Progetto di Intelligenza Artificiale 2

Question Answering

Federico Ferrantelli
Christian Simolo
Lorenzo Terribili

Magistrale Informatica
Università di Tor Vergata

Aprile 2017

Specifiche del progetto

Un **QA** è un sistema capace di tradurre domande formulate in linguaggio naturale in query SPARQL.

Il nostro sistema ruota attorno all'universo cinematografico: sfruttando l'ontologia dedicata ai `Film` siamo in grado di rispondere a quesiti relativi alle pellicole cinematrografiche, ai suoi attori, al cast tecnico e ai dettagli tecnici relativi ai film stessi (durata, incasso, budget di produzione, ...).

Vogliamo sfruttare questa conoscenza del campo cinematografico per rispondere quindi a domande del tipo:

Who is the director of Blade Runner? *Ridley Scott*

Presentazione dei 3 step implementativi

Abbiamo suddiviso il lavoro in 3 step implementativi:

1. **Template Parser:** elaborazione della domanda tramite l'analisi di alcuni *template* di esempio; opera un *pattern matching* dedito al riconoscimento di:
   - Soggetto;
   - Predicato;
   - Oggetto relativo al predicato.
2. **Synonymer:** per la gestione di quei fenomeni di *sinonimia*, *iponimia* ed *iperonimia* che, applicata alla domanda posta in input, ne permette un raffinamento dei risultati ottenuti;
3. **Parser Sintattico:** elabora la domanda tramite un'*analisi sintattica* dell'input stesso. L'impiego di questo parser assieme al synonymer, ne completa le capacità ed amplia il panorama delle frasi alle quali il QA può rispondere.

STEP I
TEMPLATE PARSER

Template Parser

Template Parser

Vogliamo rispondere a domande semplici e precise (le *Wh-questions*), effettuando una prima analisi della struttura della frase tramite *pattern matching* e riconoscendo così: predicato, oggetto e, eventualmente, il soggetto.

Ad esempio:

- *Who is the director of Mad Max: Fury Road? George Miller*
- *Is Steven Spielberg the director of Saving Private Ryan?* TRUE
- *What is the budget of E.T.? 1.05E7 usDollar*
- *What is the gross of Avatar? 2.788E9 usDollar*
- *What is the runtime of The Goonies? 114.0 minute*

Implementazione

Questo primo step consiste nella:

- Realizzazione di un parser ad-hoc dotato di alcuni parser elementari che permetteranno una veloce elaborazione delle *wh- questions*
- Derivazione della query SPARQL sfruttando le informazioni ricavate dalla domanda posta in input, tramite utilizzo delle API fornite da OWL-ART

Svantaggi:

Il sistema risulta limitato poiché è in grado di riconoscere solo quelle domande legate ai pattern forniti.

Restringerci a questi pattern ci permette di rispondere esclusivamente a domande semplici, che spesso contengono errori grammaticali.

Il pattern matching 1/5

Questa prima realizzazione è stata svolta in 5 punti:

1. Caricamento dei pattern

Consideriamo i seguenti TAG che indicano le componenti elementari di una tipica frase in lingua inglese:

- **#O** oggetto della frase
- **#P** predicato della frase
- **#S** soggetto della frase

Nel progetto viene fornito un file XML contenente i pattern relativi alle tipologie di domande riconosciute:

- (what—who) is the #P of #O?
- (what—who) is #P of #O?
- (what—who) film was #P by #O?
- (are—is) #S the #P of #O?

Il pattern matching 2/5

2. Definizione delle espressioni regolari per le frasi di tipo *soggetto-predicato-oggetto* all'interno di ciascun pattern fornito e definizione del parser per i pattern stessi. In caso di esito positivo viene restituito il match designato.

Il parser necessita di sostituire tutti i TAG con la regex: `(.+?)`.

Applicata la sostituzione, tutto il pattern risulterà come un'unica espressione regolare. Quindi, si parsa la domanda e vengono selezionati i valori dei relativi ai TAG, ovvero ciò che ci interessa per svolgere con successo la ricerca. Ad esempio, la prima regex si ferma al primo `[ of ]`.

Ad esempio:

(what—who) is the `(.+?)` of `(.+?)`

Il pattern matching 2/5

2. Definizione delle espressioni regolari per le frasi di tipo *soggetto-predicato-oggetto* all'interno di ciascun pattern fornito e definizione del parser per i pattern stessi. In caso di esito positivo viene restituito il match designato.

L'utilizzo di questa strategia basata sui template e con l'impiego di espressioni regolari generiche ci permette di tracciare una linea tra ciò che siamo in grado di rispondere e cosa no.

Vantaggi:

Who is the music composer of Jurassic Park? John Williams

Svantaggi:

Who is the composer of the music of Jurassic Park? No results.

Il pattern matching 2/5

REGULAR EXPRESSION

```
/ (What|Who) is the (.+?) of (.+?)\?
```

TEST STRING

```
Who is the music composer of Jurassic Park?

Who is the composer of the music of Jurassic Park?
```

```
Who is the music composer of The Lord of the Rings: The Fellowship of
the Ring?
```

Il pattern matching 3/5

3. Come filtro della pre-query viene utilizzato il valore restituito dal match per il TAG #P in modo da ottenere tutte le proprietà RDF correlate a tale oggetto.

Pre-query:

```
SELECT ?x WHERE {
?movie a <http://dbpedia.org/ontology/Film> .
?movie ?x ?y
FILTER (regex(str(?x), "^.*#P.*")) }
```

Inizialmente cercavamo il predicato tra tutte le proprietà di dbpedia, ma, grazie all'intervento degli ing. Fiorelli e ing. Turbati, ci siamo ristretti all'ontologia dei `Film`.

L'ontologia è configurabile da applicativo.

Il pattern matching 3/5

3. Come filtro della pre-query viene utilizzato il valore restituito dal match per il TAG #P in modo da ottenere tutte le proprietà RDF correlate a tale oggetto.

Si consideri lo standard utilizzato da *dbpedia* per le diciture delle sue *property* e *resource*, come ad esempio `dbo:director`, `dbo:editing` e `dbo:musicComposer`.

In base a quanto detto, abbiamo stabilito che:

- Se #P è costituito da una singola parola, come *director* ricerchiamo *irector* per ricercare sia nelle pagine dove compare sia con la `D` sia con la `d`;
- Se #P è costituito da più parole, come *music composer* lo ricerchiamo in camelCase: *musicComposer*.

Il pattern matching 4/5

4. La tripla identificata dal pattern viene utilizzata nella query SPARQL dove: #O corrisponde al *soggetto*, #P la *proprietà* ed #S è il valore di ritorno che ci si aspetta come risposta dalla query.

In caso di *ask-query*: *Is Steven Spielberg the director of Titanic?*

Ask-query:

```
SELECT ?movie (COUNT(?movie) AS ?countReg) WHERE {
?movie a <http://dbpedia.org/ontology/Film> .
?movie <http://dbpedia.org/ontology/abstract> ?abstract .
?movie <http://dbpedia.org/property/#P>
        <http://dbpedia.org/resource/#S> .
FILTER (regex(str(?abstract), "^.*#O.*"))
} ORDER BY DESC(?countReg) LIMIT 1
```

Al terzo step l'utente potrà impostare manualmente sia l'ontologia di riferimento che la proprietà tramite cui filtrare.

Il pattern matching 4/5

Problema risolto:

Poiché i film in dbpedia vengono accostati a desinenze come `_(film)` o `_(YYYY_film)` come ad esempio `Titanic_(1997_film)` e `Demolition_Man_(film)`, inizialmente ricercavamo la combinazione vincente accostando al titolo ogni desinenza conosciuta.

Come concordato successivamente negli incontri avvenuti, il metodo finale da noi utilizzato permette, tramite le query appena mostrate, di ricercare le occorrenze del *titolo del film* all'interno degli `abstract` di ogni `Film` in dbpedia, e di restituire la risorsa che ne ha contate di più.

