Introduction

Dear colleagues,

This document serves as a data dictionary but it also covers various aspects about the *Projet du Dictionnaire*.

As each of you will be working on specific subject, instead of getting overwhelmed by the volume of information, please study the brief description of each sections before reading the one most suited to your needs.

At the beggining of each sections are listed the contents of each subsections.

They are as following :

- **General Structure of the Results** : I describe the general contents of the zip file with all results, `complete_data.zip` .
- **Dictionary Structures, Graph Metrics and Algorithm Results** : Read this section to study the structures of dictionaries and their contents. The section also describes graph theoretic aspects.
- **Prompts Data** : Read this section to study the inter-model agreement on the prompts and the model-human agreement on 25 prompts.
- **Methodology** : Read this section if you are interested in the methodological aspects of this project OR if you are not familiar with some aspects of the project.
- **Produced Figures** : A collection of figures used throughout this document.

PLEASE NOTE : The sections are written assuming familiarity with the main concepts and ideas of the project. If you find any concept you are not familiar with, please consult the **Methodology** section.

PLEASE ALSO NOTE: The preprocessing for the data used by DeepSeek has since been improved upon. Saddly, some named entities slipped through and have been removed. If someone has the time, it would be simple however to fix by filtering out from the raw `dico` format the symbols that are named entities. Then you could re-execute my pipeline `dico` to `minset` , using the cleaned DeepSeek data.

QUAESO NOTA : All the code will be available on GitHub.

General Structure

You will find all the results and data in the zip file `complete_data.zip`.

You will find at its root 3 folders, one for each model (GEMMA, LLAMA, QWEN, DEEPSEEK) (TODO ADD B OF PARAMS HERE). They will each have a CSV file with the most important metrics and results, and a folder with various plots.

Each of these will also have 8 subfolder within, one for each dictionary (CIDE, LDOCE, MWC, WCDT, WEDT, WILD, WLDT, WN). **Please see the file `dictionnaires.txt` for an explanation in french of what each of these dictionaries are**). In these, you will find various *raw results*, suchs as the lists of words found in different structures, and everything else that was produced.

```
➜  disamb_results git:(add-digraph-reductions) ✗ tree -d GEMMA LLAMA QWEN
GEMMA
├── CIDE
├── LDOCE
├── MWC
├── WCDT
├── WEDT
├── WILD
├── WLDT
└── WN
LLAMA
├── CIDE
(...)
```

TODO?

Dictionary Structures, Graph Metrics and Algorithm Results

This section covers the results more in details.

The subsections are as follow :

- **Dictionary Structures** : A data dictionary for the main results.
- **Other Graph Metrics** : Data dictionary for various graphs metrics.
- **Reduction Algorithm Results** : Data dictionary for the results of the reduction algorithm.

Dictionary Structures

- Let's now describe the data dictionary for the main results.

For each model, a CSV file named `main_results.csv` is produced. Each row in that file refers to one of the 8 dictionaries and the columns are the most important metrics regarding dictionary structures. Do note that the following data dictionary switches this around and describe each of these columns in its own table.

Field	Type	Description
nb_of_vertices	*int*	The number of defined words in the dictionary.
nb_of_words_in_kernel	*int*	The number of words in the kernel. The kernel is obtained by removing all non-defining and non-defined word *recursively* (i.e., words that are made non-defining by removing other non-defined words are removed until none can be removed anymore). Any word in the kernel is either in a satellite or in a core.
nb_of_words_in_core	*int*	The number of words in the core. The core is the biggest strongly connected component of the kernel.
nb_of_words_in_satelites	*int*	The number of words in the various satelites. Any strongly connected component found in the kernel that is not the biggest one is named a satelite.
nb_of_words_in_rest	*int*	The number of words in the rest. The rest is made up of all words removed to obtain the kernel.
nb_of_words_mfvs	*int*	The number of words in the MFVS. This number is obtained by including words in the MFVS through the reduction algorithm, then finding the MFVS of

Field	Type	Description
		the remaining graph that could not be reduced any further using binary linear integer programming (Gurobi).
nb_of_undefined	*int*	The number of words that are not defined.
nb_of_undefining	*int*	The number of words that are not used to define any other.
size_biggest_satelite	*int*	The number of words in the biggest satelite (the second biggest strongly connected component of the kernel).

Additionally, you can find in the each of the folders described above a list of the words found in each structure : `rest_words.json`, `kernel_words.json`, `core_words.json` and `satellite_words.json`. For the words of the **MFVS**, they are contained in two different files : `included_words.json` (words included in the MFVS by the reduction algorithm) and `reduced_core_mfvs_word.json` (words included in the MFVS by the MFVS-solver by cracking the reduced core).

