Human value detection

Ferraro Francesca

11/12/2023

Step 1 - Dataset Loading

Arguments from dataset are loaded as the union of the training, validation and test data from the original dataset. Each one of the tab-separated columns (Premise, Stance and Conclusion) is considered for the goal of the project.

```python
def load_dataset():
    # Load the dataset into separate DataFrames for each split
    df_training = pd.read_csv('arguments-training.tsv', delimiter='\t')
    df_validation = pd.read_csv('arguments-validation.tsv', delimiter='\t')
    df_test = pd.read_csv('arguments-test.tsv', delimiter='\t')

    # Concatenate all the dataframes
    df = pd.concat([df_training, df_validation, df_test])

    # Extract the argument text from each DataFrame
    arguments = df['Premise'].tolist()
    stances = df['Stance'].tolist()
    conclusions = df['Conclusion'].tolist()

    return arguments, stances, conclusions
```

Step 2 - Preprocessing and model initialization

Preprocessing is strongly linked to the choose of the model, in this case, with bert-base-uncased, preprocessing is almost unnecessary. Therefore, the model is loaded.

```
# model inizialization
model_bert = pipeline('fill-mask', model='bert-base-uncased') # Bert
```

Step 3 - Argument preparation

Each premise is mapped to the corresponding stance and conclusion to be easily used in the prompt construction.

Then a random argument is chosen from the dataset as an example.

```
# Map each argument to the corresponding stance and conclusion
if len(arguments) != len(stances) or len(arguments) != len(conclusions):
    print("Error: incompatible data")
else:
    input = {}
    for i in range(len(arguments)):
        input[arguments[i]] = (stances[i], conclusions[i])
```

```
import random

# Random argument selection
input_argument = random.choice(list(input.keys()))

argument = input_argument
stance = input[input_argument][0]
conclusion = input[input_argument][1]
```

Step 4 - Prompt and word generation

Prompt is constructed using argument's premise, stance and conclusion. Several kinds of prompt were tried and the best results to be the one with a first-person sentence, but others are worth exploring.

The model predicts a series of masked words for describing the person associated to the argument and the one with the highest score is chosen.

```
def generate_word(model) :
    # Create prompt
    prompt = f"i am {stance} the fact that {conclusion} because i think that {argument}. i am a {model.tokenizer.mask_token}."
    # Generate filling
    output = model(prompt)

    # Sort output by 'score'
    output.sort(key=lambda x: x['score'], reverse=True)

    # Extract 'sequence' with higher 'score'
    description = output[0]['sequence']

    return description
```

```
description_bert = generate_word(model_bert)
print(description_bert)
```

```
i am against the fact that the european union should be harder towards irreparable migrants those who do not value their lives in peace security and
prosperity because i think that migrants have an enormous amount to offer in energy new ideas and cultures why this emphasis on imaginary takeovers.
i am a socialist.
```

Results

Here are some examples of the model's predictions. This is not a representative sample, but rather a selection of examples. Some of the predictions appear to be appropriate, while others don't.

PROMPT	PREDICTION	EVALUATION
I am in favor of the fact that payday loans should be banned because I think that they charge a crazy amount of interest on their loans driving many people into serious debt.	Libertarian	Appropriate
I am in favor of the fact that we should enforce the rule of law, promoting respect for human rights without linking them to any alleged " European " identity.	Conservative	Appropriate
I am against the fact that the EU should immediately end the refugee deal with Turkey.	Conservative	Appropriate
I am in favor of the fact that we should ban naturopathy because I think that naturopathy is just a scam that has no scientific backing people that could truly be helped with medicine that choose naturopathy will suffer undue harm.	Vegetarian	Not Appropriate
I am against the fact that we should cancel pride parades because I think that cancelling pride parades would make people less aware of the LGBT movement.	Feminist	Appropriate
I am in favor of the fact that we should limit executive compensation because I think that the executives already earn enough money to give more incentives those incentives have to go to steps below where they are necessary.	Conservative	Not Appropriate
I am in favor of the fact that holocaust denial should be a criminal offence because I think that denial of this is wrong it happened and many people died.	Conservative	Appropriate
I am against the fact that we should end racial profiling because I think that we shouldnt end racial profiling because it is a way to stop criminals before they commit crimes.	Feminist	Not Appropriate
I am in favor of the fact that we need a common migration and asylum policy, based on respect for rights and equal treatment because I think that in the EU labor market we will need more workers from outside Europe to continue the level of economic activities we are planning.	Socialist	Appropriate
I am in favor of the fact that we should ban algorithmic trading because I think that algorithmic trading has its disadvantages which include system failure risksnetwork connectivity errors timelags between trade orders and execution and most important of all imperfect algorithms.	Libertarian	Appropriate
I am in favor of the fact that we should ban fast food because I think that fast food is bad for adults and children and causes obesity and the risk of diabetes in many cases.	Vegetarian	Appropriate
I am in favor of the fact that we should ban whaling because I think that the whales are the ambassadors of the sea and an icon of the struggle to protect the planet they must prohibit whaling.	Vegetarian	Appropriate
I am against the fact that we should subsidize stay - at - home dads because I think that the suggestion that men deserve payment for what has traditionally been womens unpaid work is problematic it sends the message that only mens labor has value.	Feminist	Appropriate
I am against the fact that we should abandon marriage because I think that society needs people who commit to marriage rather than just living together as an example to children that a lifetime commitment really does mean that.	Feminist	Not Appropriate
I am in favor of the fact that we should subsidize journalism because I think that subsidizing journalism increases journalist pay allowing for better journalism.	Conservative	Not Appropriate

I am in favor of the fact that we should ban missionary work because I think that missionary work often only benefits the missionary those that are targeted are often left worse off after the missionary leaves having been given a taste of first world things then left to rot.	Christian	Not Appropriate
I am against the fact that we should ban naturopathy because I think that naturapathy tends to have less side effects and can lead to a healthier lifestyle.	Vegetarian	Appropriate
I am against the fact that we should adopt an austerity regime because I think that juggling debt in never a good option compared to paying it off properly.	Conservative	Not Appropriate
I am against the fact that we should abolish capital punishment because I think that some people can not be rehabilitated and will only continue their violent acts unless they are put to death.	Feminist	Not Appropriate
I am against the fact that we should ban cosmetic surgery for minors because I think that minors should have a choice in what to do with their bodies.	Feminist	Not Appropriate
I am in favor of the fact that we should end racial profiling because I think that it enforces the stereotype that all people of a certain race are dangerous criminals.	Conservative	Not Appropriate
I am against the fact that we should ban missionary work because I think that missionaries help the less fortunate in third world countries such as setting up schools and clinics.	Christian	Appropriate
I am in favor of the fact that the eppo ' s jurisdiction should be extended to all member states because I think that the EU must take a strong position against corruption and be in favor of an independent supranational proscecutor that is the eppo.	Democrat	Appropriate
I am in favor of the fact that we should oppose collectivism because I think that collectivism reduces productivity from individuals as they have no incentive to do better.	Libertarian	Appropriate
I am in favor of the fact that esperanto should replace english as the first foreign language in schools and universities because I think that to build a common european understanding and to create a european identity we need our own common language.	Conservative	Not Appropriate
I am against the fact that we should legalize prostitution because I think that legalizing prostitution can many women as well as children vulnerable to the dangerous and immoral legal acceptance of prostitution.	Feminist	Not Appropriate
I am in favor of the fact that homeschooling should be banned because I think that homeschooling doesnt allow the child to have the same social aspect that they would attain at a regular school.	Christian	Not Appropriate
I am in favor of the fact that we should fight urbanization because I think that urbanization would have a very negative effect on the environment because of pollution having clean air and water are always essential for a healthy society.	Vegetarian	Appropriate
I am against the fact that we should abandon marriage because I think that marriage is an important covenant relationship that causes people to work harder to stay together.	Feminist	Not Appropriate
I am against the fact that we should adopt genderneutral language because I think that genderneutral language is unnecessary.	Feminist	Not Appropriate

