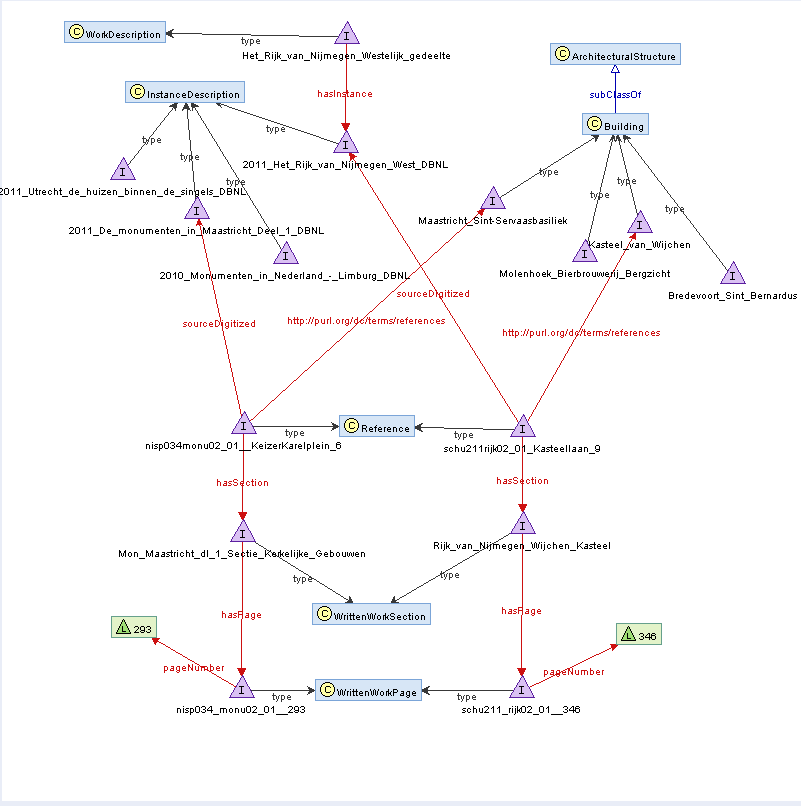
## Semantisering van monumentinformatie

### Verslag van een lopend onderzoek, Gerard Kuys met dank aan Roland Cornelissen

1. **Context en opzet**  
     
   Doel van dit project is te bereiken dat men vanaf een Wikipedia-pagina over een monument kan springen naar het (DBNL-) tekstfragment dat ook werkelijk over het relevante onderwerp gaat. Op dit moment verwijzen de externe links op Wikipedia-pagina’s bijna altijd naar een globale paragraaf in de DBNL-tekst of zelfs alleen naar de inhoudsopgave. In plaats daarvan moet een bepaald monument worden geïdentificeerd als onderwerp van de Wikipedia-pagina, moet tegelijkertijd hetzelfde object als onderwerp worden herkend van een specifieke passage in een DBNL-tekst, en moeten de ‘coördinaten’ van die specifieke passage als navigatiemiddel worden gebruikt bij het springen naar de externe tekst. In de oplossing die dit project beproeft, wordt gebruik gemaakt van het feit dat via DBpedia monumenten reeds nu als zodanig herkenbaar zijn, dat zij van een typologie voorzien zijn, en als RDF over de hele inhoud van Wikipedia bevraagbaar zijn. Wanneer nu de tekst van de DBNL-bestanden zodanig kan worden verwerkt, dat de monumenten in DBpedia in deze teksten worden herkend en dat op basis van deze herkenning eenduidige coördinaten kunnen worden gegenereerd, dan wordt het mogelijk om de DBNL-teksten van specifieke markeringspunten voor specifieke onderwerpen te voorzien.  
     
   In juli 2013 kende de Nederlandstalige Wikipedia/DBpedia ruim 100.000 Nederlandse monumenten (rijksmonumenten, alsook gemeentelijke en provinciale monumenten), definitiekwesties daargelaten. Dit aantal is ruimschoots voldoende om te dienen als de referentieverzameling, waarmee DBNL-teksten worden vergeleken om te zien of passages een monument tot onderwerp hebben. Want de stap die dit project wil maken is dat niet langer tekst met tekst wordt vergeleken, maar dat monumenten *als onderwerp* en als concept in een kennisbank belangrijk worden. Monumentinformatie zit niet meer in een tekstfragment met tal van talige problemen (naamgevings- en schrijfvarianten), maar in een resource in een monumentenontologie. Het aanklikken van de DBNL-link op een Wikipedia-pagina moet leiden tot het identificeren van de monumenten waar het om gaat. Bij elk van die monumenten kunnen vervolgens alle of enkele tekstfragmenten worden opgehaald en getoond.
2. **Op te lossen problemen**  
   Zowel in de tekstgerichte oplossingsrichting als in de oplossing die haar heil zoekt in een kennismodel is de eigenlijke aard van het probleem een matchingsprobleem: hoe herken ik een element in de ene verzameling als zijnde een element in de andere verzameling. Dit matchingsprobleem omvat niet alleen het afbeelden van het ene object op het andere (is A in verzameling X gelijk aan A’ in verzameling Y), maar behelst ook verschillen van beschrijvingsperspectief die moeten worden opgelost. Soms staat een DBNL-link bij één enkel object en soms bij een hele lijst. Naar welk monument verwijst zo’n link in laatst-genoemd geval precies?  
   Welgeteld zijn er op dit moment 160 Wikipedia-pagina’s die met een zogeheten externe link verwijzen naar een digitale monumententekst van DBNL. Van die 160 externe links zijn binnen de scope van dit project er 69 nader onderzocht. Omdat externe links in Wikipedia geplaatst kunnen worden overal waar de redacteur dat relevant vindt, is er geen eenduidige manier waarop links naar DBNL-teksten verwijzen. De verschillende manieren waarop externe links vóórkomen worden geïnventariseerd in Bijlage 1.  
     
   De vraag is, in welke gevallen een Wikipedia-link naar een DBNL-tekst kan worden gerelateerd aan één enkel monument. Bijlage 1 laat zien dat binnen de scope van het project er 38 gevallen zijn waarin de externe link op de Wikipedia-pagina min of meer rechtstreeks leidt naar een passage over een monument in een DBNL-tekst. Wanneer we daarnaast proberen om ook de monumenten in de ‘Lijsten van rijksmonumenten in…’ te matchen op de DBNL-teksten, dan explodeert het aantal gevonden monumenten. Een SPARQL-query op alle lijsten van rijksmonumenten in DBpedia (Bijlage 3, query 2) leert ons, dat deze lijsten in totaal 73.017 rijksmonumenten bevatten, waaronder een aanzienlijk aantal dat ook al op een eigen pagina van Wikipedia beschreven wordt en dus een eigen resource in DBpedia heeft. Filteren we dit zoekresultaat uit op die lijsten die binnen de scope van dit project vallen, dan blijkt voor een steekproef (query 3) dat het al snel om 3024 monumenten gaat met hieraan gekoppeld 3713 externe links. Het totale aantal monumenten binnen de scope zal nog aanmerkelijk hoger blijken te liggen, maar eigenlijk is dat niet relevant. Immers, de beperking tot die monumenten die binnen de scope van het project vallen ligt niet in de monumentenlijst die vanuit DBpedia wordt gegenereerd, maar in de verzameling DBNL-teksten waarmee ze moeten matchen. Ligt een monument niet in het gebied dat door de DBNL-teksten bestreken wordt (Gelderland, Limburg of de stad Utrecht), dan is er – als alles werkt zoals het moet – ook geen match. De toetsingslijst van monumenten in DBpedia kan daarom recht-toe recht-aan worden gegenereerd vanuit een query (query 4), die eenvoudigweg alle monumenten ophaalt die de status van rijksmonument hebben. Dat zijn er 67.796.  
     