Grazie a questa strategia raggiungiamo la risorsa del film esatta qualora ci fossero pellicole omonime alle quali difficilmente ci riferiremmo (ed in tal caso sarebbe necessario e imprescindibile specificare l'anno del film nella sintassi sopra citata).

Il pattern matching 4/5

4. La tripla identificata dal pattern viene utilizzata nella query SPARQL dove: #O corrisponde al *oggetto*, #P la *proprietà* ed #S è il valore di ritorno che ci si aspetta come risposta dalla query, ovvero il *soggetto*.

Nel caso di domande generiche: *Who is the director of Titanic?*.

Query per domande generali:

```
SELECT ?x (COUNT(?x) AS ?countReg) WHERE {
?movie a <http:// dbpedia.org/ontology/Film> .
?movie <http://dbpedia.org/ontology/abstract> ?abstract .
?movie <http://dbpedia.org/property/#P> ?x .
FILTER (regex(str(?abstract), "^.*#O.*"))
} ORDER BY DESC(?countReg) LIMIT 1
```

Il pattern matching 4/5

4. La tripla identificata dal pattern viene utilizzata nella query SPARQL dove: #O corrisponde al *soggetto*, #P la *proprietà* ed #S è il valore di ritorno che ci si aspetta come risposta dalla query.

Vantaggi nel coinvolgere il filtro sugli abstract relativi ai film:

- Ricercare titoli non precisi come *Star Wars 4* porta con successo a *A New Hope* grazie anche al mapping incluso nella resource di dbpedia
- Tempi di processamento della `FILTER` trascurabili

Svantaggi del punto 4:

Is Leonardo DiCaprio an actor of Titanic? non risponde, poiché nella risorsa `Titanic_(1997_film)` non è presente la proprietà *actor*, ma bensì *starring*

Il pattern matching 5/5

5. Analisi dei risultati ottenuti dalle query eseguite

A seconda del tipo di query effettuata, ne viene analizzato il risultato in maniera differente:

- Nel caso di una *ask-query* se viene restituito nessuno, uno o più valori saremo in grado di rispondere alla domanda con YES o NO
- Nel caso di una domanda generica invece, viene restituito il valore con più ricorrenze nella query tramite COUNT

Resoconto del primo step

Vantaggi del primo step:

Rispondiamo alle domande:

- *Who is the director of Mad Max: Fury Road*
- *Is Steven Spielberg the director of Saving Private Ryan*
- *What is the budget of Avatar?*

Svantaggi del primo step:

Non rispondiamo alle domande:

- *Who is the composer of the music of La La Land?*; l'espressione regolare seleziona erroneamente *composer* e *the music of La La Land*
- *Who is the director The Lord of the Rings?*; il film è presente con un titolo diverso
- *Who is the editor of Whiplash?*; il ruolo di *editor* è contenuto nella proprietà *editing* del film `Whiplash_(2014_film)`

STEP II
SYNONYMER

Synonymer

Synonymer

Vogliamo trovare una soluzione valida laddove l'utente richieda una proprietà il cui termine è un **sinonimo** (o **iponimo** o **iperonimo**) di una proprietà realmente contenuta nella risorsa di dbpedia.

In altre parole, vogliamo rispondere a domande come:

- *Who is the manager of La La Land?* intendendo il regista (*director*);
- *What is the fund of The Thin Red Line?* intendendo il *budget* del film;
- *What is the box office of Avatar?* intendendo il guadagno (*gross*) della pellicola;

Come funziona il synonymer 1/6

Consideriamo la domanda d'esempio: *Who is the manager of La La Land?* eseguita con il *template parser* ed il synonymer abilitato.

- *manager* è una proprietà sconosciuta a dbpedia
- *manager* è un nome comune utilizzato come sinonimo di *director*, ovvero il *regista* del film La La Land
- *director* è una proprietà conosciuta da dbpedia e contenuta nella risorsa relativa a suddetto film

Il procedimento seguente si pone antecedentemente all'esecuzione del primo step.

Come funziona il synonymer 2/6

Tramite **WordNet** navighiamo tra i sinonimi associati al *sysnet* corrispondente della risorsa *manager*, tra i suoi iponimi e tra i suoi iperonimi:

Noun

- S: (n) director, **manager**, managing director (someone who controls resources and expenditures)
  - *direct hyponym* / *full hyponym*
    - S: (n) bank manager (manager of a branch office of a bank)
    - S: (n) district manager (a manager who supervises the sales activity for a district)
    - S: (n) manageress (a woman manager)
  - *direct hypernym* / *inherited hypernym* / *sister term*
    - S: (n) administrator, decision maker (someone who administers a business)

Limitandoci a considerare *director*, viene elaborata la domanda *Who is the director of La La Land?* come spiegato nello step 1, rispondendo con *Damien Chazelle*.

Come funziona il synonymer 3/6

Cosa succede se non si ha un esito positivo cercando direttamente nel synset del termine?

Vediamo il flusso completo di esecuzione nel caso in cui il predicato sia composto da un solo termine:

Navigazione su WordNet (caso con singolo termine):

1. Si cerca tra i sinonimi del termine
2. Si cerca per due livelli di profondità tra gli iponimi del termine
3. Si cerca per due livelli di profondità tra gli iperonimi del termine
4. Si ripetono i punti 2 e 3 fino al matching con una proprietà della risorsa dbpedia

Come funziona il synonymer 4/6

Ad esempio, chiediamo al QA: *What is the fund of The Thin Red Line?*

Noun

- S: (n) **fund**, monetary fund (a reserve of money set aside for some purpose)
  - *direct hyponym* / ***full hyponym***
    - S: (n) budget (a sum of money allocated for a particular purpose)
      "the laboratory runs on a budget of a million a year"

- I sinonimi (*monetary fund, stock, store*) non ci aiutano;
- Cercando tra gli iponimi di primo livello troviamo: *savings, deposit,* ***budget***, ...;
- Troviamo in *budget* la soluzione alla domanda posta.

L'utente voleva sapere il budget investito per la produzione de *La sottile linea rossa* ed ottiene correttamente in risposta $5.2E7$. *Budget* è iponimo di *fund*.

Come funziona il synonymer 5/6

Vediamo il flusso completo di esecuzione nel caso in cui il predicato sia composto da due o più termini:
La navigazione avviene ancora su due livelli di profondità e ***termina*** *alla prima individuazione valida. Il suo ordinamento è stato accuratamente scelto dopo la fase di testing.*

Navigazione su WordNet (caso con due termini: *box office*):

1. Si cerca tra i sinonimi della stringa intera (*box office*)
2. Si cerca tra gli iponimi della stringa intera (*box office*)
3. Si cerca tra gli iperonimi della stringa intera (*box office*)
4. Si ripetono i punti da 1 a 3 per il primo termine della stringa (*box*)
5. Si ripetono i punti da 1 a 3 per il secondo termine della stringa (*office*)

Come funziona il synonymer 6/6

Ad esempio, chiediamo al QA: *What is the box office of The Thin Red Line?*

Noun

- S: (n) **box office** (total admission receipts for an entertainment)
  - ***direct hypernym*** / *inherited hypernym* / *sister term*
    - S: (n) gross, revenue, receipts (the entire amount of income before any deductions are made)
- S: (n) **box office**, ticket office, ticket booth (the office where tickets of

- I sinonimi (*ticket office, ticket booth*) non ci aiutano;
- Gli iponimi di primo livello non sono presenti;
- Cercando tra gli iperonimi di primo livello troviamo: *revenue, receipts,* ***gross***, . . . ;
- Troviamo in *gross* la soluzione alla domanda posta.

L'utente voleva sapere il guadagno de *La sottile linea rossa* ed ottiene correttamente $9.81E7$. *Gross* è iperonimo di *box office*.

