KAF to SQL and KAF Debugger

Version 1
November 3, 2010

Knowledge Yielding Ontologies for Transition-based Organization

ICT 211423

Authors:

Maurizio Tesconi, CNR-IIT Pisa, Italy, maurizio.tesconi@iit.cnr.it
Matteo Abrate, CNR-IIT Pisa, Italy, matteo.abrate@iit.cnr.it
Clara Bacciu, CNR-IIT Pisa, Italy, clara.bacciu@iit.cnr.it

Introduction

This document explains the **motivations** behind our attempt to import the data from the KAF collection of XML documents into a relational database, our **implementation of an automatic conversion and import system** and briefly the **KAF debugger**, a web application that relies upon this database to help finding errors or inconsistencies in the annotated KYOTO documents.

KAF to SQL

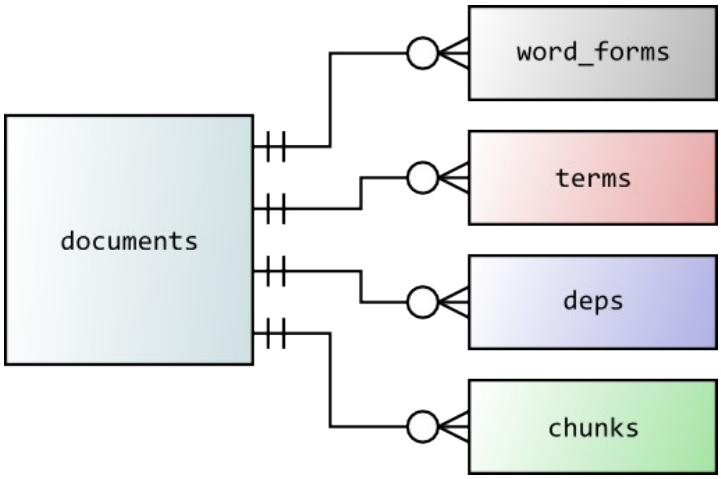
Observing a KAF file, we note that its main content consists of four sections: a list of word forms (text layer), a list of terms (term layer), a list of dependencies (deps layer) and a list of chunks (chunk layer). These sections present an **evident tabular structure**, and they can easily be mapped to classic relational tables.

It is our opinion that this new representation could be very helpful to reach a fast query performance on the KYOTO documents collection, and could also make some queries easier to write than its XQuery equivalent. Relational databases are a powerful and proven technology, so their use could provide substantial benefits, especially regarding robustness, stability, scalability and raw performance.

We have thus developed an SQL database schema for KAF, in order to do some experiments along these lines.

Database schema

Each KAF document in our database is identified by a row in the `documents` table, that has a one-to-many relationship with each of the tables representing the aforementioned KAF layers:



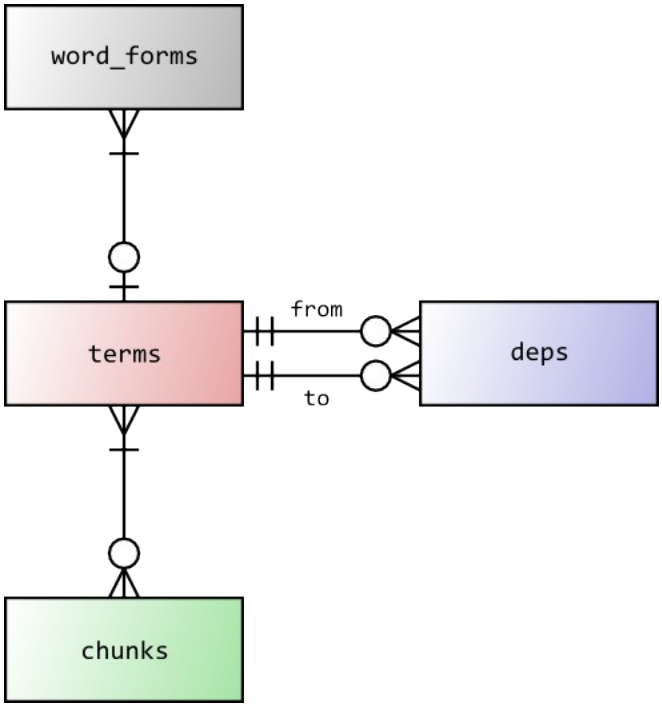
In our representation, no layer is mandatory. Furthermore, some of these relationships are not yet represented in the database using foreign key constraints.