   Het laatste op te lossen probleem betreft de aard van de match. Wanneer wordt de vermelding van een rijksmonument in een DBNL-tekst geacht over hetzelfde monument te gaan als een passage in een DBNL-tekst? We betreden hier terrein dat in hoge mate bepaald wordt door de technologie achter het zogeheten verenigen (*reconciliation*) van vocabulaires. De situatie die we hier aan de hand hebben, is er een waarin een monumentenresource moet worden vergeleken met een tekst. Computerprogramma’s als DBpedia Spotlight of LODRefine analyseren brokjes tekst of de inhoud van een veld in een matrix, en kijken of de woorden daarin die vermoedelijk overeenkomen met zelfstandige naamwoorden kunnen worden gematcht op (de naam, of andere tekstuele attributen van) een informatieresource. Wat voor informatie-eenheden moeten we in de DBNL-tekst onderkennen om met voldoende zekerheid te kunnen zeggen dat DBNL-tekst X refereert aan DBpedia-monument Y? En bevat het informatieobject aan weerszijden (dus zowel de monumentenresource in DBpedia als het tesktfragment in het DBNL-document) voldoende informatie om goed te *kunnen* matchen?
3. **Het kennismodel**

Zodra er een match is tussen een tekstfragment en een monument, moet de informatie over die match worden vastgelegd. Dit gebeurt in een uitbreiding op de monumentendata in DBpedia, een uitbreiding die hetzij in de triple store van DBpedia zelf, hetzij in een aan DBpedia gelinkte andere dataset bewaard en benaderd kan worden. De structuur van deze data komt zoveel mogelijk overeen met die van andere vocabulaires die over begrippen, personen, boeken en locaties gaan, te weten SKOS, Foaf, DCTerms, Bibo en BibFrame, en natuurlijk de eigen ontologie van DBpedia. Dbpedia-owl:Book is equivalent aan WrittenWork in het kennismodel, een klasse die staat voor het boek en zijn volledige tekst.  
  
Het kennismodel drukt de relatie uit tussen een monument en een stuk tekst. Dat is natuurlijk in de eerste plaats een onderwerpsrelatie, maar verder moet ook worden vastgelegd hoe de tekst intern georganiseerd is (met welke hoofdstukken en paragrafen), en met welke metadata de betreffende tekst in de collecties van bijvoorbeeld bibliotheken is terug te vinden. Ook van de auteurs wordt informatie vastgehouden die in staat stelt om vanuit (de extensie op) DBpedia de relatie te leggen naar datasets over auteurs, zoals bijv. VIAF (<http://www.viaf.org> ).  
  
De manier waarop in dit project de relatie tussen een stuk tekst en zijn onderwerp wordt vastgehouden is georganiseerd rondom resources van het type Reference. Dit is een klasse uit de DBpedia-ontologie die als het ware de property dct:subject van DCTerms verlengt. Behalve dat je via een Reference vasthoudt, dat er een onderwerpsrelatie bestaat tussen een boek en een monument, kun je aan diezelfde Reference ook data vastknopen die vertelt welke plaats in de tekst nu precies over dit onderwerp gaat.  
  
Het kennismodel maakt onderscheid tussen de manier waarop een boek beschreven is (zeg maar, het catalogusrecord met een groot deel van de metadata) aan de ene kant en de eigenlijke, direct digitaal te reproduceren tekst aan de andere kant. Dit onderscheid is relevant, omdat metadata op de catalogus-manier nooit gegevens bevatten over de interne organisatie van een tekst – het verste dat bibliotheken normalerwijze gaan, is het opnemen van de Inhoudsopgave in het catalogusrecord, en ook dit is vooral gebeurd onder de druk van Amazon.com en Bol.com. Maar voor dit project moet wel degelijk per tekstfragment een onderwerp kunnen worden aangewezen. Daarbij moet duidelijk zijn om welke uitgave het gaat, want verwijzingen naar een papieren of digitale editie kunnen navenant van elkaar verschillen. Voor bibliotheekcatalogi is een collectie-item een black-box object dat beschreven is om voorwerp te kunnen zijn van een uitleentransactie. Het kennismodel van *dit* project sluit aan bij het datamodel van Europeana, waarin een cultureel erfgoedobject (waaronder boeken) uitsluitend cultureel erfgoedobject is wanneer het onmiddellijk (digitaal) kan worden opgehaald en aan de lezer in gebruik gegeven.



Figuur 1: Kennismodel met de monumenten St. Servaas en Kasteel van Wijchen

Figuur 1 geeft aan de hand van de St. Servaasbasiliek in Maastricht en het Kasteel van Wijchen weer, hoe Referenties de spil zijn van het kennismodel. Bijlage 2 voegt daar nog een ander voorbeeld aan toe, alsmede een overzicht van de structuur van het kennismodel. Dit model is een graaf in RDF (zie de kennisbank Monumentensemantiek in Ontowiki/Virtuoso), en doet dienst als de referentiestructuur waarin de uit de DBNL-teksten geëxtraheerde monumenten moeten worden geboetseerd.

Het kennismodel staat centraal in de volgende stappen in het DBNL-project:

1. Het vastleggen van de manier waarop de DBNL-teksten zijn georganiseerd;
2. Het herkennen van concepten waaraan in DBNL-tekstfragmenten wordt gerefereerd en die verwijzen naar monumenten of naar de plaatsen waar monumenten zich bevinden;
3. Het extraheren van die concepten conform het kennismodel;
4. Het ‘oplijnen’ (*alignment*) van de uit DBNL geëxtraheerde ontologie met die van DBpedia;
5. Het genereren van de markers voor in de DBNL-tekst voor die monumenten waarvoor een match tussen de DBNL-tekst en DBpedia is gevonden. De data van deze markers worden vastgehouden in resources van de types uit de subhiërarchie UnitOfText in het kennismodel, een hiërarchie die op hoofdlijnen identiek is met die van Bibo.

