Un modello ed un linguaggio ad oggetti per documenti testuali

Renzo Orsini

Dipartimento di Scienze Ambientali, Informatica e Statistica
Università Ca' Foscari Venezia

Quinto Convegno Annuale AIUCD, Venezia

7-9 Settembre 2016

Sintesi

- Problemi:
  - Come **rappresentare** un testo nei suoi vari aspetti
  - Come **fare ricerche** che riguardano il testo e le varie informazioni associate
  - Come **semplificare** lo sviluppo di applicazioni sui testi
- Soluzione proposta:
  - Abbandonare l'idea di rappresentare il testo come *testo "marcato"*
  - Introdurre meccanismi di astrazione e di modellazione ad alto livello
  - Introdurre un linguaggio specifico per "query" e sviluppo di applicazioni basato su oggetti (object oriented)

Problemi dei linguaggi di *mark-up*

- Problemi della marcatura (prendendo come esempio XML):
  - Il testo può avere una struttura con più gerarchie
  - Si vogliono associare più serie di informazioni diverse al testo
  - Ambiguità e discrezione nella marcatura (es. linee guida TEI)
- Conseguenze:
  - Livello di complessità difficile (o impossibile) da gestire “manualmente” (il documento non è *“scalabile”*).
  - Difficile o impossibile integrare marcature diverse
  - Difficile effettuare ricerche che tengono conto delle varie informazioni
- Soluzione classica (nell’ambito della marcatura):
  - Notazione stand-off (non si integra)
  - Aumento della complessità con *milestones*.

Alternative proposte nella letteratura

- Rappresentare il testo come gerarchia di oggetti (Coombs et al. 1987, De Rose et al. 1997)
- Uso di *standoff properties* (Schmidt 2012):
    - Rimuovere la marcatura dal testo e costruire un generico livello di annotazioni separate (standoff property), simile a LMNL (Piez 2010) (variante della marcatura standoff)
- Uso di componenti basate su oggetti (TeiCoPhiLib, Boschetti e Del Grosso, 2014):
    - Dal documento, rappresentato con notazione XML, anche in formato standoff, si estraggono oggetti che vengono utilizzati per la sua elaborazione:
        - Visualizzazione,
        - Annotazione
        - Analisi

Il documento non è la sua rappresentazione!

- La marcatura nasce “sopra” una rappresentazione concreta di un testo
- **Ma il documento non è la sua rappresentazione!**
- Nell’atto creativo che porta alla nascita di un documento, sono indissolubilmente “mischiati” il testo e gli aspetti strutturali con cui si concepisce:
  - la sua divisione in unità significative (capitoli, paragrafi, oppure strofe, versi, oppure atti, scene, ecc.),
  - il testo a livello base è comunque un’unità logica composta da frasi, segni di interpunzione e parole,
  - la forma fisica è di solito mutabile: l’impaginazione può cambiare, le spaziature possono essere diverse (ma fanno parte della rappresentazione fisica!) e hanno a che fare con la leggibilità, la fruizione estetica del documento.

L’idea fondamentale del modello

- Un documento testuale digitale è una coppia:
  - testo sottostante (“underlying text”, o “full text”)
  - struttura di oggetti testuali (che compongono più gerarchie)
- Un oggetto testuale è una entità software, dotata di stato e di comportamento
- Lo stato:
  - definisce la precisa porzione di testo sottostante
  - definisce una serie di proprietà:

    le componenti: oggetti testuali

    gli attributi: valori di qualunque tipo e complessità (metadati, annotazioni, ecc.)
- Il comportamento: operazioni significative sul tipo dell’oggetto (metodi)

Le regole fondamentali del modello

- Relazione fra componenti:
  Un oggetto testuale *t* è una componente di un oggetto testuale *t*' se e solo se il testo sottostante di *t* è una parte del testo sottostante di *t*'.

- Esempio:

