Database Technology Project Report
Never Ending Language Learner

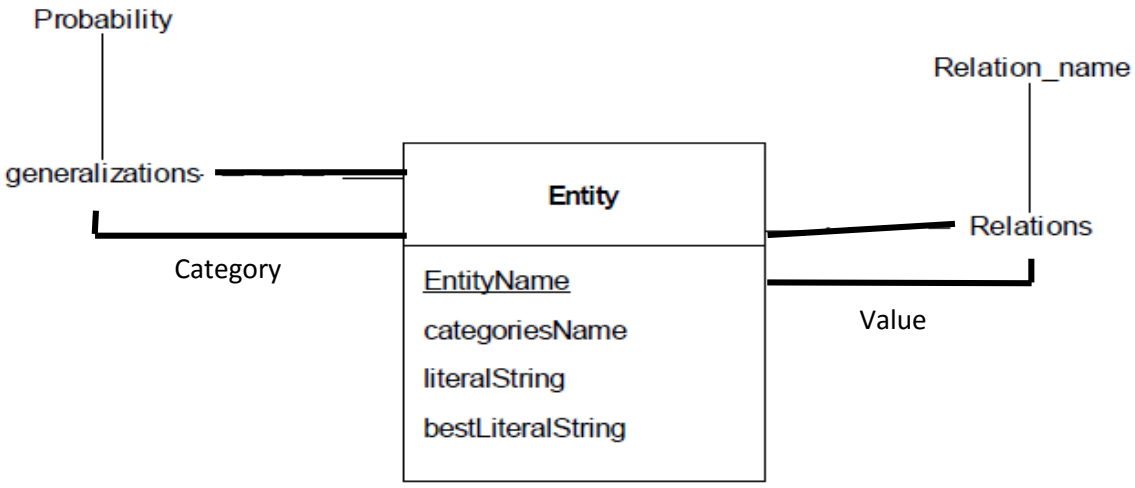
Introduction

The data that I'm using in this project is provided by the NELL program: Never Ending Language Learner. Somehow he manages to read the web, and give us various data that are likely to be true. And as Nell means, the program learns over time. The data that we get is stored into several columns. The main columns represent what we call a belief, meaning a triple of text values composed of 2 concepts and a relation between them (there is an order though, first concept is related through the relation value to the second one). The first concept is called entity then the relation and finally the second concept which is called value. In this typography, we could represent an entity that belongs to a category using the relation 'generalizations'. Then given this data, the main issue is to build a smart and efficient database model that answers to our needs, which are basically searching for some information into this knowledge database, such as an encyclopedia.

Let's see how we will proceed for the database modeling. This means that we have to build a relational model, so we could have a useful access to data. This single table that we had first is not convenient. For example we don't have access to the entity list, we have duplicates of entities, because an entity may intervene in several beliefs.

Database Modeling

The Database Model have to be built regarding to the different useful user's requests. This different requests are basically related to an entity (or a particular category) search, including information about relations between entities. According to that, a logical E-R diagram can be as followed:



We can as well add some attributes for the relation set ‘Relation’ as ‘iterationPromotion’, ‘source’ etc… Datatest is the data table given originally.
Given this E-R Diagram, we can easily compute the SQL script as followed:

```
create schema if not exists nell;
create table if not exists entities (
        entityName varchar(255) ,

         categoriesName varchar(255),
         literalstring text,
        bestLiteralString varchar(255),
        primary key (entityName));

insert  into Entities
        Select distinct `Entity`,`Categories for Entity`,`Entity literalStrings`,`Best Entity literalString`
        From datatest
        Where `Entity` is not null and Entity not in (Select entityName from Entities) ;

insert  into Entities
        Select distinct `Value`,`Categories for Value`,`Value literalStrings`,`Best Value literalString`
        From datatest
        Where `Value` not in (Select entityName from Entities) and `Value` is not null;

create table if not exists generalizations (
        entityName varchar(255),
        categoryName varchar(255),
        probability double,
        primary Key (entityName,categoryName),
        foreign key (entityName) references entities(entityName),
        foreign key (categoryName) references entities(entityName),
        check (probability<=1 and probability>=0));

insert into generalizations
        select Entity, Value, Probability
        from datatest
        where (entity, value) not in (select entityName,categoryName from generalizations) and
        relation='concept:generalizations';

create table if not exists beliefs (
        entity varchar(255),
        relation varchar(255),
        value varchar(255),
        iterationPromotion int,
        source text,
        candidateSource text,
        Primary key (entity,relation,value),
        foreign key (entity) references entities(entityName),
        foreign key (value) references entities(entityName));
insert into beliefs
        select entity,relation,value,`Iteration of Promotion`,Source,`Candidate Source`
        from datatest
        where (entity,relation,value) not in (select entity,relation,value from beliefs);
```

We notice that we don't really need id attributes for the entities (including categories) because every entity is in the form of 'concept:categoryName:entityName' and is unique. Moreover the search requests don't involve the entityName Column but the literalString column, which is logical because the literalString is the text that had been red by NELL in order to designate a particular entity.

The Queries and the actual use of the Database

The GUI code used is Java, with the JDBC driver that connects the Java program to the actual database. As we can see in the Java code, I decided to first give the possibility to the user of searching either an entity or a category (particular entities). Then the user can select one category or entity among the set of tuples returned, so he can select one of the different requests offered by the Database Model and get back the results relatively to the selected entity or category.

The queries to find the entity or category, given a name chosen by the user are:
For an entity research: *'Select distinct entityName from entities where literalString like ?'*
For a category research: *'Select distinct value from (beliefs inner join entities on entityName=value) where relation='concept:generalizations' and literalString like ?'*

The '?' refers to the unknown parameter in the SQL prepared statement that is sent to the database. It is actually the name typed by the user.

The different requests that I found important for this Database model are as followed:

1. Find all categories the selected entity belongs to.
   Select categoryName from generalizations where entityName=?
2. Find the top 50 entities that belongs to this category.
   Select distinct entityName from generalizations where category=? order by probability desc (then we just return the 50 first tuples)
3. Find all entities that are cooccurred with the selected entity in one triple.
   (select distinct entity from beliefs where value=?) union (select distinct value from beliefs where entity=?)

The Java program implements those SQL prepared statements thanks to JDBC and enables the user to interact with the database.

Optimization

The main optimization in the database design is the use of indices to speed up the search of tuples. The different Primary Keys are by default set as a B+ Tree index by MySQL. However for the query "Find the top 50 entities that belongs to this category" we have to use the index on probability so we can order more quickly the returned tuples.