Resoconto del secondo step 1/2

Vantaggi del secondo step:

- *Who is the author of The Thin Red Line?* → dal synset di *author*: *writer* → lo sceneggiatore (e regista) del film: *Terrence Malick*;
- *What is the wholesaler of Whiplash?* → dagli iperonimi di *wholesaler*: *distributor* → la casa distributrice del film *Sony Picture Classics*

Svantaggi del secondo step:

- Limitazione ereditata dall'uso dei limitati pattern predefiniti dello step 1;
- L'impiego di termini troppo vaghi rallenta e devia la ricerca della proprietà corretta;

Resoconto del secondo step 2/2

Domande che non rispondono allo step II:

- *What is the running time of Hacksaw Ridge?*; il termine *runtime* non è presente all'interno di WordNet, pertanto ciò potrebbe essere risolto solo passando per un *lexicon*. Tuttavia, la durata del film non è una proprietà contenuta nella risorsa del film;
- *Who is the main actor of Braveheart?*; poiché gli attori sono elencati in ordine alfabetico, non riusciamo a distinguere il protagonista;
- *Who is the composer of the music of La La Land?*; WordNet non riconosce il predicato *composer of the music*;
- *Who is the editor of Whiplash?*; in dbpedia è presente la proprietà *editing* non raggiungibile dal termine *editor* tramite WordNet;

STEP III
PARSER

Parser OpenNLP

Parser

Realizzazione di un parser in grado di effettuare l'analisi sintattica ed il riconoscimento degli elementi che comporranno l'albero sintattico della domanda posta in linguaggio naturale in input

- Non si è più limitati dall'uso dei template dello step I: vogliamo rispondere alla moltitudine di domande che si possono porre in ambito cinematografico in lingua inglese
- Si può usare accostato al synonymer ampliando il panorama delle domande a cui troviamo risposta
- Si vuole riconoscere le entità di una frase: soggetto, predicato e oggetto, escludendo quei termini che risultano obsoleti alla comprensione della domanda e concentrandoci su quei termini che giungono invece ad una soluzione valida

Le pipeline 1/2

Una domanda posta al sistema può passare per le seguenti operazioni:

SEGMENTER: tutte le parole, i simboli e la punteggiatura della domanda vengono considerati come singoli token

POS TAG: effettua l'analisi grammaticale; ciascun token viene considerato come *part-of-speech* ed assume un TAG specifico relativo al ruolo assunto, tramite **Penn Treebank Project**

STEMMER: processa le parole poste in forma flessa per ridurle alla forma radice, tramite **Snowball Stemmer**

producer→*produc*, *editing*→*edit*, *writer*→*writer*

LEMMATIZER: processa le parole poste in forma flessa per ridurle alla propria forma canonica (il lemma dalla quale deriva), tramite **Gate Lemmatizer**

are→*be*, *written*→*write*, *editing*→*edit*

Le pipeline 2/2

La nostra implementazione sfrutta i seguenti strumenti forniti da Apache:

- *UIMA*, framework per l'analisi e la gestione di informazioni semi-strutturate tramite JCAS, oggetto legato alle annotazioni. Il parser riceve i JCAS della frase, li elabora e restituisce i JCAS con le annotazioni correttamente assegnate;
- *OpenNLP*, il parser coinvolto per l'annotazione per UIMA

Nel QA vengono fornite le seguenti pipeline:

- **simple**: parser, POS tag & segmenter
- **stemmer**: parser, POS tag, segmenter & stemmer
- **lemmatizer**: parser, POS tag, segmenter & lemmatizer

Il risultato di questo processo ci fornisce l'**albero sintattico** della domanda posta in input.

Come funziona il parser 1/14

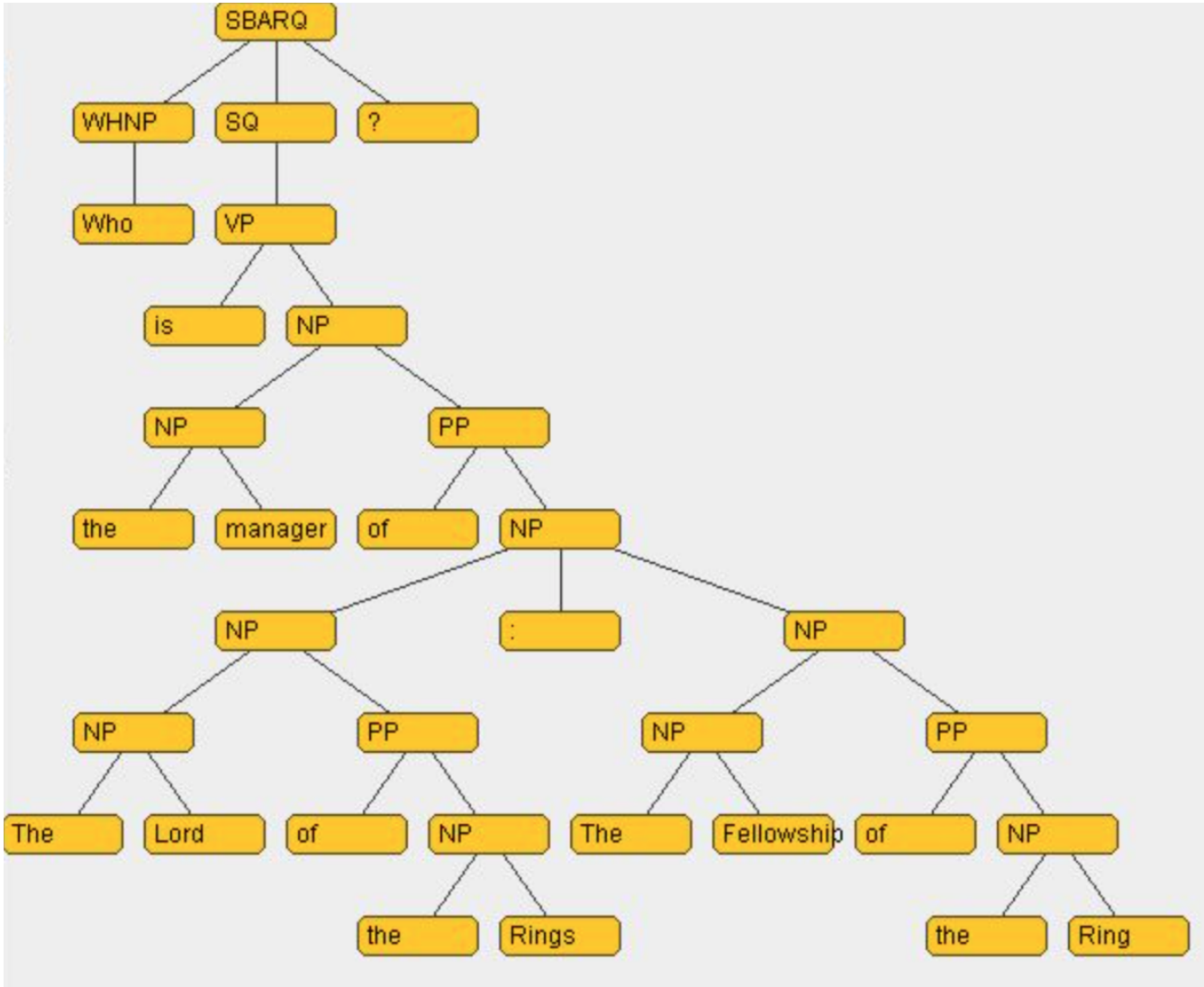
Consideriamo la domanda d'esempio: *Who is the manager of The Lord of the Rings: The Fellowship of the Ring?* con **simple pipeline** e **synonymer** risponde correttamente con *Peter Jackson*, individuando in *manager* il sinonimo di *director*.

Il primo passo, che avviene per **tutte** le pipeline è: **costruire l'albero sintattico**.