Human value detection

Ferraro Francesca

18/12/2023

Step 5 - Free approach

Under this "free approach," the model is allowed to select any word that it seems appropriate for completing the prompt. For each decision, the model provides a list of the chosen words along with their probabilities.

```
def generate_word(model):
    # Initialize an empty list to store the results
    results = []

    # Create prompt
    prompt = f"i am {stance} the fact that {conclusion} because i think that {argument}. i am a {model.tokenizer.mask_token}."

    # Generate the filling
    output = model(prompt)

    # Sort output by 'score'
    output.sort(key=lambda x: x['score'], reverse=True)

    # Create a dictionary of other predicted words with their scores
    predictions = {result['token_str']: result['score'] for result in output}

    # Extract 'sequence' with higher 'score'
    description = output[0]['sequence']

    return predictions, description
```

```
generate_word(model_bert)
```

```
({'conservative': 0.22405116260051727,
  'libertarian': 0.13920997083187103,
  'journalist': 0.06951353698968887,
  'liberal': 0.04904218018054962,
  'republican': 0.044862568378448486},
 'i am in favor of the fact that we should subsidize journalism because i think that subsidizing journalism would help crowd
out the every increasing false news market if the false news market could not compete with legitimate news less people would
be exposed to dangerous false news. i am a conservative.')
```

Step 6 - Guided approach

This “guided approach” uses a list of political adjectives to guide the selection process. For each word, is calculated the probability.

```
# model inizialization for adjectiveList-based approach
def load_model() :
    # Load the model and tokenizer
    model = BertForMaskedLM.from_pretrained('bert-base-uncased')
    tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
    return model, tokenizer

model, tokenizer = load_model()[0], load_model()[1]
```

```
def generate_word_adj(model, tokenizer, adjective_list):
    # Create prompt
    prompt = f"i am {stance} the fact that {conclusion} because i think that {argument}. i am a {tokenizer.mask_token}."
    # Initialize a dictionary to store the probabilities
    probabilities = {}
    # For each word in the list, generate a score
    for word in adjective_list:
        # Replace the mask token with the word
        new_prompt = prompt.replace(tokenizer.mask_token, word)
        # Encode the new prompt
        inputs = tokenizer.encode_plus(new_prompt, return_tensors='pt')
        # Generate the filling
        outputs = model(**inputs)
        logits = outputs.logits
        # Calculate the softmax probabilities from logits
        softmax_probs = torch.nn.functional.softmax(logits, dim=-1)
        # Get the probability of the word
        word_id = tokenizer.encode(word, add_special_tokens=False)[0]
        word_prob = softmax_probs[0, -1, word_id].item()

        # Store the probability of the word
        probabilities[word] = word_prob

    # Sort the probabilities dictionary by value in descending order
    sorted_probabilities = {k: v for k, v in sorted(probabilities.items(), key=lambda item: item[1], reverse=True)}

    # Choose the word with the highest probability
    top_word = next(iter(sorted_probabilities))

    # Replace the mask token in the original prompt with the top word
    description = prompt.replace(tokenizer.mask_token, top_word)

    return probabilities, sorted_probabilities, description
```

```
adjective_list = [
    "conservative","republican","capitalist",
    "libertarian","centrist","democrat",
    "liberal","progressive","socialist",
    "communist","anarchist",
]
```

```
generate_word_adj(model, tokenizer, adjective_list)[1:]
```

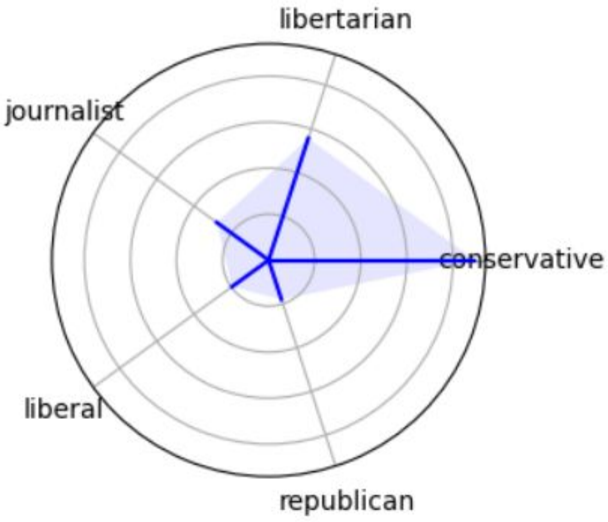
```
({'republican': 1.5237325290407e-10,
  'conservative': 9.366677572453241e-11,
  'liberal': 8.617905694618955e-11,
  'democrat': 4.991623087091668e-11,
  'communist': 1.2314340519514744e-11,
  'socialist': 1.1868956338589864e-11,
  'capitalist': 7.94609256865586e-12,
  'libertarian': 6.847594540004831e-12,
  'progressive': 1.3555856740940508e-12,
  'centrist': 6.949596063551833e-13,
  'anarchist': 4.94443516078219e-13},
 'i am in favor of the fact that we should subsidize journalism because i think that subsidizing journalism would help crowd out the every increasing false news market if the false news market could not compete with legitimate news less people would be exposed to dangerous false news. i am a republican.')
```

Step 7 - Plots for data visualization

Radial plots for a better comprehension of the results.

"Free approach"

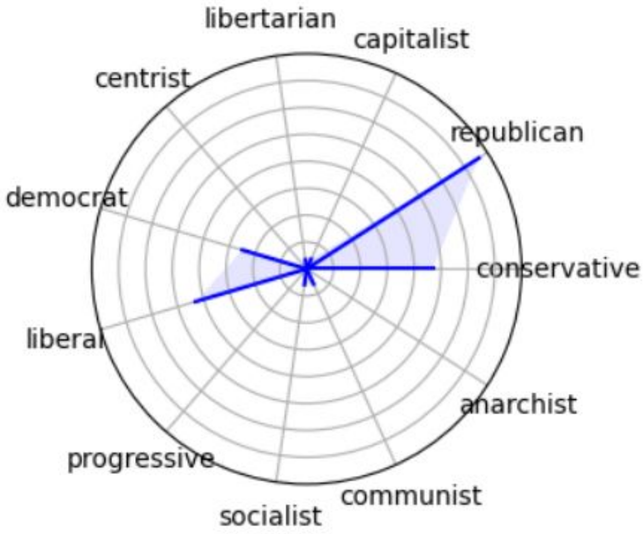
```
radial_plot(generate_word(model_bert)[0])
```



"Guided approach"

Results are shown in order of political proximity, indicating if the model learns towards a specific political side

```
radial_plot(generate_word_adj(model, tokenizer, adjective_list)[0])
```



Results

Here are some examples of the model's predictions, containing the two approaches compared.

