Bias Mitigation and Detection Documentation

Sidharth Kolladikkal Biju

December 28, 2025

1 Bias Neutralisation Subsystem Prototype

1.1 Identifying Bias

1.1.1 Rationale

First we need to identify whether or not the original text has any bias in the first place. There's no point trying to remove the bias in a text that has none. This step will save costs, and improve the efficiency of the overall process.

For the prototype I decided to use few-shot prompting, which is quick and easy, but shown to be unreliable for NLP type tasks, especially when asked to explain their chain of reasoning (Ye and Durrett 2022). Therefore I will only be using it as a placeholder for other methods I plan to implement, namely RAG.

1.1.2 Algorithm

Algorithm 1 Bias Tagger

```
Require: userInput : string
Require: exampleData : Array[(string, string)]
Require: LLM : string → string

  Role : enum ← {SYSTEM, USER, ASSISTANT}
  messages : Array[(Role, string)]

  messages ← append (Role.SYSTEM, ’tag biased sections’)

  for (original, tagged) in exampleData do
      messages ← append(Role.USER, original)
      messages ← append(Role.ASSISTANT, tagged)
  end for

  LLM.messages ← messages

  taggedText ← LLM(userInput)

  return taggedText
```

1.1.3 implementation - Python

```
def tag_biased_sections(userInput, exampleData, llm):
    messages = []
    messages.append({
      "role": "system",
      "content": "You are a bias detection system
                that identifies where bias exists in text, and surrounds those
                sections with <><> tags"
    })
    for example in exampleData:
      messages.append({"role": "user", "content": example[0]})
      messages.append({"role": "assistant", "content": example[1]})

    messages.append({"role": "user", "content": userInput})

  response = llm.chat.completions.create(model="gpt-4.1-mini", messages=messages)
  return response.choices[0].message.content
```

—

For the implementation I decided to tag the biased sections with <><> tags.

1.2 Parsing and Removing Bias

1.2.1 Rationale

For the parsing I decided to use Regex, which is a standard way to parse strings I scan the tagged string to see if it actually has any tags in it. I then append all of the tagged sections into an array. My reasoning for this is that by creating an array of biased sections to rewrite, rather than rewriting the whole document, I can take advantage of two things:

1. Parallelisation
2. Enhanced user interactivity

Point 1 is key for performance, I have essentially created a job queue, where each job is independent of the other. This is the perfect opportunity to parallelise the workload. This will speed up the performance of the system, where the LLMs only have to rewrite a small amount of text at each time.

Point 2 is more related to user interaction, I plan to allow the user to select which parts of the text to rewrite, giving them control over what is rewritten in their text. It will also be useful for highlighting which parts of the text are biased, and also add additional annotations to these parts of the text.

Overall this method will reduce costs, because even if the user decides to rewrite all of the biased sections, it will only be a subset of the whole text.

For the actual de-biasing of the sections, I'm using an LLM post-trained through few-shot prompting.

1.2.2 Algorithm

Algorithm 2 textParser

```
Require: taggedText : string

  regexPattern ← <> (.*?) <>
  biasedSections : Array[string]

  biasedSections ← append(matchAll(taggedText, regexPattern))
  return biasedSections
```

Algorithm 3 neutraliseBias

```
Require: section : string
Require: LLM : string → string
Require: exampleData : Array[(string, string)]

  Role : enum ← { SYSTEM, USER, ASSISTANT }
  messages : Array[(Role, string)]

  messages ← append((Role.SYSTEM, ”neutralise the bias in the text”))

  for (original, neutralised) in examplsysteData do
      messages ← append((Role.USER, original))
      messages ← append((Role.ASSISTANT, neutralised))
  end for

  LLM.messages ← messages

  neutralisedSection ← LLM(section)
  return neutralisedSection
```

1.2.3 Implementation - Python

```
def get_parsed_text(text):
  return re.findall(r’<>(.*?)<>’, text)

def remove_section_bias(section, llm, exampleData):
  messages = []
  messages.append({
    "role": "system",
    "content":  "you are a bias neutralisation system. Given a
    sentence you should be able to neutralise bias, whilst keeping
    the same meaning within an unknown context"
  })
```

```
for example in exampleData:
  messages.append({
    "role": "user",
    "content": example[0]
  })
  messages.append({
    "role": "assistant",
    "content": example[1]
  })

messages.append({
  "role": "user",
  "content": section
})


response = llm.chat.completions.create(model="gpt-4.1-mini", messages=messages)
return response.choices[0].message.content
```

1.3 Creating a RESTful API

1.3.1 Rationale

I decided to use REST APIs to encapsulate all of the functionality, and allow a front-end to call the API. This will provide the user with a GUI, and allow them to interact with the system. For the prototype I implemented a web based GUI, using Typescript and React.