I will also be adding the raw graphs in our `dico` format. They will be on a Google Drive link and will be shared on demand.

Other Graph Metrics

For those wanting to do a deeper dive into graph metrics, you can find for each disambiguated dictionary or first definition dictionary a file named `graph_metrics.json`.

For example, here is the complete breakdown of `graph_metrics.json` :

Field	Type	Description
nb_of_vertices	*str*	Total number of vertices (defined words) in the digraph, followed by that count

Field	Type	Description
		expressed as a percentage of itself (always 100 %).
nb_of_words_in_kernel	*str*	Size and percentage of the kernel obtained by recursively removing all non-defining / non-defined words.
nb_of_words_in_core	*str*	Size and percentage of the core (largest strongly-connected component of the kernel).
nb_of_reduced_core	*str*	Words (and %) that remain in the core **after** all reduction rules have been applied.
nb_of_words_in_satelites	*str*	Size and percentage of the union of all satellite SCCs (kernel SCCs other than the core).
nb_of_words_in_reduced_satelites	*str*	Words (and %) still present in satellites after reduction.
nb_of_words_in_rest	*str*	Size and percentage of the “rest” (vertices outside the kernel).
reduced_core_mfvs_size	*str*	Portion of the MFVS that lies **inside** the reduced core (number / %).
nb_of_words_mfvs	*str*	Total number and percentage of vertices in the Minimal Feedback Vertex Set (MFVS) produced by the reduction algorithm + MILP solver.
initial_nb_of_loops	*int*	Number of self-loop arcs in the original digraph.
initial_nb_of_arcs	*int*	Total number of directed arcs (edges) in the original digraph.

Field	Type	Description
final_nb_of_arcs	*int*	Number of arcs that remain after reduction.
initial_nb_of_out_deg_0	*int*	Vertices whose out-degree is 0 in the original digraph.
initial_nb_of_in_deg_0	*int*	Vertices whose in-degree is 0 in the original digraph.
initial_nb_of_out_deg_1	*int*	Vertices with out-degree = 1 in the original digraph.
initial_nb_of_in_deg_1	*int*	Vertices with in-degree = 1 in the original digraph.
initial_nb_of_sccs	*int*	Count of strongly-connected components (SCCs) in the original digraph.
size_of_biggest_scc	*int*	Cardinality of the largest SCC in the original digraph.
size_of_2nd_biggest_scc	*int*	Cardinality of the second-largest SCC.
nb_of_sccs_kernel	*int*	Number of SCCs that belong to the kernel.
nb_of_sccs_reduced_digraph	*int*	Number of SCCs remaining after the full reduction procedure.
size_biggest_satelite_kernel	*int*	Size of the largest satellite SCC within the kernel.
size_biggest_reduced_satelite	*int*	Size of the largest satellite SCC after reduction (0 if satellites disappear).
nb_undefined	*str*	Number and percentage of words that are **never** defined by any definition (dangling targets).
nb_undefining	*str*	Number and percentage of words that

Field	Type	Description
		never occur in any definition (dangling sources).
top n out deg	*list[dict]*	List of vertices with the highest out-degree; each element is `{"out_degree": int, "vertex": str}`.
top n in deg	*list[dict]*	List of vertices with the highest in-degree (same structure).
top n total deg	*list[dict]*	List of vertices with the highest total degree (in + out), same structure as above.

Reduction Algorithm Results

The reduction algorithm is an important part for two reasons.

1. It reduces de size of the core by identifying words not part of the grounding set, so that it is easier to find its MFVS using a solver like Gurobi
2. It creates a partial solution as some reductions include words in the MFVS
3. For a formal description in French of the reduction, please read the file `réductions.txt`, for a formal description in English please see the article `confluent_reductions.pdf`, an article of our colleagues.

To study the result of the reduction algorithm, one can find in each of the subfolder (one for each dictionary for each model) a file named `algorithm_results.json`, which contains the following fields :

Field	Type	Description
total_time	*str*	The total time it took to reduce the digraph.
total_reductions_applied	*int*	The total number of reductions

Field	Type	Description
		applied.
Number of included vertices in the solution	*int*	The number of words that were included in the MFVS by the reduction algorithm.
Number of excluded vertices in the solution	*int*	The number of words that were excluded from the MFVS by the reduction algorithm.
Number of vertices remaining to be solved	*int*	As the name implies : the remainder of words from which the MFVS is to be extracted using a MFVS solver.
reductions	*dict*	A dictionary of the used reductions.
REDUCTION_NAME.time	*str*	The total time spent on a given reduction.
REDUCTION_NAME.count	*int*	The total number of times a given reduction was applied.
REDUCTION_NAME.time_percentage	*str*	The proportion for time out of the whole to a given reduction.
REDUCTION_NAME.count_percentage	*str*	The count proportion out of the whole for a given reduction.

