Design and implementation of an ontology based algorithm for Trend Mining

Di Nicola Marco
Piccinelli Flavio

The trend mining

" Trend mining is the extraction of implicit, previously unknown and potentially useful knowledge from time-ordered patterns (of text or data).

The trend mining techniques can be used for capturing trend in order to support user in providing previously unknown information and knowledge about the general development in users field of interests."

International Conference on Knowledge Engineering and Ontology Development 2012.

Index

- A Conceptual view.
  - Topic model.
  - Trend model.
  - Algorithm(s).
- The Implementation(s).
  - Concepts as Ontologies.
  - The Application model.
  - Offline Java application.
  - Web application.

Mine a Trend

Define this function:

$$mineTrend : Topic \times Context \times Corpus \longrightarrow Trend$$

Taking as input a topic, a context and a corpus, this function extracts the trend of the topic through time.

Let's define the corpus:

$$Corpus := \{(d_0, t_0), \ldots, (d_n, t_n)\}, n \in \mathbb{N}$$

Where d_i is a document and t_i is its associated timestamp.

Topic and Context

Topic and context are defined as couples

$$Topic, Context := \langle C, R \rangle$$

Where

- $C := C_{co} \cup E \cup P \cup L$ is a set of concepts, including descriptions of events, people and locations.
- $R := \{r_0, \ldots, r_n\}, n \in \mathbb{N} \wedge r : C \times C$ is a set of relations amongst concepts.

Moreover, depending on the model, one could define them as to:

$$Topic \cap Context = \emptyset$$

Trend

Trend definition:

$$Trend := \langle T, C_{tx}, TW, A \rangle$$

Where:

- T is a Topic
- C_{tx} is a Context
- $TW := \{tw_0, \ldots, tw_n\}, n \in \mathbb{N} \wedge tw \; is \; a \; time \; window$
- $A : TW \rightarrow \mathbb{R}$

A returns a numeric value representing the importance of the topic at the given time window.

The algorithm

```
MineTrend(topic, context, corpus) {
  corpus.sortByTime();
  for doc ∈ corpus {
    words = doc.getWords();
    matches = getMatches(words, topic, context);
    for m ∈ matches {
      amp = amp + termDensity(m) ×
            conceptWeight(m, topic, context);
    }
    trend.amplitude.add(doc.timestamp, amp);
  }
  trend.amplitude.normalize(timeWindow);
  return trend;
}
```

Term Density

$$termDensity : term \rightarrow \mathbb{R}$$

A notion of how well a term is distributed inside a document.
It uses internally a list of naturals:

$$P_t := (wc_0, \ldots, wc_k)$$

wc_i is the number of occurrences of the term t in the $i-th$ phrase of the document.

getTermDensity(t) {
  $pf = \#\ \{wc \in P_t \mid wc > 0\}$;
  $tf = \sum_{wc \in P_t} wc$;
  $sd = P_t$.**getStandardDeviation()**;
  $td = \frac{tf \times pf}{sd + 1}$;
  return td;
}

Concept Weight

$$conceptWeight : concept \times topic \times context \longrightarrow \mathbb{R}$$

Depending on the trend mining model one uses, the implemented algorithm changes.

Text based model algorithm:

```
conceptWeight(concept, topic, context) {
  if concept ∈ topic
    return topic.getWeight(concept) × TOPIC COEFFICIENT;
  else if concept ∈ context
    return context.getWeight(concept) × CONTEXT COEFFICIENT;
}
```

Where context and topic associate a static weight to each of their concepts.

Text based model

Based on a static weight $\in [0, 1]$ associated to one or more keywords describing a concept.

e.g.
Complementi di Basi di Dati {
"Complementi di Basi di Dati" $\mapsto$ 1.0
"CBD" $\mapsto$ 0.7
}

The sum of Topic weights is also multiplied by a coefficient, in order to give it more importance than the Context weights.

Note that, using this model, one can easily evaluate a Topic inside different Contexts (e.g. Samsung Electronics in the televisions or smartphones market fields).

Concept Weight (2)

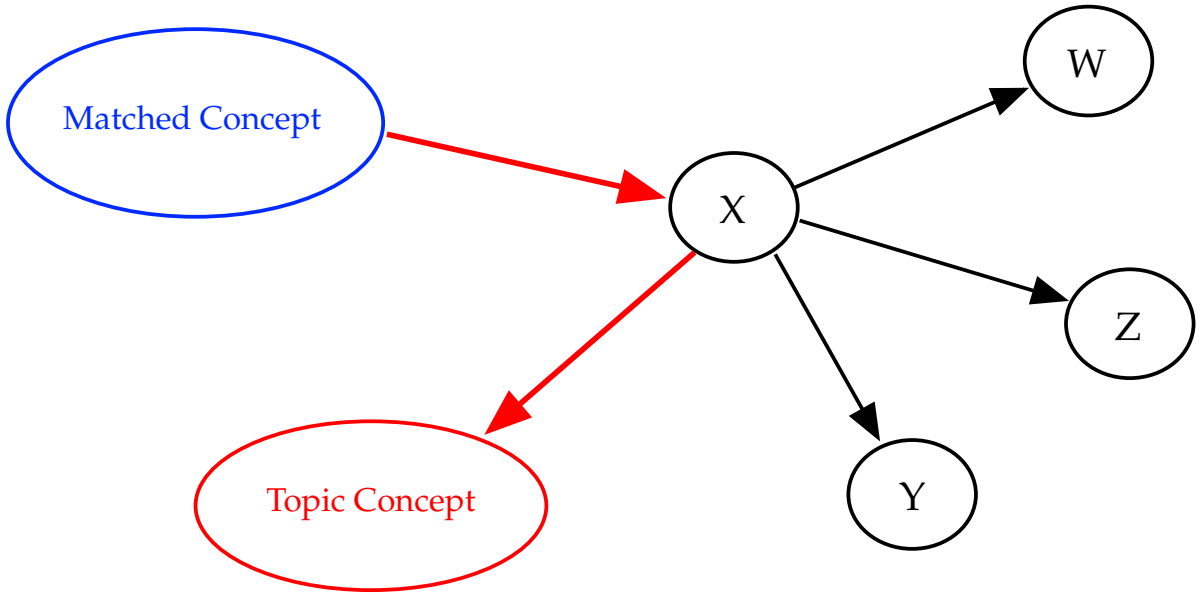
Knowledge (or relations) based model algorithm:

conceptWeight(concept, topic, context) {
  conceptDistance = **`getConceptDistance(concept, topic, 0)`**;
  return $\frac{1}{2^{conceptDistance}}$;
}

getConceptDistance(concept, targetConcepts, distance) {
  if concept $\in$ *targetConcepts* $\vee$ *distance* == *MAX DISTANCE*
    return distance;
  else {
    minDistance = *MAX DISTANCE;*
    for relatedConcept $\in$ *concept.relations()*
      minDistance = *min(getConceptDistance(relatedConcept,*
      targetConcepts, distance + 1), minDistance);
    return minDistance;
  }
}

Knowledge based model

Based on the relations graph of the concept, using the depth needed to reach a Topic concept as a measure of the concept importance.



