Identification of Human Values Behind Arguments

NLP Standard Project A.Y. 2022/2023
University of Bologna – Artificial Intelligence

Matteo Nestola, Simona Scala and Sara Vorabbi

Data Loading

- Loading the data from Zenodo repository
- Merging the data from ‘arguments-x.tsv’ and ‘labels-x.tsv’
- Splitting the data into Training, Validation and Test set

Data Loading

- Loading the data from Zenodo repository
- Merging the data from ‘arguments-x.tsv’ and ‘labels-x.tsv’
- Splitting the data into Training, Validation and Test set

- arguments-test-nahjalbalagha.tsv
- arguments-test.tsv
- arguments-training.tsv
- arguments-validation-zhihu.tsv
- arguments-validation.tsv
- labels-training.tsv
- labels-validation-zhihu.tsv
- labels-validation.tsv

Repository contents

```
df_arguments_train = pd.read_csv(arguments_train_path, sep='\t')
df_labels_train = pd.read_csv(labels_train_path, sep='\t')
df_train = pd.merge(df_arguments_train, df_labels_train, on='Argument ID')

df_arguments_val = pd.read_csv(arguments_val_path, sep='\t')
df_labels_val = pd.read_csv(labels_val_path, sep='\t')
df_val = pd.merge(df_arguments_val, df_labels_val, on='Argument ID')

df_arguments_test = pd.read_csv(arguments_test_path, sep='\t')
df_labels_test = pd.read_csv(labels_test_path, sep='\t')
df_test = pd.merge(df_arguments_test, df_labels_test, on='Argument ID')
```

Code to merge and split the data

Data Loading

- Loading the data from Zenodo repository
- Merging the data from ‘arguments-x.tsv’ and ‘labels-x.tsv’
- Splitting the data into Training, Validation and Test set

Conclusion	Stance	Premise
We should ban human cloning	in favor of	we should ban human cloning as it will only ca...

Input columns

```
print("Shape of the Training Dataset: ", df_train.shape)
print("Shape of the Validation Dataset: ", df_val.shape)
print("Shape of the Test Dataset: ", df_test.shape)
```

```
Shape of the Training Dataset:  (5393, 24)
Shape of the Validation Dataset:  (1896, 24)
Shape of the Test Dataset:  (100, 24)
```

