RosaeNLG 3.0.0

Natural Language Generation

Systeemvereisten:	node.js
Ontwikkeld door:	Ludan Stoecklé and contributors ; https://rosaenlg.org/
Contactpersoon:	joachim.ganseman@smals.be

Apache License 2.0

Functionaliteiten

RosaeNLG is een Natural Language Generation templating engine, gebouwd op de Pug templating engine voor node.js. In RosaeNLG kan je templates definiëren om JSON objecten om te zetten in een beschrijving in natuurlijke taal. RosaeNLG maakt daarbij in de achtergrond gebruik van woordenboeken om de correcte grammaticale regels toe te passen voor lidwoorden, vervoegingen, verbuigingen en naamvallen). Er worden momenteel 5 talen ondersteund: Engels, Frans, Duits, Spaans en Italiaans.

RosaeNLG is een library voor node.js, maar er bestaan wrappers voor Java en een Docker image. Die laatste komt met een API die toelaat om templates toe te voegen, en dan toe te passen op een JSON object. Er is de mogelijkheid om variabelen te definiëren in een template die hergebruikt kunnen worden, synoniemen te definiëren, if-else constructies op te zetten, en variaties in de output te bepalen. Er is een rudimentair systeem van geheugen beschikbaar waarmee kan vermeden worden in herhaling te vallen.

RosaeNLG leunt sterk op de Pug.js library. De templates genereren HTML, het is de bedoeling dat de output in een browser wordt weergegeven. Templates moeten geprogrammeerd worden in de opmaaktaal van Pug.js die wat lijkt op Markdown. Meertalige output vereist minstens één template per taal. Dat elke variatie of synoniem expliciet bepaald moet worden in het template, maakt dat templates vele malen langer kunnen worden dan de voorziene output. De grammaticale ondersteuning en de documentatie zijn soms onvolledig, maar de tool is al voldoende uitgebouwd voor rapportage-usecases.

Conclusies & Aanbevelingen

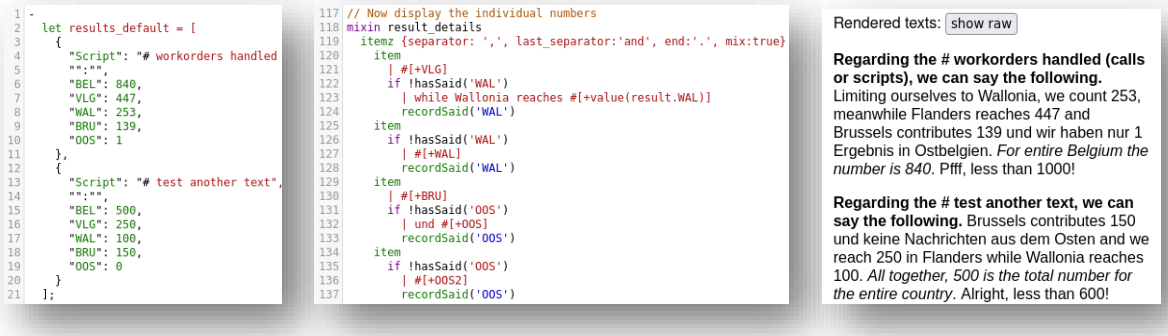
RosaeNLG wordt actief ontwikkeld en ondersteunt momenteel 5 talen. Belangrijke grammaticale functionaliteit ontbreekt nog, zo mankeren sommige werkwoordsvervoegingen. Soms moet men teruggrijpen naar de documentatie van Pug.js zelf. RosaeNLG zet JSON objecten om naar HTML-geformatteerde output, wat het vooral geschikt maakt voor rapportage. Voor andere usecases kan het nuttig zijn alternatieven zoals SimpleNLG en RiTa te overwegen. Grafische interfaces vindt men eerder bij commerciële spelers zoals AxSemantics en Narrative Science.

