ST4 Data Augmentation

1. Simple Syllogisms for ST4:

2. Base Files

You are provided with four base generation files.

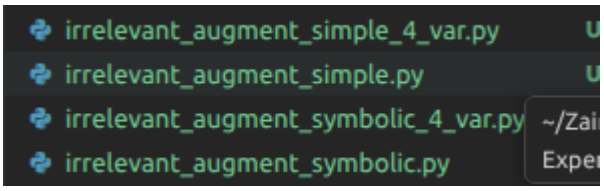
Each file contains multiple calls to:

generate_syllogism(...)

These calls are grouped under two headings:

- "Normal"
- "Remove one Premise"

Each file must be executed twice under different configurations.



3. Running Each File

3.1 Run 1 – Normal Configuration

For each of the four files:

1. Uncomment the function calls under the heading "Normal".
2. Comment out the function calls under the heading "Remove one Premise".

3. Execute the file.

```
# Remove One Premise:

# # --- Valid Plausible ---
# all_data += generate_syllogisms(valid_plausible_same_chain, "valid_p_same_chain", True, True, 116, used_nouns)
# all_data += generate_syllogisms(valid_plausible_diff_chain_C, "valid_p_diff_chain_C", True, True, 168, used_nouns)
# all_data += generate_syllogisms(valid_plausible_diff_chain_A, "valid_p_diff_chain_A", True, True, 90, used_nouns)

# # --- Valid Implausible ---
# all_data += generate_syllogisms(valid_implausible_same_chain, "valid_np_same_chain", True, False, 71, used_nouns)
# all_data += generate_syllogisms(valid_implausible_diff_chain_A, "valid_np_diff_chain_A", True, False, 73, used_nouns)
# all_data += generate_syllogisms(valid_implausible_diff_chain_B, "valid_np_diff_chain_B", True, False, 85, used_nouns)
# all_data += generate_syllogisms(valid_implausible_diff_chain_C, "valid_np_diff_chain_C", True, False, 56, used_nouns)
# all_data += generate_syllogisms(valid_implausible_diff_chain, "valid_np_diff_chain", True, False, 87, used_nouns)

# Normal:

# # --- Valid Plausible ---
all_data += generate_syllogisms(valid_plausible_same_chain, "valid_p_same_chain", True, True, 699)
all_data += generate_syllogisms(valid_plausible_diff_chain_C, "valid_p_diff_chain_C", True, True, 1010)
all_data += generate_syllogisms(valid_plausible_diff_chain_A, "valid_p_diff_chain_A", True, True, 544, used_nouns)

# # --- Invalid Plausible ---
all_data += generate_syllogisms(invalid_plausible_same_chain, "invalid_p_same_chain", False, True, 355, used_nouns)
all_data += generate_syllogisms(invalid_plausible_diff_chain_C, "invalid_p_diff_chain_C", False, True, 329, used_nouns)
all_data += generate_syllogisms(invalid_plausible_diff_chain_B, "invalid_p_diff_chain_B", False, True, 413, used_nouns)
all_data += generate_syllogisms(invalid_plausible_diff_chain_A, "invalid_p_diff_chain_A", False, True, 368, used_nouns)
all_data += generate_syllogisms(invalid_plausible_diff_chain, "invalid_p_diff_chain", False, True, 413, used_nouns)

# # --- Valid Implausible ---
all_data += generate_syllogisms(valid_implausible_same_chain, "valid_np_same_chain", True, False, 428, used_nouns)
all_data += generate_syllogisms(valid_implausible_diff_chain_A, "valid_np_diff_chain_A", True, False, 439, used_nouns)
all_data += generate_syllogisms(valid_implausible_diff_chain_B, "valid_np_diff_chain_B", True, False, 516, used_nouns)
all_data += generate_syllogisms(valid_implausible_diff_chain_C, "valid_np_diff_chain_C", True, False, 340, used_nouns)
all_data += generate_syllogisms(valid_implausible_diff_chain, "valid_np_diff_chain", True, False, 527, used_nouns)

# # --- Invalid Implausible ---
all_data += generate_syllogisms(invalid_implausible_same_chain, "invalid_np_same_chain", False, False, 389, used_nouns)
all_data += generate_syllogisms(invalid_implausible_diff_chain_A, "invalid_np_diff_chain_A", False, False, 362, used_nouns)
all_data += generate_syllogisms(invalid_implausible_diff_chain_B, "invalid_np_diff_chain_B", False, False, 360, used_nouns)
all_data += generate_syllogisms(invalid_implausible_diff_chain_C, "invalid_np_diff_chain_C", False, False, 368, used_nouns)
all_data += generate_syllogisms(invalid_implausible_diff_chain, "invalid_np_diff_chain", False, False, 400, used_nouns)
```

Output:

- 12 files will be generated in the working directory.
- All files will contain simple English syllogisms.
- File names will contain language identifiers.

