# Knowledge Graph Seminar - Team 01:

## Chosen Dimensions and Metrics

Dimensions:

- Accessibility (Weight = 0.4 ➝ The most important aspect. The KG is useless if not reachable.)
- Completeness (Weight = 0.3 ➝ Data completeness shows how much a data source adds into our KG.)
- Accuracy (Weight = 0.3 ➝ Correctness shows us if our data is wrong or not.)

Metrics:

- **Accessibility: Provisioning of public endpoint**
    - Weight = 0.45
    - 1 if SPARQL and REST API
    - 0.75 if SPARQL or REST API endpoint is publicly available
    - 0.5 any form of offline data
    - 0 otherwise
    - *Reason*: This metric is important, even in our use case, as it helps developers or other users perform queries on the knowledge graph with ease. If SPARQL is available, then nearly full points should be given, as SPARQL is a RDF query language, but having a general REST API endpoint available should also be rewarded as the data is still queryable (it just contains minor workarounds). If both SPARQL and REST API are provided, then full points should be given, since a broader reach of developers is possible.

- **Accessibility: Retrievable format**
    - Weight = 0.45
    - 1 if RDF export available
    - 0.75 if only JSON structure available
    - 0.5 if only semi–structured data available
    - 0 otherwise
    - *Reason*: In general in this work area, people know RDF and are probably working directly with RDF, meaning this structure of data should be rewarded the most. But, json-ld exists and transforming json into json-ld shouldn't be too difficult and also json is a nicely structured data form which can be mapped to RDF or other. With semi-structured data the user/developer has to adapt to the data form specifically, for instance in web scraping the data form may be structured but requires more work to convert, still this data is acceptable.

- **Accessibility: Content negotiation**
    - Weight = 0.1
    - 1 if content negotiations is supported
    - 0 otherwise
    - *Reason*: Data is often required in different formats. Just giving 2 is really good, but more is better. So if 2 or more are provided then this should be rewarded well. Different developers in a team prefer their “favorite” data format to work with.

- **Completeness: Instance completeness**
    - Weight = 0.5
    - $m = \frac{1}{N}\Sigma\frac{number\ of\ values\ from\ classes\ \&\ properties\ in\ instance\ in\ subset}{number\ of\ total\ values\ from\ \ classes\ \&\ properties\ according\ to\ DS}$; $N \dots\ subset\ size$
    - *Reason*: The weight on data completeness is relatively high since it is important to have as much data as possible to follow the properties and classes from our given predefined domain specification.
    - *Approach:* For this metric we will take a subset of the current data source and apply the formula over the subset, receiving a mean value of instance completeness of this subset.

- **Completeness: Population completeness**
    - Weight = 0.5
    - $m = \frac{number\ of\ objects\ per\ domain\ represented\ in\ the\ data\ source}{total\ number\ objects\ per\ domain}$
    - *Reason*: We use the GTKG as the gold standard simply because if we get a value above 1, then our data source has new lodging business data to enrich the GTKG. While, if the resulting value m is less than 1, then the data source has a lower chance of enriching the GTKG with more information.
    - *Approach*: For this metric we will take data from our gold standard and from our KG and compare it.
    - $if\ m > 1$ then $m = 1$

- **Accuracy: Formal Semantic validity**
    - Weight = 0.5
    - $m = \frac{|\{\ o\ |\ (s,p,o) \in r \land o \in L \land semValid(o)\ \}|}{|\{o\ |\ (s,p,o) \in r \land o \in L\ \}|}$
    - *Reason*: This metric shows us how good our source is. The source can have many data points in it, representing many new instances, but it would require a lot of work to adapt to it if the property values are incorrect. Additionally, if many property values are incorrect, then one doesn’t know if the entire source is even trustable. We would evaluate this on a subset of our data source and check if the literal values in the property values are correct (e.g. the city Vienna does not have the postal code 6020).
    - *Approach*: For this metric we will take a subset of s,p,o triples from the knowledge graph through the function $semValid()$ and apply a formal rule that checks the Formal Semantic validity of the object.

$semValid()$ rules examples:

- Postal code:
    - length of 5
    - starting from 01 to 99
- Phone number: has to start with +49 or 0049 followed by a valid area code
    - starting from 02 to 09
    - total length between 3 and 5

- **Accuracy: Syntactic validity**
    - Weight = 0.5
    - $m = \frac{|\{o \mid (s,p,o) \in r \land o \in L \land synValid(o)\}|}{|\{o \mid (s,p,o) \in r \land o \in L\}|}$
    - *Reason:* This metric shows us how good our source is. The source can have many data points in it, representing many new instances, but it would require a lot of work to adapt to it if the data types are incorrect.
    - *Approach:* This metric will be applied over a subset of the given data source, where the values are evaluated based on their given data type, compared to the data type they must have. A function $synValid()$ gets the needed data type for reference and the current property value's data type.

    $synValid()$ rules example:

    - Postal code:
        - given: *https://schema.org/Text*
        - needed: *https://schema.org/Integer*
    - Address*:*
        - if a lodging business is present in GTKG and data source then we use the Levenshtein distance to calculate the difference between two strings/numbers in order to detect syntactic errors