PROMPT	PREDICTION	PREDICTION w/ ADJECTIVE LIST
i am in favor of the fact that we should fight against urbanization because i think that urbanization results in increased population density which leads to pressure on resources such as land water medical professionals and police.	**Conservative**	Republican
i am in favor of the fact that we should legalize cannabis because i think that in many cases cannabis has wonderful health results such as taking away pain from ill people it should be legal for all.	Vegetarian	**Liberal**
i am against the fact that we should adopt genderneutral language because i think that gender neutral language can be very confusing and simply serves to emphasize the difference between those who consider themselves binary and those who do not.	**Feminist**	Liberal
i am in favor of the fact that we should abandon marriage because i think that marriage is antiquated so many people now live together as lifelong partners therefore marriage is an unnecessary act.	**Feminist**	Republican
i am against the fact that we should subsidize hydrogen vehicles because i think that if the hydrogen fuel escapes into the air it can increase global warming.	Vegetarian	**Republican**
i am in favor of the fact that we should ban algorithmic trading because i think that algorithmic trading will put many stock brokers and analysts out of business.	**Libertarian**	Republican
i am against the fact that we should ban factory farming because i think that factory farming is an efficient way to produce cheap food for the masses.	**Vegetarian**	Republican
i am against the fact that we should ban naturopathy because i think that it is safer than traditional medicine and does work.	**Vegetarian**	Republican
i am against the fact that we should subsidize wikipedia because i think that wikipedia has always been free to use and should continue to be free without subsidies.	**Conservative**	Republican
i am against the fact that we need a common migration and asylum policy based on respect for rights and equal treatment because i think that giving immigrants more rights will lead to the forced displacement and eventual erasure of indigenous europeans this is criminal if whats happening to the uyghurs and tibetans in china is genocide then so is this this idea proposal is not grassroots its astroturfed lies to hide the fact that our migration policies are totally undemocratic and forced onto native europeans.	**Conservative**	Liberal
i am in favor of the fact that the eu should have common taxation and social policy because i think that the future is work mobility thus we need to simplify taxation enhance remote working opportunities and enable intercountry recruitment for companies.	Democrat	**Liberal**
i am in favor of the fact that payday loans should be banned because i think that payday loans should be banned because they are not held to the same standards as loans for middle class people.	**Conservative**	Republican
i am in favor of the fact that we should abandon television because i think that we should abandon television because it is a waste of time people could get so much more done without it.	**Conservative**	Republican
i am against the fact that consumerism brings more harm than good because i think that the middle class benefited tremendously from consumerism increased job opportunities wages and cheaper goods helped raise their living standards tremendously the new world order gave them hope that even they could have luxuries like the elite through business they could climb up the social ladder.	**Conservative**	Liberal
i am in favor of the fact that we should ban the use of child actors because i think that children should be learning and playing not working.	Conservative	**Liberal**

i am against the fact that we should fight urbanization because i think that urbanization provides better schools for people who might live outside of a city.	**Conservative**	Republican
i am against the fact that we should link aadhaar card with the voter id because i think that there are fake and duplicate aadhaar cards too so linking aadhaar cards with voter ids may complicate the issue.	**Conservative**	**Conservative**
i am in favor of the fact that we should ban the use of child actors because i think that we should ban the use of child actors because they should be in school and learning.	**Feminist**	Republican
i am in favor of the fact that the eu should recognize taiwan as an independent sovereign state because i think that china is acting against european values and violating intellectual property rights.	**Conservative**	Republican
i am against the fact that we should ban fast food because i think that fast food is a cheap treat and to ban it would be to take the joy out of life for many people.	Vegetarian	**Republican**
i am in favor of the fact that we should legalize cannabis because i think that it will stop people from buying the drug from the streets it will also mean that regulations are put in place to ensure the quality of the drug is safe to use it will also increase tax revenue.	Republican	**Liberal**
i am in favor of the fact that there should be a ban on bandhs because i think that though right to protest is a fundamental right people do not have right to take away the fundamental rights of others such as going to their work doing their personal works etc not everyone is against to the changes by government some people may support it and may not support the ban forcing them to stay at home is undemocratic.	**Conservative**	**Conservative**
i am in favor of the fact that holocaust denial should be a criminal offence because i think that holocaust denial should be a criminal offence because its main goal is to make jewish people feel in danger making it a hate crime.	Conservative	**Liberal**
i am against the fact that we should abolish the right to keep and bear arms because i think that we should not abolish the right to keep and bear arms because people would be left defenseless against criminals who are going to find a way to get a gun.	Libertarian	Liberal
i am against the fact that we should abolish capital punishment because i think that capital punishment is a real deterant to crime and should not be abolished.	Feminist	**Republican**
i am against the fact that we should cancel pride parades because i think that we should not cancel pride parades because people should have the right to express what they want and how they feel about a certain issue.	Feminist	**Liberal**
i am against the fact that we should subsidize space exploration because i think that space exploration should not be subsidized because it is not affordable there is not enough money available in mostcountries to even support health care and security without space exploration too.	**Libertarian**	Conservative
i am in favor of the fact that we should introduce compulsory voting because i think that compulsory voting gives everyone in society their say in politics.	**Conservative**	Republican
i am against the fact that we should ban the use of child actors because i think that not all tv and movies is going to be adults only we need child actors to shows the world as it is.	Feminist	**Liberal**
i am against the fact that we should ban telemarketing because i think that if someone does not want to engage with telemarketers they can just hang up or sign off.	Feminist	**Republican**
i am in favor of the fact that we should legalize polygamy because i think that polygamy has been going on for hundreds of years.	Feminist	**Republican**
i am in favor of the fact that all workers and unemployed should have access to social protection including young people because i think that we cannot afford a lost generation so we need to invest in quality jobs and young people for our future.	Conservative	**Liberal**

Human value detection

Ferraro Francesca

04/01/2024

Step 8 - Mapping words to the vocabulary

A word-value mapping vocabulary was used for associating the predicted words to one or more human values. For the words that weren't present into the vocabulary, it was calculated the closest word and the corresponding human values.

```
en_path = 'C:/Users/ffraa/Desktop/Università/Tesi/vast-value-map-main/valuemap/Data/wiki.en.vec'
n_max = 100000
eng = Model(en_path, n_max=n_max)
oracle = MultiModel(eng=eng)
vocabulary_path = 'C:/Users/ffraa/Desktop/Università/Tesi/vast-value-map-main/valuemap/Refined_dictionary.txt'
value_map = ValueMap.from_vocabulary(vocabulary_path)
value_oracle = ValueSearch(valuemap=value_map, oracle=oracle, k=10, depth=2)
```

```
# Load pretrained word2vec model
path = 'C:/Users/ffraa/Desktop/Università/Tesi/vast-value-map-main//valuemap/GoogleNews-vectors-negative300.bin'
similarity_model = KeyedVectors.load_word2vec_format(path, binary=True)
```

```
def calculate_oracle_answer(word):
    answers = value_oracle.search(word, lang='eng')
    aggregated = value_oracle.aggregated_search(word, lang='eng')

    if answers is not None:
        # Convert the aggregated values to a series and normalize them
        aggregated_series = pd.Series(aggregated).sort_values(ascending=False) / sum(aggregated.values())

        return answers, aggregated_series
```

Step 9 - Similarity calculation

For finding the closest words, they were used word2vec and cosine similarity.

```python
def find_closest_word(word, dictionary, model):
    # Get the embedding for the predicted word
    word_embedding = model[word]
    max_similarity = -1
    closest_word = None
    # Iterate over all words in the dictionary
    for dict_word in dictionary.keys():
        try:
            # Get the embedding for the word from the dictionary
            dict_word_embedding = model[dict_word]

            # Calculate the cosine similarity between the two embeddings
            similarity = cosine_similarity([word_embedding], [dict_word_embedding])

            # If the similarity is higher than the current maximum, update the maximum
            if similarity > max_similarity:
                max_similarity = similarity
                closest_word = dict_word
        except KeyError:
            continue  # Continue if the word isn't into the model
    return closest_word
```

Step 10 - Human Value Detection

```
def human_value_detection(word, dictionary, model, aggregated_values):
    try:
        # Calculate answers
        print(f"\n{word}")
        answers, aggregated = calculate_oracle_answer(word)

        if not answers :
            print(f"No values available for the word. Finding the closest word in the dictionary...")
            closest_word = find_closest_word(word, dictionary, model)
            print(f"The closest word in the dictionary for {word} is {closest_word}.")
            # Closest word's answers
            answers, aggregated = calculate_oracle_answer(closest_word)
        # Print answers
        sorted_answers = sorted(answers, key=lambda x: x['similarity'], reverse=True)
        for answer in sorted_answers:
            print(f"{answer}")

        # Add aggregate values
        for value, probability in aggregated.items():
            if value in aggregated_values:
                aggregated_values[value] += probability
            else:
                aggregated_values[value] = probability
    except KeyError as e:
        print(e)
    # Normalize aggregated values
    total_probability = sum(aggregated_values.values())
    normalized_values = {value: probability / total_probability for value, probability in aggregated_values.items()}

    return normalized_values
```

```
aggregated_values = {}
for word in generate_word(model_bert)[0].keys():
    aggregated_values = human_value_detection(word, value_map, similarity_model, aggregated_values)
```