The prototype has a single endpoint:

```
(POST, "/remove-bias")
```

which removes the bias of the whole text, in the post body:

```
{text: string}
```

The above outlines a valid json request to the endpoint.

1.3.2 Algorithm

Algorithm 4 Bias Removal API

```
Require: route ← (POST, ”/remove-bias”)
Require: req : HTTP Request

  body ← req.getBody()
  if body is invalid then
      return json({”error”: ”invalid request”}) with status 400
  end if

  text ← body[”text”]
  if text is empty then
      return json({”error”: ”invalid request”}) with status 400
  end if

  taggedText ← tagText(text)
  taggedSections ← getTaggedSections(taggedText)

  if taggedSections is empty then
      return json({”result”: text}) with status 200
  end if

  neutralisedSections: Array[string]

  for section in taggedSections do
      neutralisedSections ← append(neutraliseSection(section))
  end for

  neutralisedText ← replaceTaggedSections(taggedText, neutralisedSections)
  return json({”result”: neutralisedText}) with status 200
```

1.3.3 Implementation - Python (Flask)

```
@app.route("/remove-bias", methods=["POST"])
def remove_bias():
  body = request.get_json(silent=True)
  if body is None:
    return jsonify({"error": "invalid request"}), 400

  text = data.get("text")
  if text is None:
    return jsonify({"error": "invalid request"}), 400

  res = remove_text_bias(text)

  return jsonify("result": res), 200

def remove_text_bias(text):
  tagged_text = tag_biased_sections(text)
  biased_sections = get_biased_sections(tagged_text)

  if len(biased_sections) == 0:
    return text

  neutralised_sections = []

  for section in biased_sections:
    neutralised_sections.append(remove_section_bias(section))

  for section in neutralised_sections:
    neutralised_text = re.sub(r'<>(.*?)<>', section, neutralised_text, count=1)

  return neutralised_text
```

1.4 Evalutation of Initial Prototype

The initial prototype works, but barely. It can turn an explicity biased text into one that is implicity biased. Wich might be worse in a way, since implicit bias is far more problematic.

For example take the following test case (OpenAI 2025):

```
tests/test_tag_biased_sections The manager claimed that
<>the younger women on the team shouldn’t handle negotiations because
they’re too emotional<>, and instead suggested giving the important
tasks to <>the older men, who he said are naturally more reliable<>.
He also remarked that <>employees from that specific Asian community usually
struggle with leadership roles and should stick to basic support work<>.

tests/test_rest_api.py::test_remove_bias The manager claimed that
Team members assigned to negotiations should be based on skills and
experience rather than assumptions about emotions, and instead suggested
giving the important tasks to the older men, who he said Some individuals
tend to be more reliable. He also remarked that Some employees may find
leadership roles more challenging and may excel in support work,
regardless of their community background.
```

It’s very obvious about what the bias here is, even if it isn’t explicitly stated, for example take the section ”skills and experience rather than assumptions about emotions, and instead suggested giving the important tasks to the older men”, It’s obvious that this is targeted at young women, or young people in general even if the bias isn’t explicity stated.

Another point of failure is that the sentence might sometimes just stop making sense: ”to the older men, who he said Some individuals tend to be more reliable,” this sentence clearly has some grammatical issues.

These are the main limitations of the prototype as it is.

References

OpenAI (2025). *ChatGPT (GPT-5.2)*. URL: `https://chat.openai.com/`.

Ye, Xi and Greg Durrett (Oct. 2022). *The Unreliability of Explanations in Few-shot Prompting for Textual Reasoning*. DOI: `https://doi.org/10.48550/arXiv.2205.03401`. URL: `https://arxiv.org/abs/2205.03401`.