- OpenNLP prende in input la *sentence*, effettua il parsing e restituisce il JCAS con le annotazioni
- Il JCAS viene visitato in profondità e, in parallelo, viene costruito l'**albero sintattico** mostrato alla slide successiva
- Il prossimo passo sarà considerare esclusivamente tutti quei terminali utili al raggiungimento di un corretto risultato

Come funziona il parser 2/14

L'albero sintattico relativo alla *sentence*:



Come funziona il parser 3/14

Il prossimo passo è: **costruire il sottoalbero relativo al predicato della frase**.

L'albero viene visitato in profondità fino all'individuazione di un nodo PP: se sotto tale nodo è immediatamente presente il predicato l'esecuzione termina restituendo il sottoalbero figlio di PP.

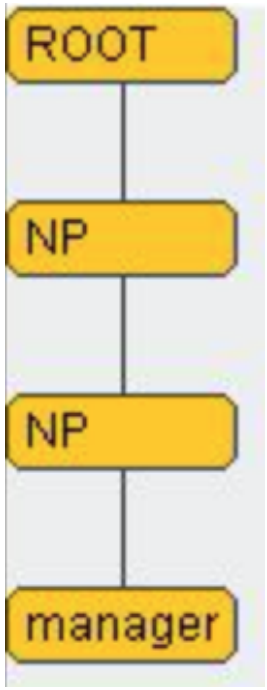
Se il predicato non è stato raggiunto, si prosegue fino alla **prima** occorrenza di un nodo VP o NP. Se è la prima volta che troviamo uno di questi due nodi, allora nel sottoalbero in lui radicato è presente il predicato. Pertanto un primo filtro agisce escludendo i verbi solitamente usati per porre domande in lingua inglese con una corretta sintassi, come ad esempio *is*, *are*, *was*, *were*, ... Questi verbi compaiono come foglie dei nodi VP.

Come funziona il parser 4/14

La visita prosegue fino all'individuazione del token corretto figlio di un nodo NP o VP.

Nel nostro esempio troncheremo il verbo *is* e, nel sottoalbero radicato in VP, troveremo il token corrispondente al predicato.

L'esempio a destra è quanto otteniamo dall'applicare il procedimento appena illustrato all'albero sintattico relativo alla *sentence*.



Come funziona il parser 5/14

Il prossimo passo è: **costruire il sottoalbero relativo all'oggetto della frase**.

L'albero viene visitato in profondità fino all'individuazione di un nodo NP o PP, perché:

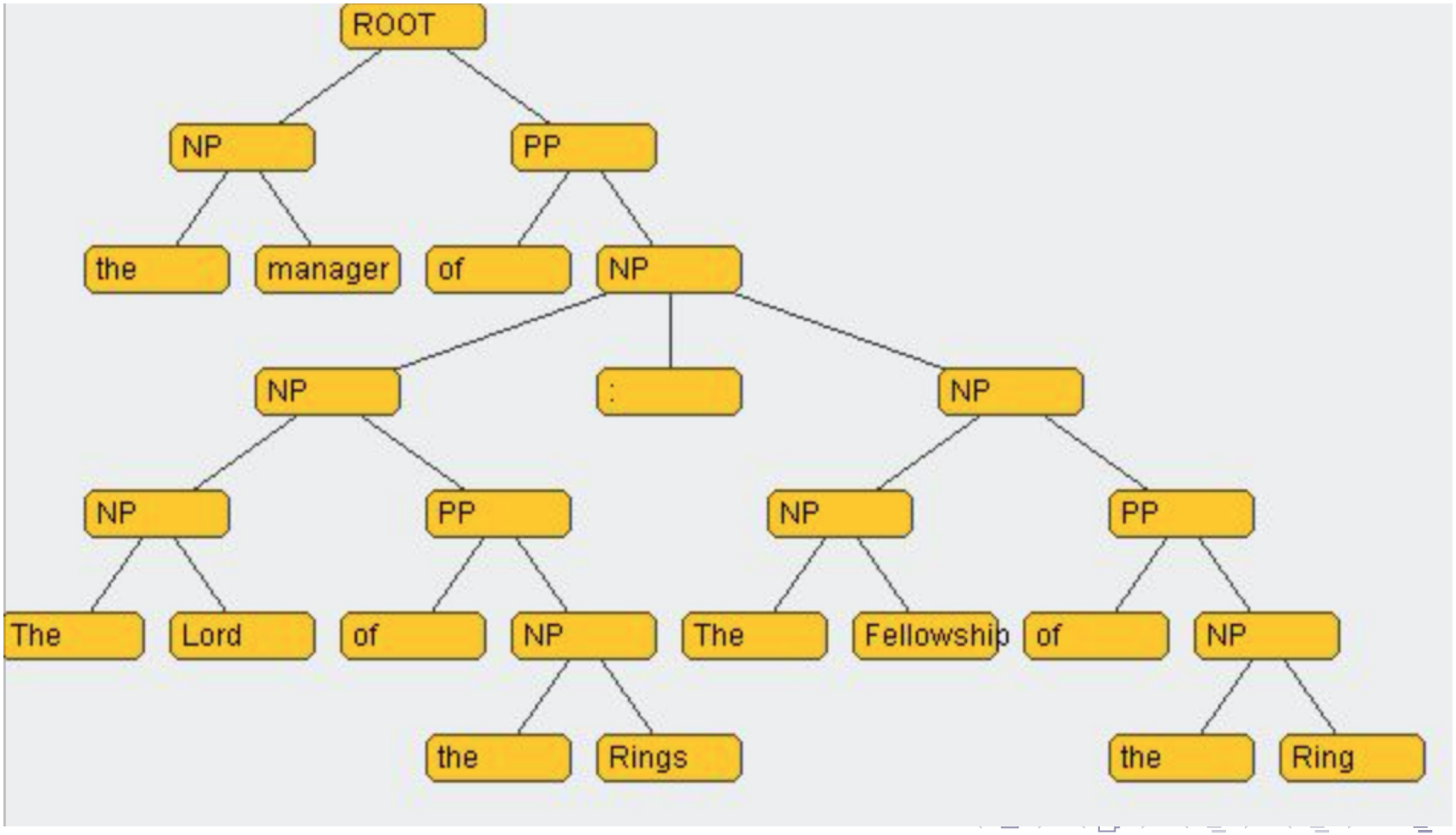
- il titolo del film sarà figlio di una coppia di questi due nodi,
- non sappiamo quale sia la coppia che conterrà il titolo,
- PP compare sempre come figlio di NP.

Pertanto consideriamo tutto il sottoalbero figlio di questa prima coppia di nodi, lasciando a posteriori la selezione dei termini che ne individuano il titolo del film escludendo il resto.

In alcuni particolari casi può capitare che il predicato si trovi come figlio di un nodo PP; viene effettuato un matching tra i due sottoalberi per escludere, eventualmente, il predicato che verrebbe altrimenti coinvolto nel sottoalbero dell'oggetto.

Come funziona il parser 6/14

Il sottoalbero relativo all'*oggetto*, ottenuto procedendo come illustrato a partire dall'albero sintattico relativo alla *sentence*:



Come funziona il parser 7/14

Il prossimo passo è: **costruire il sottoalbero relativo al soggetto della frase**.

Questo passo verrà effettuato esclusivamente nel caso di *ask-questions* come ad esempio *Is Ron Howard the director of A Beautiful Mind?*. Il soggetto, Ron Howard, verrà trovato come nodo foglia del sottoalbero radicato nel primo nodo NP incontrato durante la visita in profondità.

In qualsiasi altro caso, come quello preso in esame per il terzo step, il soggetto rappresenta la risposta che ci si aspetta dal QA al quale poniamo la domanda.

Quindi i tre sottoalberi ottenuti vengono **visitati** un'ultima volta ed i suoi terminali vengono raccolti in stringhe per la futura estrapolazione dell'informazione desiderata.