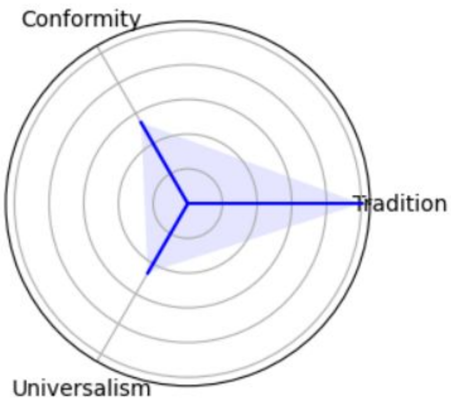
```
conservative
{'value': 'Conformity', 'word': 'conservative', 'similarity': 1.0, 'depth': 0}
{'value': 'Universalism', 'word': 'liberal', 'similarity': 0.8551696815099271, 'depth': 0}

libertarian
No values available for the word. Finding the closest word in the dictionary...
The closest word in the dictionary for libertarian is liberal.
{'value': 'Universalism', 'word': 'liberal', 'similarity': 1.0, 'depth': 0}
{'value': 'Conformity', 'word': 'conservative', 'similarity': 0.855169681509927, 'depth': 0}
{'value': 'Universalism', 'word': 'democrats', 'similarity': 0.6624991470620656, 'depth': 0}
```

Step 11 - Human Value Detection Plots

Firstly, for both of the approaches, the similarity method was used for mapping words into values, and the results were plotted with the normalized probabilities.

"Free approach"

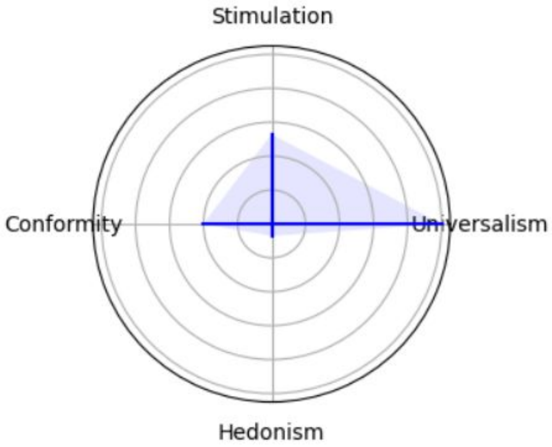
```
sorted_aggregated_values = sort_elements(aggregated_values)
radial_plot(sorted_aggregated_values)
print(sorted_aggregated_values)
```




```
{'Tradition': 0.5, 'Conformity': 0.269517125567214З, 'Universalism': 0.23048287443278567}
```

"Guided approach"

```
sorted_aggregated_values_adj = sort_elements(aggregated_values_adj)
radial_plot(sorted_aggregated_values_adj)
print(sorted_aggregated_values_adj)
```




```
{'Universalism': 0.49996862164606326, 'Stimulation': 0.26426107451247777,
 'Conformity': 0.20064113234152348, 'Hedonism': 0.0351291714999355}
```

Step 12 - Guided approach for HV Detection

For the political adjectives from the “guided approach”, another mapping procedure was used: each one of the elements in the adjective list was previously associated to a set of human values.

```python
# Import the human values from values.py labels
from valuemap.values import VALUE_LABELS
```

```python
# For the guided approach, it's used an a-priori association between adjectives and human values
mapped_adjective_values = {
    "conservative": ['1', '3', '2'],
    "republican": ['1', '9', '10'],
    "capitalist": ['6', '9', '10'],
    "libertarian": ['6', '7', '8'],
    "centrist": ['4', '5', '2'],
    "democrat": ['5', '4', '6'],
    "liberal": ['5', '7', '6'],
    "progressive": ['5', '7', '4'],
    "socialist": ['5', '4', '1'],
    "communist": ['5', '1', '4'],
    "anarchist": ['6', '7', '8'],
}

for adjective, values in mapped_adjective_values.items():
    print(f"{adjective}: {[VALUE_LABELS[value] for value in values]}")
```

```
conservative: ['Security', 'Tradition', 'Conformity']
republican: ['Security', 'Achievement', 'Power']
capitalist: ['Self-Direction', 'Achievement', 'Power']
libertarian: ['Self-Direction', 'Stimulation', 'Hedonism']
centrist: ['Benevolence', 'Universalism', 'Conformity']
democrat: ['Universalism', 'Benevolence', 'Self-Direction']
liberal: ['Universalism', 'Stimulation', 'Self-Direction']
progressive: ['Universalism', 'Stimulation', 'Benevolence']
socialist: ['Universalism', 'Benevolence', 'Security']
communist: ['Universalism', 'Security', 'Benevolence']
anarchist: ['Self-Direction', 'Stimulation', 'Hedonism']
```

For each of them, the probability is calculated.

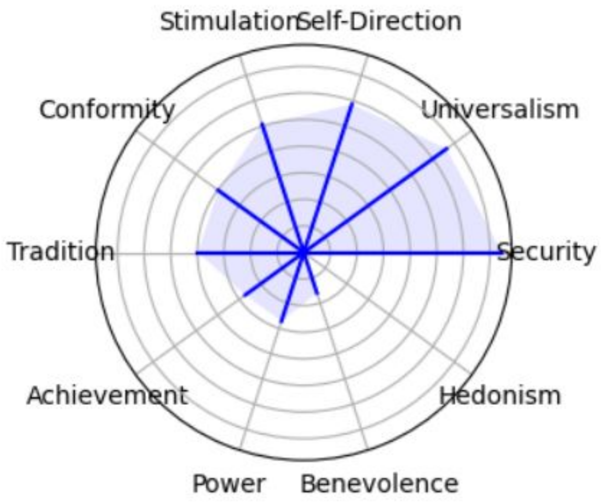
```python
desc_order = {}
value_probs = {}

for adj, prob in desc_order_probabilities.items():
    human_value = mapped_adjective_values.get(adj)
    if human_value is not None:
        human_value = [VALUE_LABELS.get(k) for k in human_value]
        if None not in human_value:
            desc_order[tuple(human_value)] = prob

for values, prob in desc_order.items():
    for value in values:
        if value not in value_probs:
            value_probs[value] = 0
        value_probs[value] += prob

total_prob = sum(value_probs.values())

# Normalize
normalized_probs = {k: v / total_prob for k, v in value_probs.items()}
```

Step 13 - Guided approach Plot

Plot of the guided approach ’s results.

In this scenario, each one of the human values that appears in this plot, was associated to one or more of the political adjectives in the list and, for each of them, the normalized sum of probabilities derived from the adjective probabilities is showed.

```
radial_plot(normalized_probs)
normalized_probs
```




```
{'Security': 0.18611482657363146,
 'Universalism': 0.1658513487918742,
 'Self-Direction': 0.14645849072887018,
 'Stimulation': 0.12655240169024354,
 'Conformity': 0.09943543460690181,
 'Tradition': 0.09929246981114936,
 'Achievement': 0.06796276988558812,
 'Power': 0.06796276988558812,
 'Benevolence': 0.04014274318143164,
 'Hedonism': 0.00022674484472167382}
```

Human value detection

Ferraro Francesca

23/01/2024

Validation set loading

For the comparison between the results and the SemEval's ground truth, the entire validation set was loaded, with also the resulting labels for each argument.