  - Atto ⇒ Prologo
    Scene
    Epilogo

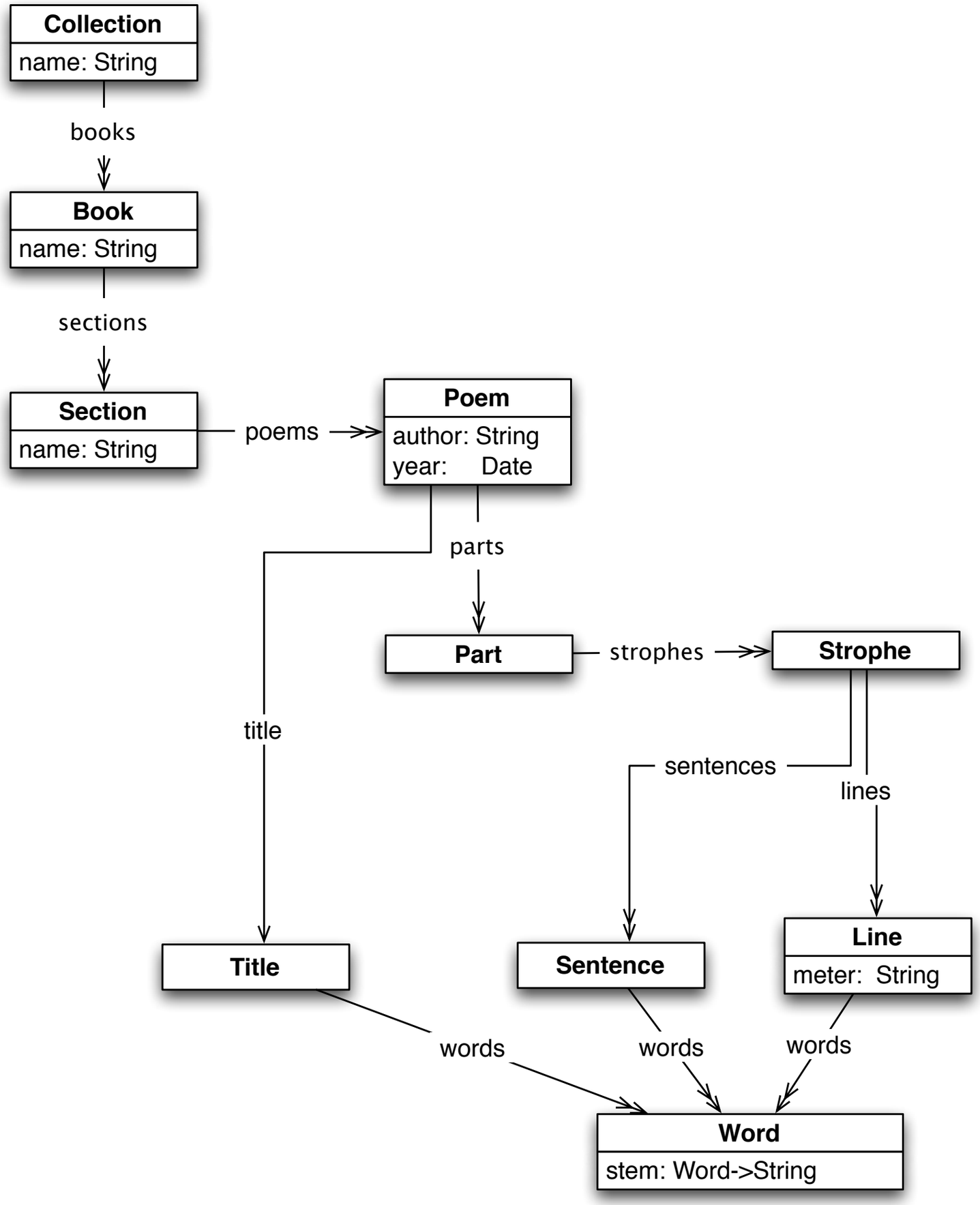
- Oggetti testuali ripetuti:
  Costituiti da elementi omogenei, condividono le caratteristiche del singolo oggetto e delle sequenze di oggetti.

- Esempio:

  - La prima frase di ogni sonetto di una serie di sonetti

Come descrivere il modello di un’opera

- L’idea fondamentale è che il modello dell’opera sia rappresentato con una serie di “tipi oggetti testuali”: lo *schema* del testo (tipo: termine tecnico informatico!)
- Il tipo descrive le caratteristiche degli oggetti testuali che compongono il testo (componenti), i suoi attributi, i suoi metodi
- I tipi prevedono anche sottotipi, che descrivono oggetti testuali più “specializzati” rispetto ad altri (es. un *endecasillabo* è un tipo specializzazione di *verso*)
- I tipi vengono dati con un linguaggio formale specifico, ed esiste una notazione grafica per rappresentarli in maniera sintetica e comprensibile.