We have two sets of reductions for now : a confluent one (meaning you can apply any of the reductions in the set as much as possible in any order and you still obtain the same reduced graph) and a non-confluent one, where that property is lost.

Prompts Data

This section described the data for those interested in the disambiguation of dictionaries and the performances of various LLMs at performing the task of disambiguating words in dictionary definitions.

- **Main Results** : This is a data dictionary for the main results about dictionary disambiguation.
- **Human Results** : Read this section if you are interested the agreement between human disambiguations and LLM disambiguations.
- **Tools** : Read this section for technical details on how these disambiguation were conducted.

Main Results

The main results we have are the inter-model agreement (how much the different models agree on the answers for various prompts) and the number of retries it took for each model.

For each dictionary, you will find pairs of CSV files of the same length, where one file has all the prompts and the other the answer from the model and the number of times it took it to get it right. This was necessary as some prompt files are CSV files of above 2GB!

You can find them in the folder `prompts_data`. **You will find the prompts in a Google Drive link that I will share on demand** (with one file per dictionary in there) and the disamb answers for the three models for all dictionaries. **Do note** that words with 20 retries were word that could be disambiguated by the models and for which we opted to give the first definition as an answer.

The answer files have the following fields (where `MODELNAME` will be either `llama`, `qwen`, or `gemma`) :

- `MODELNAME_retries` : THe number of wrong answers before the model provided a possible answer
- `MODELNAME_answers` : The final answer by the model

The prompt files have the following fields :

- `prompt` : The prompt fed to the model.
- `defined_word` : The defined word in the current definition
- `possible_answers` : The list of numbered words that are possible answers.

Additionnally, it is important to note that some prompts could never be solved by our various models (i.e. even when given a selection of possible answers to chose from, it could not pick one). This resulted in assuming the first definition was the right one. Here is the total numbers of such prompts in each dictionary for each model

```
Ordered summary (dictionary × model)
────────────────────────────────────
WLDT
  GEMMA: 0 /  17810  missing
  LLAMA: 2 /  17810  missing
  QWEN : 54 /  17810  missing

WILD
  GEMMA: 0 /  23298  missing
  LLAMA: 19 /  23298  missing
  QWEN : 13 /  23298  missing

CIDE
  GEMMA: 0 / 261334  missing
  LLAMA: 163 / 261334  missing
  QWEN : 3207 / 261334  missing

LDOCE
  GEMMA: 1 / 365412  missing
  LLAMA: 4597 / 365412  missing
  QWEN : 10764 / 365412  missing

MWC
  GEMMA: 253 / 954363  missing
  LLAMA: 21252 / 954363  missing
  QWEN : 21251 / 954363  missing

WEDT
  GEMMA: 12 / 313858  missing
  LLAMA: 1525 / 313858  missing
  QWEN : 1200 / 313858  missing

WN
  GEMMA: 0 / 658837  missing
  LLAMA: 3869 / 658837  missing
  QWEN : 10037 / 658837  missing

WCDT
  GEMMA: 0 /  82470  missing
  LLAMA: 61 /  82470  missing
  QWEN : 620 /  82470  missing
```

Humans Results

In addition to LLMs disambiguations, we have 15 subjects who disambiguated 25 definitions.

Each subject has its own CSV file in the Results folder with the following two fields :

- `word` : The token currently being disambiguated
- `selected_answer` : This answer needs to be parsed using simple code as answers have the following string format : `Modified vocable: well_3   POS: adverb`, so you need to retrieve the value between vocable: and POS:

Tools

TODO describe the tools used for disambiguation

Methodology

This section is describes in more details some of the methodological details of this document. The *Projet du Dictionnaire* is a multidisciplinary one aimed at furthering our knowledge of Minimum Grounding Sets. The following is an **informal** overview of the most important methodological aspects.

- Graph Theory : Read this for a barebones introduction to graph theory.
- Dictionaries as Graphs : Read this for an explanation on how to see dictionaries as graphs
- Graph Reductions : What are graph reductions and why are they so cool?
- Minimum Feedback Vertex Set : **IMPORTANT**, as the Minimum Feedback Vertex Set is the same as the **Minimal Grounding Set**.

Graph Theory

TODO DESCRIBE GRAPH THEORY

Dictionaries as Graphs

TODO Explain Dictionaries as Graphs

Graph Reductions

TODO

Minimum Feedback Vertex Set

TODO

Produced Figures

TODO ADD USED FIGURES HERE