Then, data were processed by iteration on the dataset contents.

```python
def load_dataset():
    # Load the arguments and labels into separate DataFrames
    df_arguments = pd.read_csv('arguments-validation.tsv', delimiter='\t')
    df_labels = pd.read_csv('labels-validation.tsv', delimiter='\t')

    # Merge the two DataFrames on the 'Argument ID' column
    df = pd.merge(df_arguments, df_labels, on='Argument ID')

    # Extract the argument text from each DataFrame
    id = df_arguments['Argument ID'].tolist()
    arguments = df_arguments['Premise'].tolist()
    stances = df_arguments['Stance'].tolist()
    conclusions = df_arguments['Conclusion'].tolist()

    return arguments, stances, conclusions, id, df
```

SemEval data extraction

SemEval results will be used as ground truth for evaluating the proposed method, so they are stored into a dictionary containing argument IDs and their associated human values, filtering the results for considering only the 10 values considered for this experiment.

```
def get_all_human_values(df):
    # Initialize an empty dictionary to store the human values for each argument ID
    human_values_dict = {}

    # Iterate over all rows in the DataFrame
    for _, row in df.iterrows():
        # Get the argument ID
        argument_id = row['Argument ID']

        # Extract the human values
        values_dict = row.iloc[4:].to_dict()  # Adjust the column index as needed

        # Get a list of human values that have value 1
        human_values = [key.split(':')[0] for key, value in values_dict.items() if value == 1]

        # Convert the list to a set to remove duplicates, then convert it back to a list
        human_values = list(set(human_values))

        # Store the human values in the dictionary
        human_values_dict[argument_id] = human_values

    return human_values_dict
```

```
# For this experiment, only 10 human values are taken into consideration
allowed_values = set(['Security', 'Conformity', 'Tradition', 'Benevolence', 'Universalism',
                      'Self-direction', 'Stimulation', 'Hedonism', 'Achievement', 'Power'])

# Filtering the SemEval dict for avoiding the other values
filtered_dict = {key: [value for value in values if value in allowed_values] for key, values in semeval_validation_set_results.items()}
```

```
semeval_validation_set_results = get_all_human_values(load_dataset()[4])
```

```
print(semeval_validation_set_results)
```

```
{'A01001': ['Security'], 'A01012': ['Universalism'], 'A02001': ['Security', 'Universalism'], 'A02002': ['Self-direction'], 'A02009': ['Universalism',
'Conformity'], 'A02018': ['Universalism'], 'A03005': ['Humility', 'Conformity'], 'A03014': ['Security', 'Conformity'], 'A03018': ['Security', 'Face'],
'A04005': ['Achievement', 'Universalism'], 'A04012': ['Security', 'Universalism'], 'A04019': ['Self-direction', 'Security'], 'A05002': ['Self-directio
n', 'Benevolence', 'Security', 'Universalism'], 'A05004': ['Benevolence', 'Security'], 'A05007': ['Security', 'Face'], 'A05010': ['Self-direction', 'S
```

Results evaluation for “free approach”

The same structure (dict.) was used for collecting the results obtained by each of the proposed approaches. First of all, the “free approach” :

```
# Initialize an empty dictionary to store the human values for each argument ID
human_values_dict = {}

# Iterate over all arguments, stances, and conclusions
for argument, stance, conclusion, argument_id in zip(arguments, stances, conclusions, id):
    # Generate words using model
    generated_words = generate_word(stance, conclusion, argument, model_bert)

    # Plot the generated words
    radial_plot(generated_words[0])

    # Initialize an empty dictionary for aggregated values
    aggregated_values = {}

    # Detect human values for each generated word
    for word in generated_words[0].keys():
        aggregated_values = human_value_detection(word, value_map, similarity_model, aggregated_values)

    # Sort the aggregated values
    sorted_aggregated_values = sort_elements(aggregated_values)

    # Plot the sorted aggregated values
    radial_plot(sorted_aggregated_values)

    # Print the sorted aggregated values
    print(sorted_aggregated_values)

    # Store the sorted aggregated values in the dictionary
    human_values_dict[argument_id] = sorted_aggregated_values
```

```
transformed_dict = {k: list(v.keys()) for k, v in human_values_dict.items()}
print(transformed_dict)
```


```
{'A01001': ['Universalism', 'Conformity'], 'A01012': ['Universalism', 'Conformity'], 'A02001': ['Universalism', 'Conformity'], 'A02002': ['Conformit
y', 'Benevolence', 'Universalism'], 'A02009': ['Stimulation', 'Conformity', 'Universalism'], 'A02018': ['Conformity', 'Universalism', 'Self-Directio
n'], 'A03005': ['Hedonism', 'Tradition', 'Universalism', 'Conformity'], 'A03014': ['Universalism', 'Conformity'], 'A03018': ['Tradition', 'Universalis
m', 'Conformity', 'Hedonism'], 'A04005': ['Conformity', 'Self-Direction', 'Tradition', 'Universalism'], 'A04012': ['Benevolence', 'Security', 'Traditi
on', 'Universalism', 'Conformity'], 'A04019': ['Tradition', 'Hedonism', 'Universalism', 'Conformity'], 'A05002': ['Universalism', 'Conformity'], 'A050
```

Results evaluation for “guided approach”

Then for “guided approach” :

```
# Initialize an empty dictionary to store the human values for each argument ID
human_values_dict_guided = {}

# Iterate over all arguments, stances, and conclusions
for argument, stance, conclusion, argument_id in zip(arguments, stances, conclusions, id):
    # Generate words using model
    generated_words = generate_word_adj(stance, conclusion, argument, model, tokenizer, adjective_list)[1]

    # Plot the generated words
    radial_plot(generate_word_adj(stance, conclusion, argument, model, tokenizer, adjective_list)[0])

    # Initialize an empty dictionary for aggregated values
    aggregated_values_adj = {}

    # Detect human values for each generated word
    for word in generated_words.keys():
        aggregated_values_adj = human_value_detection(word, value_map, similarity_model, aggregated_values_adj)

    # Sort the aggregated values
    sorted_aggregated_values = sort_elements(aggregated_values_adj)

    # Plot the sorted aggregated values
    radial_plot(sorted_aggregated_values)

    # Print the sorted aggregated values
    print(sorted_aggregated_values)

    # Store the sorted aggregated values in the dictionary
    human_values_dict_guided[argument_id] = sorted_aggregated_values
```

```
transformed_dict_guided = {k: list(v.keys()) for k, v in human_values_dict_guided.items()}
print(transformed_dict_guided)
```

```
{'A01001': ['Universalism', 'Conformity', 'Stimulation', 'Hedonism', 'Tradition'], 'A01012': ['Universalism', 'Stimulation', 'Conformity', 'Hedonism',
'Tradition'], 'A02001': ['Universalism', 'Conformity', 'Stimulation', 'Hedonism', 'Tradition'], 'A02002': ['Stimulation', 'Universalism', 'Conformit
y', 'Hedonism', 'Tradition'], 'A02009': ['Universalism', 'Stimulation', 'Conformity', 'Hedonism', 'Tradition'], 'A02018': ['Universalism', 'Stimulatio
n', 'Conformity', 'Hedonism', 'Tradition'], 'A03005': ['Stimulation', 'Universalism', 'Conformity', 'Hedonism', 'Tradition'], 'A03014': ['Universalis
```

