De datascientist als nieuwe doelgroep binnen de FEWS community

Gebruikersdag FEWS-NL 6 juni 2023

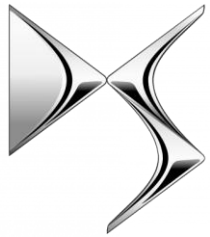
Roger de Crook, HDSR

Rudie Ekkelenkamp, Deltares

FEWS into the future

Inhoud

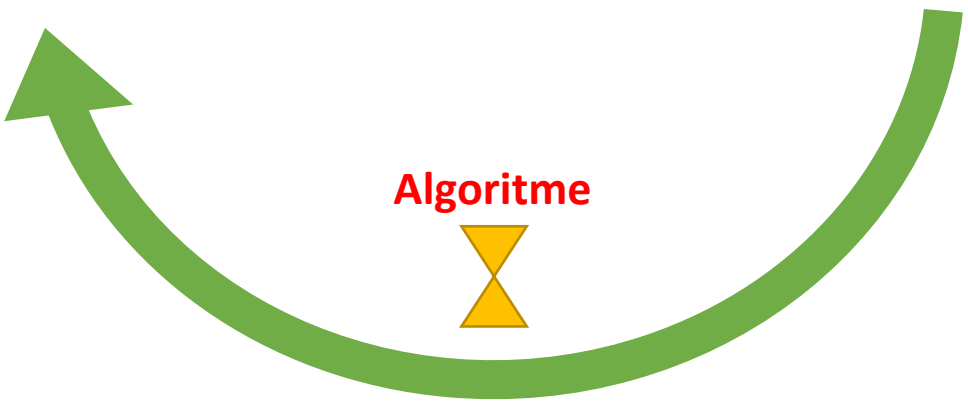
- FEWS in de context van Data Science
- Gebruik & Performance
- Client Side Restriction
- Server Side Restriction
- Verschil met Open Data



Data Science versus Business Intelligence

Klassieke BI
= metingen hebben bekend veronderstelde gedragingen en die worden getoetst met bekende kennisregels

Data Science
= “trial & error” binnen Research & Development om patronen te ontdekken in historische data die nog onbekend waren



Wat is er anders aan de datascientist?

- Requests van datascientists aan PI-webservice zijn onvoorspelbaar in grootte, frequentie en tijdstip
- Bulkbevragingen aan PI-webservice kunnen requests van andere bevragers in de weg zitten
- Risico voor externe partijen zoals Slim Water Management portalen of Perceelwijzer portaal of Droogteportaal of Lizard/HydroNet
- Reden van bevraging: andere analysetechnieken kunnen toepassen die (nog) niet door FEWS gefaciliteerd worden

Verschil tussen FEWS-componenten

OC

⇒ analytische expertgebruiker van grafieken en tabellen en kaarten

⇒ uitgebreide voorgedefinieerde analysetechnieken, visualisaties en workflows

WebOC

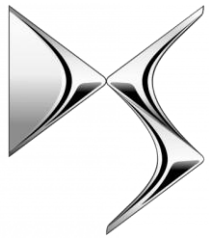
⇒ dagelijkse bediener of waterbeheerder (hier & nu)

⇒ beperkte voorgedefinieerde analysetechnieken, visualisaties en workflows

PI-webservice & Archive

⇒ scripts door data scientists of externe/interne portalen

⇒ zelf te definiëren ‘on-the-fly‘ analysetechnieken, visualisaties en workflows



Welke info wil je weten over het gebruik?

Serverbelasting: grafieken met drempelwaarden

ICT-monitoringstools: voor inzicht in de grootte van requests, responsetijd en afhandelduur (bijvoorbeeld via proxy service zoals NGINX)

Gebruikerspecifieke statistieken: over gebruik van elke PI-webservice en bevraagde data

Bescherming: tegen teveel simultane bevragingen

Hoe kan je grip houden op het gebruik?

- Bij voorkeur dual MC of datakopie naar standalone met localdatastore en eigen PI-webservice, altijd op een aparte virtuele machine zodat andere FEWS-processen niet verstoord kunnen worden

- Voorkom extra vertraging door altijd een specifiek Filter-ID op te geven, want dat maakt het bevragen van locaties, parameters en tijdreeksen veel efficiënter.