Collection
name: String
books
Book
name: String
sections
Section
name: String
poems
Poem
author: String
year: Date
parts
Part
strophes
Strophe
title
sentences
lines
Title
Sentence
Line
meter: String
words
words
words
Word
stem: Word->String

Il linguaggio

- Il linguaggio prevede sia delle ricerche in stile “database SQL”, cioè con una serie di comandi che descrivono le proprietà dei dati che si vogliono ritrovare (query language), sia una parte più generale per lo sviluppo di applicazioni di qualunque complessità
- La parte di interrogazione ha una soglia di difficoltà relativa, e come per SQL, almeno per le interrogazioni più semplici, è alla portata di un non-informatico con un breve corso
- La parte di ricerca sugli oggetti testuali permette di combinare condizioni sia sulle componenti (testi contenuti) sia sugli attributi (metadati, annotazioni, ecc.)
- Introdurre un nuovo linguaggio ha vantaggi e svantaggi:
  - permette di avere uno strumento specializzato e molto efficiente per sviluppare applicazioni specifiche per analisi di testi
  - costringe ad imparare un nuovo linguaggio di programmazione
- Altra possibilità: estensione (API: tipi + operatori) in un linguaggio “tradizionale”

Esempi

```
let collection : Collection = usedatabase "montale";
let allbooks : Books = get books of collection;

get name of collection;          #=> "Poesie di Montale":String

get name of allbooks;            #=> [’Ossi di Seppia’, ’La Bufera e Altro’, ...]:[String]


let words = getall Word of books[1..3];

sort unique (text of words); #=> ["A", "ALBUM", "ALTA", "ANGUILLA", ...]:[String]


let short_poems : Poems =
  select p from p in poems where size of text of p < 300;

text of (get title of short_poems)
  #=> ["LONGOMARE", "LASCIANDO UN DOVE", ...]:[String]
```

Un esempio complesso

Trovare i titoli delle poesie dove si parla di amore e il numero dei versi nella poesia in cui la parola con radice “amor” è presente

```
let loveVerses : Verses =
select v
    from v in (getall Verse of collection)
    where some w in (getall Word of v)
      with (get stem of w) = "amor";

select {title = poemTitle, numberOfLoveVerses = (size of partition)}
from v in loveVerses
groupby {poemTitle = get title of (parent Poem of v)};

  #=> [{poemTitle = "L’ANGUILLA", numberOfLoveVerses = 1},
         {poemTitle = "INCANTESIMO", numberOfLoveVerses = 2},
          ...] : [{poemTitle:String, numberOfLoveVerses:Int}]
```

Architettura di un sistema di “Digital Library”

- Definizione di una libreria di schemi per testi di tipi diversi.
- Definizione di libreria di annotazioni di tipo diverso, applicabili a tutti i tipi di documenti:
  - ad es. annotazioni grammaticali, traduzioni, varianti, commenti, ecc.
- Uso di un formato per l’interscambio dei documenti (basato su XML)
- Sistema di visualizzazione di tipo “dichiarativo”
  - per le annotazioni come nelle mappe: si possono/escludere “livelli” diversi di informazioni sul terreno (in questo caso il terreno è il testo, i livelli sono i diversi tipi di annotazioni)
- Sistema multi-utente: permette annotazioni da parte di tanti utenti, è scalabile

Problemi aperti

- Definire i tipi può essere troppo vincolante / poco significativo in alcune situazioni ?
- Il linguaggio deve essere messo a punto su molti esempi
- Serve una cornice di riferimento per l'interoperabilità (uso di XML)
- Sarebbe molto utile avere una architettura di riferimento di un tale sistema, in modo da permetterne lo sviluppo di moduli diversi e istanze diverse di tali sistemi (TBMS, come DMBS)
- Il problema forse più importante: occorre un sistema funzionante e abbastanza completo per convincere della sua effettiva utilità.
- Stato attuale: esistono delle componenti separate, che devono essere integrate. Non c'è ancora un esecutore completo del linguaggio. Sono stati fatti degli esperimenti su alcuni corpus di testi.

Conclusioni

- Problemi attuali nella rappresentazione di testi
- Non è tanto importante **questa** specifica proposta, è importante rendersi conto che lo sviluppo delle tecnologie per la rappresentazione, gestione ed uso dei testi **potranno crescere abbandonando la marcatura come mezzo fondamentale di memorizzazione dei testi!**
- La proposta è quella di usare dei **meccanismi di astrazione**, che hanno avuto un grande successo nell'applicazione della tecnologia informatica a settori molto diversi
- Solo in questo modo si potranno far crescere in maniera significativa le possibilità di elaborazione dei tesi e ridurre la necessità di avere a disposizione degli informatici impiegati per costruire ogni volta un **nuovo** sistema per ogni progetto che si vuole realizzare.
- Questa riduzione dei costi passa attraverso una sforzo iniziale, congiunto fra umanisti e informatici, per la definizione di modelli e sistemi generali.