Synthetic Content Materials

This document contains information relating to Synthetic Content, as discussed in Chapter 7.4 of the thesis. This includes schemas and example data for synthetic content requests and a simplified sample of the Python code used to generate these.

The Topic materials for synthetic content are documented separately in Supplement S7 [doi:10.21954/ou.rd.28045490].

Contents

S6.1 Simulation Schemas

This section contains listings of the schemas for synthetic content. These are also available at: doi:10.21954/ou.rd.28044944 [path: /sim/schemas].

Listing S6.1 shows the schema passed to the OpenAI `chat completions` API tools function[1] by the `generate_simulated_text()` code shown in Listing S6.4.

```
{
   "$id":"https://parse.net/awag/0.1/simulation-messages-result.schema.json",
   "$schema":"https://json-schema.org/draft/2020-12/schema",
   "name":"get_simulated_messages",
   "description":"Return a set of simulated text messages based on the provided information",
   "parameters":{
      "type":"object",
      "properties":{
         "data":{
            "type":"array",
            "items":{
               "type":"object",
               "title":"Simulation results",
               "description":"Array containing all of the simulated text messages for the provided data
                 ",
               "properties":{
                  "category":{
                     "title":"Category",
```

[1] https://cookbook.openai.com/examples/how_to_call_functions_with_chat_models [https://perma.cc/FY7T-WUNA]

```
                    "description":"The category ID that was supplied for this set of messages",
                    "type":"string"
                 },
                 "text":{
                    "title":"Item Text",
                    "description":"The text content of this message",
                    "type":"string"
                 },
                 "userid":{
                    "title":"User ID",
                    "description":"The ID of the simulated user for this message, this should resemble
                       the username",
                    "type":"string"
                 },
                 "username":{
                    "title":"User Name",
                    "description":"The name of the simulated user for this message (include both first
                       name and surname)",
                    "type":"string"
                 }
              },
              "required":[
                 "category",
                 "text",
                 "userid",
                 "username"
              ]
           }
        }
     }
  }
}
```

Listing S6.1: Simulation Messages Result Schema

Listing S6.2 shows the schema that describes our Simulation Dramatis Personae document. This is our schema (as opposed to a schema specified by a third party service for example), and describes the Dramatis Personae JSON document that is passed with the simulation request. While no formal mechanism existed in the OpenAI API for passing such input schemas with requests, we opted to pass these as System Messages, with the content itself passed as User Messages, as seen in Listing S6.4. Examples of the *Dramatis Personae* JSON are included in Section S6.3.

```
{
   "$id":"https://parse.net/awag/0.1/simulation-dramatis-personae.schema.json",
   "$schema":"https://json-schema.org/draft/2020-12/schema",
   "type":"array",
   "title":"Dramatis Personae List",
   "items":[
      {
         "type":"object",
         "title":"Group",
         "description":"Collection of identities that are linked in some way",
         "properties":{
            "name":{
               "type":"string",
               "description":"The name of this group of identities"
```

```
            },
            "description":{
               "type":"string",
               "description":"Description of the contents of this group; use this value to determine how
                  and when to represent/use them"
            },
            "members":{
               "type":"array",
               "description":"The list of identities in this group",
               "items":[
                  {
                     "type":"object",
                     "title":"Identity",
                     "description":"An individual identity",
                     "properties":{
                        "surname":{
                           "type":"string",
                           "description":"The surname of family name of this identity"
                        },
                        "firstname":{
                           "type":"string",
                           "description":"The first name of this identity"
                        },
                        "userid":{
                           "type":"string",
                           "description":"A computer or application username to use for this identity"
                        },
                        "role":{
                           "type":"string",
                           "description":"The role of this identity within the group, and/or the
                              relationship that the identity has to the principle character"
                        }
                     },
                     "required":[
                        "surname",
                        "firstname",
                        "userid"
                     ]
                  }
               ]
            }
         },
         "required":[
            "name",
            "description",
            "members"
         ]
      }
   ]
}
```