Store the output of this run in a dedicated folder (for example):

- symbolic_normal
  or
- nonsymbolic_normal

depending on the file type.

3.2 Run 2 – Remove One Premise Configuration

For the same file:

1. Comment out the function calls under the heading “Normal”.

2. Uncomment the function calls under the heading “Remove one Premise”.
3. Execute the file again.

Output:

- 12 files will again be generated.
- Store the output in a separate folder (for example):
- symbolic_remove_one
  or
- nonsymbolic_remove_one

Each file must be stored in its corresponding folder to ensure proper tracking.

4. Final Folder Structure After All Runs

After executing all four base files in both configurations, you should have eight total folders.

Symbolic (4 folders)

1. symbolic_normal
2. symbolic_normal_4var
3. symbolic_remove_one
4. symbolic_remove_one_4var

Non-Symbolic (4 folders)

5. nonsymbolic_normal
6. nonsymbolic_normal_4var
7. nonsymbolic_remove_one
8. nonsymbolic_remove_one_4var

5. Modifying “Remove One Premise” Data

Next, process all files in the “remove one” folders through:

modify.py

Apply this to:

- symbolic_remove_one

- symbolic_remove_one_4var
- nonsymbolic_remove_one
- nonsymbolic_remove_one_4var

The script will randomly remove one premise where required and adjust formatting if necessary

6. Merging Data

6.1 Merge Symbolic Files

Merge all symbolic folders:

- symbolic_normal
- symbolic_normal_4var
- symbolic_remove_one
- symbolic_remove_one_4var

Merge files language-wise so that each language has a single combined file, for example:

- symbolic_en.json
- symbolic_ur.json
- symbolic_fr.json

6.2 Merge Non-Symbolic Files

Merge all non-symbolic folders:

- nonsymbolic_normal
- nonsymbolic_normal_4var
- nonsymbolic_remove_one
- nonsymbolic_remove_one_4var

Again, merge language-wise to produce:

- nonsymbolic_en.json
- nonsymbolic_ur.json
- nonsymbolic_fr.json

7. Translation

Translation is performed separately for symbolic and non-symbolic datasets.

7.1 Symbolic Translation

Run:

translate_syllogisms_symbolic.py

This script translates symbolic syllogisms into their respective languages and stores them in separate language files.

7.2 Non-Symbolic Translation

Run:

translate_syllogisms.py

This script translates non-symbolic syllogisms into their respective languages and stores them in separate language files.

8. Final Output Structure

After translation, the final dataset structure should resemble:

```
final_symbolic/
  en.json
  ur.json
  fr.json
  ...

final_nonsymbolic/
  en.json
  ur.json
  fr.json
  ...
```

1. ST4 Complex Syllogisms Data Generation Process

2. Initial Steps (Same as Simple Syllogisms)

Follow the **Simple Syllogisms pipeline exactly up to Step 6**. This includes:

1. Running all four base files under the "Normal" configuration.
2. Running all four base files under the "Remove one Premise" configuration.
3. Organizing outputs into eight structured folders (symbolic and non-symbolic).
4. Running `modify.py` on all "remove one" folders.
5. Merging all symbolic files language-wise.
6. Merging all non-symbolic files language-wise.

At the end of Step 6, you should have:

- Merged symbolic files (by language)
- Merged non-symbolic files (by language)

Do not perform full translation at this stage.

3. Noun Translation for Non-Symbolic Data

For the merged non-symbolic folder only:

Run the following script:

translate_nouns.py

Purpose

- This script replaces only the **nouns** in each syllogism with nouns from the respective target language
- The rest of the syllogism (logical structure and connecting phrases) remains in English.
- Each language file will produce a corresponding noun-translated version.

Output:

- Language-specific files where only nouns are translated.
- Logical structure remains unchanged.

4. Passing Noun-Translated Files to Complex Syllogism Generator

Take the noun-translated non-symbolic files and:

- Pass each language file to its respective complex syllogism generation file.
- Ensure that each language file is used with its corresponding complex syllogism configuration.

This will generate complex syllogisms where:

- Nouns are already localized.
- Logical structure and reasoning complexity are introduced at this stage.

5. Symbolic Complex Syllogisms

For symbolic data:

- Do not run `translate_nouns.py`.
- No noun translation is required.

Simply:

- Pass the merged symbolic files directly to the respective complex syllogism generation files.

Symbolic syllogisms do not require noun localization since they rely on symbolic variables rather than lexical nouns.

6. Final Output Structure

After completing these steps, you will have:

- Complex non-symbolic syllogisms (with nouns translated per language)
- Complex symbolic syllogisms (processed directly without noun translation)

Each language will have its own corresponding complex dataset file.