A distance of 2

Note: intermediate nodes don't have to be necessarily in the Context definition (more on this later).

Comparing two models

Text based

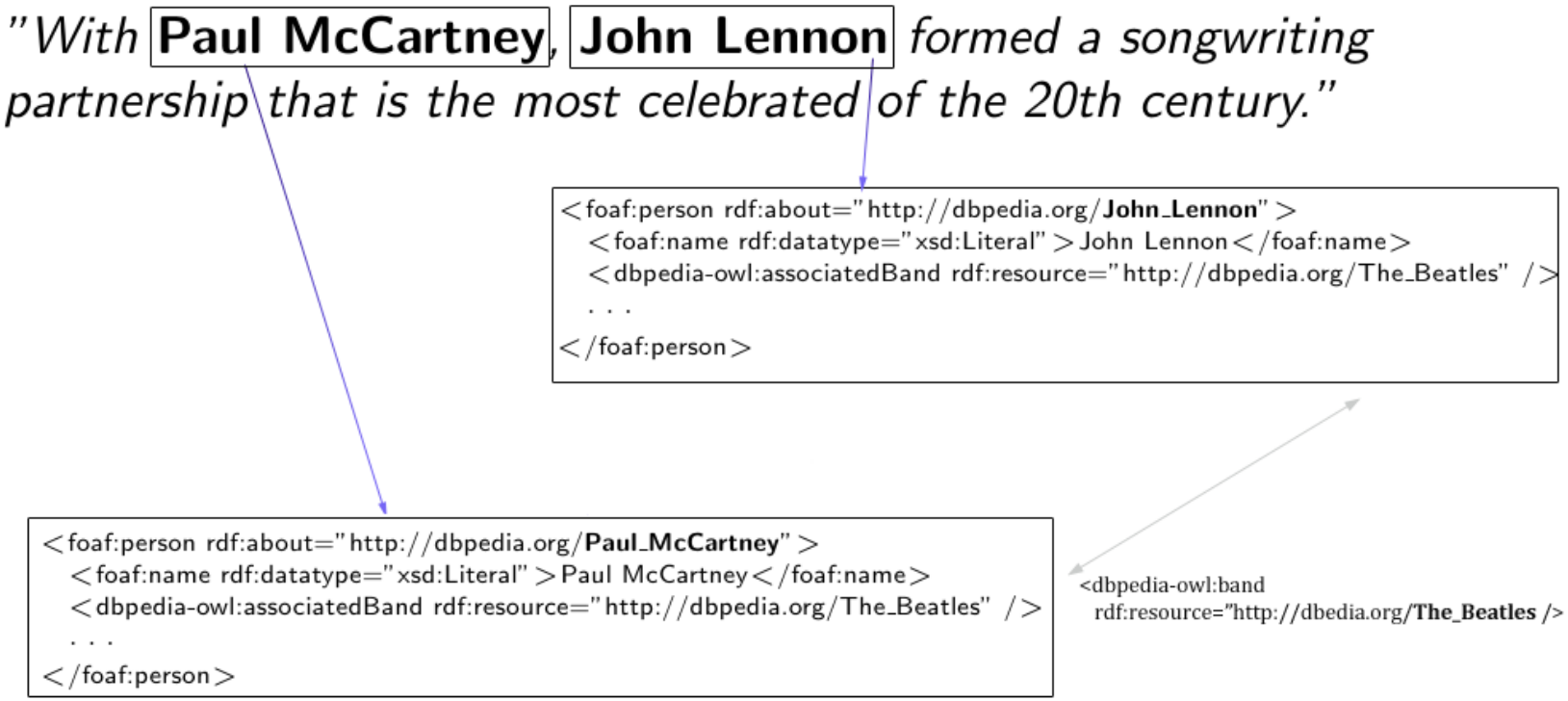
- √ Easy (also computationally: check an Hash Map for the weight).
- √ Allows a better separation between Topic and different Contexts.
- × Must specify keywords for each concept in Topic and/or Context.

Knowledge Based

- √ Scalability: easy to connect relation graphs between concepts.
- √ Depending on the implementation, can use external concepts (not defined by user).
- × Computationally harder: traverse a graph.

