Class Definitions in Manchester Syntax

1. Large Language Model (LLM)

Class: LLM
SubClassOf: LM
EquivalentTo:

$$\text{LM} \sqcap (\text{wasTrainedOn some BigData}) \sqcap (\text{architectureType value Neural})$$

Explanation: An LLM is a Language Model (LM) that has been trained on **BigData** and has an architecture type of **Neural**.

2. Small Language Model (SLM)

Class: SLM
SubClassOf: LLM
EquivalentTo:

$$\text{LLM} \sqcap (\text{hasSize value “small”})$$

Explanation: An SLM is an LLM that has a size characterized as "small".

3. BigData

Class: BigData
SubClassOf: Data
EquivalentTo:

$$\text{Data} \sqcap (\text{hasVolume some xsd} : \text{integer}[\geq 1,000,000])$$

Explanation: BigData is a type of Data that has a volume of at least 1,000,000 units.

4. TrainingData

Class: TrainingData
SubClassOf: Data
EquivalentTo:

$$\text{Data} \sqcap (\text{usedFor value Training})$$

DisjointWith: TestingData, ValidationData
Explanation: TrainingData is data used for training purposes. It is disjoint with TestingData and ValidationData.

5. TestingData

Class: TestingData
SubClassOf: Data
EquivalentTo:

$$\text{Data} \sqcap (\text{usedFor value Testing})$$

DisjointWith: TrainingData, ValidationData
Explanation: TestingData is data used for testing models. It is disjoint with TrainingData and ValidationData.

6. ValidationData

Class: ValidationData
SubClassOf: Data
EquivalentTo:

$$\text{Data} \sqcap (\text{usedFor value Validation})$$

DisjointWith: TrainingData, TestingData
Explanation: ValidationData is data used for validating models during training. It is disjoint with TrainingData and TestingData.

7. TextualData

Class: TextualData
SubClassOf: Data
EquivalentTo:

$$\text{Data} \sqcap (\text{producedBy value Writing})$$

DisjointWith: SpokenData
Explanation: TextualData is data produced through writing. It is disjoint with SpokenData.

8. SpokenData

Class: SpokenData
SubClassOf: Data
EquivalentTo:

$$\text{Data} \sqcap (\text{producedBy value Speaking})$$

DisjointWith: TextualData
Explanation: SpokenData is data produced through speaking. It is disjoint with TextualData.

9. InputLayer

Class: InputLayer
SubClassOf: NNLayer
EquivalentTo:

$$\text{NNLayer} \sqcap (\text{layerPosition value “first”})$$

Explanation: InputLayer is a neural network layer that comes first in the network architecture.

10. HiddenLayer

Class: HiddenLayer
SubClassOf: NNLayer
EquivalentTo:

$$\text{NNLayer} \sqcap (\text{isDeep value true})$$

Explanation: HiddenLayer is a neural network layer that is part of the deep layers in the network.

11. OutputLayer

Class: OutputLayer
SubClassOf: NNLayer
EquivalentTo:

$$\text{NNLayer} \sqcap (\text{layerPosition value “last”})$$

Explanation: OutputLayer is the neural network layer that comes last in the network architecture.

Additional Definitions and Notes

Properties Introduced

wasTrainedOn
Type: Object Property
Domain: LM
Range: Data

architectureType
Type: Data Property
Domain: LM
Range: xsd:string

hasSize
Type: Data Property

Domain: LM
Range: xsd:string or xsd:integer

hasVolume
Type: Data Property
Domain: Data
Range: xsd:integer

usedFor
Type: Object Property
Domain: Data
Range: UsagePurpose

producedBy
Type: Object Property
Domain: Data
Range: ProductionMethod

layerPosition
Type: Data Property
Domain: NNLayer
Range: xsd:string

isDeep
Type: Data Property
Domain: NNLayer
Range: xsd:boolean

Individuals Introduced

UsagePurpose
Individuals: Training, Testing, Validation

ProductionMethod
Individuals: Writing, Speaking

Additional Classes

LM (Language Model)
Class: LM
Explanation: A general class for language models.