- Buiten FEWS inregelen van:
  - throttling (rate limiting per user)
  - load balancing (prioriteren van user requests of bijschakelen van resources)

Voorbeeld van bescherming via client side

- hdsrfewspy van Renier Kramer
  - Gebaseerd op hkvfewspy (o.a. Mattijn van Hoek) en fewspy (Daniel Tollenaar)
- Extra functionaliteiten:
  - Client side authenticatie (gebruikersnaam en token)
  - Client side autorisatie

  => check rechten in organisatie-specifieke Github-repo (permissies en filters)
  - Opknippen request in kleine brokken (nu timeseries, op termijn ook samples)
  - Wachttijd tussen requests van minimaal 1 seconde
  - “in memory” (JSON, XML of Pandas-dataframe) & “to file” (JSON, XML of CSV)

Toegang beperken tot PI-webservice (client side)

Script
>>> import hdsr_fewspy
>>> api = hdsr_fewspy.Api()

Credentials (email + token):
als API argumenten
of in een .env file

Script
>>> api.get_timeserie()

hdsr_fewspy.Api

1 loop

hdsr_fewspy_auth

hdsr_pygithub

Request /timeseries met onlyHeaders=True + showStatistics=True

get_time_series

N loops

Eventueel start + end aanpassen
om N timestamps
binnen bandbreedte te krijgen

Request /timeseries met onlyHeaders=False + showStatistics=False

Download timeseries
(unieke location-
parameter-qualifier)

Lijstje met PI_XML responses

Lijstje met PI_JSON responses

1 pandas dataframe

1 csv (padverwijzing)

Aandachtspunten

- Houtje-touwtje 3.0
- PiPy: pip install (en eventueel conda install)
- Ondersteuning: hoe actueel wordt een pakket bijgewerkt en wijzigingsverzoeken doorgevoerd?
- Github: account vereist met eigen losstaande repo voor beheer van rollen en rechten (via pygithub)
- Wees voorzichtig met het parallel afvuren van requests

Hoe van client side naar organisatie centraal?

- Autorisatie & authenticatie (OpenIDConnect) toepassen in Python-scripts via Active Directory om rechten voor gebruik van PI-webservice te faciliteren

- Buiten FEWS inregelen van throttling (rate limiting per user) en load balancing (prioriteren van user requests of bijschakelen van resources)

- Benodigde ontwikkelingen binnen & buiten FEWS

Uitdagingen binnen FEWS → Deltares

- Bij synchrone afhandeling van requests is continu monitoren van belasting essentieel, anders krijgen gebruikers een timeout
- Wens: als de PI-webservice ook asynchrone afhandeling van requests kan ondersteunen, worden taken één voor één in een wachtrij gezet en krijgt de gebruiker een trigger zodra de data klaar staat (download-URL)

- Momenteel geeft de PI-webservice de volledige respons terug
- Wens: met paginatie kan een request opgeknipt worden en middels seperate pagina's als respons teruggegeven worden (goede performance aan client-kant)

Uitdagingen buiten FEWS → IHW/HWH

- Digitale Delta API's ondersteunen geen qualifiers
- Wens: voor een tijdreeks van een parameter op een locatie moet je idealiter middels een qualifier kunnen aangeven bijv. welk tijdinterval gebruikt is, welke meettechniek, welke meetbron, welke validatietechniek, etc.

FEWS into the future

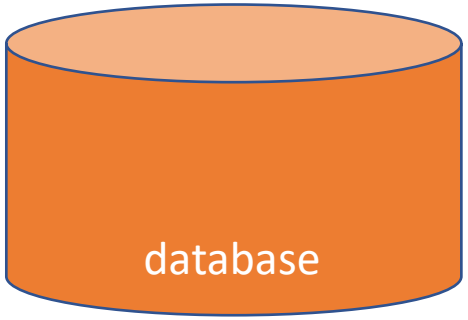
OpenIDconnect via Active Directory

Requesting:
1. Locations
2. Parameters
3. Qualifiers
4. Timeseries
5. Periods
6. Data

Checking:
1. #requests/minute
2. Size of request
3. Priority of user
4. Response time
5. Handling time

Handling:
1. Pool of requests
2. Async execution 1-by-1
3. Chop into pieces
4. Return download trigger

Processing:
1. Execute queries
2. Cache
3. Return data



Wat is het verschil met Open Data?

- Alleen bedoeld voor intern gebruik aan data-scientists met vertrouwen, niet buitenwereld
- Open Data aan buitenwereld bij voorkeur via FEWS Archive met standalone PI-webservice

Uitsmijter

- Naast REST API zou een GraphQL API interessant kunnen zijn voor datascientists

- Voordelen & nadelen:

[GraphQL vs. REST in 2023: Top 4 Advantages & Disadvantages (aimultiple.com)]