1. **Tekstorganisatie DBNL-bestanden**

De monumentenboeken die door DBNL zijn gedigitaliseerd zijn aangeleverd als XML-bestand. Het grote voordeel hiervan is, dat aan de hand van de tags de indeling van de tekst kan worden vastgesteld. Die indeling wordt conform het kennismodel ingericht op de manier van het bibo-vocabulaire: er is een boek, en dat is opgedeeld in hoofdstukken die elk weer uit een aantal secties bestaan. Dit levert voor het voorbeeld van het eerste deel van het boek van Marceline Dolfin e.a. een verzameling RDF-resources op, die naast het boek zelf en zijn auteurs de lijst van hoofdstukken is:

****

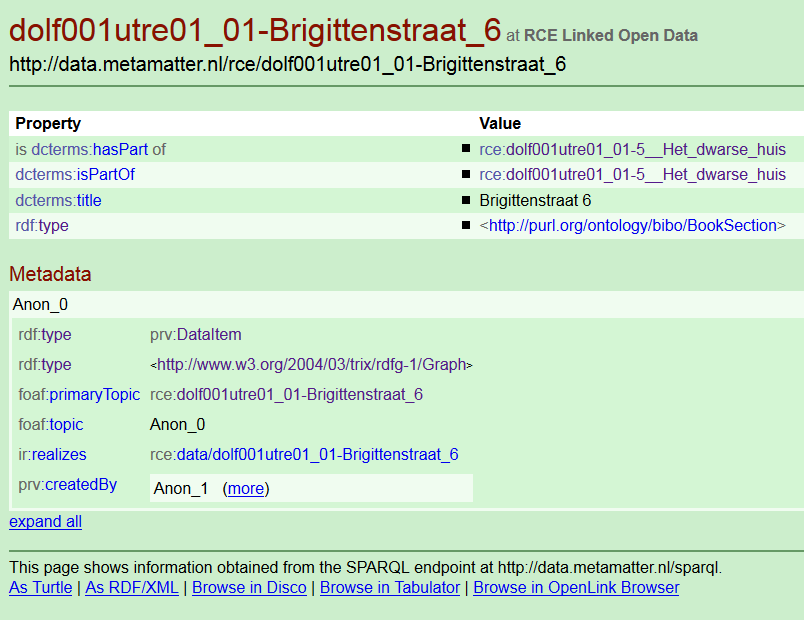
Figuur 2: Hoofdstukken in een monumententekst

Elk hoofdstuk is op zijn beurt de container waarin de paragrafen, secties in het jargon, worden bewaard. Hoofdstuk *Het dwarse huis* bevat secties waarvan een aantal een monument beschrijft:



Figuur 3: Secties in een hoofdstuk

Secties kunnen over verschillende onderwerpen gaan, zo blijkt uit bovenstaand overzicht: naast een inleiding en een bronnenlijst kan het gaan over een enkel monument of een straat. Is een sectie te groot, dan kunnen secties binnen secties worden onderkend, secties zijn de meest fijnkorrelige teksteenheid. Een sectie (in het kennismodel een *WrittenWorkSection*, equivalent aan *bibo:BookSection*) komt tot stand door op basis van de tags in het XML-bestand tekstblokken te onderkennen, waarvan de voorkomende termen worden verzameld en geanalyseerd. In een of meer analyserondes wordt vastgesteld, welke term de grootste kanshebber is om als onderwerp van het tekstblok te gelden. In een volgende slag kan dan worden gekeken, of een bepaalde term een adres representeert, enz.  
  
Op deze manier worden de properties van de sectie opgebouwd (Figuur 4), properties die dienen om te kunnen matchen met de properties van een monument uit de vergelijkingslijst.

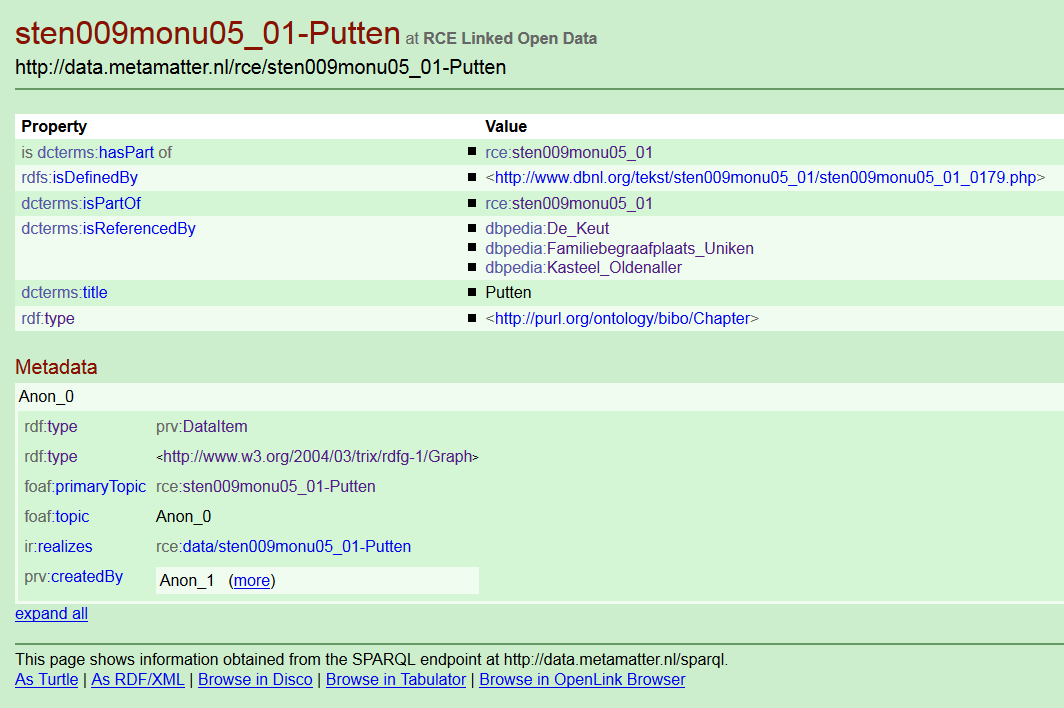


Figuur 4: De eerste properties van een sectie

Wij weten nu, dat er een sectie *Brigittenstraat 6* is binnen het hoofdstuk over *Het dwarse huis* van het boek *Utrecht: de huizen binnen de singels*. Twee vragen resteren:

1. Hoe komen we nu (geautomatiseerd) te weten, dat de sectie ‘Brigittenstraat 6’ inderdaad een monument is op <http://nl.dbpedia.org/page/Lijst_van_rijksmonumenten_in_Utrecht_(stad)/Brigittenstraat> en dat de secties ‘Inleiding’, ‘Geschiedenis’ en ‘Bronnen’ dat niet zijn?
2. Bevat de resource over deze sectie voldoende informatie om in het XML-bestand naar de juiste tag te springen?
3. **Conceptherkenning en –extractie: de praktijk**  
   Het herkennen van de monumentobjecten die door de DBNL-teksten worden besproken gebeurt door properties, waarven eerst de titel van de sectie (dcterms:title), te vergelijken met de eigenschappen van een verzameling objecten die met zekerheid als monument kunnen gelden. Hoe die referentie-verzameling tot stand komt is in paragraaf 2 al besproken, dit zal altijd een query op DBpedia-monumenten zijn.  
     