Testen & Resultaten

We nemen een spreadsheet met statistische gegevens, metingen en tellingen, waarvan we ook graag een tekstuele beschrijving hebben, bijvoorbeeld om een inhoudelijk relevante push-notificatie te kunnen sturen als er een nieuwe lijn met gegevens wordt toegevoegd. Na omzetting van de spreadsheet via CSV in een reeks JSON objecten (1 per rij, 1 veld per kolom), kunnen we een template maken om zo'n object te beschrijven. Dat komt neer op het schrijven van een reeks "mixin" functies in Pug. Werk bottom-up: maak eerst eenvoudige basisfuncties die een paar alternatieve beschrijvingen genereren voor elk van de velden. Combineer deze later in complexere constructies, die de volgorde van vermelding variëren, of met bepaalde beschrijvingen activeren op basis van de inhoud van de velden, of lijsten opsommen of zinnen samenvoegen met komma's en voegwoorden. Om te vermijden daarbij iets te genereren dat in een eerdere variatie al werd vermeld, definiëren we een token na iedere beschrijving, waarop nadien getest kan worden.

Het template wordt afgerond met een geschikte intro en afsluiting afhankelijk van de inhoud van de data. Met intro, afsluiting, en genereren van tekst voor een JSON met 6 velden, waarbij er ongeveer 3 opties per veld zijn vermeld en de volgorde kan variëren, is een template die gemiddeld 5 volzinnen genereert al snel 140 lijnen code lang (zonder witregels en commentaar).

Met een lokale Docker image van RosaeNLG kunnen we via de API deze template uploaden, lokaal opslaan, en aanroepen met de JSON-objecten als parameter. Daarvoor moet eerst nog de template gepackaged worden waarvoor zelf nog code moet geschreven worden. De API call responses bevatten referenties die bewaard moeten worden in de applicatie voor verder gebruik. De afbeeldingen hieronder tonen 2 JSON objecten, een klein deel van een template, en de resulterende gerenderde tekst.



```
-
  let results_default = [
    {
      "Script": "# workorders handled
      "":"",
      "BEL": 840,
      "VLG": 447,
      "WAL": 253,
      "BRU": 139,
      "OOS": 1
    },
    {
      "Script": "# test another text",
      "":"",
      "BEL": 500,
      "VLG": 250,
      "WAL": 100,
      "BRU": 150,
      "OOS": 0
    }
  ];
```

```
// Now display the individual numbers
mixin result_details
  itemz {separator: ',', last_separator:'and', end:'.', mix:true}
    item
      | #[+VLG]
      if !hasSaid('WAL')
        | while Wallonia reaches #[+value(result.WAL)]
        recordSaid('WAL')
    item
      if !hasSaid('WAL')
        | #[+WAL]
        recordSaid('WAL')
    item
      | #[+BRU]
      if !hasSaid('OOS')
        | und #[+OOS]
        recordSaid('OOS')
    item
      if !hasSaid('OOS')
        | #[+OOS2]
        recordSaid('OOS')
```

Rendered texts: show raw

Regarding the # workorders handled (calls or scripts), we can say the following. Limiting ourselves to Wallonia, we count 253, meanwhile Flanders reaches 447 and Brussels contributes 139 und wir haben nur 1 Ergebnis in Ostbelgien. *For entire Belgium the number is 840.* Pfff, less than 1000!

Regarding the # test another text, we can say the following. Brussels contributes 150 und keine Nachrichten aus dem Osten and we reach 250 in Flanders while Wallonia reaches 100. *All together, 500 is the total number for the entire country.* Alright, less than 600!

We merken toch nog verschillende mankementen op. Zo vormen JSON veldnamen met spaties of de HTML karakters <> erin al een probleem. Het resetten van de zelf gedefinieerde geheugentokens lijkt niet altijd te lukken en het is uit de documentatie niet duidelijk hoe ver de scope ervan reikt of waar je ze best aanmaakt en verwijdert. Andere outputformaten dan HTML zijn er vooralsnog ook niet.

Gebruiksvoorwaarden & Budget

RosaeNLG is open source software uitgebracht onder Apache 2.0 licentie, die ook gebruik in private en commerciële projecten toelaat. Wie het wilt integreren in een bestaand project kan de npm library gebruiken, wie de functionaliteit liever aanroept via API kan de Docker image gebruiken.