Come funziona il parser 8/14

Il passo conclusivo è: **isolare gli elementi utili alla costruzione della query**. Le stringhe costruite vengono processate per ottenere le componenti della tripla RDF nel seguente modo:

- **Oggetto:** il titolo di un film in lingua inglese ha sempre la prima lettera e l'iniziale di ogni parola che lo compone in maiuscolo, ad eccezione di articoli e pronomi.

  Quando si pone una domanda al QA è necessario rispettare questa regola, in modo da ottenere la risposta corretta.

  L'espressione regolata usata dunque è `([A-Z].+)`

  Nell'esempio verrà selezionato *The Lord of the Rings: The Fellowship of the Ring*.

Come funziona il parser 9/14

- **Predicato:** il primo filtro applicato ha rimosso dal sottoalbero del predicato il verbo *is*; un successivo filtro viene applicato alla stringa, in modo da eliminare quei termini che risultano prettamente legati alla grammatica della frase e no alla sintassi della domanda posta.

  Nell'esempio, dalla stringa del predicato *the manager of* viene rimosso l'articolo *the* e la preposizione *of*, identificando in *manager* il predicato voluto.

- **Soggetto:** nel caso in cui sia presente il sottoalbero del soggetto (e quindi di una *ask-question*) verrà selezionato tutto ciò che compare nella stringa dall'*albero relativo al soggetto* ottenuto in output.

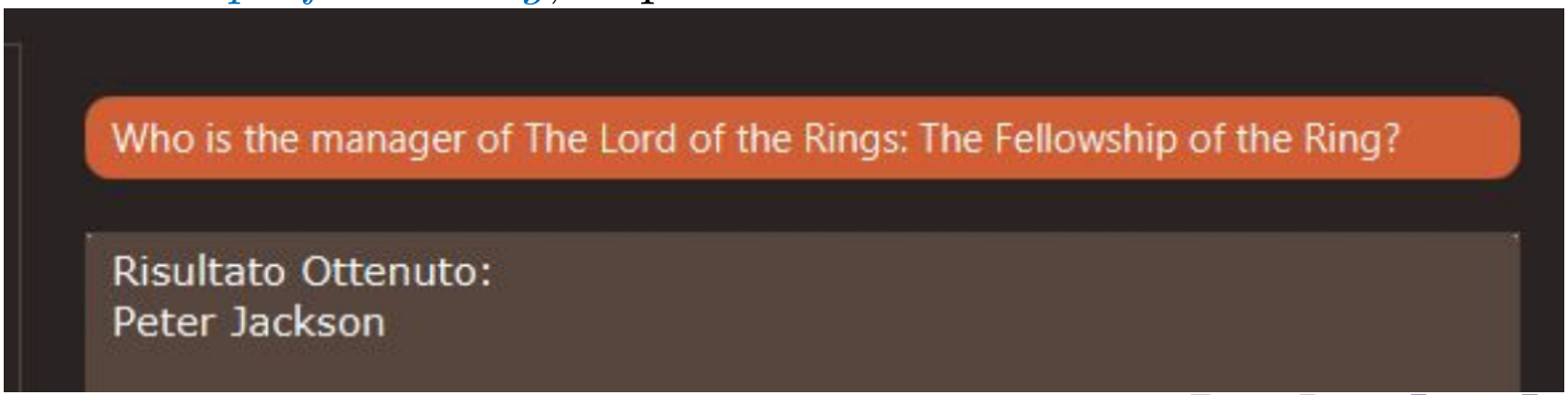
  Is Ron Howard the director of A Beautiful Mind? viene selezionato *Ron Howard*.

Come funziona il parser 10/14

A questo punto, il flusso esecutivo intrapreso a seguire è il medesimo discusso nei precedenti due step del progetto.

Il generatore di query SPARQL riceve ed elabora le informazioni catturate in questo procedimento mostrato, derivando la corretta query da eseguire.

Quindi: *Who is the manager of The Lord of the Rings: The Fellowship of the Ring?*, dopo aver individuato *manager*→*synonymer*→*director* e *The Lord of the Rings: The Fellowship of the Ring*, risponde correttamente così:



Come funziona il parser 11/14

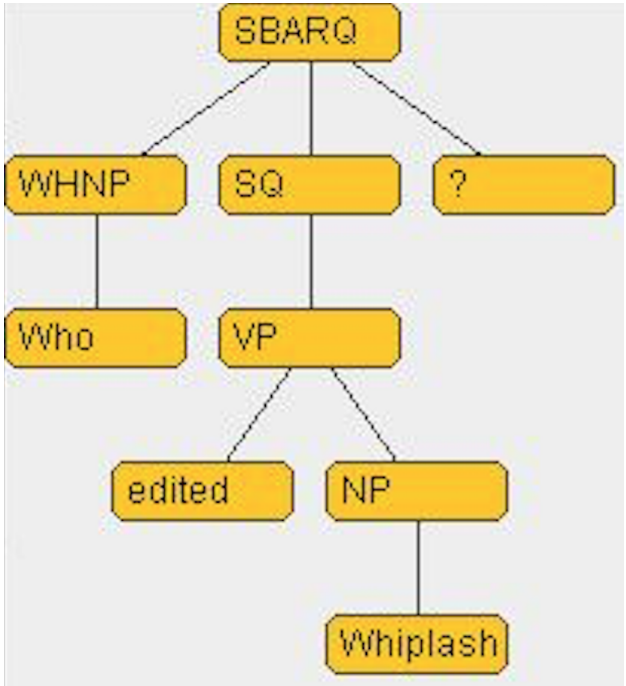
Con i seguenti due brevi esempi discuteremo le ultime sfaccettature del parser progettato, risolvendo le precedenti problematiche incontrate negli step I e II.

Si consideri, ad esempio, la domanda: *Who edited Whiplash?*

Prima di questo terzo step non avremmo potuto rispondere a questa domanda a causa del fatto che il termine *edited* non è presente come proprietà nella risorsa dbpedia di Whiplash rendendo così impossibile raggiungere la voce *editing*.
Attualmente invece, il sistema con **stemmer pipeline** e senza considerare il synonymer, risponde correttamente con *Tom Cross*, l'addetto al montaggio del film Whiplash.

Come funziona il parser 12/14

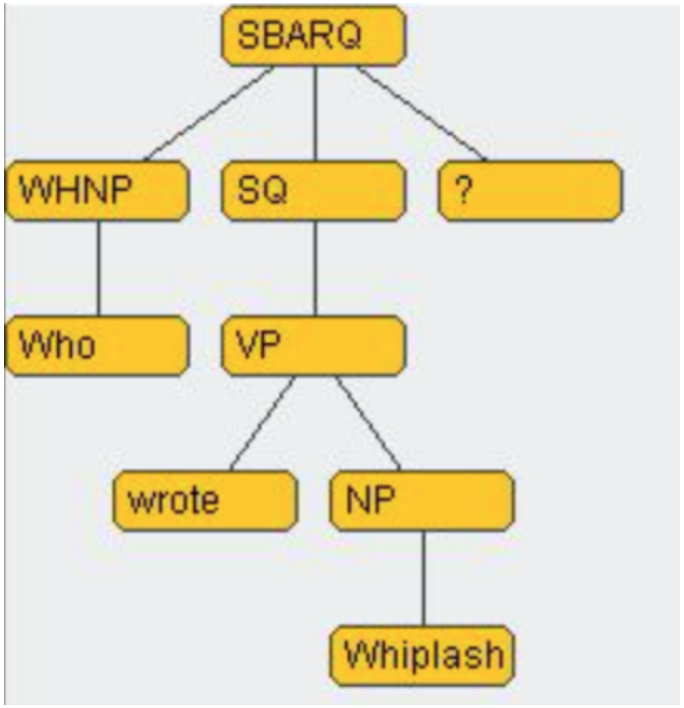
Il procedimento eseguito è analogo a quello precedentemente illustrato con la differenza che avviene un ultimo step prima della costruzione della query SPARQL: viene coinvolto lo *stemmer* che agisce sul predicato riconducendo il termine *edited* a *edit* (forma radice) rendendo raggiungibile la proprietà *editing*.