   Hoe meer monumentproperties vergeleken kunnen worden met properties uit de tekstsectie, des te beter kan de match worden gemaakt of verworpen. Een term in de DBNL-tekst die herhaald wordt in de titel, is vrijwel zeker het onderwerp. Voor het hierboven gekozen voorbeeld hebben we echter maar een magere basis om te vergelijken. Van het monument Brigittenstraat 6 weten we uit Wikipedia/DBpedia, dat het geen eigen naam heeft maar alleen een omschrijving, die aangeeft dat er een bouwlaag is bijgekomen. Voorts weten we dat het een woonhuis is of was, en dat het in de 17e eeuw gebouwd is (DBpedia verhaspelt het bouwjaar tot ‘0017’). Tenslotte beschikken we over de coördinaten, het rijksmonument-nummer en een of meer foto’s.   
   Dat is niet veel, het monument heeft dan ook geen eigen pagina op Wikipedia maar alleen een plaats in een lijst. Dat is echter nog steeds meer dan de informatie die de boeksectie aan de DBNL-kant ons geeft, want aan die kant is alleen de titel (dcterms:title) geïdentificeerd en zouden we kunnen weten dat het om een ‘dwars huis’ gaat. Om de informatie uit de *Lijst van rijksmonumenten…* beter geschikt te maken, is het een mogelijkheid om het adres als de naam van het monument te gaan gebruiken. Dat doet de DBNL-tekst per slot van rekening ook. In het gegeven voorbeeld is er echter al een omschrijving die als naam dient, en er is geen machine die kan weten dat de naam hier een omschrijving is. Bijgevolg zou de naam ‘Brigittenstraat 6’ vergeleken worden met de veronderstelde naam ‘Huis met oorspronkelijk twee, maar nu drie bouwlagen’, en dat levert geen match op.  
     
   Teneinde de kansen op een match te vergroten bij dit soort monumenten (monumenten in een lijst met geen duidelijke eigen naam) zijn verschillende maatregelen mogelijk:  
     
   1. Verbeteren van de mapping van Wikipedia naar DBpedia: overal waar een monument dat alleen in  
    een lijst voorkomt (een ‘list item’) en geen eigen naam heeft het adres opvoeren als naam  
   2. Een extra matchingsronde uit te voeren, waarin voor secties waar mogelijk ook adressen worden  
    herkend. Dit zou ‘matchen op adres’ aan de mogelijkheden toevoegen.  
     
   Beide maatregelen kunnen echter niet binnen het tijdvenster van dit project worden doorgevoerd. We moeten dus naar matchingsmogelijkheden gaan kijken voor de rijksmonumenten waarvoor wel een complete set van gegevens aanwezig is. Dit zijn doorgaans de monumenten met een eigen pagina, hoe kort verder ook. Nemen we als voorbeeld boerderij De Keut uit het boek van Stenvert e.a. *over Monumenten in Nederland. Gelderland*. Het boek van Stenvert kent een groot aantal hoofdstukken, waarvan hier de eerste:  
     
   

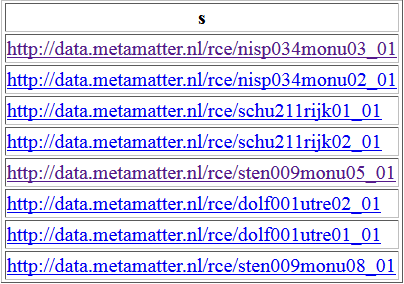
Figuur 5: Hoofdstukken uit Stenvert, Gelderland

Van die hoofdstukken is het hoofdstuk over de plaats Putten als volgt ingedeeld:  
  


Figuur 6: Properties van het hoofdstuk 'Putten'

Hier vinden we meer properties dan alleen dcterms:title, er is ook een drietal DBpedia-resources onderkend die kennelijk binnen de het hoofdstuk aan de orde komen (eigenlijk had hier de inverse van dcterms:isReferencedBy moeten worden gebruikt, dcterms:references). Voor het hoofdstuk ‘Putten’ is een match gevonden met drie monumenten: boerderij De Keut, de familiebegraafplaats Uniken en Kasteel Oldenaller. Van deze drie is de tweede match onjuist, onduidelijk is nog op grond van welke overeenkomsten deze begraafplaats uit Wildervank aan een hoofdstuk over Putten gerelateerd is kunnen worden. Maar nu kan in ieder geval voor twee van deze drie Wikipedia/DBpedia-monumenten een link worden gegenereerd naar dit specifieke stuk tekst.  
  
Named entities voor monumenten worden uit de DBNL-tekst geëxtraheerd door te kijken of een of meer properties van de tekstsectie overeenkomen met zoveel mogelijk dezelfde properties van de monumentresource in DBpedia. Behalve de naam-property is ook de adres-property meteen kandidaat, en een locatie-property als die er is. Bijvoorbeeld de locatie Wageningen in een sectie waarin de naam van ‘Hotel De Wereld’ voorkomt. Wat precies de mogelijkheden zijn, hangt af van de technologie die en dus ook van het computerhulpmiddel dat wordt gebruikt. De meeste programma’s voor concept-herkenning kijken naar de mate van verschil tussen groepen van lettertekens (strings) aan elk van beide zijden: hoeveel lettertekens mogen anders zijn om nog te kunnen zeggen dat het alleen om een variant in schrijfwijze gaat? En natuurlijk ook: hoe maken we onderscheid tussen alle monumenten die ‘Hervormde Kerk’ heten?

Is eenmaal de sectie-resource gelinkt aan een monument, dan moet nog voldoende informatie vastgehouden worden om naar de tekst van deze sectie te kunnen springen (vraag 2 op blz. 5). Dit is betrekkelijk eenvoudig, omdat secties en hoofdstukken immers onderscheiden worden op basis van tags in het XML-bestand: <div type=chapter|section>. Als we bij iedere sectie ook vasthouden welke tag(s) in welk bestand erbij horen, dan wordt het springen eenvoudig. Omdat XML hiërarchisch is georganiseerd, zal dat vaak een reeks van tags zijn. Een probleem dat nog moet worden opgelost, is het geval waarin één tekstfragment verwijst naar meer monumenten, en een monument voorkomt in meer dan één tekstfragment. Hier zal met een soort van prioritering op basis van frequentie van de verwijzing moeten worden gewerkt (*ranking*). De referentie-resources uit het kennismodel kunnen hier nuttige diensten bewijzen.  
  