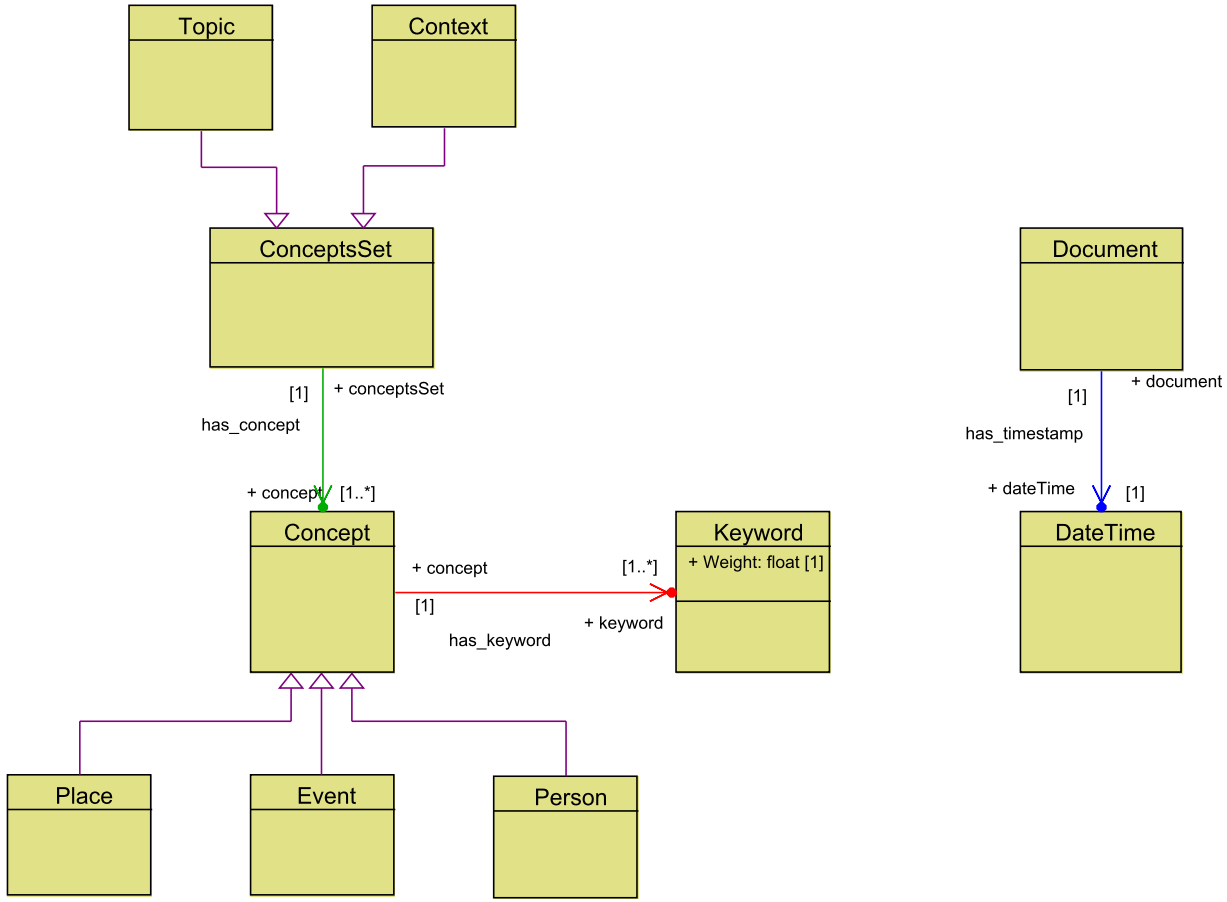
Concepts as Ontologies

Exploit the web 3.0 and the idea of linked data to describe concepts and their relations.



Topic ontology schema

http://mdinicol.web.cs.unibo.it/ontologies/topic-model.owl



Topic example

A more concise Turtle syntax:

@prefix topic-model: <http://mdinicol.web.cs.unibo.it/ontologies/topic-model.owl#>
@prefix : <http://mysite.it/mytopic.owl#>

:**MyTopic** a topic-model:Topic
; topic-model:has_concept :MyConcept
; topic-model:has_concept :MyPerson .

:**MyConcept** a topic-model:Event
; rdfs:label "MWC 2013"
; topic-model:has_keyword :MyConceptKeyword .

:**MyConceptKeyword** a topic-model:Keyword
; topic-model:word: "MWC 2013"^^xsd:Literal
; topic-model:weight "0.9"^^xsd:float .

:**MyPerson** a topic-model:Person
; **owl:sameAs** <**http://anothersite.com/ExternalPerson**> .

Using the built-in OWL property owl:sameAs makes it possible to use external entities as Topic concepts.

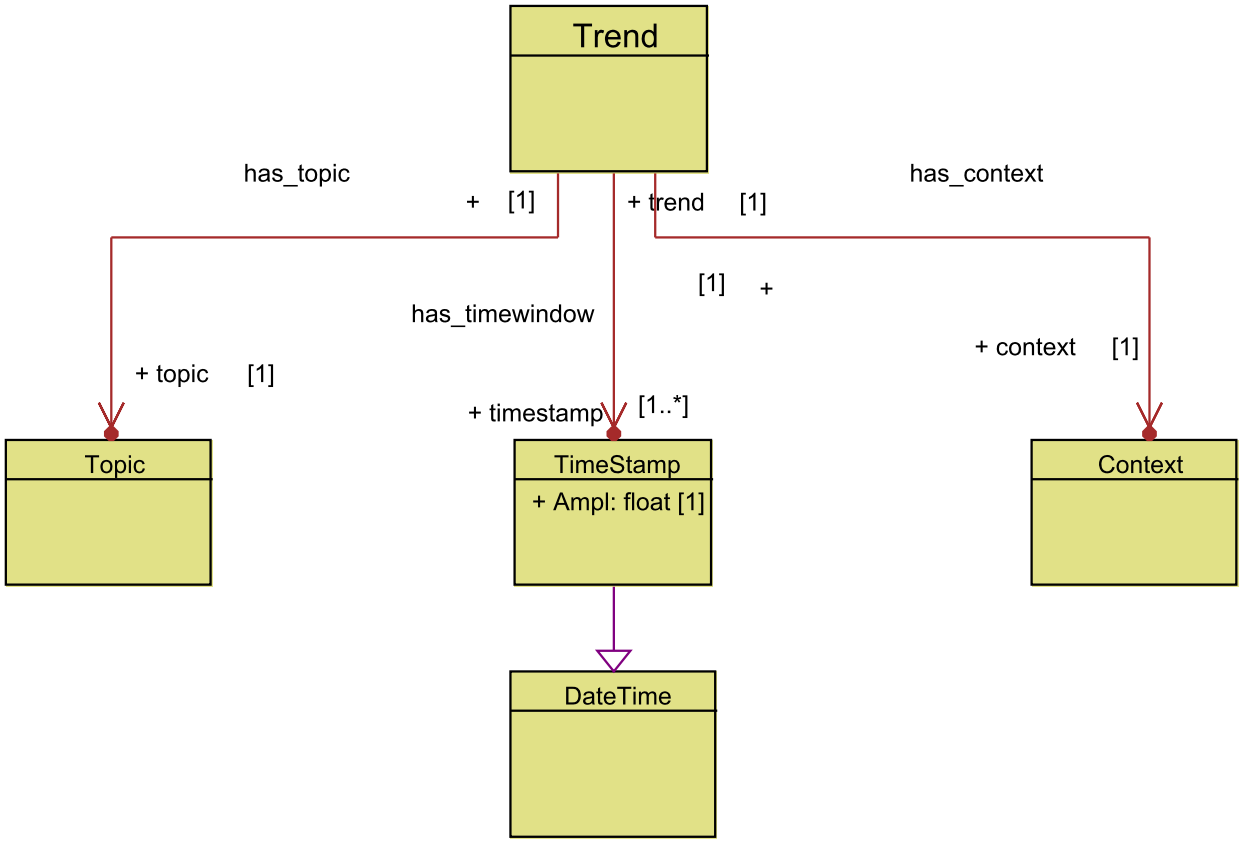
Document example

:**myDoc** a topic-model:Document
; topic-model:content "The Fox and the Hound is a 1981 American animated film loosely based on the Daniel P. Mannix novel of the same name, produced by Walt Disney Productions"^^xsd:string
; topic-model:has_timestamp :myDocTimestamp .

