Writing Evaluation Project

Erdös Institute Bootcamp, Spring 2022

Mariner Team

Irati Hurtado, Konstantinos Karatapanis, and Sammy Sbiti

Project description

- The current project comes from a Kaggle Competition titled "Evaluating Student Writing".
- The dataset contains argumentative essays written by U.S students in grades 6-12.
- The essays were annotated by expert raters for elements commonly found in argumentative writing.
- The goal of the project is to predict human annotations.

Project description (cont’d)

- Each essay must be segmented into discourse elements.
- Each discourse element must be classified as one of the following:
    - Lead
    - Position
    - Claim
    - Counterclaim
    - Rebuttal
    - Evidence
    - Conclusing statement

Exploratory data analysis

Overall number of discourse elements

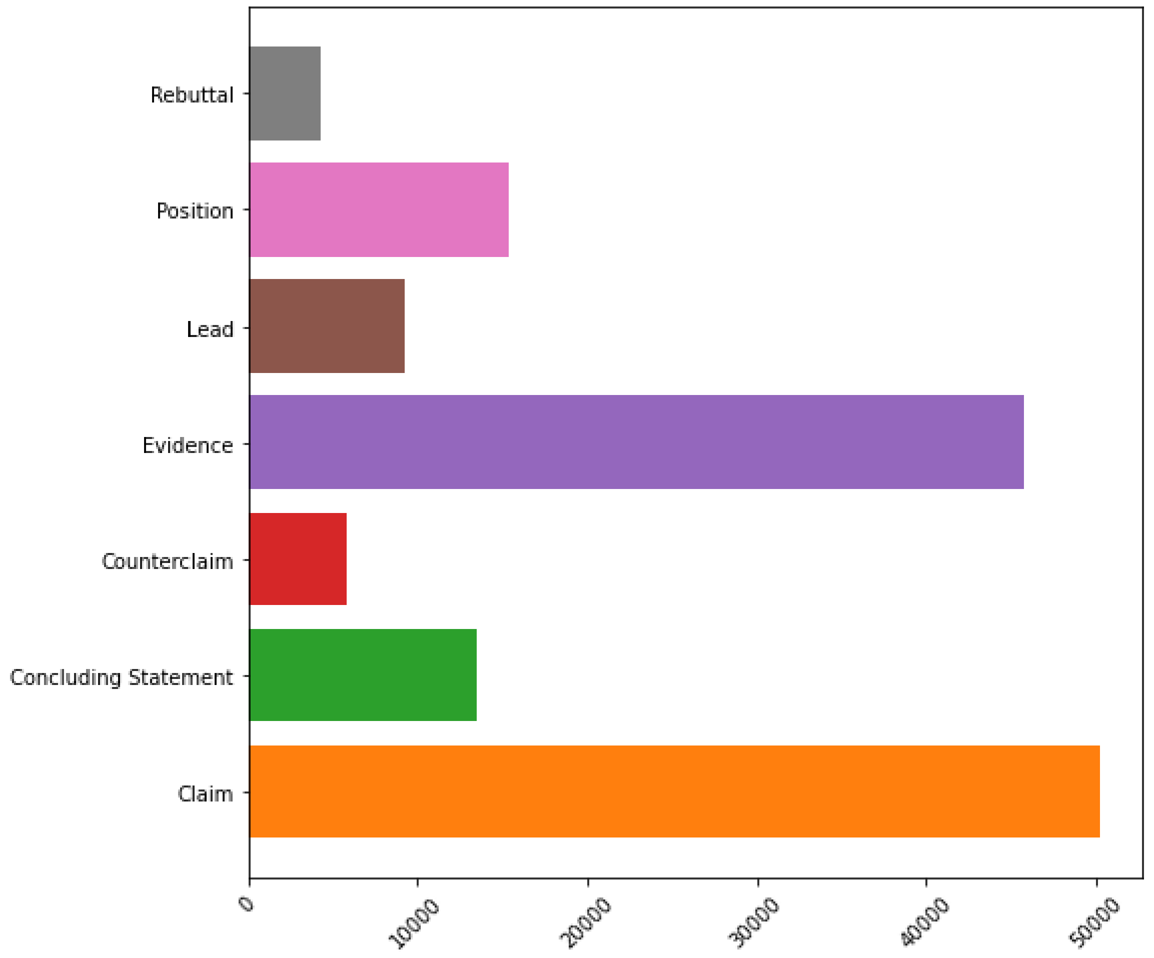
Claim and Evidence predominate

Overall number of discourse elements

Claim and Evidence predominate

In [4]:

```
df_elements = df.groupby(['discourse_type']).size().reset_index()
fig, ax = plt.subplots(1)
fig.set_size_inches(8,8)
ax.barh(df_elements['discourse_type'], df_elements[0], color = ['C1', 'C2', 'C3', 'C4', 'C5', 'C6', 'C7'])
ax.tick_params(axis='x', labelrotation = 45)
```