In het oorspronkelijke plan van aanpak was het de bedoeling om de conceptherkenning met DBpedia Spotlight te doen. De gedachte was, dat bij uitstek Spotlight in staat moet zijn om tekst te matchen op het grote bestand aan monumentresources in DBpedia. Dit blijkt echter weerbarstig in de praktijk. Wanneer bijvoorbeeld de DBNL-tekst voor het Utrechtse Huis Kranestein in Spotlight wordt ingevoerd, herkent Spotlight de begrippen ‘Oude Gracht’ (<http://nl.dbpedia.org/page/Oudegracht_(Utrecht)>) en ‘zadeldak’ (<http://nl.dbpedia.org/resource/Zadeldak>), maar niet het Huis Kranestein (Cranesteyn, Then Craen) als het monument dat het is. Dat ligt niet alleen aan de schrijfwijze, zoals blijkt wanneer de DBNL-tekst voor Huis ter Hallen in Schinveld wordt ingevoerd: herkend worden de termen ‘vakwerk’, ‘segment-boogvenstes’ en ‘Wilhelminaplein’, maar niet de naam van het monument. Dit terwijl het monument in Wikipedia gewoon <http://nl.wikipedia.org/wiki/Huis_ter_Hallen> heet.  
  
Ook na herhaalde proefnemingen blijkt Spotlight onvoldoende te presteren om ons DBNL- project tot een goed einde te brengen. Dat ligt waarschijnlijk minder aan het concept achter het programma als aan het stadium van ontwikkeling waarin het computerprogramma verkeert. Voor het tweede achtereenvolgende jaar heeft Spotlight een vliegende start gekregen doordat het als ontwikkelproject is geselecteerd voor de Google Summer of Code. Maar ook dan nog blijven er zoveel aspecten te ontwikkelen over (projectleider Pablo Mendes wil wel vraaggericht ontwikkelen, maar dan moet de opdrachtgever ontwikkelcapaciteit bij hem huren), dat Spotlight althans op dit moment afvalt als kandidaat-hulpmiddel voor conceptherkenning in ons project.  
  
Bij de keuze voor vervangende technologie hebben we ons in hoge mate laten leiden door ontwik-kelingen op het gebied van Natural Language Processing (NLP) en text engineering, meer speciaal de GATE-architectuur. Voor een korte uiteenzetting over GATE zie Bijlage 4.  
  
Tekstherkenning met het Java-programma Groovy bleek een effectief, zij het specialistische kennis vereisend middel om:   
  
1. De DBNL-teksten op te delen in onderdelen waaraan eenduidig onderwerpen gekoppeld kunnen  
 worden;  
2. Monumenten als monumenten te herkennen in een (in XML-formaat aangeleverde) DBNL-tekst;  
3. De gevonden matches als export in RDF-formaat aan te leveren, zodat ze kunnen worden  
 toegevoegd aan het kennismodel van dit project..  
  
Extractie van monumentinformatie op deze manier leidt een RDF-graaf die conform het kennismodel tekstfragmenten laat refereren aan monumentresources die kunnen worden gelinkt aan zowel DBpedia als aan de dataset van RCE. Hoe deze graaf eruit ziet, is te zien op [http://data.metamatter.nl/rce/](https://webmail.bibliotheek.nl/owa/redir.aspx?C=RNI1TdNCCEyuPte1fDgLz3kDDqhccdAIlAqgIw7AeSqEvSrytIF7rU-R3btCxl4q3YoG8ksPbPE.&URL=http%3a%2f%2fdata.metamatter.nl%2fsparql%3fdefault-graph-uri%3dhttp%253A%252F%252Fdata.metamatter.nl%252Frce%252F%26query%3dselect%2bdistinct%2b%253Fs%2b%257B%253Fs%2b%253Chttp%253A%252F%252Fpurl.org%252Fdc%252Fterms%252Fcreator%253E%2b%253Fo%257D%26format%3dtext%252Fhtml%26timeout%3d0%26debug%3don). Al naar gelang de browse-ingang tot wat we maar de DBNL-ontologie zullen noemen, zien we hier ofwel de acht monumentenboeken (of delen daarvan), als in onderstaande figuur, ofwel een lijst met de auteurs.



Figuur 7: De monumentenboeken als ingang tot de DBNL-ontologie

Als we op een van deze monumentenboeken doorklikken zien we dat de auteur als een dbpedia-owl:Person is getypeerd, dat de tekst zelf een dbpedia-owl:Book is, en dat het boek bestaat uit een aantal onderdelen, die hoofdstuk (bibo:Chapter) dan wel paragraaf (bibo:BookSection) zijn. Omdat de referentielijst is beperkt tot die monumenten die een betekenisvolle naam hebben (dus geen naamloze monumenten uit lijsten), refereren de secties en hoofdstukken in de graaf nu aan 5130 monumenten, met in totaal 2365 links naar DBNL-monumentenboeken (Bijlage 3, query 4). De kwaliteit van deze links moet verbeterd, met homoniem-monumenten als de Antoniuskapel wordt niet goed genoeg omgegaan.

1. **Verrijking door inferentie**  
     
   In het kennismodel is de weergave van de monumenten bewust beperkt gehouden. De relevante informatie over de monumenten zelf moet, zoals in paragraaf 3 gezegd, uit de erfgoedthesaurus komen, en die mapping is nog niet gemaakt. De in de offerte voorziene vastlegging in SKOS van de relaties tussen het kennismodel en de erfgoedthesaurus is derhalve evenmin aan de orde geweest.  
   Alles wat verder nog gaat over de relatie tussen tekst en monumentresource zit in de verwijzing naar de monumenten toe, dus in de Verwijzingsobjecten etc. Daarnaast kan veel informatie door inferentie worden gegenereerd. Voor elk boek dat vóórkomt in een Verwijzing (Reference-object) naar een monument, kan bijvoorbeeld een dcterms:subject relatie tussen monument en boek worden geconstrueerd.

1. **Conclusies**Als in het leven, de bekendste monumenten krijgen de beste links naar de DBNL-teksten. Het vergt echter geen massieve inspanningen om ook de ‘tweede garnituur’ van monumenten te relateren aan de DBNL-teksten die ze beschrijven. Want wij denken dat de voorgestelde opzet in grote lijnen is geslaagd, en dat het vooral een zaak van kwaliteitsverbetering is om het aantal correcte tekstreferenties significant op te voeren. Kwaliteitsverbetering aan de kant van Wikipedia/DBpedia: betere data door betere infoboxen, door meer informatie in de lijsten te zetten en door in DBpedia adressen als naam op te geven waar het monument geen eigen naam heeft. Kwaliteitsverbetering ook aan de kant van de informatie die uit hoofdstukken en secties wordt gehaald: meer betekenisvolle sectie-properties, onder meer door nieuwe informatie af te leiden uit oude. Acties van beiderlei soort zullen eraan bijdragen het matchingspercentage te verhogen.  
     
   Wanneer eenmaal het percentage geïdentificeerde monumenten en de betrouwbaarheid van de matching groot genoeg is, komen de mogelijkheden tot vergelijken op een ander niveau. Eenmaal ‘herkende’ monumenten kunnen dan immers worden gelinkt aan andere datasets, op basis van een monument ID of een ander onderscheidend kenmerk, en dan kan al naar gelang de behoefte populaire of wetenschappelijke data worden opgehaald. Data die op zijn beurt weer kan worden gebruikt om bijv. bouwjaargegevens in DBpedia te verbeteren, en om structureel beter om te gaan met onzekere dateringen. Voor het beter omgaan met jaartallen-bij-benadering is in het kennismodel een klasse ApproximateTimePeriod gedefinieerd, die nog uitermate nuttig kan blijken te zijn.  
     