:**myDocTimestamp** a topic-model:DateTime
; topic-model:date "2002-12-05"^^xsd:date
; topic-model:time "22:05"^^xsd:time .

Trend ontology schema

http://mdinicol.web.cs.unibo.it/ontologies/trend-model.owl



A Trend example

@prefix topic-model: <http://mdinicol.web.cs.unibo.it/ontologies/topic-model.owl#>
@prefix trend-model: <http://mdinicol.web.cs.unibo.it/ontologies/trend-model.owl#>
@prefix : <http://datacenter.it/mytrend.owl#>

:**MyTrend** a trend-model:Trend
; trend-model:has_topic <**http://mysite.it/mytopic.owl#MyTopic**>
; trend-model:has_context <**http://mysite.it/mytopic.owl#MyContext**>
; trend-model:has_timewindow :MyTrendTimestamp0
; trend-model:has_timewindow :MyTrendTimestamp1 .

:**MyTrendTimestamp0** a trend-model:Timestamp
; topic-model:date "2002-12-05"^^xsd:date
; topic-model:time "22:05"^^xsd:time .
; trend-model:amplitude "3.5"^^xsd:float .

:**MyTrendTimestamp1** a trend-model:Timestamp
; topic-model:date "2004-08-12"^^xsd:date
; trend-model:amplitude "1.2"^^xsd:float .

Concepts as Ontologies: overview

Pros:

- √ A standard format which uses a wide set of vocabularies.
- √ Possibility to connect to remote resources (concepts) to gain precision.
- √ Results on Trend mining can be easily shared and enhanced across the Web.

Cons:

- × Traversing the relations graph of a concept implies multiple connections to external web sources: slower.
- × Associations terms-entities can be ambivalent.

The Application(s)

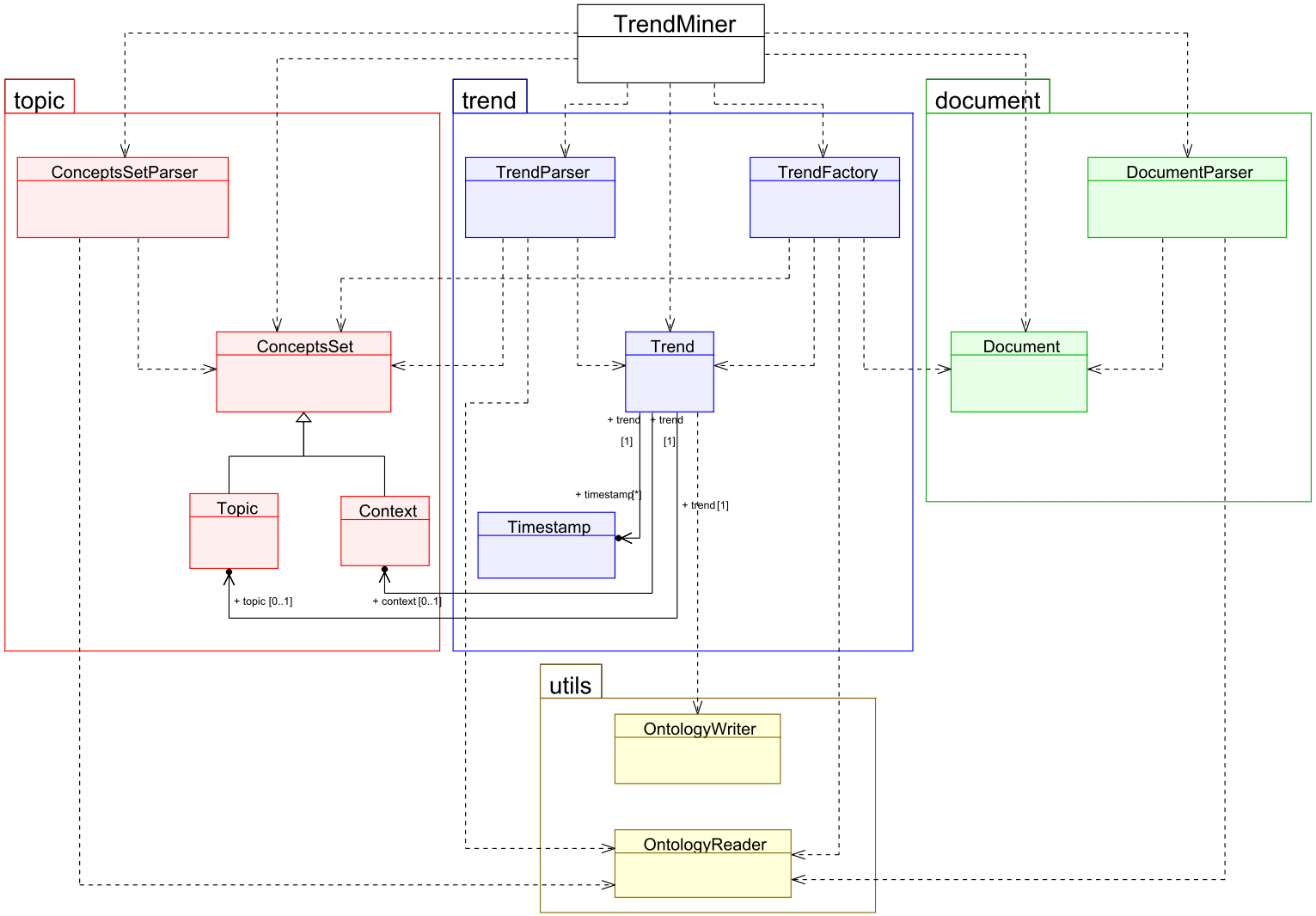
Features:

- Parses files on the web and creates data structures (classes) describing Topic, Context and Documents.
- Implements the algorithm described earlier, with it's variants, to compute a Trend description.
- Saves the Trend to file as multiple formats (RDF/XML, Turtle, ...) and optionally plots a chart to describe it graphically.
- Reads a pre-computed Trend structure from file.