An integer ID is assigned incrementally to each document imported into the database, defining the primary key of the `documents` table. The `word_forms`, `terms` and `chunks` tables each use this ID and the corresponding KAF element ID (wid, tid and cid) as primary key; these IDs are stored as strings in the current implementation, as in KAF.

Users of the database must pay attention to the fact that KAF IDs do not identify a single row in these three tables by themselves (e.g. a query for terms with tid=”t13” yields more than one result: a term for each document in which the tid is used).

Values from KAF attributes are stored in columns with the same name, with the exception of the `pos` attribute. Since the `pos` attribute in KAF is structured (formed by concatenating values separated by a dot), it has been divided into `pos` and `pos_ext` fields. The value of `pos` is the first character of the former `pos` attribute, while `pos_ext` contains the characters beyond the second (beyond the dot).

The relationships between the four layers are depicted below:



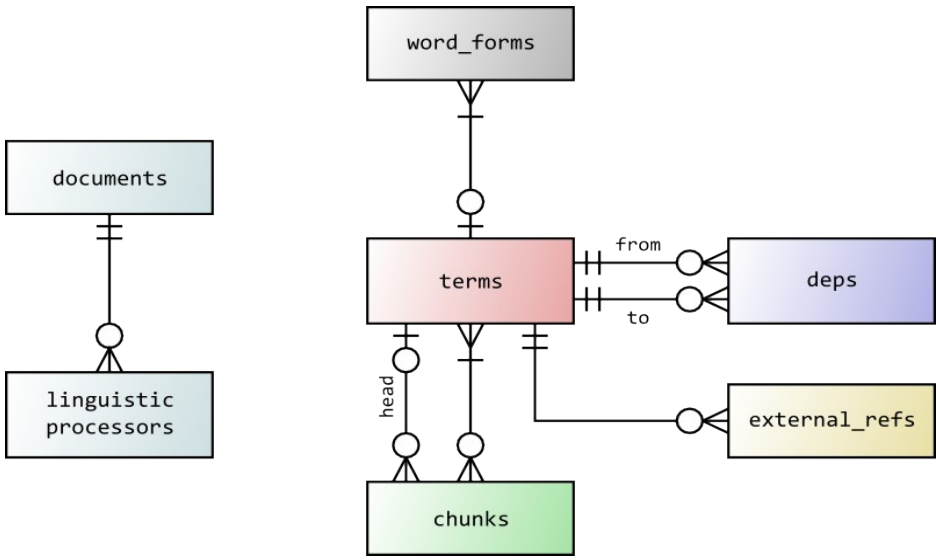
The `deps` table is implemented as a many-to-many relationship between the `terms` table and itself. With the exception of `case`, all the original KAF attributes of a dep define the primary key of the table, allowing different types of dependencies between the same ordered couple of terms.
Given a document::

- each word form MUST have zero (for punctuation or other nonterm tokens) or one associated term within the same document, and each term MUST have at least one or more (for multiwords or other terms spanning more than one token) associated word forms within the same document;
- each term have zero, one or more (in case of overlapping chunks) associated chunks within the same document, and each chunk MUST have at least one or more associated terms within the same document;
- each dep MUST have exactly one term as starting point and exactly one term as ending point, all of which within the same document; each term can participate as starting point or ending point zero or more times.