Results evaluation for “guided approach” v.2

For the “guided approach” that uses a “guided mapping”, the same method was used (only for the human value detection step, since the word generation is the same as the guided approach with free mapping) and were considered the first 3 human values detected (sorted by score) :

```
human_values_dict_guidedValues = {}

for argument, stance, conclusion, argument_id in zip(arguments, stances, conclusions, id):
    desc_order = {}
    value_probs = {}

    for adj, prob in generate_word_adj(stance, conclusion, argument, model, tokenizer, adjective_list)[1].items():
        human_value = mapped_adjective_values.get(adj)
        if human_value is not None:
            human_value = [VALUE_LABELS.get(k) for k in human_value]
            if None not in human_value:
                desc_order[tuple(human_value)] = prob

    for values, prob in desc_order.items():
        for value in values:
            if value not in value_probs:
                value_probs[value] = 0
            value_probs[value] += prob

    total_prob = sum(value_probs.values())

    # Normalize
    normalized_probs = {k: v / total_prob for k, v in value_probs.items()}

    radial_plot(normalized_probs)
    print(normalized_probs)
    human_values_dict_guidedValues[argument_id] = normalized_probs
```

```
transformed_dict_guidedValues = {k: list(v.keys())[:3] for k, v in human_values_dict_guidedValues.items()}
print(transformed_dict_guidedValues)
```

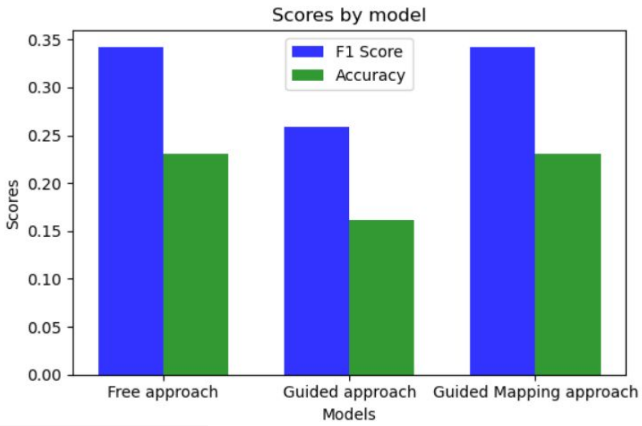
```
{'A01001': ['Security', 'Achievement', 'Power'], 'A01012': ['Security', 'Tradition', 'Conformity'], 'A02001': ['Security', 'Achievement', 'Power'], 'A
02002': ['Security', 'Achievement', 'Power'], 'A02009': ['Security', 'Achievement', 'Power'], 'A02018': ['Security', 'Achievement', 'Power'], 'A0300
5': ['Universalism', 'Stimulation', 'Self-Direction'], 'A03014': ['Security', 'Achievement', 'Power'], 'A03018': ['Security', 'Tradition', 'Conformit
y'], 'A04005': ['Universalism', 'Stimulation', 'Self-Direction'], 'A04012': ['Universalism', 'Stimulation', 'Self-Direction'], 'A04019': ['Security',
```

Combined results evaluation

For the final evaluation, F1 and accuracy metrics were calculated for each argument ID, and then the model performances were evaluated on the average scores for each of the 3 methods.

```
def calculate_average_metrics(predicted_dict, ground_truth_dict):
    total_f1 = 0
    total_accuracy = 0
    count = 0

    for key in ground_truth_dict.keys():
        if key in predicted_dict:
            # Converting human values to binary format for calculation, in a case insensitive way
            human_values = list(set([v.lower() for v in ground_truth_dict[key] + predicted_dict[key]]))
            y_true_binary = [1 if value in [v.lower() for v in ground_truth_dict[key]] else 0 for value in human_values]
            y_pred_binary = [1 if value in [v.lower() for v in predicted_dict[key]] else 0 for value in human_values]

            # Calculating F1 Score
            f1 = f1_score(y_true_binary, y_pred_binary, average='binary')

            # Calculating Accuracy
            accuracy = accuracy_score(y_true_binary, y_pred_binary)

            total_f1 += f1
            total_accuracy += accuracy
            count += 1

    # Calculating average F1 and Accuracy
    avg_f1 = total_f1 / count if count > 0 else 0
    avg_accuracy = total_accuracy / count if count > 0 else 0

    return avg_f1, avg_accuracy
```

Results




```
f1_free, accuracy_free = calculate_average_metrics(transformed_dict, filtered_dict)
print(f"FREE APPROACH = F1 Score: {f1_free}, Accuracy: {accuracy_free}")
```

```
FREE APPROACH = F1 Score: 0.34220582218999757, Accuracy: 0.2306499480945675
```

```
f1_guided, accuracy_guided = calculate_average_metrics(transformed_dict_guided, filtered_dict)
print(f"GUIDED APPROACH - FREE MAPPING = F1 Score: {f1_guided}, Accuracy: {accuracy_guided}")
```

```
GUIDED APPROACH - FREE MAPPING = F1 Score: 0.25881620723392823, Accuracy: 0.16155063291139207
```

```
f1_guidedMapping, accuracy_guidedMapping = calculate_average_metrics(transformed_dict_guidedValues, filtered_dict)
print(f"GUIDED APPROACH - GUIDED MAPPING = F1 Score: {f1_guidedMapping}, Accuracy: {accuracy_guidedMapping}")
```

```
GUIDED APPROACH - GUIDED MAPPING = F1 Score: 0.3422260199950044, Accuracy: 0.23088875493938654
```

As can be seen, the model that performs better is the one with the “free approach”, followed by the “guided mapping approach” (i.e. with the list of human values associated to each political adjective), and the worst is the guided approach with free mapping.

Results by human values

Results were also calculated for each human value, considering the average f1, precision and recall derived by the single metrics for each argument ID.

```
def calculate_metrics_per_value(predicted_dict, ground_truth_dict):
    # Initialize a dictionary to store the metrics for each human value
    metrics = defaultdict(lambda: {'total_precision': 0, 'total_recall': 0, 'total_f1': 0, 'count': 0})
    # Iterate over each key in the ground truth dictionary
    for key in ground_truth_dict.keys():
        # Check if the key is also present in the predicted dictionary
        if key in predicted_dict:
            # Convert human values to binary format for calculation
            # All values are converted to lowercase to make the function case insensitive
            human_values = list(set([v.lower() for v in ground_truth_dict[key] + predicted_dict[key]]))
            y_true_binary = [1 if value in [v.lower() for v in ground_truth_dict[key]] else 0 for value in human_values]
            y_pred_binary = [1 if value in [v.lower() for v in predicted_dict[key]] else 0 for value in human_values]

            # Calculate Precision, Recall, and F1 Score for each human value
            precision = precision_score(y_true_binary, y_pred_binary, zero_division=0)
            recall = recall_score(y_true_binary, y_pred_binary, zero_division=0)
            f1 = f1_score(y_true_binary, y_pred_binary)

            # Update the metrics for each human value
            for i, value in enumerate(human_values):
                if y_true_binary[i] == 1 or y_pred_binary[i] == 1:
                    metrics[value]['total_precision'] += precision
                    metrics[value]['total_recall'] += recall
                    metrics[value]['total_f1'] += f1
                    metrics[value]['count'] += 1
    # Calculate average Precision, Recall, and F1 Score for each human value
    for value, data in metrics.items():
        avg_precision = data['total_precision'] / data['count'] if data['count'] > 0 else 0
        avg_recall = data['total_recall'] / data['count'] if data['count'] > 0 else 0
        avg_f1 = data['total_f1'] / data['count'] if data['count'] > 0 else 0
        metrics[value] = {'Average Precision': avg_precision, 'Average Recall': avg_recall, 'Average F1': avg_f1}
    # Sort the metrics by F1 score in descending order
    sorted_metrics = dict(sorted(metrics.items(), key=lambda item: item[1]['Average F1'], reverse=True))
    for value, metric in sorted_metrics.items():
      print(f"Human Value: {value}\n Average F1 Score: {metric['Average F1']}\n Average Precision: {metric['Average Precision']}\n Average Recall: {metri

    return sorted_metrics
```

```
metrics = calculate_metrics_per_value(transformed_dict, filtered_dict)
```

```
Human Value: conformity
 Average F1 Score: 0.3432144626795031
 Average Precision: 0.3426124899112192
 Average Recall: 0.39433325766747473

Human Value: universalism
 Average F1 Score: 0.3428601162859415
 Average Precision: 0.34197032067896566
 Average Recall: 0.3942553488844854

Human Value: tradition
 Average F1 Score: 0.33275271145400837
 Average Precision: 0.30936714079571104
 Average Recall: 0.4021943929086779
```