   Niet gelukt is de conceptherkenning met behulp van DBpedia Spotlight. Dit computerhulpmiddel lijkt nog steeds veelbelovend, omdat het specifiek op DBpedia is toegesneden en omdat het belooft conceptherkenning binnen het bereik te brengen van mensen die geen NLP-specialist zijn. Voor onze doeleinden hier en nu bleek Spotlight echter niet voldoende uitontwikkeld te zijn. We hebben Spotlight goed kunnen vervangen door programmatuur uit de hoek van de NLP, maar het zou onjuist zijn te verwachten dat iedereen even goed hiermee kan omgaan als met Spotlight of soortgelijke tools. Voor conceptherkenning zullen we voorlopig nog wel een beroep moeten doen op specialistische kennis uit de NLP-hoek. Is echter specifiek voor monumenten deze NLP-programmatuur ontwikkeld, getest en getuned, dan is dat natuurlijk wel een uitnodiging aan de minder-specialisten om hiervan op grotere schaal gebruik te gaan maken.

## Bijlage 1: Afbakening van het project

In overleg met de RCE is het project beperkt tot de volgende DBNL-teksten:

Van de reeks *Monumenten in Nederland*:  
\* R. Stenvert e.a., Monumenten in Nederland : Gelderland

\* R. Stenvert e.a., Monumenten in Nederland : Limburg  
  
Van A.J. Schulte:

\* Het Rijk van Nijmegen : Westelijk gedeelte

\* Het Rijk van Nijmegen : Oostelijk gedeelte en de Duffelt  
  
Van O.A.E. van Nispen tot Sevenaer:

\* De monumenten in de gemeente Maastricht, deel 1 en deel 2

Van Marceline Dolfin e.a.:  
\* Utrecht : de huizen binnen de singels.

Voorts is de tekst geanalyseerd van twee Jaarboeken:  
\* Jaarboek Monumentenzorg 2003 : Boerenbedrijvigheid, voortgang en behoud

\* Jaarboek Monumentenzorg 1994 : Monumenten van een nieuwe tijd. Architectuur en stedebouw  
 1850-1940

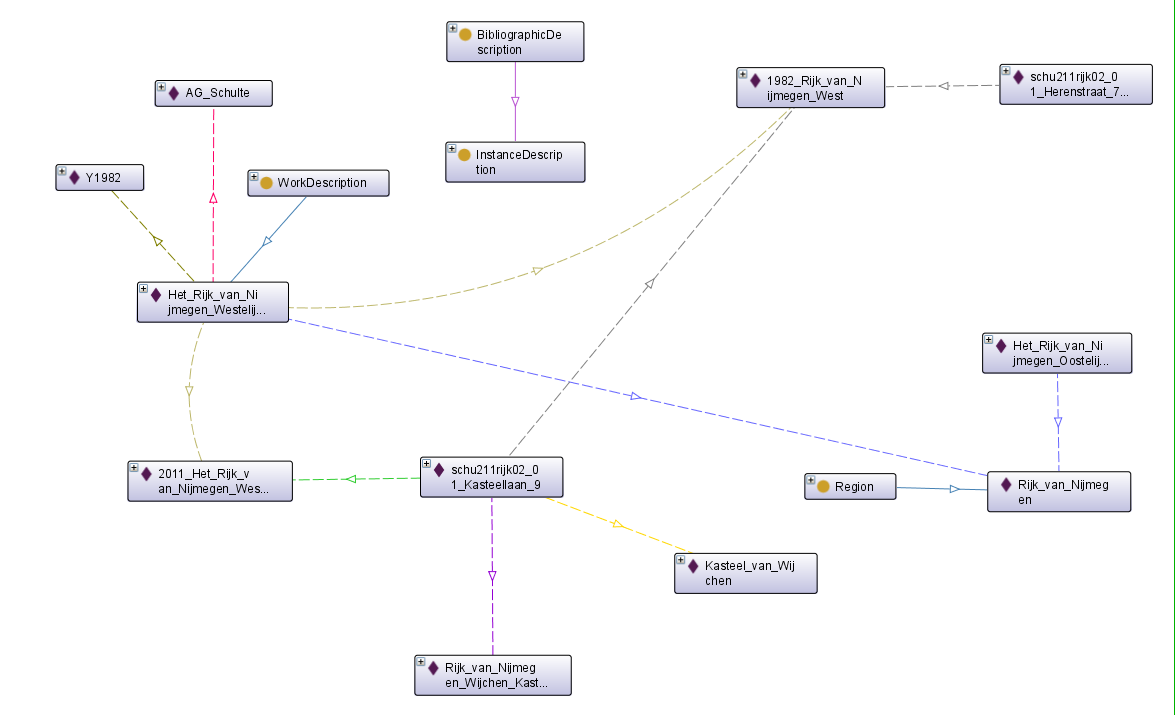
De jaarboeken komen niet voor als link op een Wikipedia-pagina, maar hebben in het project gediend als voorbeeld van een tekst die geautomatiseerd op monumentinformatie kan worden onderzocht.

De 69 externe links die op Wikipedia-pagina’s naar bovengenoemde teksten verwijzen, kunnen als volgt worden onderverdeeld:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **# links** | **Type externe link** | **Voorbeeld** | |
| Op Wikipagina over enkelvoudig object (één monument) | | | | |
|  | 26 | Doelbepaling in tekst met link-extensie of query-parameter | | Huis De Spyker |
|  | 12 | Doeltekst opgedeeld in hoofdstukken, link naar hoofdstuk | | Stadhuis Maastricht |
| Op Wikipagina over onderwerp anders dan monument | | | | |
|  | 9 | Wikipagina met ‘Lijst van rijksmonumenten in …’ | | Apeldoorn\_(stad) |
|  | 1 | Wikipagina over type gebouwen | | Stadskastelen Utrecht |
|  | 14 | Wikipagina over streek | | Rijk van Nijmegen, Zuid-Limburg, Plateau van Margraten |
|  | 1 | Wikipagina over stad | | Maastricht |
|  | 1 | Wikipagina over stadsdeel | | Utrecht binnenstad |
|  | 4 | Wikipagina over straat | | Achter St. Pieter Utr. |

Over de rubriek ‘enkelvoudige objecten’ kan daarbij nog aangetekend worden, dat in de meeste van die 26 gevallen nog steeds alleen de globale context van de DBNL-tekst geboden wordt. Bij het Huis Drakenburg in Utrecht bijvoorbeeld (<http://nl.wikipedia.org/wiki/Drakenburg_(huis))> wordt doorgelinkt naar een algemene passage over stadskastelen in de stad Utrecht, niet naar de eigenlijke tekst over Drakenburg op blz. 395 van de DBNL-tekst van Dolfin e.a.   
De rubriek ‘onderwerp anders dan monument’ bevat de externe links naar een DBNL-boek op plaatsen die zelf niet over een monument gaan, maar die een lijst bevatten of over een streek gaan. In twee gevallen, <http://nl.wikipedia.org/wiki/Lijst_van_rijksmonumenten_in_Limburg> en <http://nl.wikipedia.org/wiki/Lijst_van_rijksmonumenten_in_Gelderland> gaat het zelfs niet om een lijst van monumenten, maar om een lijst van lijsten – hetgeen een extra trap toevoegt aan het geheel.  
  
