Dies sind die (Zwischen)-Ergebnisse meiner Bearbeitung der Accessibility-Feature-Liste aus *Traveling to Unknown Buildings: Accessibility Features for Indoor Maps*.
gez.: Max Wendler

Zunächst war zu entscheiden, in welcher Struktur die Features vorliegen sollten. Im Paper wurde bereits festgestellt, dass eine hierarchische Struktur ihre Schwächen (sinngemäß "signpost weit verteilt") hat, und eine flache Struktur bzw. objektorientierte mit A11yJSON vorgeschlagen. Der Blick in die Liste aller Features in **Alignment.csv** verrät, dass viele von ihnen nicht via A11yJSON strukturiert wurden (dort leere Zellen). Gleiches gilt für die Paper-eigene Hierarchie *category-feature-property* (usw.). Also: flach, d.h. Feature-Liste im Sinne der ersten Spalte von **Alignment.csv.** Im Folgenden habe ich die Features anhand ihrer eigenständigen Semantik betrachtet, d.h. ohne Berücksichtigung der vorhandenen Strukturierungen. Dabei ergab sich die Liste bereits als hierfür geeignet, d.h. die Features haben tatsächlich eigenständige Semantik.

Anmerkung: Im Original **Alignment.csv** liegt beim Feature *type of services* ein syntaktischer Fehler vor - es erstreckt sich über zwei Zeilen. Dessen Korrektur fand vor allen weiteren Bearbeitung statt.

Welche Prozesse habe ich nun angewandt?

- Unterteilung der Datei **Alignment.csv** in:
    - Outdoor-Features: die (meines Erachtens) ausschließlich für den Außenbereich genutzt werden sollten = **exklusiv Outdoor** (außer Doppel)
    - Building-Features: Features (meines Erachtens) ausschließlich zur Annotation von als Gebäude getaggten Features + Grobunterteilung durch *building parts* = **exklusiv Building** (außer Doppel)
    - Indoor-Features: alle Features, die auch im Innenbereich eingesetzt werden können = **auch Indoor**

  wobei es zu Doppelzuordnungen zu Building & Outdoor kam, die in der neuen Datei **buildings.csv** via eigener Spalte (mit Begründungen) vermerkt sind. Man könnte auch sagen, ich habe alles aussortiert, was nicht für Indoor in Frage kommt.

- Dokumentation möglicher Implikationen der Building-Features für Indoor-Features in dem Gebäude via eigener Spalte in **buildings.csv**

- Annotation aller Features mit einem oder mehreren *Semantic tags*, eine Art eigene Kategorisierung (wo Zugehörigkeit zu mehreren möglich ist) zum späteren Gruppieren, Zusammenfassen, Filtern. (Die im Paper beschriebene Kategorisierung fehlt für zahlreiche Features → Spalte *AccessibleMaps Categories*). Dabei wurde nicht jede passende Kategorie gewählt, sondern subjektiv nach Potential für genannte künftige Tätigkeiten entschieden. Ihre Bedeutungen und Auftreten befinden sich in **semantic_tags.csv.**

- Feature-Umbenennungen aus verschiedensten Gründen, mit dem grundsätzlichen Ziel: Jeder Name soll als semantisch klarer "primary key" der Tabelle dienen können, während im Original viel der Semantik nur durch Betrachtung der OSM-Tags zu

erschließen ist. Änderungen und Gründe sind **renamed.csv** dokumentiert. Die wohl häufigsten waren Uneindeutigkeit worauf es sich bezieht (“backrest does not exist” wobei nach Tag die Lehne einer Bank gemeint war); fehlende Ersichtlichkeit, was überhaupt gemeint ist (“beep-beep in most countries” für Fußgängerüberwege mit Audio-Barrierefreiheit); Orthographie.

- Feature-Korrekturen über Umbenennungen hinaus, bspw. der OSM-Tags, und Löschungen von Features aus Redundanz-Gründen (existiert scheinbar oft aufgrund der hierarchischen Strukturierung im Sinne des Papers.) Eine davon erschien mir nur sinnvoll, konnte mir aber nicht sicher sein, da es die Paper-Hierarchie betrifft. Dazu nur ein Vorschlag am Ende der Liste.

