GFMed: Question Answering over BioMedical Linked Data with Grammatical Framework

Anca Marginean

Technical University of Cluj-Napoca, Romania

September 17, 2014
Outline

1. Context
   - "Journey" to GF
   - Grammatical Framework
   - Description Logic

2. Building a grammar for DL inspired blocks
   - Grammar modules
   - SPARQL resource module
   - Building Trees for Property Restrictions
   - Transformation functions. Functions for Queries
   - Pre- and Post-processing

3. Lexicon extracted from collections

4. Results and Their Analysis. Conclusions
Translation of Natural Language to meaning representation language (Machine learning research group, University of Texas)

- CHILL - Prolog based language (1996), ILP

a first personal attempt in querying DrugBank with Stanford Parser and Patterns - 2013

SQUALL - Montague grammar: each rule for syntactic structure is decorated by a \( \lambda \)-term, (2012), Sebastien Ferré

Grammatical Framework (GF)
GF - Grammatical Framework


- multilingual translation systems - Multilingual Online Translation MOLTO [http://www.molto-project.eu/](http://www.molto-project.eu/)
- language-based human-computer interaction
- creation of computational linguistic resources - 36 languages

GF grammar = abstract module + concrete modules

Abstract module: trees that capture semantically relevant structure
Concrete module: relates trees with linear strings
GF - basic example

**cat** Person, Action, Statement

**fun** John, Mary : Person;

Read : Action;

Do : Person $\rightarrow$ Action $\rightarrow$ Statement;

<table>
<thead>
<tr>
<th>abstract</th>
<th>John Learn</th>
</tr>
</thead>
<tbody>
<tr>
<td>french</td>
<td>Jean apprend.</td>
</tr>
<tr>
<td>english</td>
<td>John learns.</td>
</tr>
</tbody>
</table>
Building a grammar for DL inspired blocks
Lexicon extracted from collections
Results and Their Analysis. Conclusions

"Journey" to GF
Grammatical Framework
Description Logic

GF - basic example

cat  Person, Action, Statement

fun  John, Mary : Person;
      Read : Action;
      Do : Person → Action → Statement;

abstract  John Learn
french  Jean append.
english  John learns.

lincat Person, Action, Statement = Str;

lin  John = "Jean";
     Learn = "apprend";
     Do  p  a = p ++ a ++".";

lincat Person, Action, Statement = Str;

lin  John = "John";
     Learn = "learns";
     Do  p  a = p ++ a ++".";
Language libraries used in concrete grammars

\( \text{lincat} \) \( Person = NP; \quad Action = VP; \quad Statement = Cl; \)
\( \text{lin} \)
\( Learn = mkVP(P.mkV \quad "apprendre"); \)
\( John = mkNP(P.mkPN \quad "Jean"); \)
\( Do \quad p \quad a = mkCL \quad p \quad a; \), \( where \ P=\text{ParadigmsFrench} \)

Abstract syntax in multilingual grammar

Abstract syntax need not care about features like morphology, agreement, word order, and discontinuity.
### Description Logic

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Syntax</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>negation</td>
<td>( \neg C )</td>
<td></td>
</tr>
<tr>
<td>conjunction</td>
<td>( C \sqcap D )</td>
<td></td>
</tr>
<tr>
<td>disjunction</td>
<td>( C \sqcup D )</td>
<td></td>
</tr>
<tr>
<td>existential restriction</td>
<td>( \exists R \cdot C )</td>
<td>( \exists \text{PossibleDrug}.\text{ApprovedDrugs} )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( \exists \text{PossibleDrug}^{-1}.\text{DiseasesWithDegree1} )</td>
</tr>
<tr>
<td>universal restriction</td>
<td>( \forall R \cdot C )</td>
<td>( \forall \text{PossibleDrug}.\text{Bextra} )</td>
</tr>
<tr>
<td>individual assertion</td>
<td>( a : C )</td>
<td>( \text{Rickets}:\text{Disease} )</td>
</tr>
<tr>
<td>role assertion</td>
<td>( (a, b) : R )</td>
<td>( (\text{Rickets}, \text{Calcitriol}) : \text{PossibleDrug} )</td>
</tr>
</tbody>
</table>