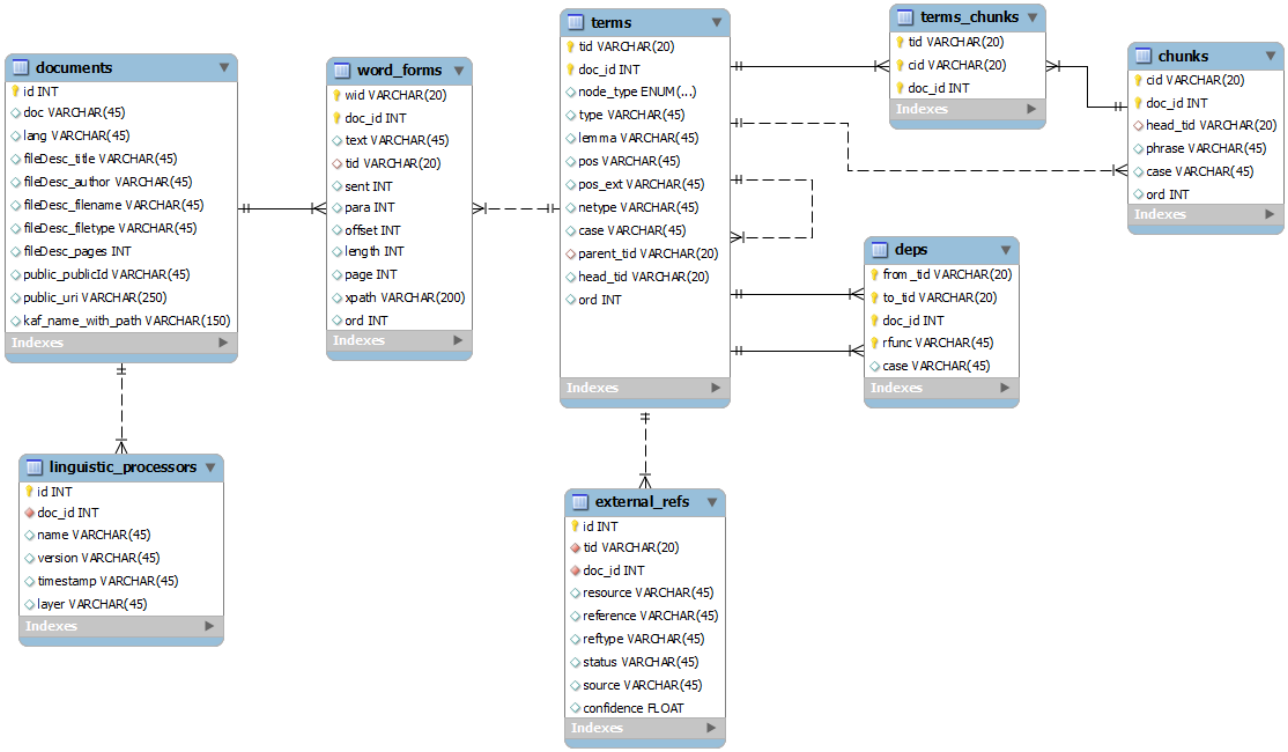
In our current implementation there is no enforcement for some of these constraints.

Users of the database must pay attention to the fact that the document ID must be taken into consideration in queries that resolve KAF ID references (e.g. a chunk referring the term “t13” is associated to the term “t13” within the same document, not to other terms that have tid=”t13” in other documents).

Two more tables represent the list of lingustic processors the document has been through and the external references. The picture below depicts all the tables and relationships in our database; some relationships between the documents table and the others are omitted for clarity:



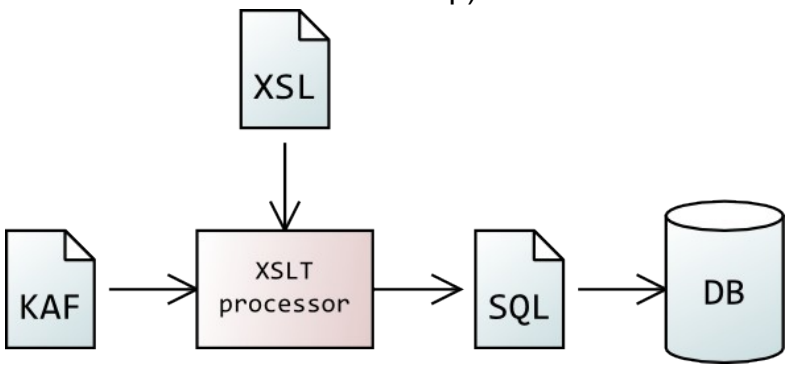
The following image is included for completeness, and shows the actual MySQL Workbench ER diagram used to create the database schema:



XSL Transformation

In order to populate the database, we have implemented an automatic process that import KAF documents from a directory in the filesystem. A shell script traverse the directory tree to find *.kaf files. For each file found:

- it feeds the KAF document into an XSLT processor, that applies an XSL transformation that produces an SQL representation of the document;
- the SQL script is then executed to update the database with the new document, saving errors in a log file. This log is very useful for debugging purposes, since it reports if a schema constraint fails (e.g. a reference to a nonexistent ID is detected in this step).

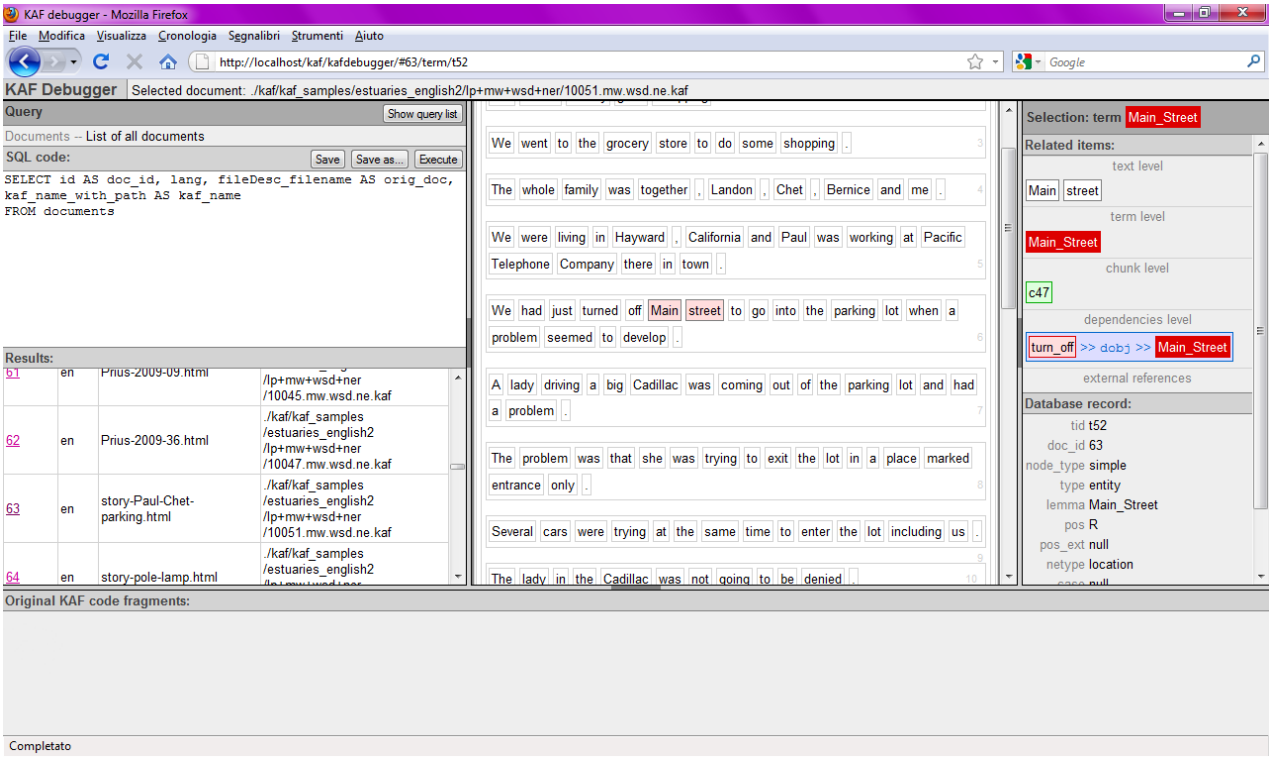


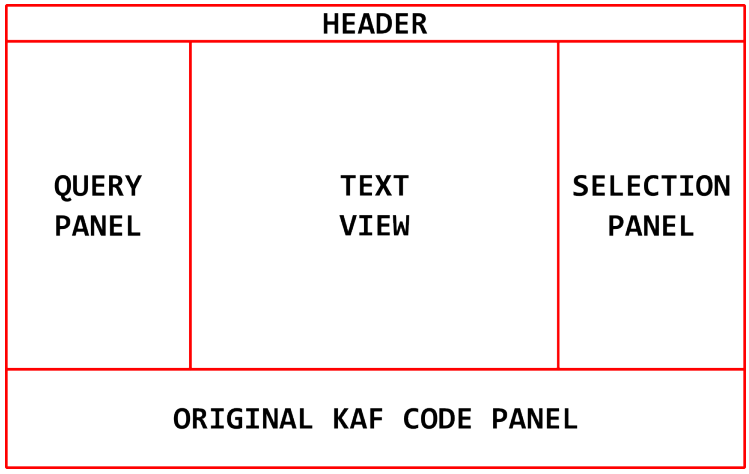
The handling of missing values for some attributes is not homogeneous: by default, if the XSLT processor encounters a missing value sends an empty string to the output document; in some cases we modified this default behavior to send `null` values instead.

KAF debugger

KAF is difficult to inspect manually because there are lots of ID references between annotation layers. In order to simplify debugging and semantic validation of documents, we have developed the KAF debugger, a web application for **navigating** and **querying** KAF documents.

It shows informations about word forms, terms, chunks, dependencies and external references, and it could prove to be a very useful tool to designer of linguistic processors and designers of kybots.





The interface is organized in five areas:

- the **header**, that shows informations about the selected document;

- the **query panel** (left), that could be used to execute arbitrary SQL queries;
- the **text view** (center), that displays the document content;
- the **selection panel** (right), that gives information about a particular item in the document;
- the **original KAF code panel** (bottom)

The three panels could be resized by dragging their borders and collapsed by clicking their gray handles.

Navigating KAF