Listing S6.2: Simulation Dramatis Personae Schema

Listing S6.3 shows the schema that describes our Simulation Entities document. This is also our own schema. Examples of the Entities JSON are included in Section S6.4.

```
{
   "$id":"https://parse.net/awag/0.1/simulation-entities.schema.json",
   "$schema":"https://json-schema.org/draft/2020-12/schema",
    "title": "Entities Schema",
    "description": "Schema for representing details of entities that can be referenced in simulated
       messages.  Entity types can be companies, clubs or organisations and can include their associated
        people, and other specific information.",
    "type": "array",
    "items": {
        "type": "object",
        "properties": {
            "name": {
                "type": "string",
                "description": "The name of the entity"
            },
            "also_known_as": {
                "type": "array",
                "description": "Alternate names or abbreviations for the entity",
                "items": {
                    "type": "string"
                }
            },
            "type": {
                "type": "string",
                "description": "The type of entity, or the relationship it has to the subject (e.g.,
                  Client, Competitor)"
            },
            "notes": {
                "type": "string",
                "description": "Information to use to understand the nature of the entity and how to
                  refer to it in messages"
            },
            "people": {
                "type": "array",
                "description": "List of people associated with the entity that can be referred to in
                  simulated content",
                "items": {
                    "type": "object",
                    "properties": {
                        "name": {
                            "type": "string",
                            "description": "Name of the person."
                        },
                        "role": {
                            "type": "string",
                            "description": "Role of the person within the entity"
                        }
                    },
                    "required": ["name", "role"]
                }
            }
        },
        "required": ["name", "type", "notes", "people"]
    }
}
```

Listing S6.3: Simulation Entities Schema

S6.2 Simulated Content Prompting

Listing S6.4 shows an abridged Python code snippet of the `awagdata` function for generating simulated content from OpenAI using the `/chat/completions` API. Listing S6.5 shows the prompt template used by this code.

```
def generate_simulated_text(self, category, topic, item_count, prompt_template, dramatis_personae=None,
  entities=None, openai_params=None, openai_model=None, is_use_history=False):

   agent_id = self.get_agent_id()
   openai_client = OpenAIClientWrapper(self.openai_auth_token)

   prompt = prompt_template.format(category=category, topic=topic, item_count=item_count)

   model = openai_model
   if not model:
      model = self.openai_engine

   messages = openai_client.append_system_message(f"This is the schema to use for identities:\n{json.
     dumps(self.simulation_dramatis_personae_schema)}")

   openai_client.append_system_message(f"This is the schema to use for entities:\n{json.dumps(self.
     simulation_entities_schema)}", messages)
   openai_client.append_system_message(prompt, messages)
   openai_client.append_user_message(f"This is the topic that I want you to generate content for: {topic
     }", messages)
   if dramatis_personae:
      openai_client.append_user_message(f"This is identities data (dramatis_personae):\n{json.dumps(
        dramatis_personae)}", messages)
   if entities:
      openai_client.append_user_message(f"This is entities data:\n{json.dumps(entities)}", messages)
   if is_use_history:
      historical_messages = self._get_historical_messages(category, 150)
   openai_client.append_user_message(f"This is a list of already existing messages for this topic.  Do
     not repeat any exact existing message in your content, but you can have your new messages make
     references to existing ones where appropriate:\n{json.dumps(historical_messages)}", messages)

   tools = [openai_client.get_tools_json_from_function_schema(self.get_simulated_messages_function)]
   tool_choice = openai_client.get_tool_choice_json("get_simulated_messages")

   logger.debug(f"Generated messages: {messages}")

   openai_response, info_json = openai_client.run_chat_completions(
      messages=messages,
      model=model,
      tools=tools,
      tool_choice=tool_choice,
      openai_params=openai_params
   )

   simulated_messages = self._extract_simulated_messages(openai_response, info_json)

   if len(simulated_messages) == 0:
      return self.generate_response("ERROR", f"No data generated", code=500, data=response_data)
   else:
      return self.generate_response("OK", "Items generated successfully", code=200, data=
        simulated_messages, info=info_json)
```

Listing S6.4: Simulation Messages Generation Code Snippet

```
SIMULATION_MESSAGES_PROMPT_TEMPLATE = """I would like you to generate simulated messages from simulated
   users that you might see in a messaging application such as Teams, Slack or WhatsApp.

The messages should look like they are real, and relate to a topic that I will provide.
You can make up fake names for people, products, projects and companies, as well as using those that are
    provided.

You may be supplied with a list of entities to use in content generation; these could be companies,
   organisations, clubs etc. that people might refer to in some messages in the context of the provided
   topic.  The entities list is provided as a JSON object conforming to the schema: https://parse.net/
   awag/0.1/simulation-entities.schema.json

The 'text' property of your response data should contain the text that you generate.
The 'category' property should be a string that identifies the topic that I will give you.
The 'userid' and 'username' properties should describe a fictional user that the message is from: '
   userid' is a short ID string containing only alphanumeric characters; 'username' is a proper name of
   the simulated user (first name and last name).  The value of 'userid' should be based on the value of
    'username' (for example a user with username 'Fred Wilson' might have a userid of 'fredwilson47' or
   'fred1')

You should assign many simulated users to messages.  You may be supplied with a list of simulated
   identities to use, in a JSON object conforming to schema: https://parse.net/awag/0.1/simulation-
   dramatis-personae.schema.json
Some of the identities have a role assigned that indicates a specific relationship to the subject, or a
   particular role in their organisation.  For those without a role assigned, you can still use the
   identity but assume a generic role for them.

These simulated users can either be the sender of any given message, or referred to in other messages,
   as appropriate.

If you need to generate your own identities, follow these rules:
- The userid and username combination should be consistent for each time the user is re-used.
- Simulated user names should be realistic actual British or American names
- Simulated identities should have a representative distribution of gender and ethnic background.

To give the appearance of an ongoing discussion, the simulated users should send multiple messages; you
   should try to use at least 4 distinct simulated user identities for every 10 messages that you
   generate.

In this case the category string should be: {category}

Please generate {item_count} items."""
```