NNLayer (Neural Network Layer)
Class: NNLayer
Explanation: A class representing layers in a neural network.

Disjointness Declarations

DisjointClasses: TrainingData, TestingData, ValidationData

DisjointClasses: TextualData, SpokenData

Defining Word Embeddings

WordEmbedding
Class: WordEmbedding
SubClassOf: Model
EquivalentTo:

$$\text{Model} \sqcap (\text{usesMethod some EmbeddingMethod})$$

Explanation: WordEmbedding is a type of Model that uses some EmbeddingMethod.

GloVe
Class: GloVe
SubClassOf: WordEmbedding
EquivalentTo:

$$\text{WordEmbedding} \sqcap (\text{usesMethod value GlobalVectors})$$

Explanation: GloVe is a WordEmbedding model that uses the Global Vectors method.

EmbeddingMethod
Class: EmbeddingMethod
Individuals: GlobalVectors, Word2VecMethod, FastTextMethod

Clarifying Neural and Probabilistic Models

Since some models use both neural and probabilistic methods (e.g., GloVe), you might avoid declaring NeuralModel and ProbabilisticModel as disjoint. Instead, define models that can have multiple architecture types.

Model
Class: Model

ArchitectureType
Class: ArchitectureType
Individuals: Neural, Probabilistic, NeuralProbabilistic

Revised LLM Definition
Class: LLM

SubClassOf: LM
EquivalentTo:

LM⊓(wasTrainedOn some BigData)⊓(architectureType some ArchitectureType)

Explanation: LLM is an LM trained on BigData and can have one or more architecture types.

Defining Languages

If you want to include languages that LLMs support:

Language
Class: Language
Individuals: English, French, Spanish, etc.

Property: speaksLanguage
Type: Object Property
Domain: LLM
Range: Language

Updated LLM Definition with Language Support
Class: LLM
SubClassOf: LM
EquivalentTo:

LM⊓(wasTrainedOn some BigData)⊓(architectureType some ArchitectureType)⊓(speaksLanguage some Lang

Explanation: LLM is an LM that is trained on BigData, has some architecture type, and supports at least one language.

Activation Functions

ActivationFunction
Class: ActivationFunction
SubClassOf: ArchitectureComponent
Individuals: ReLU, Sigmoid, Tanh
Explanation: ActivationFunction represents functions used in neural network layers.

ArchitectureComponent
Class: ArchitectureComponent
Explanation: General class for components of neural network architectures.

Example Definitions for Authors and Publications

Author
Class: Author

ScientificPaper
Class: ScientificPaper

Conference
Class: Conference

Properties
writes
Type: Object Property
Domain: Author
Range: ScientificPaper
InverseOf: writtenBy

presentedIn
Type: Object Property
Domain: ScientificPaper
Range: Conference

Final Notes and Recommendations

- **Consistency Checking:** After defining your classes and properties, use an ontology editor like Protégé to input these definitions. Run a reasoner (like HermiT or Pellet) to check for logical consistency.
- **Disjointness:** Be cautious with declaring classes as disjoint unless you are certain that no individual can belong to both classes.
- **Property Characteristics:**
  - *Functional Properties:* If a property can have only one value for a given individual (e.g., `hasSize` for an SLM), you can declare it as a Functional Property.
  - *Inverse Properties:* Defining inverse properties (e.g., `writes` and `writtenBy`) can help with data retrieval and inference.
  - *Datatype Properties:* Ensure that your datatype properties have the correct range (e.g., `xsd:string`, `xsd:integer`, `xsd:boolean`).
- **Annotation:** Add annotations to classes and properties to provide definitions, comments, and other metadata.

Example of a Complete Class with Annotations

Class: LLM

Annotations: `rdfs:comment` "A Large Language Model is a language model trained on big data using neural network architectures."

SubClassOf:

LM,
wasTrainedOn some BigData,
architectureType some ArchitectureType,
speaksLanguage some Language