Come funziona il parser 13/14

In modo analogo, consideriamo adesso la domanda *Who wrote Whiplash?*

Prima di questo terzo step non avremmo potuto rispondere a questa domanda a causa del fatto che il verbo *wrote* non è presente come proprietà nella risorsa dbpedia di Whiplash rendendo così impossibile raggiungere la voce *writer*. Attualmente invece, il sistema con **lemmatizer pipeline** e senza considerare il synonymer, risponde correttamente con *Damien Chazelle*, lo sceneggiatore (e regista) del film Whiplash.



Come funziona il parser 14/14

Stavolta come ultimo step prima della costruzione della query SPARQL, invece dello stemmer, viene coinvolto il *lemmatizer* che agisce sul predicato riconducendo il termine *wrote* a *write* (forma canonica) rendendo raggiungibile la proprietà *writer*.

Ne consegue naturalmente che bisogna scegliere accuratamente quale pipeline coinvolgere per ottenere la risposta corretta.

Resoconto del terzo step 1/2

Vantaggi del terzo step:

Rispondiamo finalmente ad una gran vastità di domande:

- *Who directed Whiplash?* ⇒ lemmatizer pipeline, senza synonymer ⇒ *directed*→*direct*→*director* ⇒ *Terrence Malick*;
- *Who composed the music of Demolition Man?* ⇒ simple pipeline, senza synonymer ⇒ *music*→*musicComposer* ⇒ *Elliot Goldenthal*;
- *Who is the main star of Demolition Man?* ⇒ simple pipeline, con synonymer ⇒ *star*→*starring* ⇒ *Billy Bob Thornton* (condannato dall'assenza di un riferimento sui ruoli ricoperti dagli attori);
- *Who made the editing of Interstellar?* ⇒ stemmer pipeline, senza synonymer ⇒ *editing*→*edit*→*editor* ⇒ *Lee Smith*;
- *Who produced Aliens?* ⇒ lemmatizer pipeline, senza synonymer ⇒ *produced*→*produce*→*producer* ⇒ *Gale Anne Hurd*;
- *Ulteriori domande particolari trattate in una maniera specifica: elencare e quantificare i film di un regista. Gli esempi sono mostrati nelle slide successive.*

Resoconto del terzo step 2/2

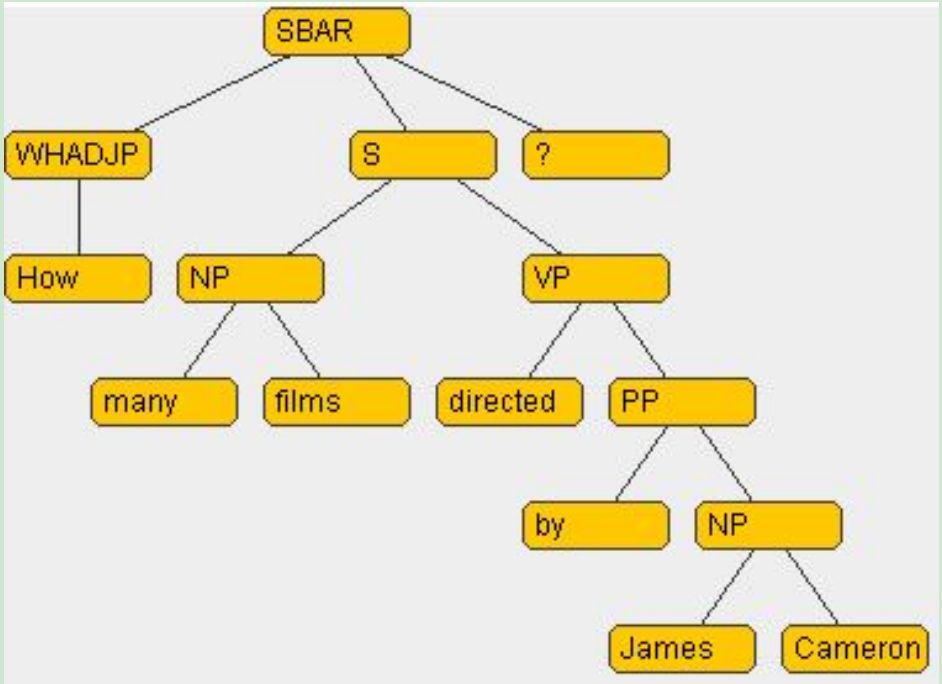
Svantaggi e possibili sviluppi del terzo step:

- Saranno discussi in sede d'esame alcuni esempi posti al termine di questa presentazione.
- *Sviluppo futuro:* la domanda *How much did gained Ben Hur?* non trova risposta poiché non viene riconosciuto il quantificatore *how much*. Ciò che ha richiesto l'utente, ossia il guadagno di Ben Hur al botteghino, è una proprietà esistente (`gross`) raggiungibile impostando diversamente la domanda o aggiungendo il pattern `<how much did #P #O?>` o riconoscendo nodi di tipo `WHADJP` nel Penn Tree.
- *Applicazione aggiuntiva:* parser ad-hoc per rispondere a domande relative ai premi ambiti nell'ambiente cinematografico, scoprendo così quando, quali e quanti premi sono stati vinti dai film alle cerimonie annuali.

Casi particolari considerati dal QA 1/2

Quantificare i film di un regista:

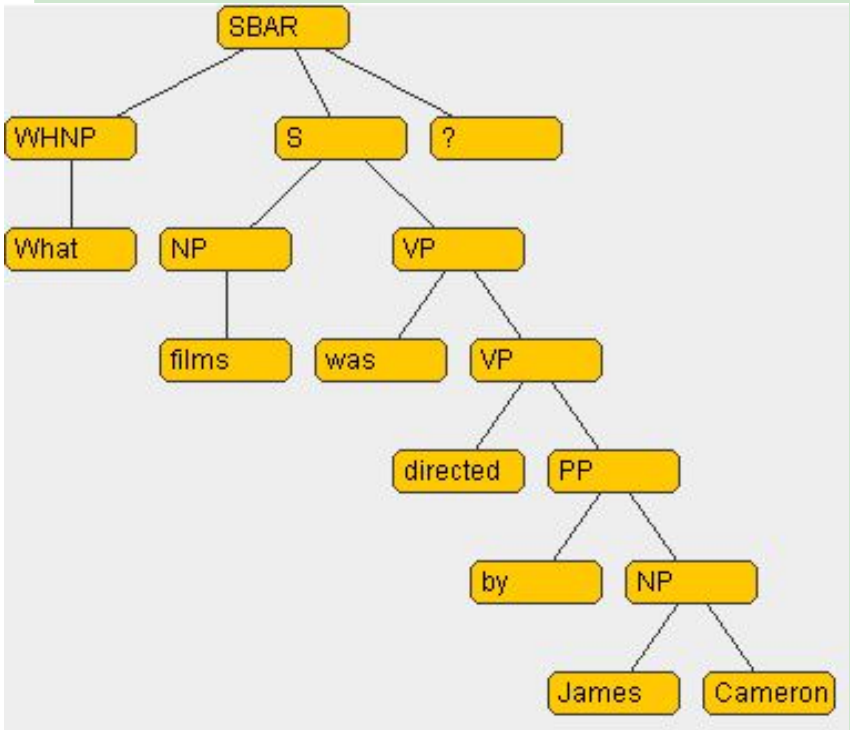
How many films directed by James Cameron?