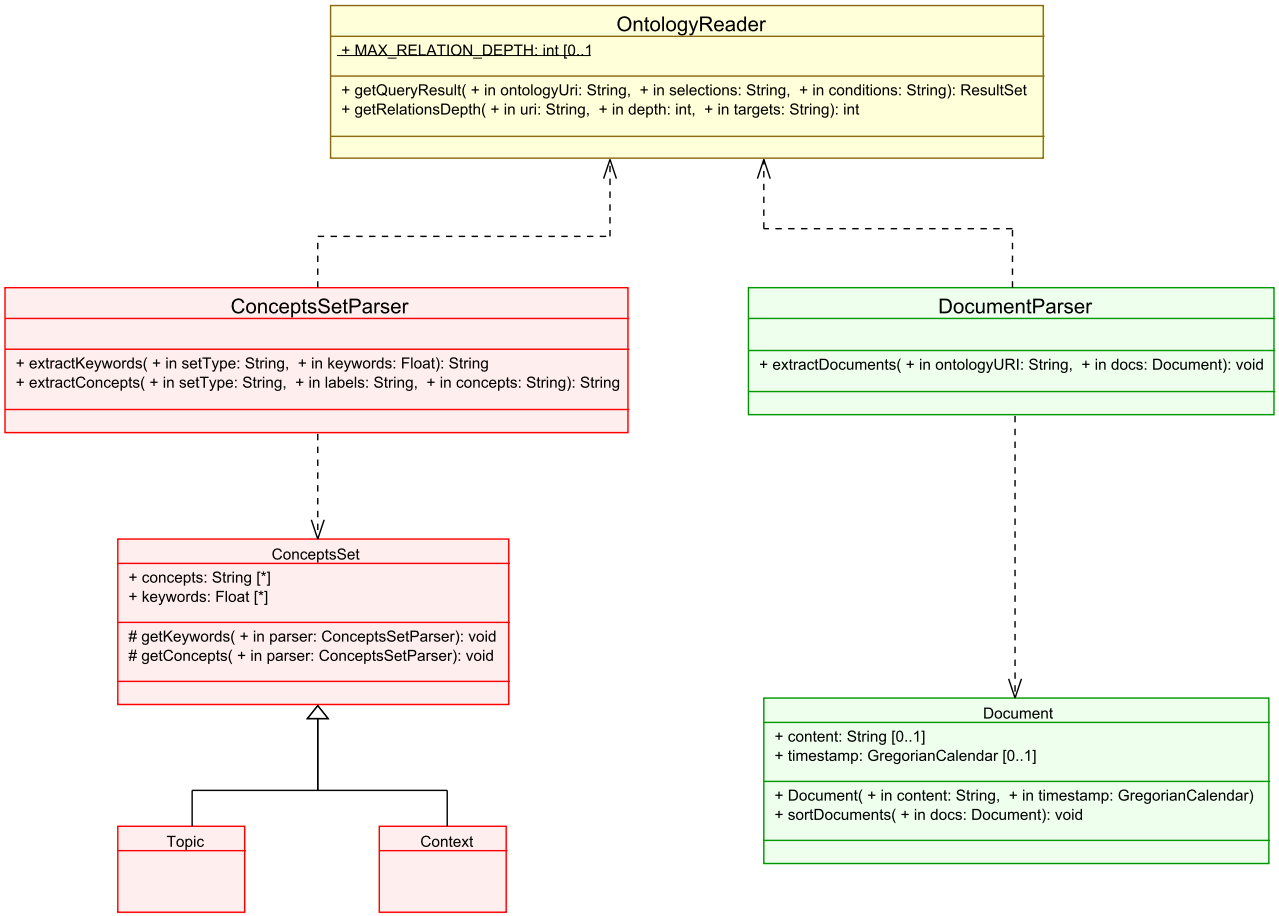
Two versions:

- Java application: allows to compute trends offline.
- Web application: stores the ontologies schemas and publishes results on the web.