Code to print the shape of each dataset

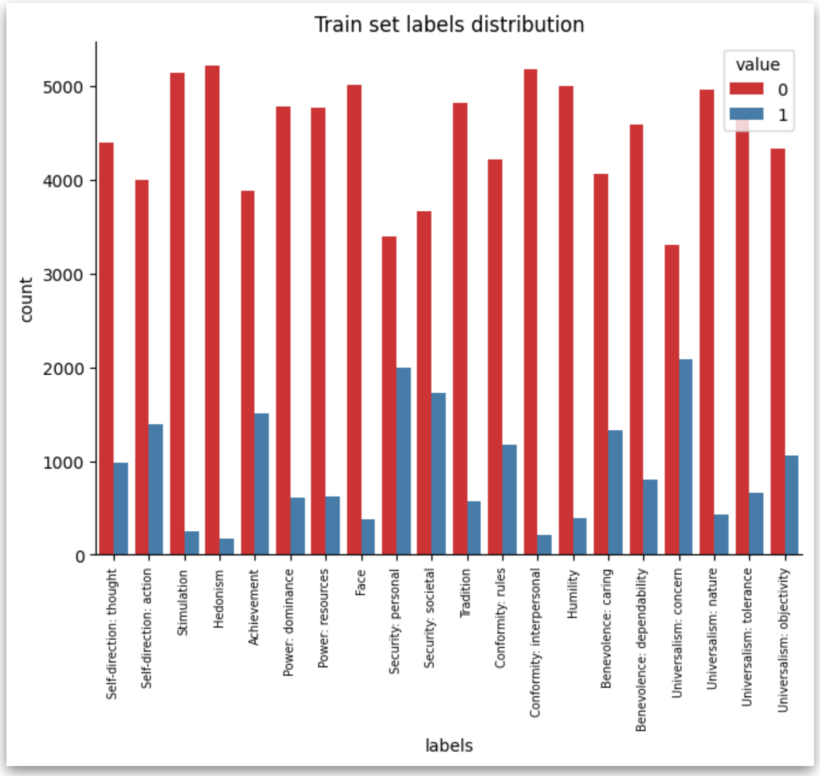
Data Exploration

- value_counts operations
- Seaborn’s catplots to show labels distributions

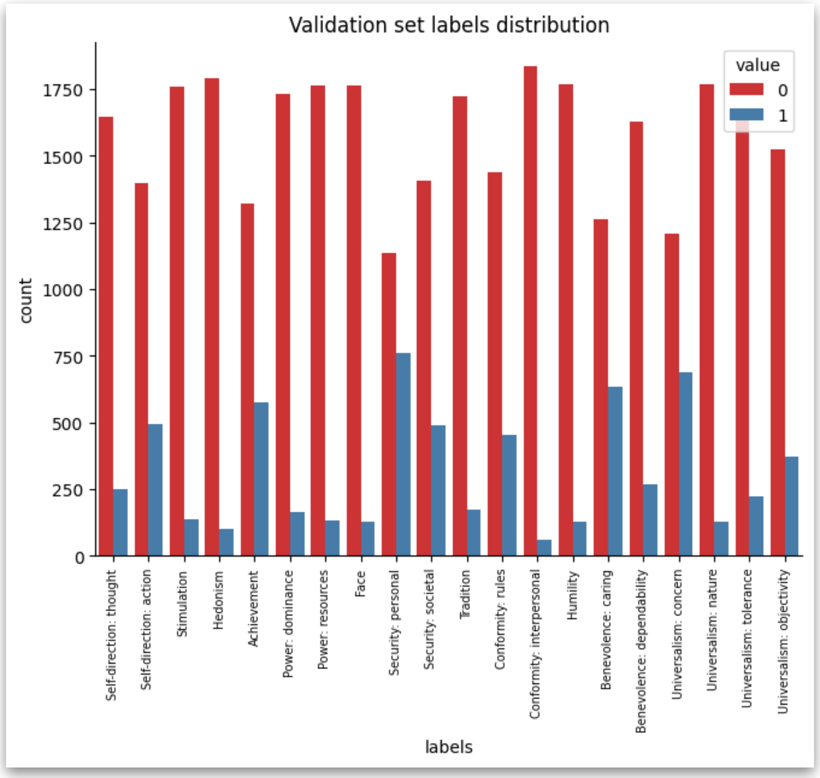
```
columns = df_train.columns.tolist()

print("Distribution of pos/neg samples for each label in the training set: ")
print()
for c in columns[4:]:
  print(df_train[c].value_counts())
  print()
```