Value restrictions on nominals: \( \exists \text{PossibleDrug} \cdot \{ \text{Bextra} \} \)

Example: \( \exists \text{PossibleDrug} \cdot \exists \text{SideEffect} \cdot \{ \text{Fever} \} \)

diseases that are treated with drugs that cause Fever.
Examples of questions

Which drugs interact with

• Valdecoxib? - a drug name
Examples of questions

Which drugs interact with

- **Valdecoxib**? - a drug name
- **ApprovedDrugs**? - a named drug class
Examples of questions

Which drugs interact with

- Valdecoxib? - a drug name
- ApprovedDrugs? - a named drug class
- drugs used for Rickets? - a drug class from a restriction
- drugs that have fever as side effect?
Examples of questions

Which drugs interact with

- **Valdecoxib**? - a drug name
- **ApprovedDrugs**? - a named drug class
- **drugs used for Rickets**? - a drug class from a restriction
- **drugs that have fever as side effect**?
- **drugs that treat diseases that are associated with EDNRB**? - a drug class from a restriction with values in a disease class
Grammar modules

- a resource library for SPARQL
- a controlled abstract language for BioMedical Data + concrete English and SPARQL languages

Figure: Main GFMed grammars and used resources
SPARQL resource module

- Triplet: Type = \{subj, obj: Str; prop: PropertyT\};
- Statement : Type = \{s: Str; extra: Str; aggreg: AggregationType\};

\[
\text{addSubj : Str} \rightarrow \text{Triplet} \rightarrow \text{Triplet} = \\
\text{ss, vp} \rightarrow \{\text{subj} = \text{ss}; \text{prop} = \text{vp}\.\text{prop}; \text{obj} = \text{vp}\.\text{obj}\};
\]

\[
\text{mkStatement : Triplet} \rightarrow \text{Statement} = \\
\text{vp} \rightarrow \{\text{s} = \text{vp}\.\text{subj} + + \text{vp}\.\text{prop}\.\text{s} + + \text{vp}\.\text{obj}; \text{extra} = ""; \text{aggreg} = \text{no}\};
\]

\[
\text{mkFilterStatement : PropertyT} \rightarrow \text{Str} \rightarrow \text{Str} \rightarrow \text{Statement} = \\
\text{p, x, v2} \rightarrow \text{case } \text{p}.\text{vt} \text{ of } \{ \\
\text{String} \Rightarrow \{\text{s} = " FILTER(regex(" + + v2 + +"," + + x + +",'i'))"}; \text{extra} = ""; \text{aggreg} = \\
\text{Number} \Rightarrow \{\text{s} = " FILTER(" + + v2 + +" = " + + x + +")"}; \text{extra} = ""; \text{aggreg} = \\
\}
\]
Grammar modules

SPARQL resource module
Building Trees for Property Restrictions
Transformation functions. Functions for Queries
Pre- and Post-processing

SPARQL Resource

\( mkQuery : Str \rightarrow Statement \rightarrow Str = \)
\[ \var, b \rightarrow \text{case } b.aggreg \text{ of } \]
\[ \text{yes } \Rightarrow \text{"select" } + + \var + +\text{"where\{" } + + b.s + +\text{"\}}" + + b.extra; \]
\[ \text{no } \Rightarrow \text{"select } \text{distinct" } + + \var + +\text{"where\{" } + + b.s + +\text{"\}}" \]
\]
### Main categories