Application's UML model

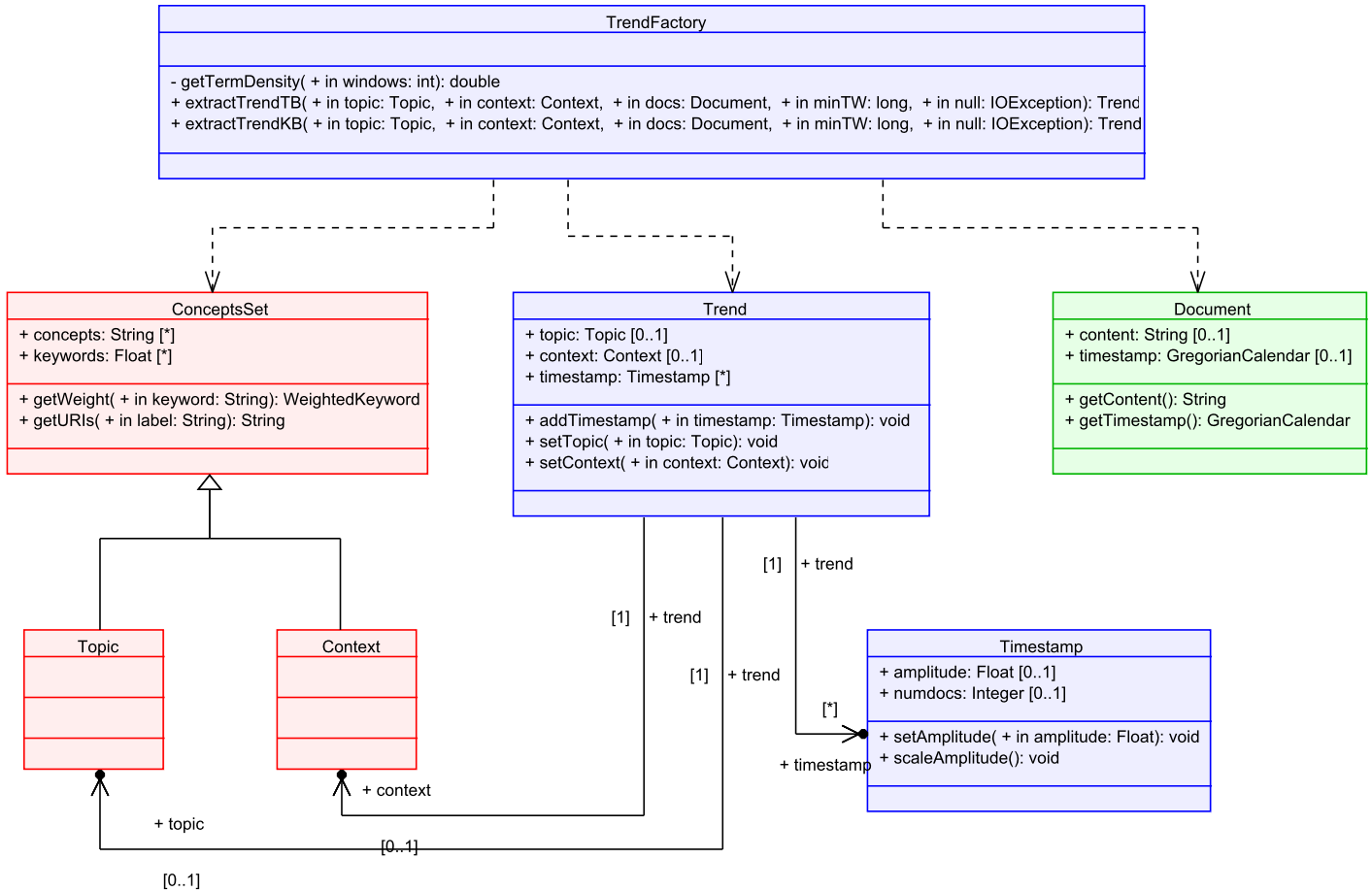


Ontology parsing

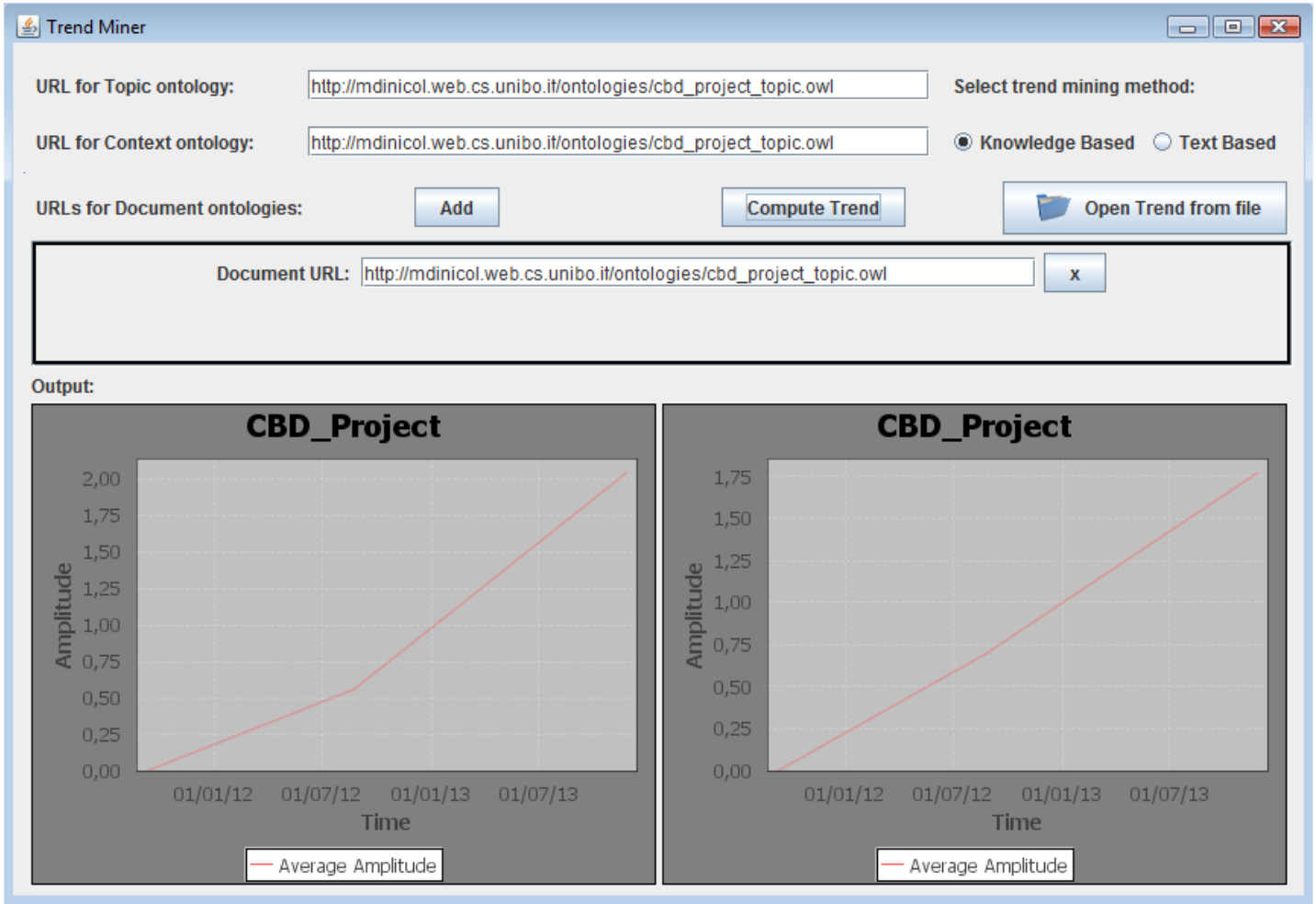


A pool of ontologies models is kept in memory, in order to reduce the number of TCP connections.