Code to count positive and negative samples for each label in the training set

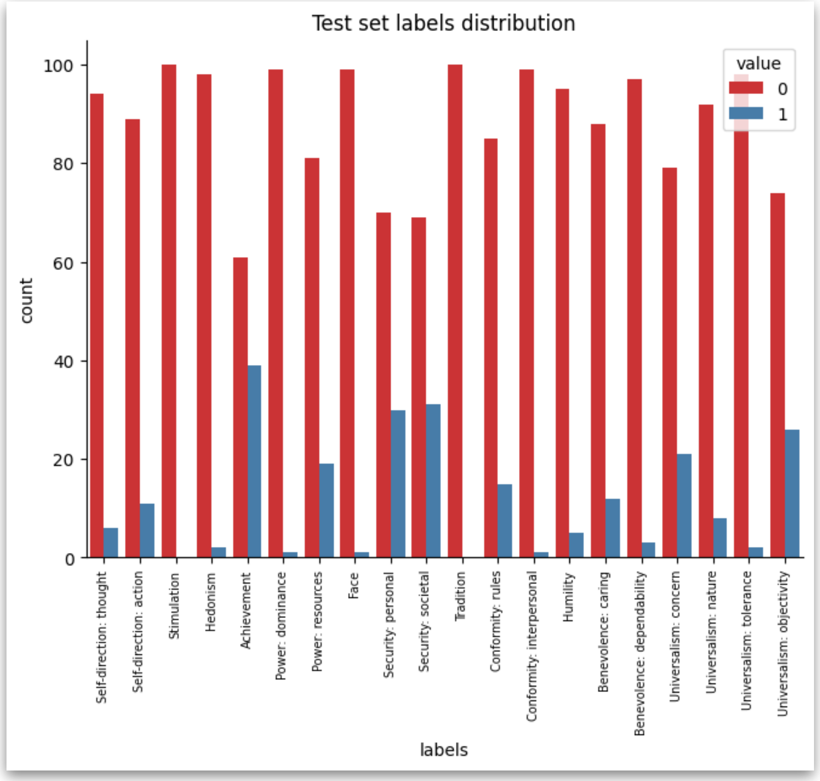
Data Exploration



Distribution of the labels occurrences in the training set



Distribution of the labels occurrences in the validation set



Distribution of the labels occurrences in the test set

Data pre-processing

Thanks, Federico! 🍻

- Pre-processing pipeline with the lab functions
- Redefinition of the stop words

```
good_stopwords = ['favor','against']

try:
    stopwords = set(stopwords.words('english'))
    stopwords = stopwords - set(good_stopwords)
except LookupError:
    nltk.download('stopwords')
    stopwords = set(stopwords.words('english'))
    stopwords = stopwords - set(good_stopwords)
```

Code to keep “favor” and “against” from the stop words

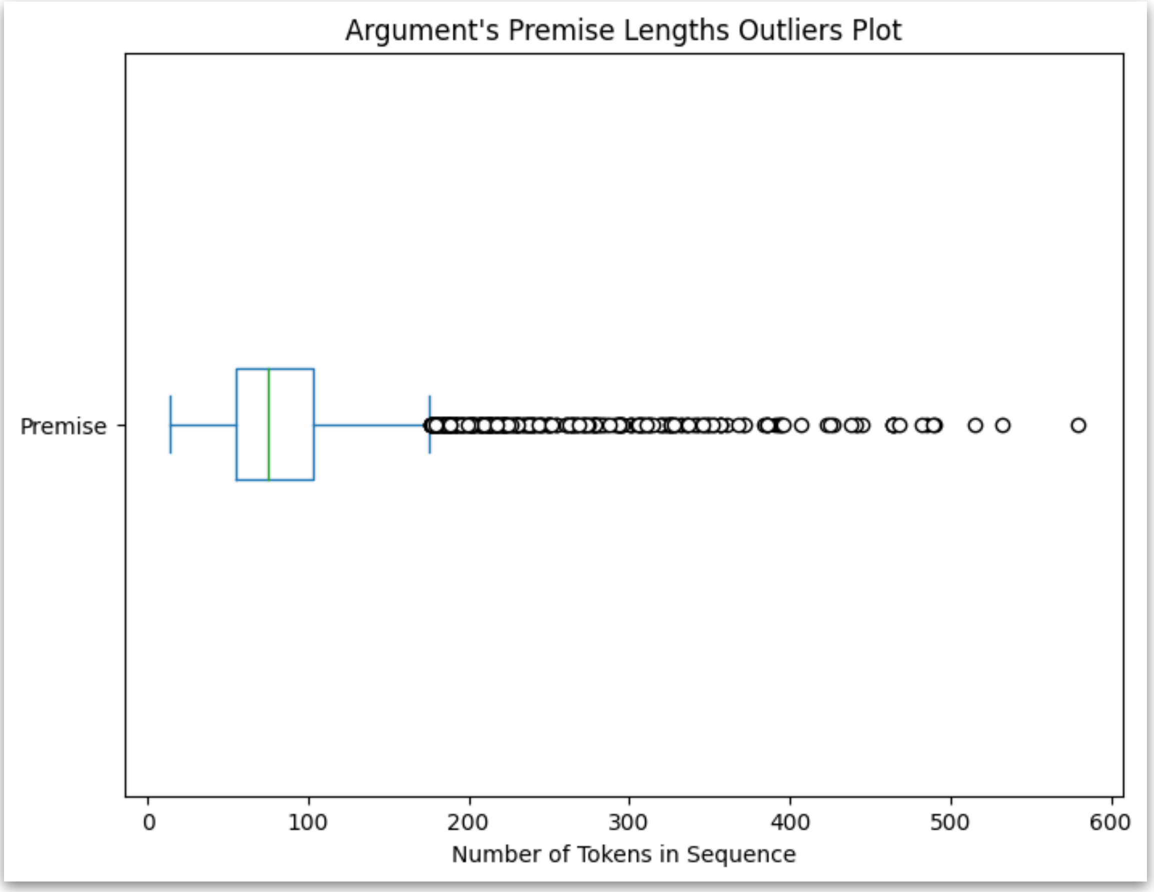
Conclusion	Stance	Premise
We should ban human cloning	in favor of	we should ban human cloning as it will only ca...