Rebuttal
Position
Lead
Evidence
Counterclaim
Concluding Statement
Claim
0
10000
20000
30000
40000
50000

Distribution of discourse elements per essay

The previous imbalance is also observed at the essay level

Distribution of discourse elements per essay

The previous imbalance is also observed at the essay level

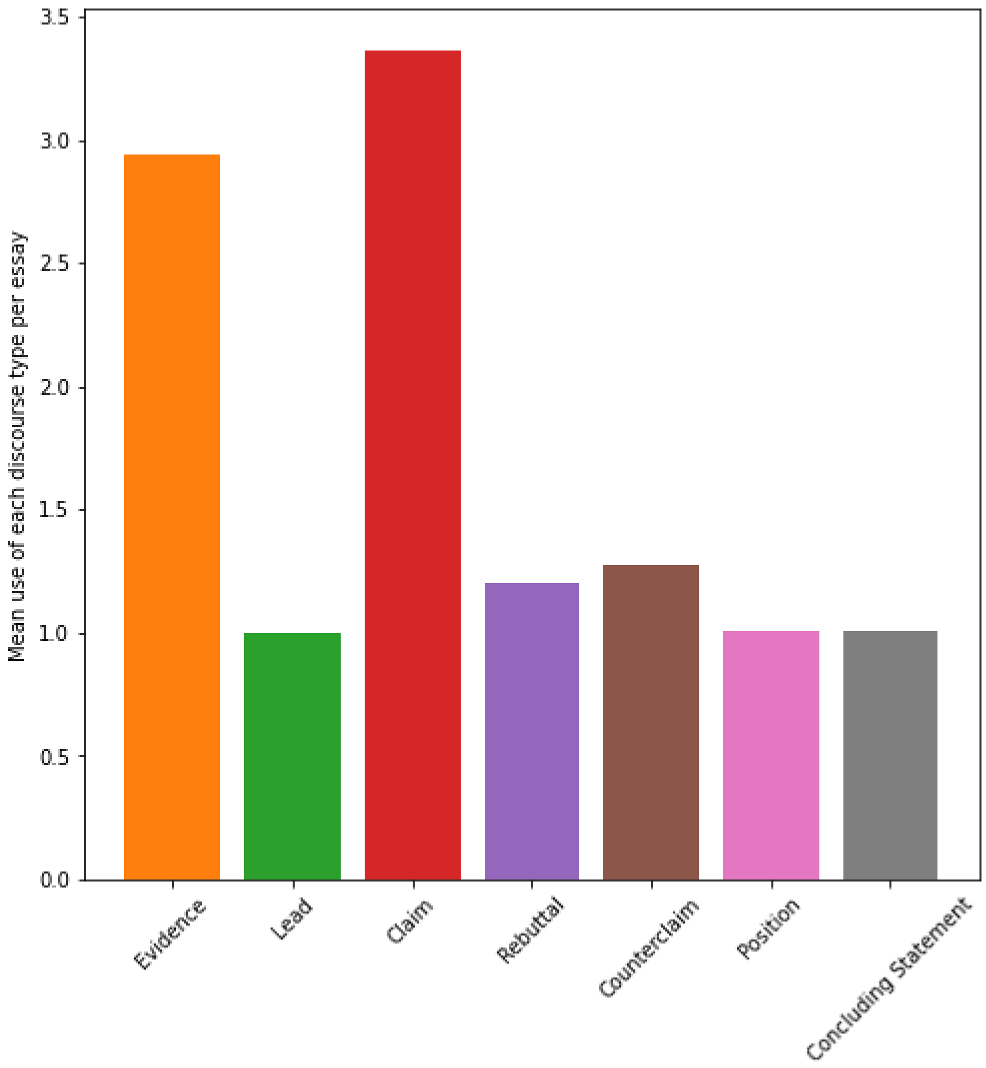
In [5]:

```
# Step 1: Grouping by id and discourse type
df_ind_distribution = df.groupby(['id','discourse_type']).size().reset_index()

# Step 2: Creating a pivot table based on column values
df_ind_distribution = df_ind_distribution.pivot(index = 'id', columns='discourse_type', values = 0)

# Step 3: Calculating descriptive stats for each discourse type
distribution_summary = df_ind_distribution.describe()
distribution_summary['stats'] = ['count', 'mean', 'std', 'min', '25%', '50%', '75%', 'max']
distribution_summary = (distribution_summary.melt(id_vars=["stats"],
                           value_vars =["Claim",'Concluding Statement','Counterclaim',
                                       'Evidence', 'Lead', 'Position', 'Rebuttal'],value_name="values")
 .replace('', np.nan,)
 .dropna()
 .sort_values('stats'))
distribution_summary = distribution_summary[distribution_summary['stats'] == 'mean']

# Step 4: Visualize
fig, ax = plt.subplots(1)
fig.set_size_inches(8,8)
ax.bar(distribution_summary['discourse_type'], distribution_summary['values'], color = ['C1', 'C2', 'C3', 'C4', 'C5', 'C6'
ax.set_ylabel('Mean use of each discourse type per essay')
ax.tick_params(axis='x', labelrotation = 45)
```