Casi particolari considerati dal QA 2/2

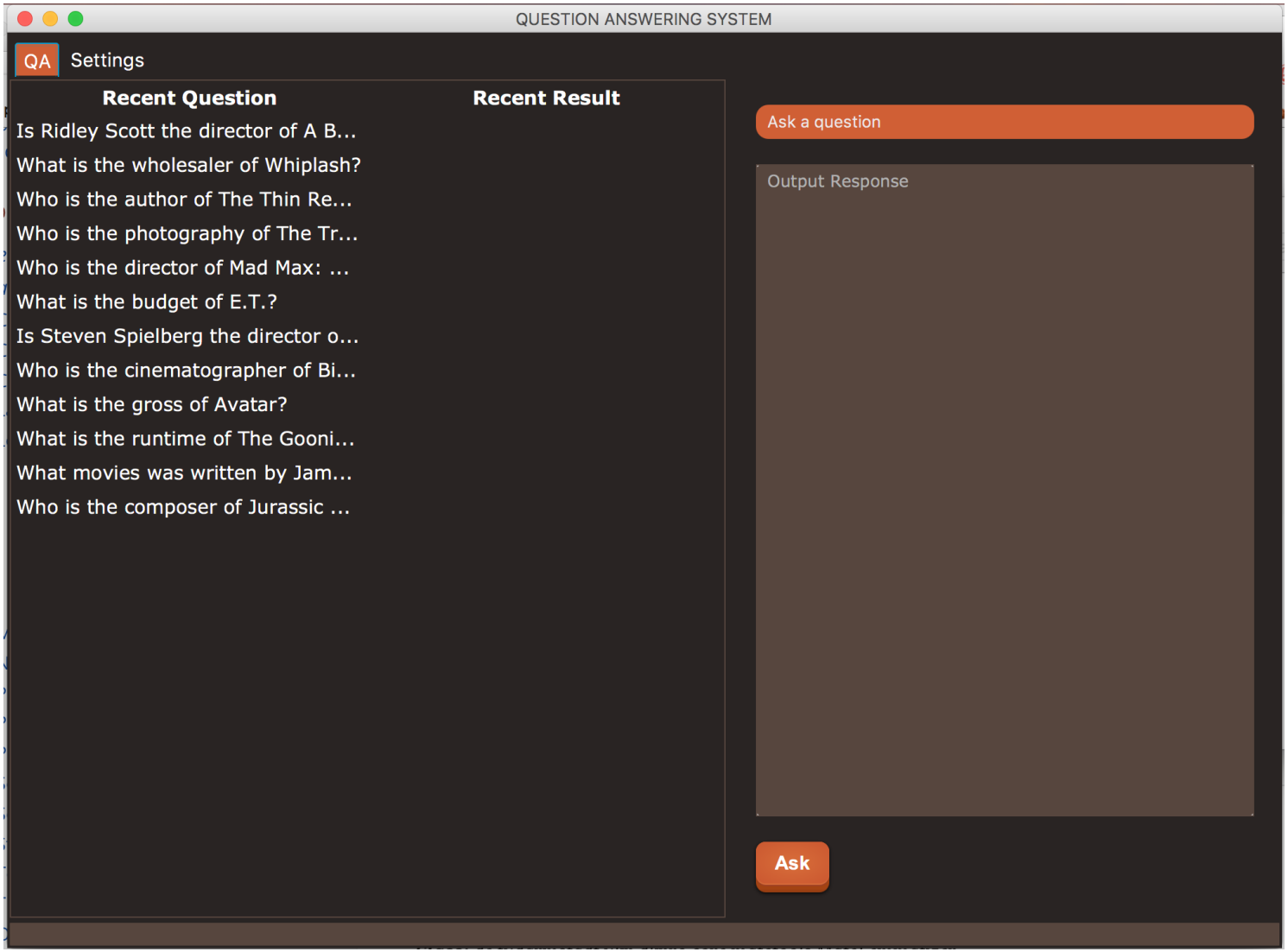
Elencare tutti i film di un regista:

What films was directed by James Cameron?

