**GML-INSPIRE**: Sprache erkennen

# Einführung

Das INSPIRE-Attribut „language“ soll automatisch, ohne manuellen Aufwand, gefüllt werden. Dazu soll der Inhalt der .gml-Datei analysiert und die verwendete Sprache erkannt werden.

Dafür wurden zwei Skripte erstellt, die unterschiedlich eingesetzt werden können, aber beide dieselbe Logik zur Spracherkennung einer .gml-Datei beinhalten. Für beide Skripte muss die ISO-Codetabelle *langcode.csv*, mit den Sprach-Codes (z.B. de:ger oder en:eng), im selben Verzeichnis wie das Python-Skript befinden.

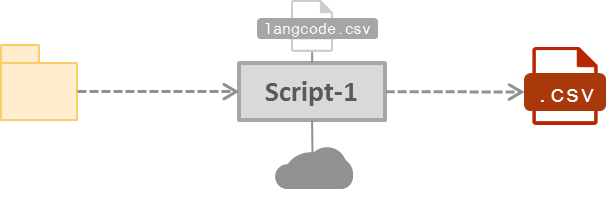
1. **Script-1\_Folder-IN\_\_csv-OUT.py**

Das erste Skript ist dafür gedacht mehrere .gml-Dateien innerhalb eines Verzeichnisses auf die Sprache zu analysieren und anschließend das Ergebnis in eine .csv-Datei abzuspeichern, die sich dann in dem Verzeichnis des Python-Skripts befindet.

Zur Auswahl des Verzeichnisses gibt es zwei Ansätze.

* 1. Entweder das Python-Skript wird in das Verzeichnis geschoben, in dem sich auch die .gml-Dateien befinden, oder
  2. der Pfad wird selbst bestimmt (Zeile 12).

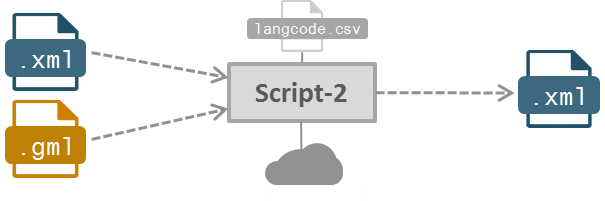
Letzteres bietet den Vorteil, dass das Python-Skript nicht redundant in mehreren Verzeichnissen kopiert werden muss. Nachteil ist, dass der Code immer bearbeitet, statt nur verschoben werden muss.



1. **Script-2\_Files-IN\_\_xml-OUT.py**

Das zweite Skript ist für den produktiven Einsatz gedacht und analysiert nur eine ausgewählte .gml-Datei auf die Sprache und erzeugt eine entsprechende .xml-Datei aus einem Template, welches mit dem Metatag „language“ befüllt wird. Um die .xml-Datei zu erzeugen muss das Template entweder im Verzeichnis liegen oder entsprechend als Pfad definiert werden. Der vollständige Pfad der zur analysierenden Datei muss in Zeile 12 des Skripts eigetragen werden. Zum Beispiel:

file = "D:/Profiles/User/Desktop/Datei.gml".



Die Spracherkennung wurde mit Hilfe einer externer Library (Textblob) durchgeführt.

TextBlob ist eine Python-Bibliothek zur Verarbeitung von Textdaten. Es bietet eine einfache API für das Eintauchen in allgemeine Natural-Language-Processing-Aufgaben (NLP, siehe Abschlussbericht Kapitel 4.3), zum Beispiel für die Spracherkennung. Neben Textblob wurden zwei weitere Libraries überprüft (langdetect, nltk.corpus (stopwords)), bei der Auswahl der Library zur Bestimmung der Sprache hat sich nach einem Benchmark allerdings die Library „Textblob“ durchgesetzt. Es ist jedoch zu beachten, dass die Library einen Internetzugang benötigt, um die Sprache zu erkennen.

# Vorgehensweise des Skripts

Zur Veranschaulichung wird im Folgenden der Ablauf des Skripts an einer Beispieldatei erklärt. Die Datei, welche auf die Sprache analysiert werden soll, lautet:

* **HH\_WFS\_AuBS-8E9D9AB6-950D-4957-9EDB-40A4CCCAFDAC-aubs\_einzelbiotope-2.0.0.gml**

Um die Sprache eines .gml-Dokuments zu analysieren, muss vorerst der Inhalt der jeweiligen Datei extrahiert werden. Dazu wird im ersten Schritt der komplette „text“ einer .gml-Datei herausgefiltert und einer Variable zugeordnet. Das heißt, dass alle Values innerhalb der Attribute extrahiert wird. Die Attribute selber werden in diesem Case nicht betrachtet, da nicht abzusehen ist, ob die Tags immer mit der Sprache des Inhalts übereinstimmen. Sollte dies der Fall sein, gäbe es die Möglichkeit auch die Tags miteinzubeziehen. Der Output aus dem ersten Schritt sieht wie folgt aus:

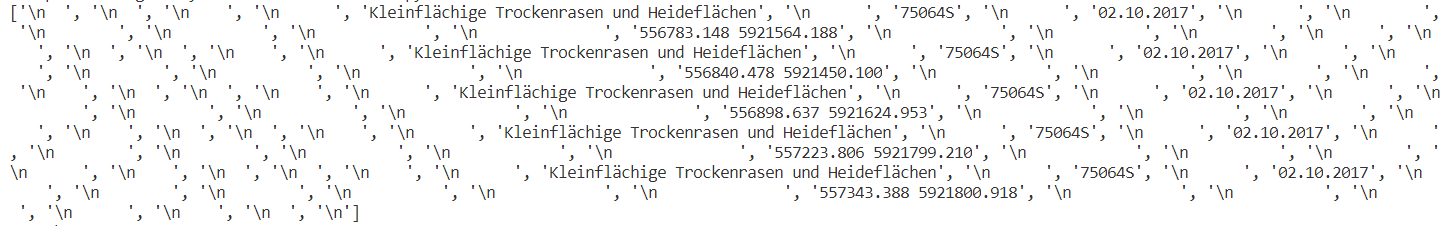


Abbildung 1 Reine Textausgabe aus der .gml-Datei

Jeder value wird in eine Liste gespeichert und mit einem Komma von anderen Values getrennt. Um die Sprache korrekt zu analysieren, müssen anschließend alle numerischen Werte herausgefiltert werden und die Struktur gesäubert werden. Alle Ziffern und Zeilenumbrüche werden entfernt, weil sie für die Spracherkennung unerheblich sind. Das vorläufige Resultat sieht so aus:



Abbildung 2 Entfernung der unnötigen Zeichen

Der Text wird im nächsten Schritt in Tokens gespalten. Doppelte Tokens und alle Tokens, die eine Länge von weniger als drei haben, werden aussortiert, da die doppelte Betrachtung von Wörtern nicht relevant ist. Weiterhin sind Wörter, die weniger als 3 Zeichen haben, meist nicht eindeutig einer Sprache zuzuordnen und werden daher entfernt. Übrig bleibt in diesem Fall:



Abbildung 3 Finaler Text zur Analyse der Sprache

Abschließend wird eine Spracherkennung über den nicht-leeren und über 2 Wörter enthaltenen Text durchgeführt. Nach der Spracheausgabe (z.B. „de“) wird diese noch über ein Matching in die bestehende ISO-Norm umgeschrieben (z.B. „ger“).



Abbildung 4 Sprachausgabe nach dem Matching nach ISO

Nach dem Erkennen der Sprache wird nun wieder in den beiden Skripten unterschieden:

1. **Script-1\_Folder-IN\_\_csv-OUT.py**

Im ersten Skript X wird der Dateiname, die erkannte Sprache und die extrahierten Wörter jeweils in getrennt durch einen Doppelpunkt in eine .csv-Datei gespeichert. Der Prozess wird für alle im Ordner befindlichen .gml-Dateien wiederholt.

1. **Script-2\_File-IN\_\_xml-OUT.py**

Im zweiten Skript wird die erkannte Sprache letztlich in eine durch ein Template erzeugte .xml-Datei als Metatag festgehalten.

# Test-Auswertung

Mit Hilfe von 1. Script-1\_Folder-IN\_\_csv-OUT.py konnte eine Auswertung erstellt werden. Als Abschlussresultat über den gesamten Korpus hat sich folgendes Bild ergeben (siehe Abbildung 5).

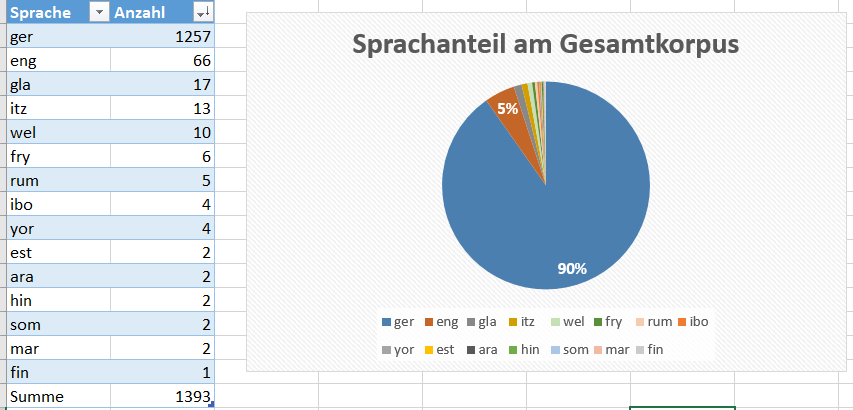


Abbildung 5 Performance der Spracherkennung

Deutsch ist die am häufigste erkannte Sprache, gefolgt von Englisch und einigen falsch klassifizierten Sprachen, die sich aufgrund der schlechten Datenqualität (Kauderwelsch in den Dateiinhalten) ergeben.

Die 5% Fehlerquote ergibt sich größtenteils aus Texten wie in Abbildung 6, die nicht eindeutig einer Sprache zugeordnet werden kann. Theoretisch wäre es möglich diese Fälle jeweils als Sonderfall zu betrachten und nicht in die Analyse einzubinden, allerdings gäbe es zu viele Sonderfälle, die einzeln herausgefiltert werden müssten, sodass sich der Code ziemlich stark aufbläht und an der Realität vorbeigeht.



Abbildung 6 Sonderfälle, die normalerweise keiner Sprache zugeordnet werden können

# Ausblick

Wie eingangs erwähnt, besteht die Möglichkeit die Tags auch zur Analyse einzubeziehen, dazu müsste aber eindeutig geklärt werden, ob im internationalen Kontext englische Attributnamen verwendet werden oder es sich immer um die Sprache des Textes handelt.

Weiterhin können Sonderfälle eingebaut werden, um die Genauigkeit weiter zu steigern und „ungültige Wörter“ aus der Analyse auszuschließen. Zudem müssten mehr diversere Sprachdaten zur weiteren Analyse verfügbar sein, damit die Sprachbibliothek auf weitere Sprachen getestet werden kann.