Trend extraction

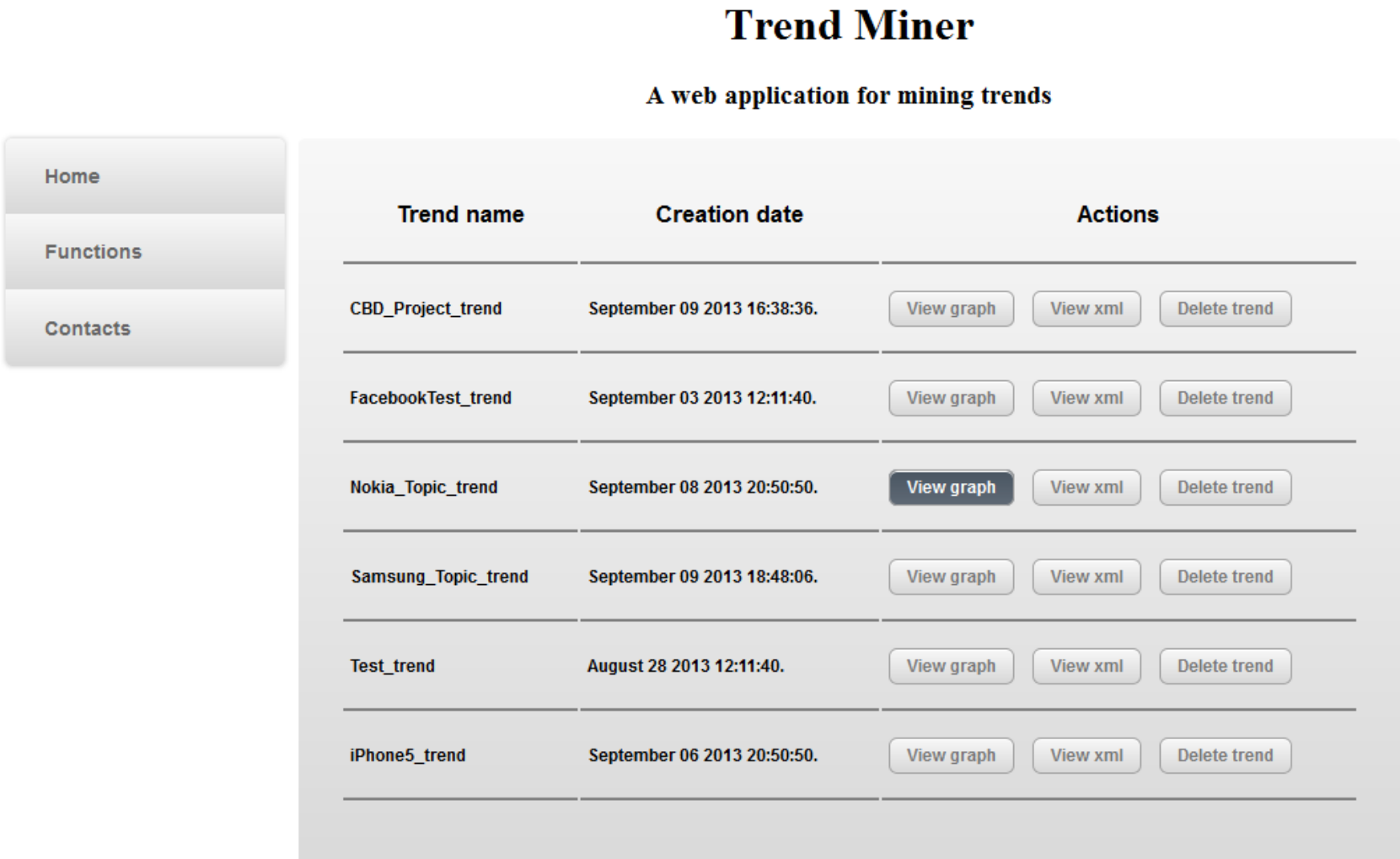


Java application



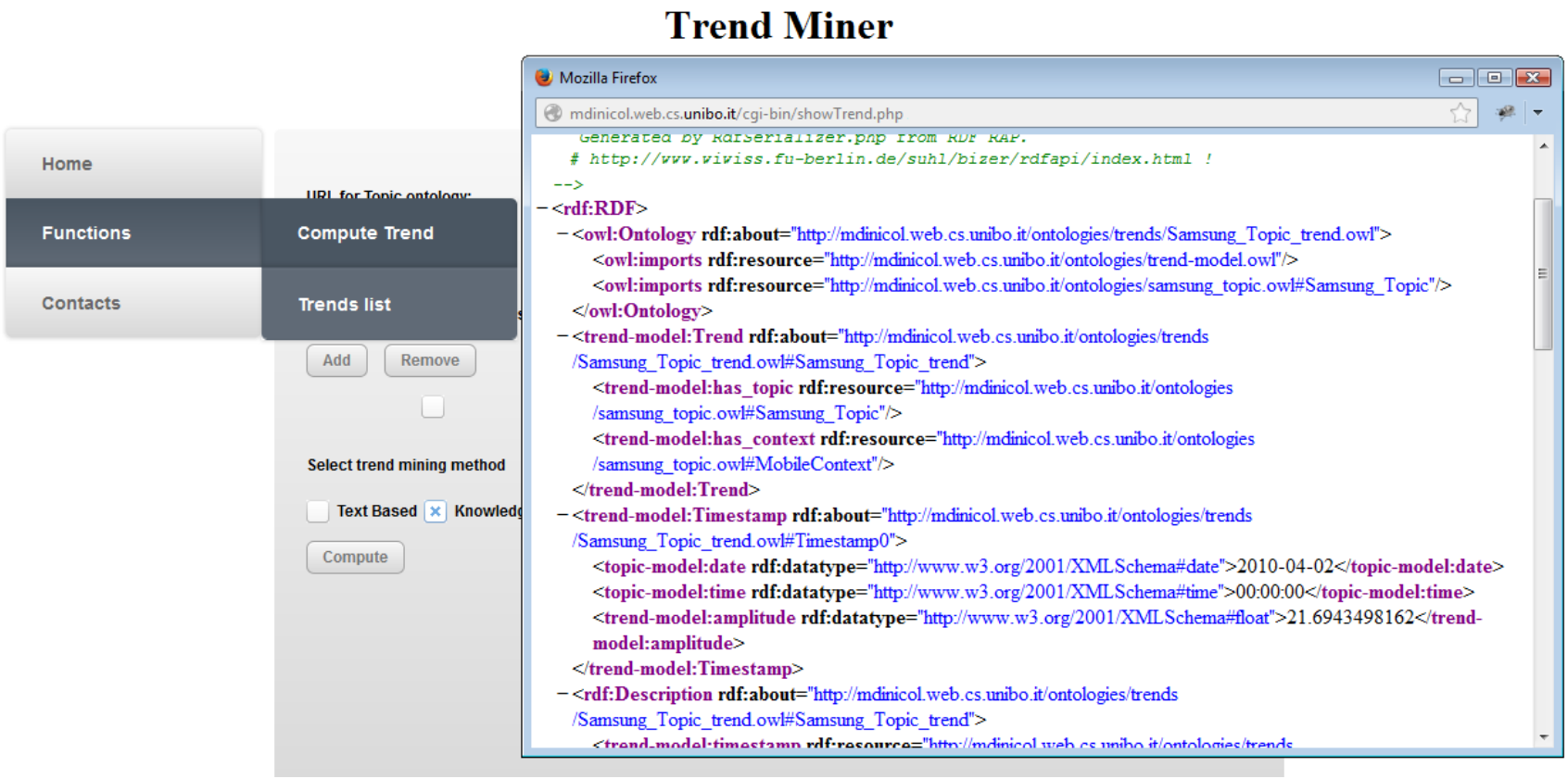
Web Application

Accessible at: http://mdinicol.web.cs.unibo.it/



Web application (2)

Generated results are placed on the web and made accessible to users or other applications.



An use case: Mobile market

Context: **Mobile market** := {
  Companies (Nokia, Samsung, Apple, LG, ...)
  Devices (Galaxy S4, iPhone 5, Wildfire S, ...)
  Features (Tethering, NFC, Wi-Fi Direct, ...)
  Relations {
    produces (Samsung $\mapsto$ Galaxy S4 , ...)
    produced_by (Wildfire S $\mapsto$ HTC , ...)