Vastgesteld moet worden, dat er Wikipedia-pagina’s zijn die nu nog geen externe link naar een DBNL-tekst hebben waar dat goed mogelijk zou zijn geweest: <http://nl.wikipedia.org/wiki/Kasteel_van_Wijchen>. Deze zijn met een SPARQL-query eenvoudig te achterhalen, waarna reparatie kan volgen.

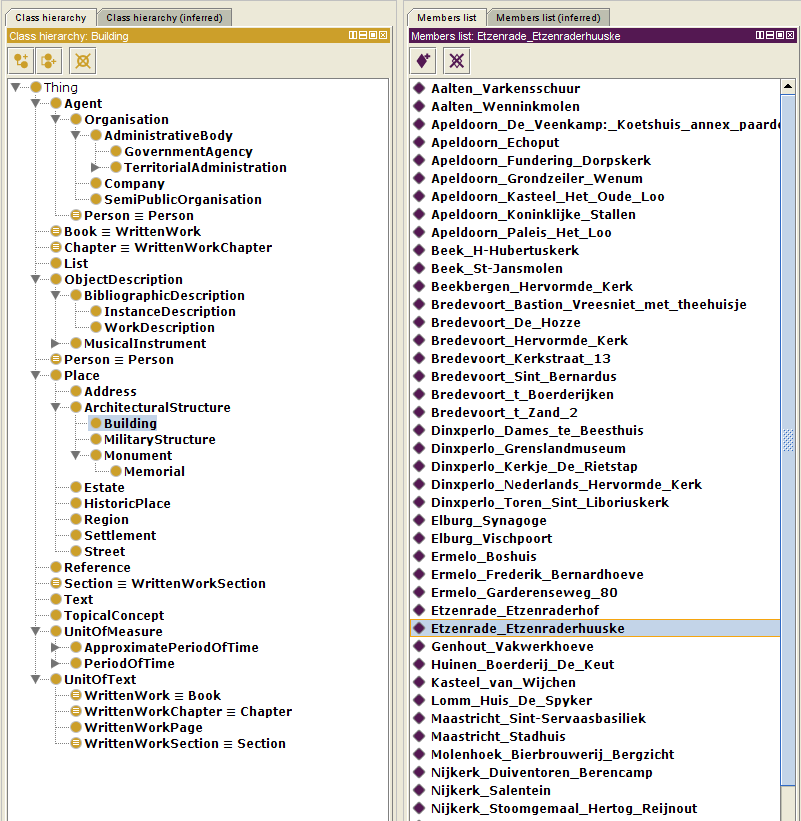
## Bijlage 2: Een uitsnede uit het kennismodel



Centraal in het model staat de Verwijzing (de klasse Reference, rode cirkel). Het verwijzingsobject verwijst naar een Sectie, dat is een object dat een tekstfragment kan bevatten (groene cirkel) maar altijd een verwijzing heeft naar de fysieke dan wel digitale pagina. Het verwijzingsobject verwijst tegelijkertijd naar het onderwerp, dat is in dit geval het object dat het Kasteel van Wijchen representeert (gele cirkel). De Sectie ‘weet’ uiteraard van welke uitgave van welk werk (blauwe cirkel) het een tekstfragment is, en ‘weet’ welke fysieke pagina erbij hoort. Gaat het om de verwijzing naar een paragraaf in een digitale uitgave, dan ‘weet’ de Sectie welke XML-tag erbij hoort.  
  
In bovenstaand diagram zijn omwille van de overzichtelijkheid niet alle relaties weergegeven. De auteur A.G. Schulte heeft ook het werk *Het Rijk van Nijmegen. Oostelijk gedeelte en de Duffelt* geschreven, terwijl de streek Het Rijk van Nijmegen eveneens onderwerp is in bredere zin (dcterms:spatial) van het werk over Gelderland in de reeks Monumenten in Nederland (R. Stenvert e.a.).

De klassenhierarchie van het kennismodel ziet men in onderstaande afbeelding. Aandacht verdienen:

* De scheiding tussen BibliographicDescription en UnitOfText
* ArchitecturalStructure als een equivalente klasse van de gelijknamige klasse in DBpedia – hetgeen de *reconciliation* gemakkelijker maakt
* De ondiepe structuur voor monumenten. Wel wordt onderscheid gemaakt tussen gebouwen, gedenktekens, militaire werken en landgoederen, maar van een verdere typologie is afgezien. Die moet waar nodig in het kennismodel worden binnengehaald via de link naar de erfgoedthesaurus van RCE
* Metadata hoeven niet beperkt te blijven tot boekbeschrijvingen/catalogusrecords. Via de superklasse Object Description kunnen ook andere soorten van collectie-items worden gekoppeld aan tekstfragmenten.
* Auteurs in het kennismodel hebben een property met een VIAF ID, hetgeen het linken en ophalen van data eenvoudig maakt.



## Bijlage 3: SPARQL-queries op DBpedia

**Query 1**: Geef alle externe links verwijzend naar de DBNL-teksten die binnen de afgesproken scope vallen

SELECT ?exLink ?monumentPag  
WHERE  
 {  
 ?monumentPag dbpedia-owl:wikiPageExternalLink ?exLink .  
 FILTER ( regex(str(?exLink), "^http://www.dbnl.org/tekst/", "i") AND (  
 regex(str(?exLink), "^http://www.dbnl.org/tekst/sten009monu05\_01/", "i") OR # Gelderland  
 regex(str(?exLink), "^http://www.dbnl.org/tekst/sten009monu08\_01/", "i") OR # Limburg  
 regex(str(?exLink), "^http://www.dbnl.org/tekst/schu211rijk02\_01/", "i") OR # Rijk van Nijmegen W  
 regex(str(?exLink), "^http://www.dbnl.org/tekst/schu211rijk01\_01/", "i") OR # Rijk van Nijmegen O  
 regex(str(?exLink), "^http://www.dbnl.org/tekst/nisp034monu02\_01/", "i") OR # Maastricht. Deel 1  
 regex(str(?exLink), "^http://www.dbnl.org/tekst/dolf001utre01\_01/", "i") # Utrecht binnen singels  
 )  
 )  
 }  
  
ORDER BY ?exLink

**Query 2:** Aantal rijksmonumenten opgevoerd in alle ‘Lijst(en) van rijksmonumenten in …’

SELECT count(?lijstMonument)  
WHERE  
 {  
 ?lijstMonument dbpedia-owl:listItemOf ?monPag .  
 FILTER ( regex(str(?monPag), "^http://nl.dbpedia.org/resource/Lijst\_van\_rijksmon", "i" ) )   
 }