Conclusion	Stance	Premise
ban human cloning	favor	ban human cloning cause huge issues bunch huma...

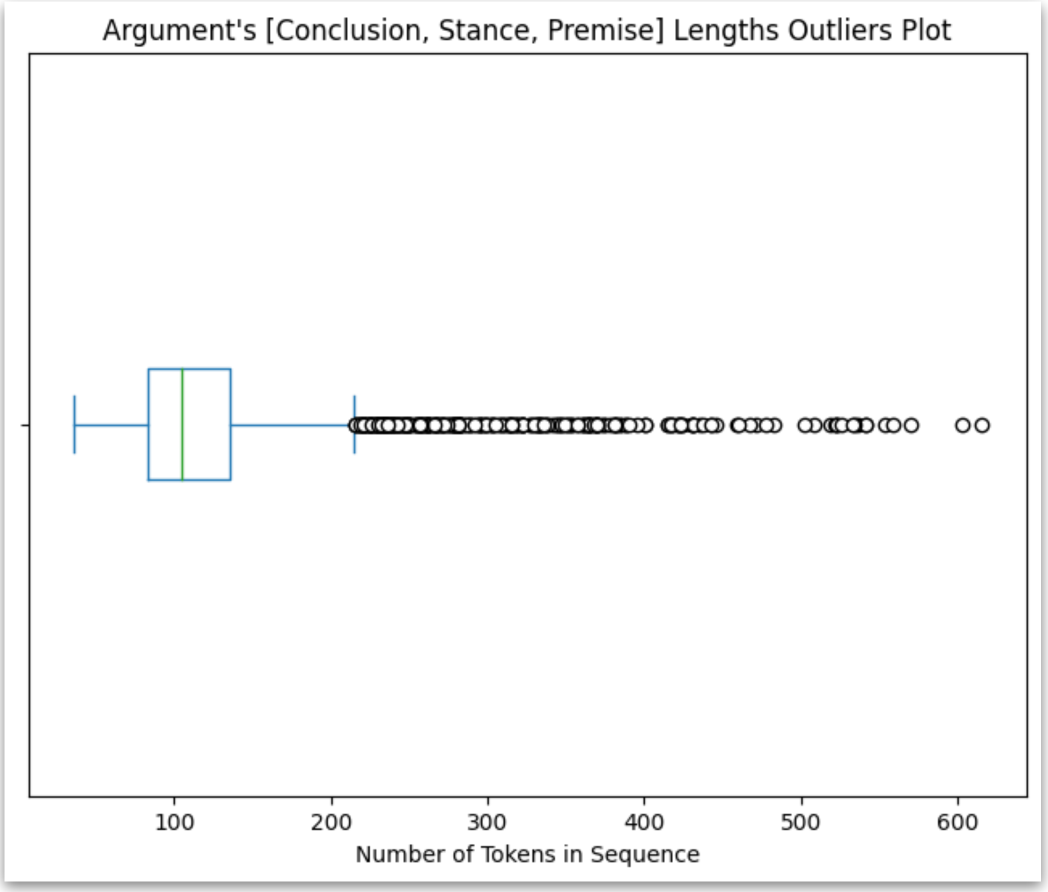
Text before and after pre-processing

Choosing the max length value

- Box plot to show the sentences’ lengths
- Set max_value to a value that covers the 95% of all the sentences



Box plot that shows the lengths of Premise sentences



Box plot that shows the lengths of Conclusion+Stance+Premise sentences

Tokenization

- Initialize an AutoTokenizer.from_pretrained()
- Three model checkpoints
  - bert-base-uncased
  - roberta-base
  - allenai/multicite-multilabel-scibert

Model Definition

- Initialize an AutoModelForSequenceClassification.from_pretrained()
- Three model checkpoints
  - bert-base-uncased
  - roberta-base
  - allenai/multicite-multilabel-scibert

Experimental Setup

Global variables to run the experiments

- Two global variables
  - model_name
  - concat

```
concat = False
#concat = True
```

Code to define the global variable ‘concat’

```
model_names = [
              #"bert-base-uncased",
              #"roberta-base",
               "allenai/multicite-multilabel-scibert"
              ]

model_name = model_names[0]
```

Code to define the global variable ‘model_name’