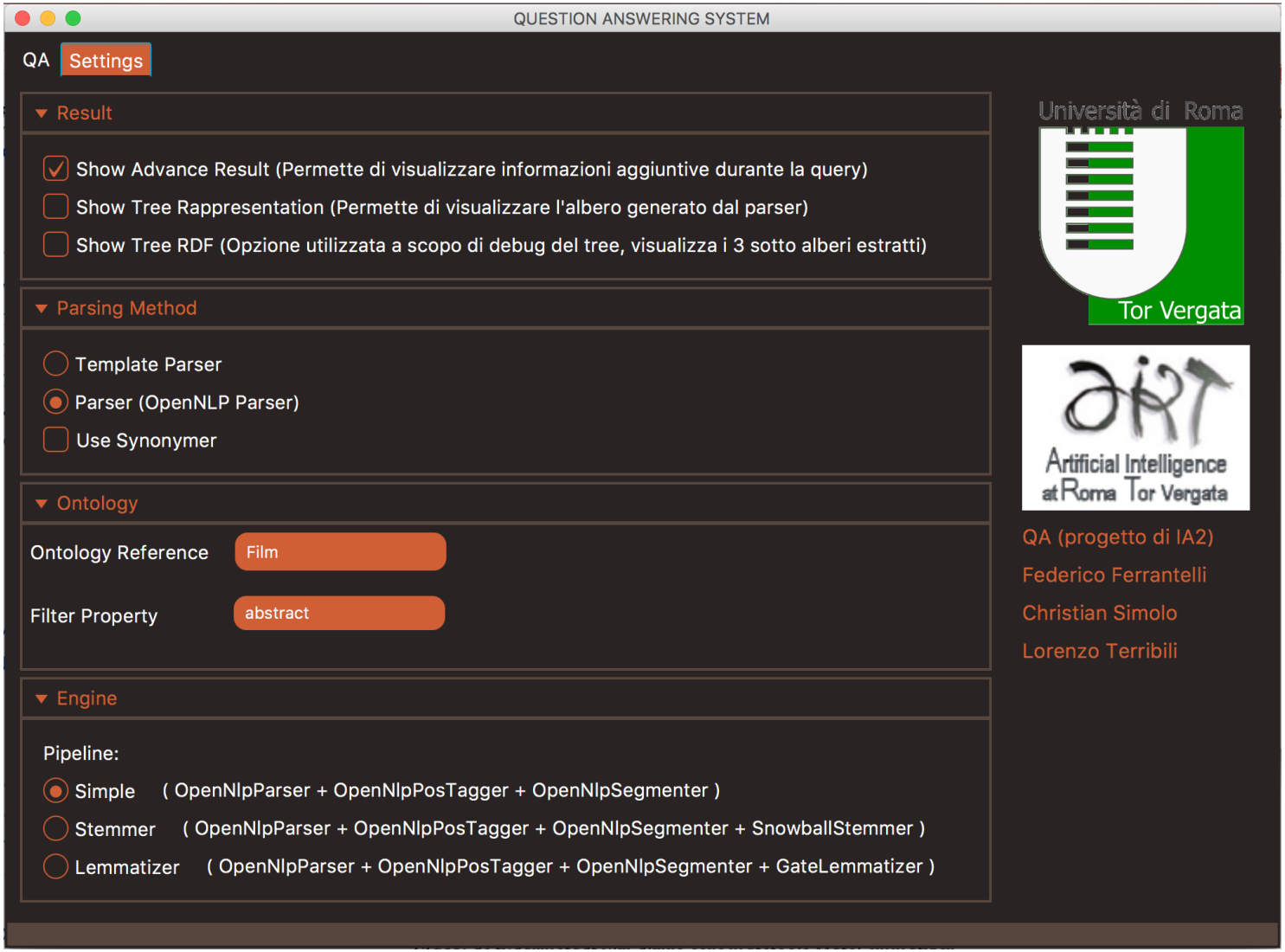


SCREENSHOT DEL QA
&
ESEMPI DI ALBERI
PROBLEMATICI

Conclusione 1/2

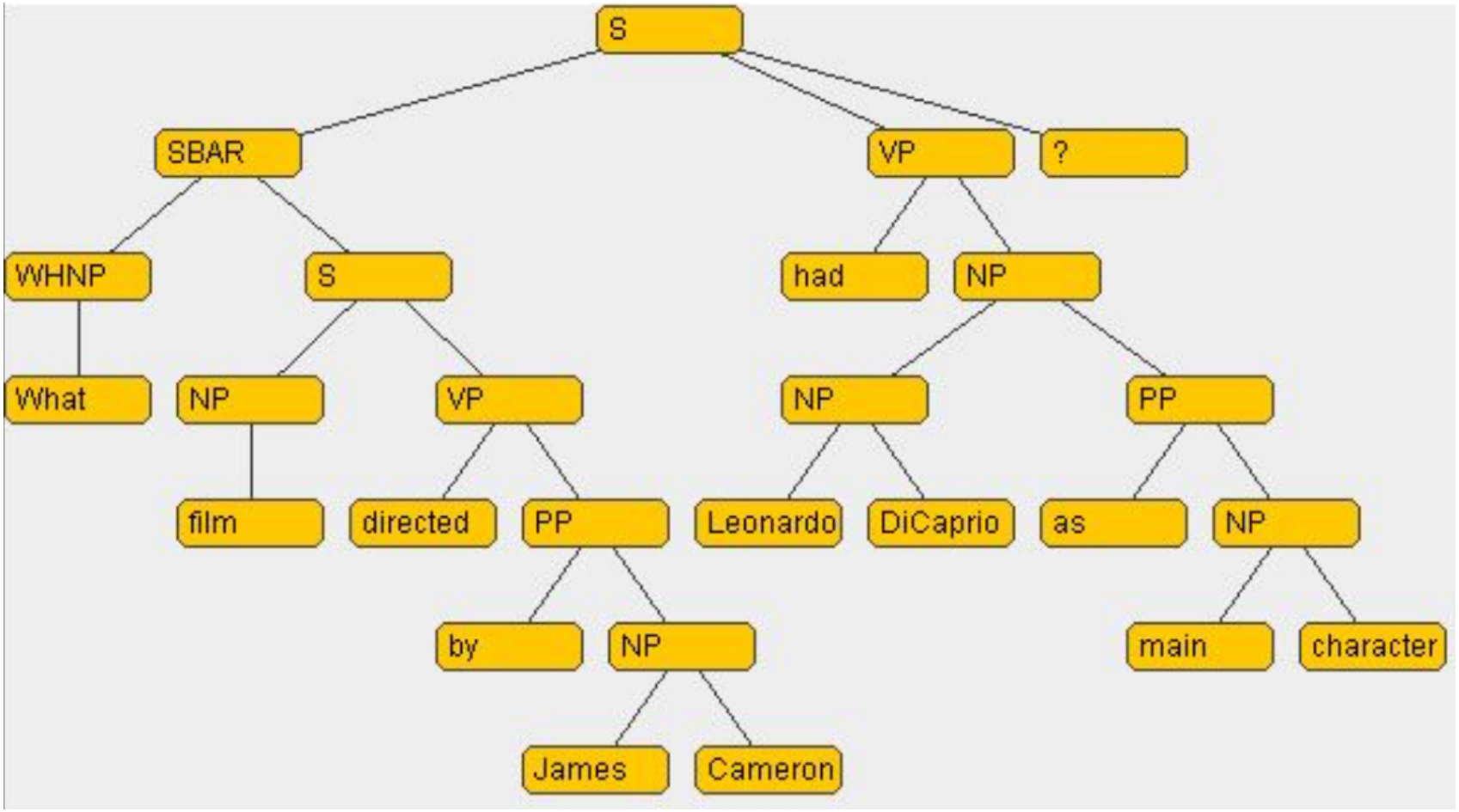


Conclusione 2/2



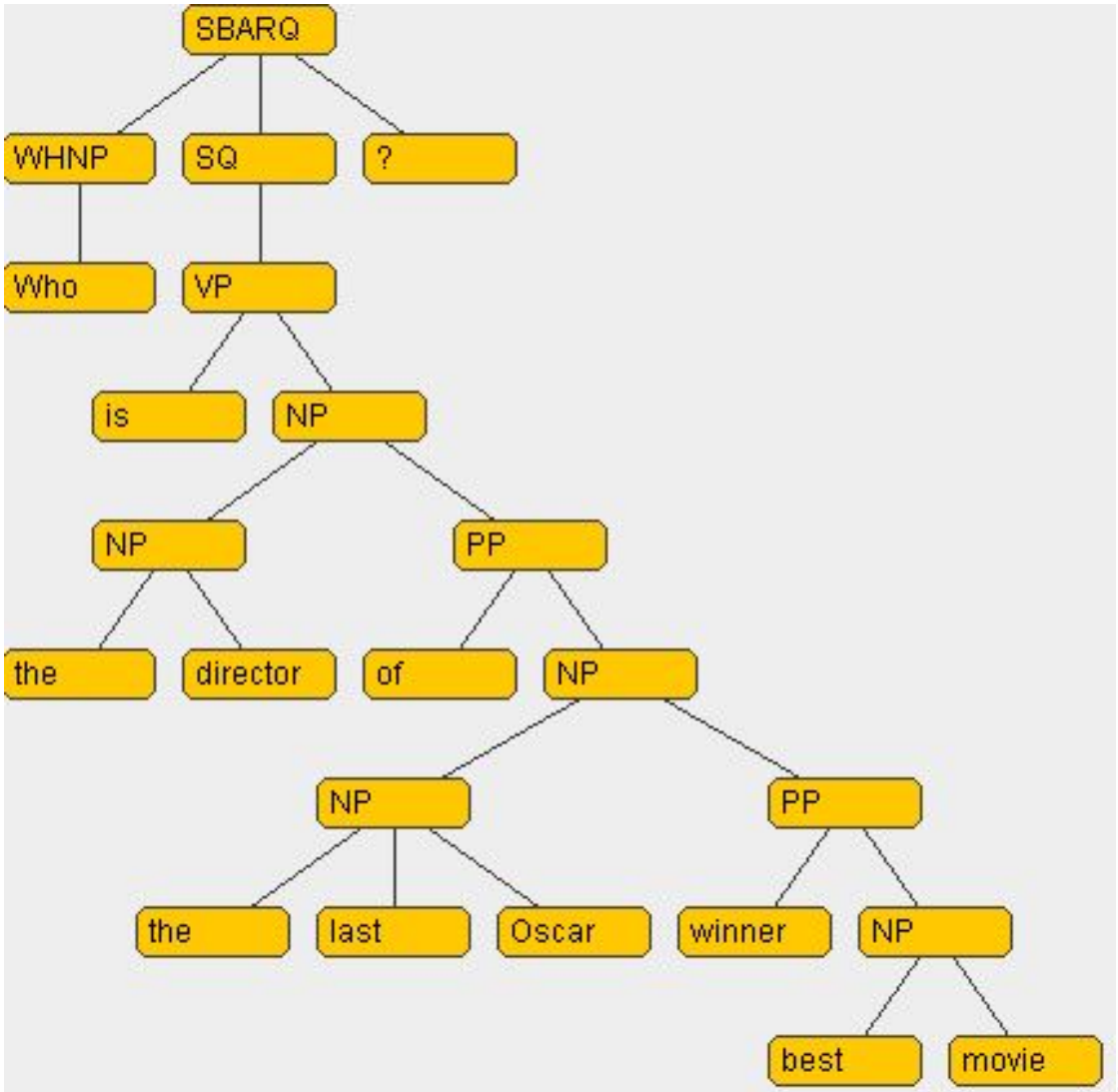
Esempio 1

What film directed by James Cameron had Leonardo DiCaprio as main character?



Esempio 2

Who is the director of the last Oscar winner best movie?



Esempio 3

Who is the director that won the last Oscar?