**Query 3**: Aantal rijksmonumenten voor een steekproef uit de monumenten binnen de scope

SELECT count(?lijstMonument )  
WHERE  
 {  
 ?lijstMonument dbpedia-owl:listItemOf ?monPag .  
 FILTER ( regex(str(?monPag), "^http://nl.dbpedia.org/resource/Lijst\_van\_rijksmon", "i" ) )   
 FILTER ( regex(str(?monPag), "Maastricht", "i" ) OR   
 regex(str(?monPag), "Gelderland", "i" ) OR  
 regex(str(?monPag), "Limburg", "i" ) OR  
 regex(str(?monPag), "Bredevoort", "i" ) OR  
 regex(str(?monPag), "Apeldoorn", "i" ) OR  
 regex(str(?monPag), "Putten", "i" ) OR  
 regex(str(?monPag), "Wageningen", "i" ) OR  
 regex(str(?monPag), "Arnhem", "i" ) OR  
 regex(str(?monPag), "Nijmegen", "i" ) OR  
 regex(str(?monPag), "Culemborg", "i" ) )  
 }

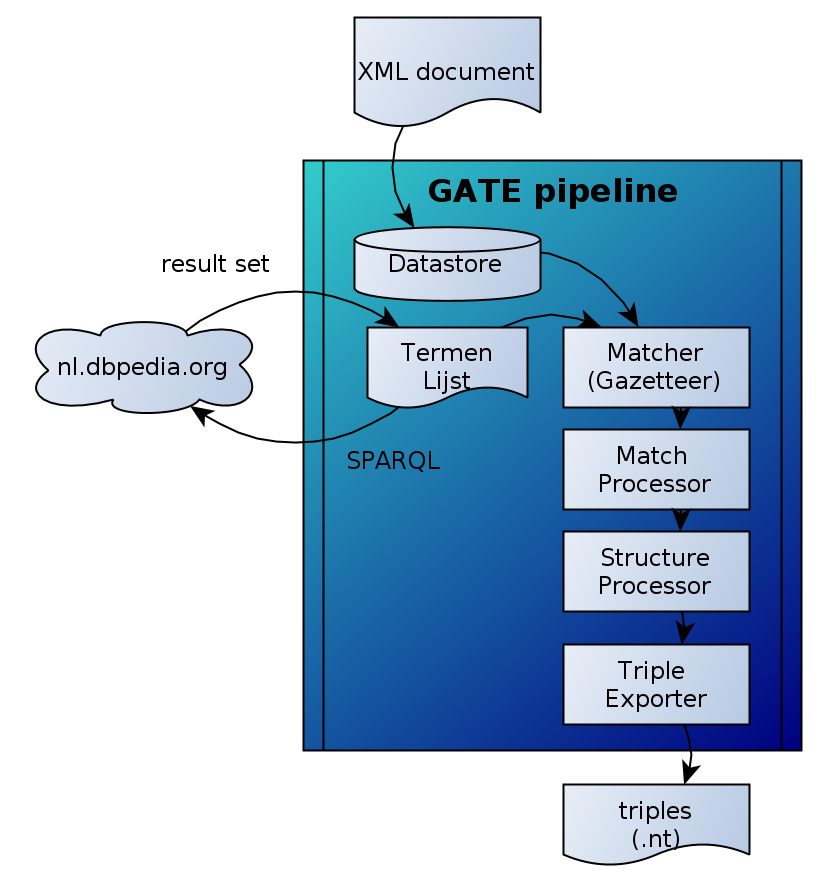
**Query 4**: Rijksmonumenten die in DBpedia als zodanig zijn aangemerkt

SELECT ?label ?pag  
WHERE {  
 {  
 ?pag dbpedia-owl:protectionStatus ?status .  
 ?pag rdfs:label ?label .  
 FILTER (regex(str(?status), "Rijksmonument", "i") )  
 }  
UNION {  
 ?pag dbpedia-owl:status ?status .  
 ?pag rdfs:label ?label .  
 FILTER (regex(str(?status), "Rijksmonument", "i") )  
 }  
UNION {  
 ?pag dbpedia-owl:status ?status .  
 ?pag foaf:name ?label .  
 FILTER (regex(str(?status), "Rijksmonument", "i") )  
 }  
UNION {  
 ?pag dbpedia-owl:protectionStatus ?status .  
 ?pag foaf:name ?label .  
 FILTER (regex(str(?status), "Rijksmonument", "i") )  
 }   
 }

## Bijlage 4: Conceptherkenning, NLP en GATE

De technische realisatie van de software die zoekt naar links tussen Dbpedia en DBNL monumentenboeken is gebaseerd op het GATE framework[[1]](#footnote-1), een raamwerk voor het analyseren van tekst of Natuurlijke Taalanalyse.

Het geheel van GATE applicaties is schematisch weergegeven in onderstaande figuur:



Deze GATE 'pipeline' is een samenstel aan software componenten in een bepaalde volgorde bewerkingen op een document uitvoeren. Resultaten van bewerkingen worden aan het document toegevoegd in de vorm van annotaties. Een document wordt dus door de 'pipeline' gestuurd en krijgt bij de verschillende onderdelen in de 'pipeline' aanvullende annotaties.

XML Documenten worden ingelezen (geïmporteerd) en toegevoegd aan een corpus. Bij het importeren wordt de oorspronkelijke structuur van het document 'vertaald' naar annotaties die beschikbaar zijn voor analyse.

De documenten en de beschrijving van het corpus worden in een datastore opgeslagen ter ondersteuning van de organisatie van documenten.

Aan live.nl.dbpedia.org/sparql wordt een SPARQL query gevraagd waarvan het resultaat wordt verwerkt als lijst t.b.v. een 'gazetteer'. De 'gazetteer' zoekt naar matchende termen in teksten op basis van de lijst met resultaten van de query. Als een match wordt gevonden wordt een annotatie aan die term in de tekst toegevoegd inclusief een eigenschap waarin de URI van DBpedia wordt aangegeven. Het resultaat van de 'gazetteer' is een verzameling met annotaties in de teksten van DBNL naar resources van DBpedia.

In de oorspronkelijke informatie uit de XML-documenten staat de structuur van een document aangegeven. Die informatie is het aanknopingspunt om metadata over de structuur van het document te annoteren en te exporteren als RDF (triples). Het resultaat is een document dat in onderdelen beschreven wordt, ieder onderdeel als aparte resource. De geannoteerde matches met DBpedia kunnen met deze informatie gerelateerd worden aan het (detail) onderdeel van het document waarin de match voorkomt. De Matchprocessor relateert DBpedia URI's aan gevonden onderdelen binnen het document. De Structure processor tast de structuur van het document af en beschrijft deze in annotaties. De Triples Exporter construeert RDF van de gevonden annotaties en schrijft deze weg in een file.

1. http://gate.ac.uk/ [↑](#footnote-ref-1)