<table>
<thead>
<tr>
<th>Category</th>
<th>English Category</th>
<th>Examples and Short Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X$</td>
<td>NP</td>
<td>$X \in {\text{Drug, TargetConcept, Gene, Disease, SideEffect, SiderDrug}}$</td>
</tr>
<tr>
<td>DrugBank-Property</td>
<td>CN</td>
<td>MeltingPoint, GeneralFunction, DosageForm, PredictedWaterSolubility, Manufacturers, Indication, Targetacting, FoodInteraction,…</td>
</tr>
<tr>
<td>Sider-Property</td>
<td>CN</td>
<td>SideEffect</td>
</tr>
<tr>
<td>Diseaseome-Property</td>
<td>CN</td>
<td>AssociatedGene, PossibleDrug, ClassDegree, Degree, Class, Size, SubtypeOf, ChromosomalLocation</td>
</tr>
<tr>
<td>Property</td>
<td>CN</td>
<td>any kind of property from the above three</td>
</tr>
<tr>
<td>$X$ Class</td>
<td>NP</td>
<td>classes formed of a single named $X$ entity: <em>Lepirudin</em>, or from drugs described by a criterion: <em>drugs that Prothrombin</em>, <em>DrugClass, TargetClass, SideEffectClass, SiderDrugClass, DiseaseClass, GeneClass, PropertyClass</em></td>
</tr>
<tr>
<td>Criterion-ForXClassY</td>
<td>NP</td>
<td>criteria for getting a class of $X$, expressed by an $Y$ syntactic structure</td>
</tr>
<tr>
<td>Question</td>
<td>QS</td>
<td><em>Lepirudin as possible drug</em></td>
</tr>
<tr>
<td></td>
<td></td>
<td><em>with Lepirudin as possible drug</em></td>
</tr>
<tr>
<td></td>
<td></td>
<td><em>treated with Lepirudin, indicated for Fever</em></td>
</tr>
<tr>
<td></td>
<td></td>
<td><em>treat Tuberculosis</em></td>
</tr>
<tr>
<td></td>
<td></td>
<td><em>whose possible drug is Lepirudin, whose possible drugs interact with Lepirudin</em></td>
</tr>
<tr>
<td></td>
<td></td>
<td><em>Lepirudin is used for</em></td>
</tr>
<tr>
<td>Utterance</td>
<td>Utt</td>
<td>utterances from affirmative clauses <em>List…</em> or from question clauses <em>What…</em></td>
</tr>
</tbody>
</table>

Anca Marginean

GFMed: Question Answering over BioMedical Linked Data with
Object Property Restrictions

∃possibleDrug.DrugClass - AP, NP, Adv, VP, RCI

WithPossibleDrugCriterion : DrugClass
   → CriterionForDiseaseClass; – treated with Lepirudin;

WithPossibleDrugCriterionClSlash : DrugClass
   → CriterionForDiseaseClassClSlash; – Lepirudin is used for

WithPossibleDrugCriterionNP : DrugClass
   → CriterionForDiseaseClassNP; – Lepirudin as possible drug

WithPossibleDrugCriterionAdv : DrugClass
   → CriterionForDiseaseClassAdv; – with Lepirudin as possible drug

WithPossibleDrugCriterionRCl : DrugClass
   → CriterionForDiseaseClassRCl; – whose possible drug is Lepirudin

WithPossibleDrugCriteriaRCl_VP : CriterionForDrugClassVP
   → CriterionForDiseaseClassRCl; – whose possible drug interacts with

WithPossibleDrugCriteriaRCl_Adj : CriterionForDrugClassAdj
Object Property Restriction

SPARQL linearization

WithPossibleDrugCriterion\{X\} dc =

let disc = addSubj "\?dis" (addObj dc.var PossibleDrug)

in\{ var = "\?dis" ; body = addStatement2

( addStatement2 (mkStatementdisc)

( mkDiseaseStatement "\?dis")

 dc.body\};

PossibleDrug = mkEmptyTriplet( mkDiseasomeProperty "possibleDrug" String)

English linearization

WithPossibleDrugCriterion dc = mkAP Treated_A2 dc;

Treated_A2 = P.mkA2 "treated" by8means_Prep

| P.mkA2 "treated" with_Prep;
Linearization example

linearize WithPossibleDrugsCriterion (SingleDrug DB00001)

- treated by lepirudin
- treated by refludan
- treated by hirudin variant-1
- treated with lepirudin ...