Mean use of each discourse type per essay
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Evidence
Lead
Claim
Rebuttal
Counterclaim
Position
Concluding Statement

Data pre-processing

- All essays were tokenized.
- Each token received a number from 0 to 7 based on the type of discourse element it belonged to (according to the training dataset):
  - 0 = Lead
  - 1 = Position
  - 2 = Evidence
  - 3 = Claim
  - 4 = Concluding statement
  - 5 = Counterclaim
  - 6 = Rebuttal
  - 7 = None of the above

Model form

- The basic form of our model takes as input the list of tokens in a given document, and outputs the class probability vectors for each token.

- We use the pre-trained `Longformer` model from the transformers module as a first layer in a neural network.
- The longformer is a pretrained word embedding based off the well known transformer `BERT`, however its attention mechanism scales linearly with document length as opposed to `BERT` which scales quadratically, making `Longformer` useful for analyzing large documents.
- We then feed the word embeddings to a dense layer followed by a classification layer using softmax to output class probabilities for each token.
- We then use these class probabilities along with certain prior assumptions about the distribution of discourse elements throughout the essays to predict classes at the sentence level.

In [9]:

```
from IPython.display import Image
Image("slide.png")
```

Out[9]:

Color codes:
Nothing
Lead
Claim
Counterclaim
Rebuttal
Evidence
Position
Concluding Statement

When people ask for advice, they often talk to more
than one person. For example, when you are in need
of someone in a difficult situation, you probably talk to
more than one person to see who has the best
answer to your problem. I think seeking multiple opinions is
better because it gives you a choice on what you
want to do,you have another piece of advice that you
can rely on, and it can help you see that
the first opinion does not make any sense First, if
you get more than one opinion you get to choose
what you feel is the best option for your problem.
I remember when i was young i only ever asked
one person for advice and that was my mom. But
she didnt always have the best advice and i paid
the price for that one time. I was really small
then and i asked my mom if i could play
tackle football and she said that i could. but I
was way too small back then so i broke my
leg on my first play. but if i asked for
another opinion i probably would've played one more season of
flag football. Next, if you get more than one opinion,
you have another piece of advice that you can rely
on. If you have a opinion that sounds really bad
but you have another opinion that sounds really good, you
go with the good advice. but if you never asked

Results

- We use the metric provided by Kaggle to evaluate performance.
- We calculate the F1 score for each class and then take the average across all classes.
- The F1 score is the harmonic mean of precision and recall, calculated by

$$2\frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

Discourse Type	**F1-score**
Claim	.084
Concluding Statement	.32
Counterclaim	0.0
Evidence	.56
Lead	.42
Position	.21
Rebuttal	0.0
Avg. across classes	.23

User interface

- We further developed a web app on `streamlit` that allows users to input their essays and outputs an annotated essay.
- We hope this can be helpful to students who want quick feedback on the structure of their argumentative writing.
- We imagine that it could be used as a part of a larger program for providing feedback on students' writing.

```
In [ ]:
```