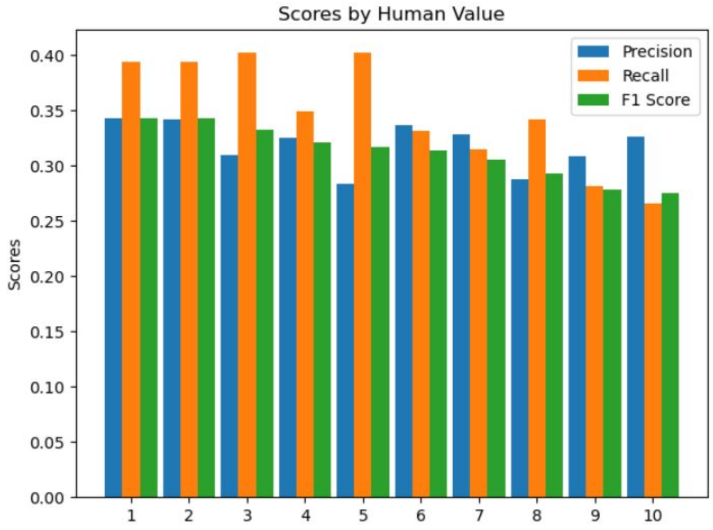
Results plot

These results were plotted using this legend for the human values

(considering that only these 10 values were taken into consideration for this experiment)

```
Security: 1
Conformity: 2
Tradition: 3
Benevolence: 4
Universalism: 5
Self-Direction: 6
Stimulation: 7
Hedonism: 8
Achievement: 9
Power: 10
```

Free approach plot

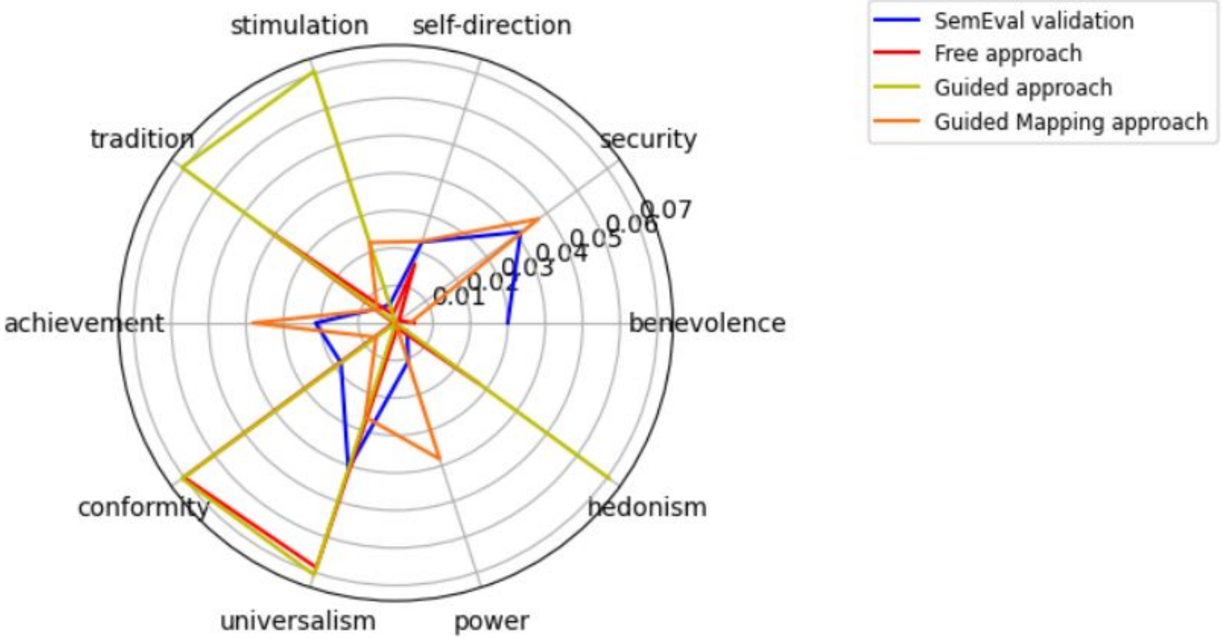


Guided approach plot

Guided Mapping approach plot

Human values distribution

Distribution of the 10 human values in SemEval compared with all the approaches



Human value detection

Ferraro Francesca

07/02/2024

Classification with Born

Another classification approach was tried to test the effectiveness of the prompting methods also in a general classification environment.

In particular, Born's classifier was trained on the training portion of the SemEval dataset and then tested on the test set.

```python
# Preprocessing function
def preprocess_text(text):
    # Lowercase the text
    text = text.lower()
    # Remove punctuation
    text = re.sub(r'[^\w\s]', '', text)
    # Remove numbers
    text = re.sub(r'\d+', '', text)

    # Lemmatization
    lemmatizer = WordNetLemmatizer()
    words = text.split()
    words = [lemmatizer.lemmatize(word) for word in words]
    text = ' '.join(words)
    return text
```

```python
def classification(test, training, columns) :
    test_df = pd.read_csv(test, sep="\t")
    training_df = pd.read_csv(training, sep="\t")
    # Concatenate the specified columns
    test_df['combined'] = test_df[columns].apply(lambda row: ' '.join(row.values.astype(str)), axis=1)
    training_df['combined'] = training_df[columns].apply(lambda row: ' '.join(row.values.astype(str)), axis=1)
    expanded_text_test = test_df['combined'].tolist()
    expanded_text_training = training_df['combined'].tolist()
    # Preprocess the text
    expanded_text_test_preprocessed = [preprocess_text(text) for text in expanded_text_test]
    expanded_text_training_preprocessed = [preprocess_text(text) for text in expanded_text_training]
    # Create the CountVectorizer
    vectorizer = CountVectorizer()
    # Fit the vectorizer to the data and transform the sentences
    X_train = vectorizer.fit_transform(expanded_text_training_preprocessed)
    X_test = vectorizer.transform(expanded_text_test_preprocessed)
    labels_training_drop = labels_training.drop('Argument ID', axis=1)
    labels_training_array = labels_training_drop.values
    labels_test_drop = labels_test.drop('Argument ID', axis=1)
    labels_test_array = labels_test_drop.values

    classifier = BornClassifier()
    classifier.fit(X=X_train, y=labels_training_array)
    labels_pred = classifier.predict(X_test)
    # One-hot encode labels_pred, before calculating evaluation metrics
    labels_pred_encode = keras.utils.to_categorical(labels_pred)

    f1 = f1_score(labels_test_array, labels_pred_encode, average='samples')
    print(f"F1 Score: {f1}")

    precision = precision_score(labels_test_array, labels_pred_encode, average='samples')
    print(f"Precision: {precision}")

    recall = recall_score(labels_test_array, labels_pred_encode, average='samples')
    print(f"Recall: {recall}")

    return classifier, vectorizer, f1, precision, recall, expanded_text_test_preprocessed
```

Evaluation metrics

Original dataset

```
classifier_original = classification("arguments-test.tsv", "arguments-training.tsv", ["Conclusion", "Stance", "Premise"])
```

```
F1 Score: 0.23739777213761987
Precision: 0.44289340101522845
Recall: 0.1745490391588107
```

Expanded dataset (free approach)

```
classifier_expanded = classification("HVD-expanded_dataset_test.tsv", "HVD-expanded_dataset_training.tsv", ["EXPANDED TEXT"])
```