?dis a diseasome:diseases .
Restrictions on datatype properties

Generic trees with a dummy string for string and number values:
(drugs with) a solubility of 3.24e-02 mg/mL $\rightarrow$ XX=3.24e-02 mg/mL

ValueRestriction : DrugBankProperty $\rightarrow$ CriterionForDrugClass
- - solubility of XX

ValueRestrictionRCl : DrugBankProperty $\rightarrow$ CriterionForDrugClassRCl
- - whose route of elimination involves XX

DiseaseValueRestriction : DiseasomeProperty $\rightarrow$ CriterionForDiseaseClassNP
- - chromosomal location of XX

DiseaseValueRestrictionRCl : DiseasomeProperty $\rightarrow$ CriterionForDiseaseClassRCl
- - whose subtype involves XX
Value restriction examples on datatype property

linearize (ValueRestriction Solubility)
- solubility of XX
- ?d drugbank:Solubility ?vp . FILTER(regex( ?vp , xx , 'i'))

linearize (ValueRestriction MolecularWeightAverage)
- molecular weight average of XX
- ?d drugbank:molecularWeightAverage ?vp . FILTER( ?vp = xx )
Value Restriction with aggregation

\[ \text{LowestNumber} : \text{Property} \rightarrow \text{CriterionForDrugClass} \]
- - lowest number of side effects

\[ \text{DiseaseWithLowestValue} : \text{DiseasomeProperty} \rightarrow \text{CriterionForDiseaseClass} \]
- - with lowest size

\[ \text{LowestNumberValue} : \text{Property} \rightarrow \text{PropertyClass}; \]
- - least common chromosome location

\[ \text{HighestValue} : \text{Property} \rightarrow \text{PropertyClass}; \]
- - highest toxicity
Examples

I PQuestion (WhichSiderDrugs2 (ToSiderDrugClass (SiderHighestNumber SideEffect)))

- which are the drugs with the highest number of side effects
- select count(distinct ?vp) as ?c, ?drug
  where { ?drug sider:sideEffect ?vp }
  group by ?drug order by desc(?c) limit 1

I PQuestion (WhatPropertyValue (HighestValue (DbToProperty MeltingPoint)))

- which is the highest melting point
- select distinct max(?v) as ?maxv
  where { ?d drugbank:meltingPoint ?v }
Transformation functions

From Criterion to a Class

- Adjectival Phrase *used from Rickets* $\rightarrow$ Noun Phrase *drugs used for Rickets*
- AdjToDrugClass: CriterionForDrugClassAdj $\rightarrow$ DrugClass
- AdjToDrugClass cc=cc (*SPARQL concrete grammar*)
- AdjToDrugClass cc=mkNP the Det (mkCN Drug N cc) (*English concrete grammar*)

All the English alternatives for expressing a conceptual DL constructor have the same SPARQL linearization
Transformation between Datasets

- English linearization does not alter the object of transformation.
- SPARQL linearization introduces new variables and *sameAs* statements.
- \( \text{DBToSiderDrug} : \text{DrugClass} \rightarrow \text{SiderDrugClass} \);
- \( \text{DBToSiderDrug} d = \{ \text{var} = "\text?siderdrug" ;
\text{body} = \text{addStatement2}
(\text{mkSameAsStatement} "\text?siderdrug" d.\text{var})
\text{d.body}) \}; \)
Example

- PQuestion (WhatSiderProperty SideEffect (DBToSiderDrug (PossibleDrugsForCriteria (SingleDisease Dis1004))))
- which are the side effects of possible drugs for rickets
- select distinct ?v0 where
  {?siderdrug sider:sideEffect ?v0 .
   ?siderdrug owl:sameAs ?possDrug .
   diseasome:diseases/1004 diseasome:possibleDrug ?possDrug }

Anca Marginean
GFMed: Question Answering over BioMedical Linked Data with
Functions for Queries

WhichDisease2 : DiseaseClass → Question;
- - which are the diseases caused by Lepirudin?