Experimental Setup

Evaluation metrics

- Define a compute_metrics function
- Computes the evaluation metrics based on the best threshold

```python
def compute_metrics(eval_predictions):

  best_f1 = 0.0
  best_precision = 0.0
  best_recall = 0.0

  for threshold in np.arange(0.1, 0.95, 0.05):

    predictions, labels = eval_predictions

    # Normalize the predictions
    predictions = torch.from_numpy(predictions).sigmoid().numpy()

    # Convert the predicted probabilities to binary labels
    eval_preds = np.where(predictions > threshold, 1.0, 0.0)

    # Compute the evaluation metrics
    f1 = f1_score(labels, eval_preds, average='macro', zero_division=0)
    precision = precision_score(labels, eval_preds, average='macro', zero_division=0)
    recall = recall_score(labels, eval_preds, average='macro', zero_division=0)

    # Update the threshold based on the best F1 score seen so far
    if f1 > best_f1:
        best_f1 = f1
        best_precision = precision
        best_recall = recall
        global best_threshold
        best_threshold = threshold

  return {'precision': best_precision, 'recall': best_recall, 'f1': best_f1}
```

Code to define the ‘compute_metrics’ function

Experimental Setup

Training

- Hyperparameters setting
  - batch_size = 128
  - num_train_epochs = 20
  - lr = 2e-5
  - weight_decay = 0.1
  - seed = 42
- Define a Custom Trainer to compute the BCEWithLogitsLoss()

Experimental Setup

🏃💨 Run!

- Two experiments
  - concat = True
  - concat = False

Results

Model	Precision	Recall	F1-score	Best Threshold	Concatenation
bert-base-uncased	0.3059	0.5776	0.3790	0.15	False
bert-base-uncased	0.3271	0.5481	0.3681	0.15	True
roberta-base	0.3462	0.5892	0.4203	0.15	False
roberta-base	0.3327	0.6189	0.4246	0.15	True
multicite-multilabel-scibert	0.3176	0.5867	0.3948	0.15	False
multicite-multilabel-scibert	0.3060	0.5861	0.3855	0.15	True

Table 1: Results of the Evaluation

Model	F1-score	Best Threshold	Concatenation
bert-base-uncased	0.2657	0.15	False
bert-base-uncased	0.2732	0.15	True
roberta-base	0.2761	0.15	False
roberta-base	0.2752	0.25	True
multicite-multilabel-scibert	0.2704	0.20	False
multicite-multilabel-scibert	0.2672	0.30	True

Table 2: Results of the Prediction

Inference

```python
def inference(my_sentence, threshold):

  # Pre-process the sentence
  input_text = text_prepare(my_sentence)

  # Tokenize it
  input_text = tokenizer(input_text, return_tensors = 'pt' ) #dizionario con key 'input_ids'

  for key in input_text:
    input_text[key] = input_text[key].to(device)

  # Give it to the model
  with torch.no_grad():
    output_predictions = model(input_text['input_ids'])
    output_predictions = {key: output.cpu() for key, output in output_predictions.items()}

  # Normalize and binarize the output
  output_predictions = output_predictions['logits'].sigmoid().numpy()
  output_predictions = np.where(output_predictions > threshold, 1.0, 0.0)

  # Use the lab2input dictionary to obtain the predicted labels
  output_labels = np.where(output_predictions == 1.0)

  # Transform them in a list of indeces
  output_labels = output_labels[1].tolist()
  output_labels = [id2label[idx] for idx in output_labels]
  print("The predicted labels are:\n", output_labels)
```

Code to define the ‘inference’ function which uses one of the models to classify any sentence

Inference

- concat = False
- model_name = “allenai/multicite-multilabel-scibert”
- my_sentence = “We should legalize the use of marijuana since it is shown in scientific studies to have a beneficial effect for people with anxiety conditions.”

```
inference(my_sentence, best_threshold)

The predicted labels are:
 ['Self-direction: action', 'Security: personal', 'Benevolence: caring', 'Universalism: objectivity']
```

The output of the function ‘inference’

Inference

- Self-direction: action – It is good to determine one's own actions.
- Security: personal – It is good to have a secure immediate environment.
- Benevolence: caring – It is good to work for the welfare of one's group's members.
- Universalism: objectivity – It is good to search for the truth and think in a rational and unbiased way.

Thanks for your attention! 🤖