The **text view** shows all the word forms from the selected document, grouped in sentences, paragraphs and pages. To inspect a word form, it is sufficient to click the gray box containing it: this action will select the item.
When an item is selected, many informations about it are shown in the rightmost panel:

- A list of **related items**, organized by annotation layers. This list gives to the user the ability to navigate between KAF layers, resolving ID references simply by clicking on an item.
- The **database record** of the selection.
- The **original KAF code** (in a collapsed panel at the bottom of the window). This shows fragments of the original KAF code that refer to the selected item, in order to find the actual problem (e.g. a duplicate ID is not imported into the database, but is still visible in the original code).

The current selected document and selected item form the part of the URI of the application beyond the hash (#), always visible in the address bar of the browser. For example, the string `#68/wf/w25` identifies the document with ID 68 and the word form “w25”, and the string `#68/chunk/c21` identifies the document with ID 68 and the chunk “c21”. This allows the user to navigate KAF documents and items like normal web pages, going back and forward, saving a history or bookmarking a particular selection; other applications or even text documentation can benefit from this approach and include simple web links to a KAF document and its items.
By using standard browser’s interfaces (e.g. CTRL+F), it is also possible to perform a full-text search in the opened document.

Querying the DB

Using the query panel a user can write an SQL query and execute it: results will be shown in a table in the lowest part of the panel. If the resulting table contains both a document ID column and one of the KAF element IDs (wid, tid or cid) column, it’s also possible to click on the IDs to load the document and select the corresponding item to begin a navigation session.
Queries can also be saved into a separate database, that is preloaded with examples.