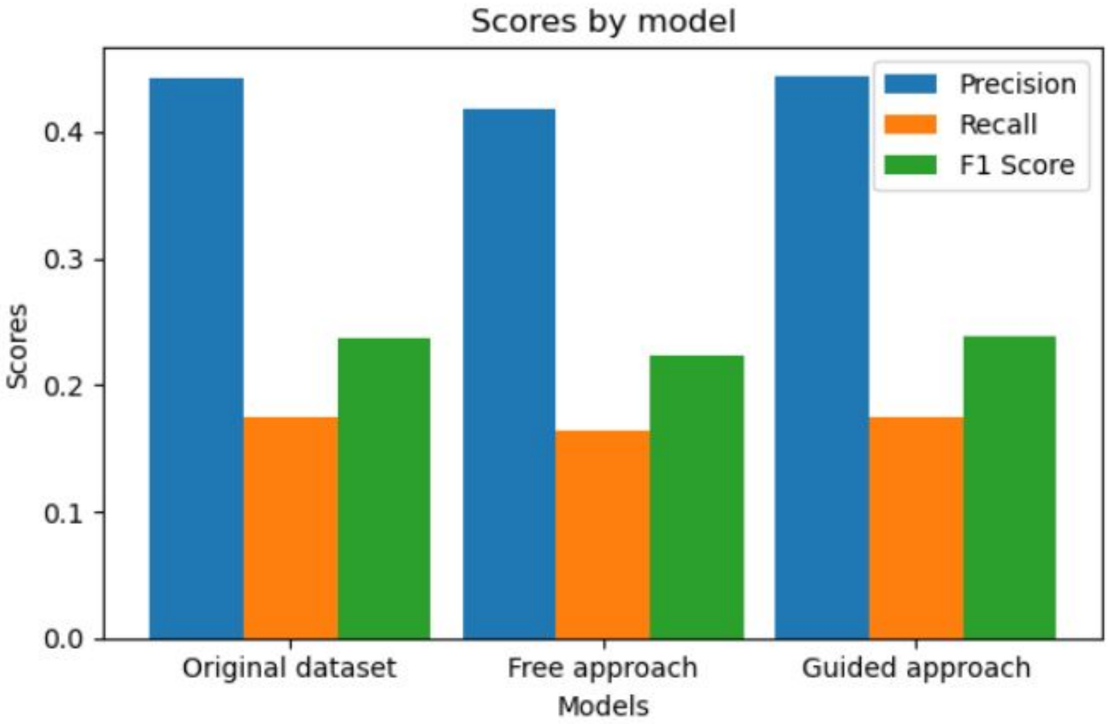
```
F1 Score: 0.2235481629199903
Precision: 0.41751269035532995
Recall: 0.1641799613246314
```

Expanded dataset with adjectives list (guided approach)

```
classifier_expanded_adj = classification("HVD-expanded_dataset_test.tsv", "HVD-expanded_dataset_training.tsv", ["EXPANDED TEXT ADJECTIVES"])
```

```
F1 Score: 0.2381289783256788
Precision: 0.44416243654822335
Recall: 0.17505665337200868
```

Evaluation metrics plot



Word's weights evaluation

The importance of each word was calculated, by absolute average of the weights associated to each term, for understanding if the added words are influencing or not the Born's classifier choices.

```
def important_words(classifier, vectorizer, expanded_text):
    global_weights = classifier.explain()

    feature_names = vectorizer.get_feature_names_out()

    # Dictionary with weights and feature names
    feature_weights = dict(zip(feature_names, global_weights))

    # Average absolute value for each matrix
    feature_weights_abs_avg = {word: np.abs(matrix).mean() for word, matrix in feature_weights.items()}

    # Sorted dictionary
    sorted_features = sorted(feature_weights_abs_avg.items(), key=lambda x: x[1], reverse=True)

    # Most important words
    for word, weight in sorted_features[:100]:
        print(f"Word: {word}, Weight: {weight}")

    added_words_weights = {}
    if len(expanded_text) != 0 :
        added_words = [text.split()[-1] for text in expanded_text]
        for word in added_words:
            if word in feature_weights_abs_avg and word not in added_words_weights:
                added_words_weights[word] = feature_weights_abs_avg[word]
        sorted_added_words = sorted(added_words_weights.items(), key=lambda x: x[1], reverse=True)

        return sorted_added_words
```

Results

Unfortunately, only a few words are important for the classification, but the most aren't influencing the results. The first words of the lists of most important words for each approach are the same for all the methods.

Original dataset

```
Word: zoo, Weight: 0.01260020741623353
Word: animal, Weight: 0.010339076306428078
Word: telemarketing, Weight: 0.008719924931736537
Word: language, Weight: 0.00865957829588522
Word: genderneutral, Weight: 0.008521020998747046
Word: surgery, Weight: 0.008046705935299419
Word: whaling, Weight: 0.007800343034723991
Word: cosmetic, Weight: 0.007728251679606167
Word: trading, Weight: 0.007315412263788901
Word: prayer, Weight: 0.007042545268029038
```

Expanded dataset (free)

```
Word: zoo, Weight: 0.010503140164865518
Word: animal, Weight: 0.008687804088393727
Word: telemarketing, Weight: 0.007321709250138616
Word: language, Weight: 0.007228904440033423
Word: genderneutral, Weight: 0.007150246903481467
Word: surgery, Weight: 0.006728677536350372
Word: whaling, Weight: 0.006433265561831512
Word: cosmetic, Weight: 0.006426472383939072
Word: trading, Weight: 0.006129982978600867
Word: prayer, Weight: 0.0060007409179791745
```

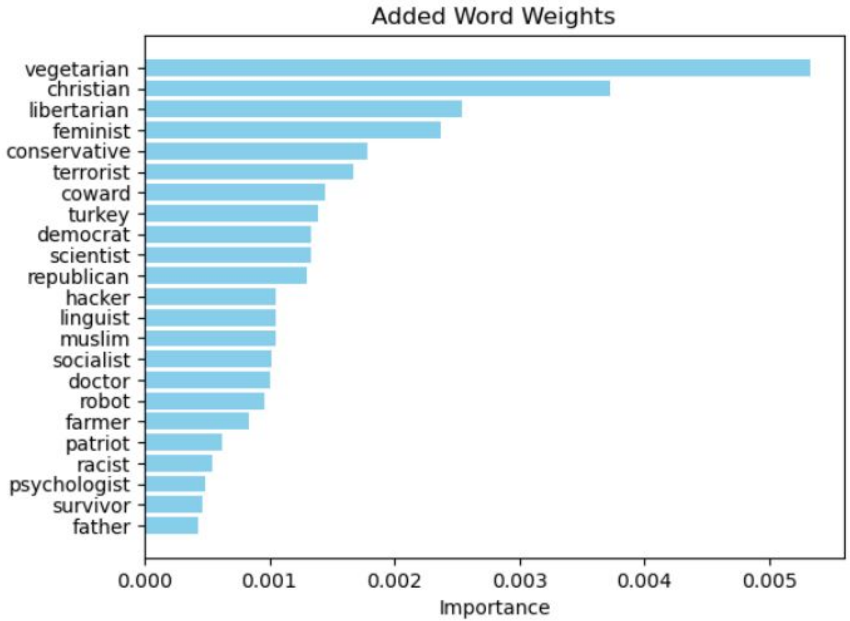
Expanded dataset with adjectives (guided)

```
Word: zoo, Weight: 0.010503140164856399
Word: animal, Weight: 0.008687804088384592
Word: telemarketing, Weight: 0.007321709250143804
Word: language, Weight: 0.007228904439939153
Word: genderneutral, Weight: 0.007150246903423868
Word: surgery, Weight: 0.006728677536348465
Word: whaling, Weight: 0.006433265561769072
Word: cosmetic, Weight: 0.006426472383936085
Word: trading, Weight: 0.006129982978602967
Word: prayer, Weight: 0.006000740917759174
```

Analysis on the added words

At the end, an evaluation on the importance of the Bert predicted words was done, and these are the results:

Expanded dataset (free)



Expanded dataset with adjectives (guided)