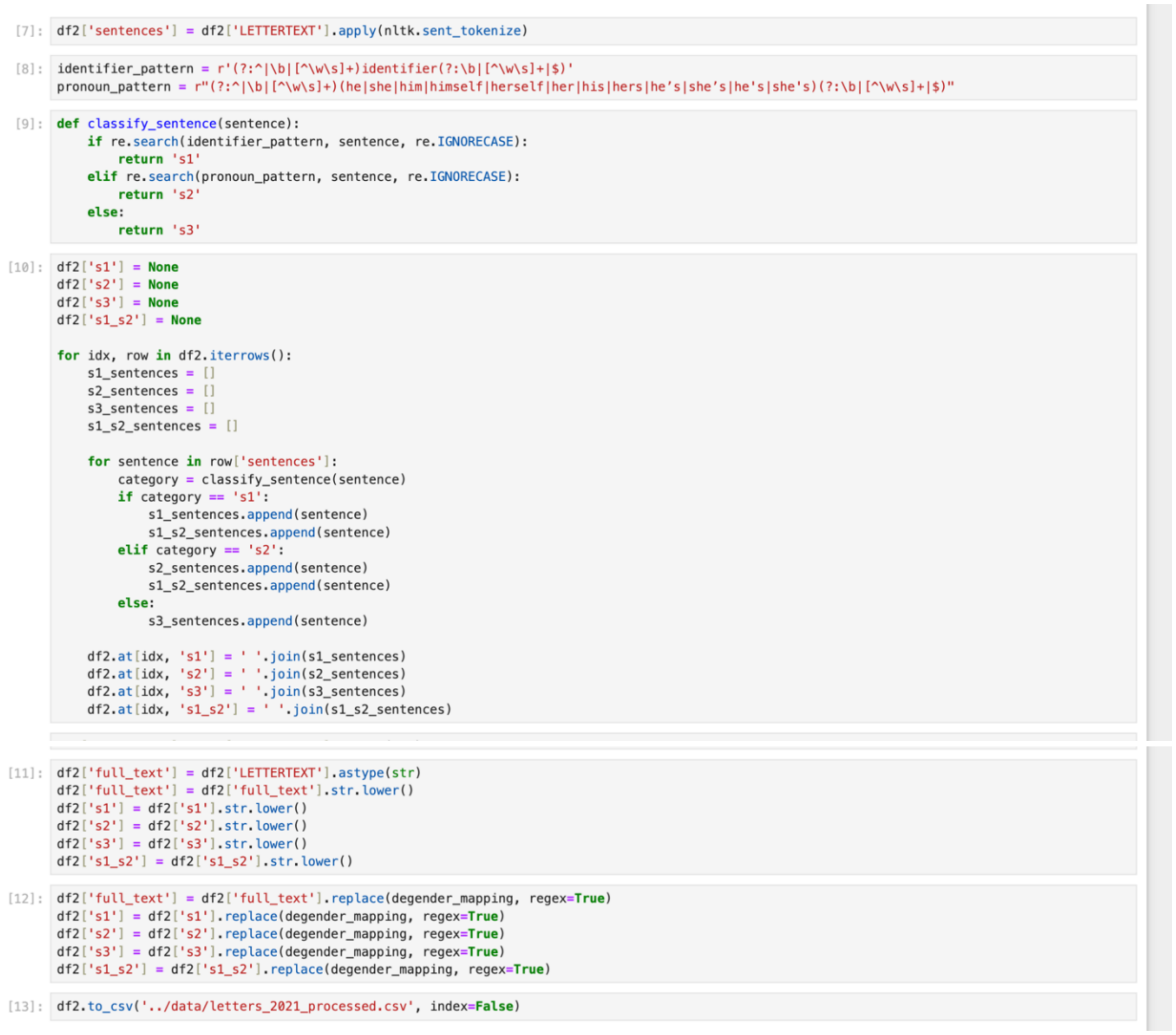
Validation of method

Action	Justification	Plan if Action Fails	Status
Compare experiment results using different classification metrics. We can use confusion matrix, F1 score, Precision, and Recall.	Because we have a lot more male labels than female labels, we want to use metrics that capture how well we can classify both groups. These metrics will be able to tell us which model is performing best.		To Do
Create Saliency Maps	Saliency maps tell us how much each input variable contributes to the output of the model. This can allow us to interpret why the model is making the decisions that it does. This can also allow us to come up with new approaches into improving the model. https://pair-code.github.io/lit/tutorials/text-salience/	We can use other model interpretability methods like LIME or SHAP values. We can also look at weights associated with each token.	To Do

User-interface/ User-experience development

Action	Justification	Plan if Action Fails	Status
NA	NA	NA	NA

Screenshot of code:



```
[7]: df2['sentences'] = df2['LETTERTEXT'].apply(nltk.sent_tokenize)
```

```
[8]: identifier_pattern = r'(?:^|\b|[^\w\s]+)identifier(?:\b|[^\w\s]+|$)'
     pronoun_pattern = r"(?:^|\b|[^\w\s]+)(he|she|him|himself|herself|her|his|hers|he’s|she’s|he's|she's)(?:\b|[^\w\s]+|$)"
```

```
[9]: def classify_sentence(sentence):
         if re.search(identifier_pattern, sentence, re.IGNORECASE):
             return 's1'
         elif re.search(pronoun_pattern, sentence, re.IGNORECASE):
             return 's2'
         else:
             return 's3'
```

```
[10]: df2['s1'] = None
      df2['s2'] = None
      df2['s3'] = None
      df2['s1_s2'] = None

      for idx, row in df2.iterrows():
          s1_sentences = []
          s2_sentences = []
          s3_sentences = []
          s1_s2_sentences = []

          for sentence in row['sentences']:
              category = classify_sentence(sentence)
              if category == 's1':
                  s1_sentences.append(sentence)
                  s1_s2_sentences.append(sentence)
              elif category == 's2':
                  s2_sentences.append(sentence)
                  s1_s2_sentences.append(sentence)
              else:
                  s3_sentences.append(sentence)

          df2.at[idx, 's1'] = ' '.join(s1_sentences)
          df2.at[idx, 's2'] = ' '.join(s2_sentences)
          df2.at[idx, 's3'] = ' '.join(s3_sentences)
          df2.at[idx, 's1_s2'] = ' '.join(s1_s2_sentences)
```

```
[11]: df2['full_text'] = df2['LETTERTEXT'].astype(str)
      df2['full_text'] = df2['full_text'].str.lower()
      df2['s1'] = df2['s1'].str.lower()
      df2['s2'] = df2['s2'].str.lower()
      df2['s3'] = df2['s3'].str.lower()
      df2['s1_s2'] = df2['s1_s2'].str.lower()
```

```
[12]: df2['full_text'] = df2['full_text'].replace(degender_mapping, regex=True)
      df2['s1'] = df2['s1'].replace(degender_mapping, regex=True)
      df2['s2'] = df2['s2'].replace(degender_mapping, regex=True)
      df2['s3'] = df2['s3'].replace(degender_mapping, regex=True)
      df2['s1_s2'] = df2['s1_s2'].replace(degender_mapping, regex=True)
```

```
[13]: df2.to_csv('../data/letters_2021_processed.csv', index=False)
```

This code breaks up the recommendation letter text into three sets: S1, S2, and S3. S1 consists of sentences that use the applicants anonymized name, S2 consists of pronouns that refer to the applicant, and S3 consists of all other sentences. The text is further processed by replacing all gender-identifying language with a gender-neutral equivalent.