WhichDisease : CriterionForDiseaseClass → Question;
- - which diseases are caused by Lepirudin?

WhichTargetAdj : ValueRestrictionAdj → Question;
- - which targets are involved in XX?

WhatPropertyValue : PropertyClass → Question;
- - which is the least common chromosome location?

What, Which, Give, List, Ask
Pre- and Post-processing

- InputQuestion - JavaModule - GF REST services of parsing and linearization - JavaModule - OutputAnswer
- JavaModule: failure handling method: when the translation module gets a failure from the server, it repeatedly trims the last word of the question and replaces the trimmed sequence with the dummy string XX
- drugs with water solubility of 3.24e-02 mg/mL \(\rightarrow\) drugs with water solubility of XX
- drugs whose mechanism of action involves norepinephrine and serotonin \(\rightarrow\) drugs whose mechanism of action involves XX and YY
## Lexicon extracted from collections

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Resource Type</th>
<th>Distinct IDs</th>
<th>Distinct names</th>
<th>Considered properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>DrugBank</td>
<td>Drug</td>
<td>1470</td>
<td>22872</td>
<td><code>drugbank : name, drugbank : synonym, drugbank : brandName</code></td>
</tr>
<tr>
<td>DrugBank</td>
<td>Target</td>
<td>4553</td>
<td>3784</td>
<td><code>drugbank : name</code></td>
</tr>
<tr>
<td>Diseasome</td>
<td>Disease</td>
<td>4213</td>
<td>3642</td>
<td><code>diseasome : name</code></td>
</tr>
<tr>
<td>Diseasome</td>
<td>Gene</td>
<td>3919</td>
<td>4328</td>
<td><code>rdfs : label, owl : sameAs</code></td>
</tr>
<tr>
<td>SIDER</td>
<td>SideEffect</td>
<td>1737</td>
<td>2398</td>
<td><code>sider : sideEffectName</code></td>
</tr>
</tbody>
</table>
Results and their analysis

<table>
<thead>
<tr>
<th>Total</th>
<th>Processed</th>
<th>Right</th>
<th>Partially</th>
<th>Recall</th>
<th>Precision</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>25</td>
<td>24</td>
<td>1</td>
<td>0.99</td>
<td>1</td>
<td>0.99</td>
</tr>
</tbody>
</table>

Yet, the task is not solved: the grammars are manually generated, even though they follow a DL oriented approach in identifying the abstract trees $\rightarrow$ DL based structured methodology to built it.

Next:

- build a layer on top of the ontologies (collections schema) that allow users to introduce linguistic expressions for DL expressions ($\exists$ possibleDrug.Classs)
- automatically derive the abstract and the concrete grammars from this annotated ontology
Multilingual GFMed

- Abstract Grammar
- Incomplete concrete Grammar - concrete grammar without any language specific element
- Lexicon for English and Romanian
- Problems:
  - incomplete language paradigms in GF
  - translation of properties (mechanism of action - mecanism de actiune, half life - timp de injumatatire)
  - translation of Named Entities (Diseases, SideEffects, Drugs...)

GFMed: Question Answering over BioMedical Linked Data with
Conclusions

- Controlled natural language
- A grammatical based approach with DL perspective over the natural language
- The concrete English grammar is split in Incomplete Concrete Grammar (language independent) and specific to English functions
- The incomplete concrete grammar was tested with a different language - Romanian

Thank you!