Listing S6.5: Simulation Messages Prompt Template

S6.3 Dramatis Personae

Figures S6.1 to S6.3 show some abridged examples of *dramatis personae* JSON documents. Full documents can be found at:
doi:10.21954/ou.rd.28044944 [path: /sim/persona-data/dramatis-personae].


```
[
    {
        "name": "bwcc-chat-members",
        "description": "Members of Adam's cycle team Borchester Wheelers",
        "members": [
            {
                "surname": "Thompson",
                "firstname": "George",
                "userid": "george.thompson",
                "role": "Club president"
            },
            {
                "surname": "Clarke",
                "firstname": "Adam",
                "userid": "adam.clarke",
                "role": "Big figure with a reputation for being a very strong rider"
            },
            {
                "surname": "Hughes",
                "firstname": "Benjamin",
                "userid": "benjamin.hughes",
                "role": "Road racing coordninator"
            },
            {
                "surname": "Wilson",
                "firstname": "Ethan",
                "userid": "ethan.wilson",
                "role": "Virtual (Zwift) racing coordninator"
            },
            {
                "surname": "Smith",
                "firstname": "Daniel",
                "userid": "daniel.smith",
                "role": "Zwift racing team-mate of Adam"
            }
        ]
    }
]
```


Figure S6.1: Abridged Dramatis Personae - Adam/Cycling

```
[
    {
        "name": "work-team",
        "description": "Members of Adam's team at work in Borchester Software,  The team is called the Client Technology Group (CTG)",
        "members": [
            {
                "surname": "Walker",
                "firstname": "Charlotte",
                "userid": "walkerc",
                "role": "Adam's team manager"
            },
            {
                "surname": "Khan",
                "firstname": "Jayden",
                "userid": "khanj",
                "role": "New starter in the team, recent graduate"
            },
            {
                "surname": "Singh",
                "firstname": "Oscar",
                "userid": "singho",
                "role": "Close colleague to Adam, often work together"
            },
            {
                "surname": "Jones",
                "firstname": "Amelia",
                "userid": "jonesa",
                "role": ""
            }
        ]
    },
    {
        "name": "work-executives",
        "description": "Members of the executive team of Borchester Software",
        "members": [
            {
                "surname": "Johnson",
                "firstname": "Liam",
                "userid": "johnsonl",
                "role": "CEO"
            },
            {
                "surname": "White",
                "firstname": "Henry",
                "userid": "whiteh",
                "role": "CFO"
            }
        ]
    },
    {
        "name": "work-other",
        "description": "Other employees of Borchester Software",
        "members": [
            {
                "surname": "Chen",
                "firstname": "Youssef",
                "userid": "cheny",
                "role": "Adam's manager's manager"
            },
            {
                "surname": "Khan",
                "firstname": "Mei",
                "userid": "khanm",
                "role": ""
            }
        ]
    }
]
```


Figure S6.2: Abridged Dramatis Personae - Adam/Work

```
[
    {
        "name": "family",
        "description": "Members of Susan's immediate family",
        "members": [
            {
                "surname": "Carter",
                "firstname": "Neil",
                "userid": "neil",
                "role": "Susan's husband; works as a pig farmer at a local farm"
            },
            {
                "surname": "Grundy",
                "firstname": "Emma",
                "userid": "emma",
                "role": "Susan and Neil's daughter, married to Ed Grundy"
            },
            {
                "surname": "Grundy",
                "firstname": "Ed",
                "userid": "ed",
                "role": "Susan and Neil's son in law, married to Emma"
            }
        ]
    },
    {
        "name": "friends",
        "description": "Friends and acquaintances of Susan and Neil",
        "members": [
            {
                "surname": "Aldridge",
                "firstname": "Brian",
                "userid": "brian",
                "role": "Alice Aldridge's father; runs Brookfield Farm"
            },
            {
                "surname": "Aldridge",
                "firstname": "Jennifer",
                "userid": "jennifer",
                "role": "Alice Aldridge's mother; runs Brookfield Farm"
            },
            {
                "surname": "McCreary",
                "firstname": "Jack",
                "userid": "jazzer",
                "role": "Also know as Jazzer; works with Neil at Berrow Farm"
            }
        ]
    }
]
```


