Challenge 1: LLM-to-AutoML query translator

AutoML uses a list of symbols such as ('LoadData', 'ProcessData', 'TrainModel') to build a series of ML pipelines. The list can be obtained from parsing natural language (with reduced syntax and vocabulary, based on the English language). For example,

- (AutoML) English: "`mean building age X dataset`"

would be parsed into:

- (AutoML) system input: `[Load(dataset=X), Select(column=age, column=type), Filter(type=building), Apply(mean)]`

The challenge is to build a translator from generic (LLM) English to the reduced vocabulary (AutoML) English so that can it be used as input for the AutoML system. So, from English to 'AutoML' English. For example,

- (LLM) English: "`How old are buildings on average considering the X dataset`"

would be translated to:

- (AutoML) English: "`mean building age X dataset`"

Challenge 2: LLM-to-AutoML query translator

Additionally, one could provide recommendations if the initial English queries cannot be directly translated to (AutoML) English on how to reduce ambiguity. This means using the LLM to recommend 'correct' alternative questions for ambiguous inputs. For example,

- (LLM English): "`How old are buildings considering the X dataset`"

Would generate the follow-up questions:

- (LLM recommendation 1): "`Are you referring to building average?`" (so that AutoML would return the average building age)

and

- (LLM recommendation 2): "`Should all ages of buildings be returned?`" (so that AutoML would select and return the age of buildings)

Note: The examples above employ a simple (toy) usage of the AutoML system where a single processing pipeline would be returned.

Note on implementation: Any implementation would have to use open LLM models such as

- ollama: https://github.com/ollama/ollama

Potentially using libraries such as:

- txtai library: https://github.com/neuml/txtai or spacy https://github.com/explosion/spacy-llm

Appendix (more details)

1. Dataset – we shall assume a dataset `(D)` with four columns: `id` (integer), `surface in square meters` (float), `height in square meters` (float) and mass in kilograms (float). The dataset is assumed to have some simple, visible structure (clusters).

2. AutoML Symbols: hierarchical structure of symbols `(S)` available to AutoML and what they correspond to:

- `LoadData` – load a dataset
- `SplitTrainTestData` – splits a dataset into a training and testing dataset
- `PreprocessData` – coresponds to any of the symbols associated with it below
    - `CalculateVolume` – calculates volume as a product of a height and surface
    - `CalculateDensity` – calculates the density as the mass divided by volume
    - `RemoveMissingRows` – removes rows with missing data
    - `RemoveMissingColumns` – removed columns with missing data
    - `ImputeValuesColumn` – imputes values for a column
- `ModelData`
    - `ClusterData` – clusters data with a clustering algorithm (algorithm not important)
    - `KnnRegression` – learns a KNN-based regression model (model is just training data and a similarity metric)
- `TestModel`
    - `ClusterCount` – counts clusters out of a clustering result
    - `RegressionMSEError` – calculates the mean squared error of a regression model given a test dataset

3. Reduced English vocabulary for LLM

The LLM needs to re-write input questies into equivalents using a reduced vocabulary of tokens:

(T): `load, dataset, calculate, volume, cluster, regression, data, count, clusters, a, and, perform, calculate, using, id, surface, height, mass`

4. Approach to consider

The desired behavior is to transform a (reasonable) NLP query into a sequence of AutoML symbols. A more complicated approach is to try to generate two sequences, the second containing also the data columns for the dataset that are to be fed to an operation.

Example:

If we consider as input (to the LLM base system) the question:

`(Q1): Given the objects dataset, how many types of objects can be found, according to their volume?`

To answer the question, the AutoML system would have to have as input, the following series of symbols:

`(S1) LoadData, CalculateVolume, ClusterData, ClusterCount.`

Additionally, one could associate CalculateVolume to (`SurfaceInSquareMeters` and `HeightInMeters`) and `ClusterData` to the output of `CalculateVolume`.

Arriving at the sequence (S1) of symbols described above using an LLM one can use an intermediary step in which the LLM rewrites into a subset of English (which is easily parsable) the initial query. The rewriting would be performed by the LLM given a prompt which instructs it to re-write queries using specific terms. An intermediate (or rewritten query) using (T) for (Q1) could be:

(T1): `load dataset, calculate volume using the surface and height columns, cluster and count clusters`

Wrapping everything together:

- consider as known: (D), (S), (T) and several questions like (Q1). One should invent a couple more questions and corresponding symbol lists, based on the types of available symbols in (S). This is necessary in order to make sure that the LLM prompt devised can generalize over question types. Tip: one should also support commands like: `Train a model that can estimate density based on surface and mass given the objects dataset.`
- The objective is to make the LLM model rewrite the initial query into easily parsable phrases that contain only terms from (T). Think on how such a parser could be implemented. Try implementing a very simple one (by associating for example words from (T) to symbols from (S))
- The translation of initial queries to intermediate queries must not be necessarily correct - it is assumed that the AutoML systems can detect incorrect sequences of symbols that are obtained by parsing the intermediate queries. One could generate several intermediate queries and associated sequences such as (S1) in order to have better chances of finding a correct one that can be executed.