Anmerkung: Entsprechend der Prozesse gab es rigorose Betrachtung nur bei den Namen. Im Sinne von Korrektur steht dies für alle weiteren Spalten noch aus (nicht durch mich). Bei Betrachtung der Ergebnisse zur Erstellung dieser Dokumentation sind mir schon wieder einige Schreibfehler in den OSM-Tags aufgefallen.

- Implementierung des (ausführlich kommentierten) Python-Skripts **integrity.py,** das überprüft, ob auf dem langen Weg der manuellen Bearbeitung Features verloren gingen (Funktion `validate_forward_existance`), keine neuen Features aus dem Nichts dazu kamen (`validate_backward_existance`) und alle dokumentierten Löschungen tatsächlich vorgenommen wurden (`validate_deletion`). Möchte man nur eines davon überprüfen, kann man die Funktionen in der main-Funktion auskommentieren. Output der erkannten Probleme erfolgt je Schritt in **[Schrittbezeichner]_issues_log.csv**.
  Unabhängig dieser Überprüfungen zählt das Skript die Features der Original-Datei (**Alignment.csv**), der Output-Dateien und die Anzahl der Löschungen und errechnet eine Differenz, die 0 sein sollte und ist (**feature_counts.csv)**
  Es enthält auch Prüfung der Tabellensyntax, die auf Eingabe schlecht strukturierter CSV-Daten (z.B. kein kontinuierlich gefüllter Header) hinweisen soll, aber i.d.R. unerheblich ist.

Anmerkung: Die Ausgangsdatei (und vermutlich auch andere Dateien der KIT-Datenbank) verwendet Semikolons als Trennzeichen und in den Einträgen. Durch Einschließung letzterer in Anführungszeichen hat z.B. Excel beim Datenimport keine Probleme. Allerdings wollte ich damit im Skript nicht umgehen müssen, d.h. basierend auf einem Trennzeichen parsen, das ausschließlich als solches genutzt wird. Daher wurden alle Input-Dateien für **integrity.py** (Original und Ergebnisse) mit Trennzeichen “|” abgespeichert.
(Via Excel: Neue Excel-Datei → Reiter “Daten” → “Aus Text/CSV” → im Fenster nach Auswahl der Datei “Daten transformieren” → ggf. “Erste Zeile als Überschriften verwenden” → “Schließen und Laden” → Export des Tabellenblatts als CSV mit Windows-System-Spracheinstellungen (Systemsteuerung → Region → Weitere Einstellungen) auf Listentrennzeichen “|” gesetzt).

Damit ergeben sich folgende Dateien, die ich hiermit liefere:

- **Filtered Alignment.xlsx**: Unterteilung von Alignment in die Tabellen(blätter) Indoor, Buildings, Outdoor

    - als Input für **integrity.py** als **indoor.csv, buildings.csv, outdoor.csv** in **/data**

- **semantic_tags.csv:** Auflistung und Erklärung der zur Kategorisierung der Features benutzten semantischen Tags

- **Rückverfolgbarkeit.xlsx**: Umbenennungen, Korrekturen, Löschungen
    - als Input für **integrity.py** als **renamed.csv, edited.csv** in **/data**

- **integrity.py** mit Tests und Testdaten in **/test**
    - **table_verif_test.py:** vollautomatischer Test für Überprüfung des Header-Syntax' mit Testdaten in **/test/data/headers.csv**
    - **validators_test.py:** Skript zur Erzeugung der Validationslogs für jeweils eine Variante des (korrekten / falschen) Outputs pro Validationsmethode mit Testdaten in diversen CSVs in **/test/data/** (alle neben **headers.csv**; genauer: analog zum **integrity.py**-Input sowie Testdaten, um diese leeren Initialisierungsdaten zu überschreiben)
        - Output d. Logs zur manuellen Überprüfung korrekten Verhaltens in **/test/data/results**

- Ergebnisse der letzten Integritätsprüfung in **/results** mit Dokumentation der Auflösung fälschlich erkannter Konflikte je Validationsschritt in ***_issues_log_resolution.csv**.