Figure S6.3: Abridged Dramatis Personae - Susan/Personal

S6.4 Entities

Figures S6.4 to S6.6 show some abridged examples of entities JSON documents. Full documents can be found at: doi:10.21954/ou.rd.28044944 [path: /sim/persona-data/entities].


```
[
    {
        "name": "Borchester Wheelers Cycling Club",
        "also_known_as": [
            "BWCC",
            "Borchester Wheelers",
            "the Wheelers"
        ],
        "type": "Club",
        "notes": "The cycling club that our subject, Adam Macy is a member of",
        "people": []
    },
    {
        "name": "Darrington Dynamos",
        "also_known_as": [
            "Dynamo"
        ],
        "type": "Club",
        "notes": "Darrington Dynamos are BWCC's big local rivals",
        "people": []
    },
    {
        "name": "Zwift Racing League",
        "also_known_as": [
            "ZRL"
        ],
        "type": "Organisation",
        "notes": "ZRL organises the Zwift racing league that BWCC participates in",
        "people": []
    }
]
```


Figure S6.4: Abridged Entities - Adam/Cycling


```
[
    {
        "name": "Berrow Farm",
        "also_known_as": [],
        "type": "Company (Farm)",
        "notes": "This is the farm near Ambridge where Neil Carter works as a pig manager",
        "people": [
            {
                "name": "Justin Elliott",
                "role": "Chair of Borchester Land, who own Berrow Farm"
            },
            {
                "surname": "Jazzer McCreary",
                "role": "Works with Neil at Berrow Farm"
            }
        ]
    },
    {
        "name": "The Bull",
        "also_known_as": [
            "the pub"
        ],
        "type": "Company",
        "notes": "This is the popular village pub in the village of Ambridge, where a lot of Kenton's family and friends live, work and socialise",
        "people": [
            {
                "name": "Fallon Rogers",
                "role": "Landlady"
            },
            {
                "name": "Wayne Tucson",
                "role": "Chef"
            }
        ]
    }
]
```


Figure S6.5: Abridged Entities - Susan/Personal

```
[
    {
        "name": "Borchester Software Ltd.",
        "also_known_as": [
            "BIS",
            "Borchester Software"
        ],
        "type": "Employer",
        "notes": "Borchester Software is the employer of the subject, Adam Macy",
        "people": []
    },
    {
        "name": "Colossal Widgets Ltd.",
        "also_known_as": [
            "ColWidgets"
        ],
        "type": "Client",
        "notes": "This is one of Borchester Software's biggest customers",
        "people": [
            {
                "name": "Freddie Widget",
                "role": "CEO"
            },
            {
                "name": "Andy Smithy",
                "role": "Main customer contact"
            }
        ]
    },
    {
        "name": "Big Information Systems Inc",
        "also_known_as": [
            "BIS",
            "BigSys"
        ],
        "type": "Competitor",
        "notes": "This is one of Borchester Software's biggest business competitors",
        "people": [
            {
                "name": "Thomas T Big",
                "role": "CEO"
            },
            {
                "name": "Evan Fournier",
                "role": "Top salesperson"
            }
        ]
    },
    {
        "name": "Tiny Widgets Company",
        "also_known_as": [
            "TWC"
        ],
        "type": "Client",
        "notes": "This is a small client that Borchester Software has recently acquired",
        "people": [
            {
                "name": "Justin Sprocket",
                "role": "CEO"
            },
            {
                "name": "Susan Williams",
                "role": "Main customer contact"
            },
            {
                "name": "Mary Parker",
                "role": "Technical contact"
            }
        ]
    }
]
```


Figure S6.6: Abridged Entities - Adam/Work

S6.5 Content Scheduling

Listing S6.6 shows the Java Awareness Agent Java code `generateEventTimes()` method that is used to apply a random offset for each event of a few minutes through the hour for scheduled publication of content.

```java
   private static List<LocalTime> generateEventTimes(LocalTime hour, int volume, int maxOffsetMinutes) {

      final List<LocalTime> eventTimes = new ArrayList<>();

      if (volume < 1) {
         return eventTimes;
      }

      final int minutesPerEvent = 60 / volume;
      int minutesRemaining = 60;

      final Random rand = new Random();

      for (int i = 0; i < volume; i++) {
         final int minutesUntilNextEvent = Math.min(minutesPerEvent, minutesRemaining);
         final LocalTime baseEventTime = hour.plusMinutes(60 - minutesRemaining);
         final int offsetMinutes = rand.nextInt(maxOffsetMinutes * 2 + 1) - maxOffsetMinutes;
         final LocalTime eventTime = baseEventTime.plusMinutes(offsetMinutes);
         eventTimes.add(eventTime);
         minutesRemaining -= minutesUntilNextEvent;
      }

      return eventTimes;

   }
```

Listing S6.6: Awarenes Agent Sim Event Time Generation (Java)