    has_feature (Galaxy S4 $\mapsto$ NFC , ...)
  }
}

Topic: { **Samsung** }

Informations about mobile devices gathered by scripts reading web pages and linked together into an ontology:
http://mdinicol.web.cs.unibo.it/ontologies/samsung_topic.owl

Mobile market - Corpus

Corpus: Selection of twenty documents depicting the state of the art of mobile devices through the last four years.

Documents are grouped together in **one month** time windows.

Documents were extracted from different sources:
news portals, generic reviews on mobile market state of the art.

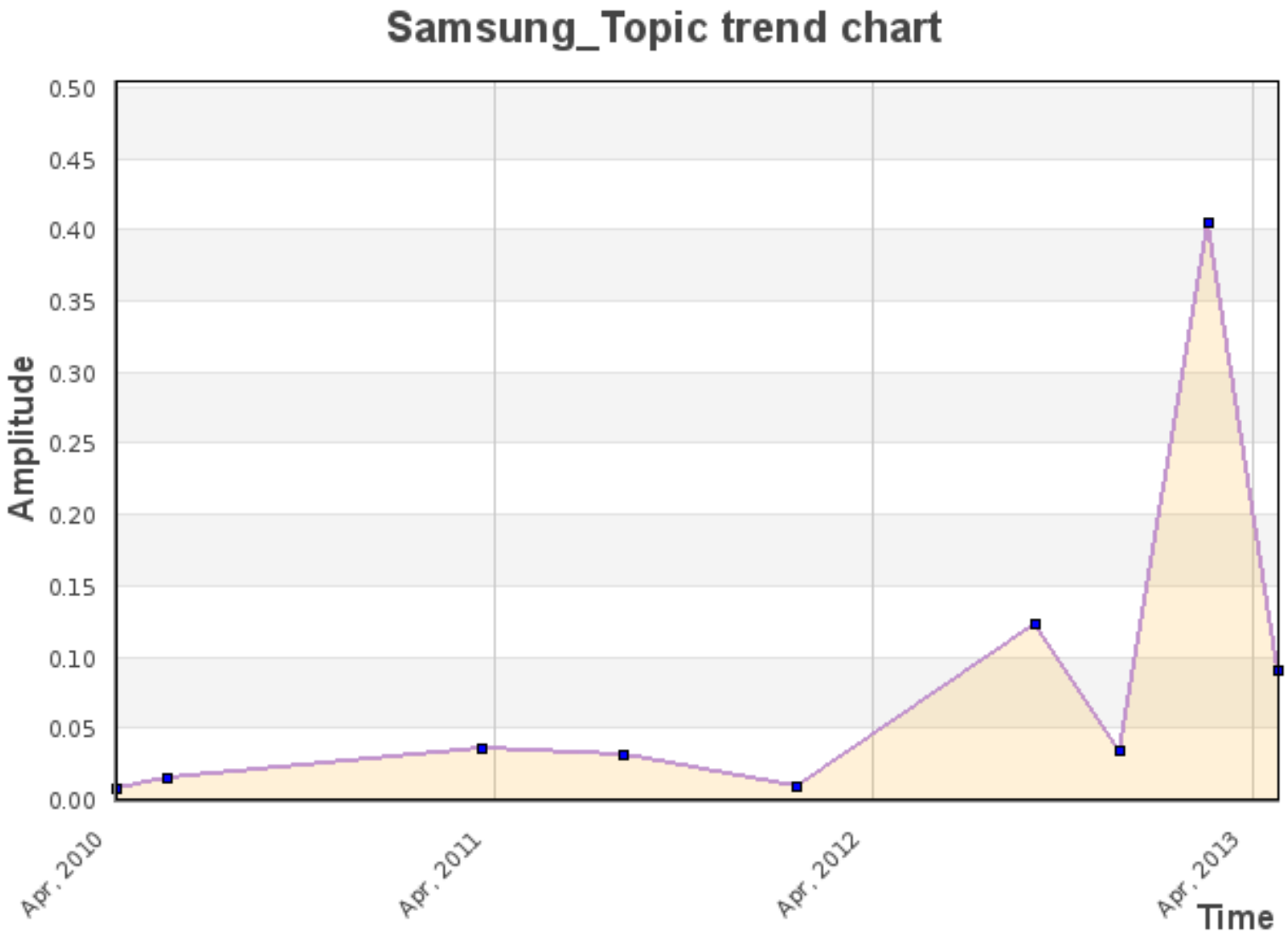


Chart generated using the knowledge based model

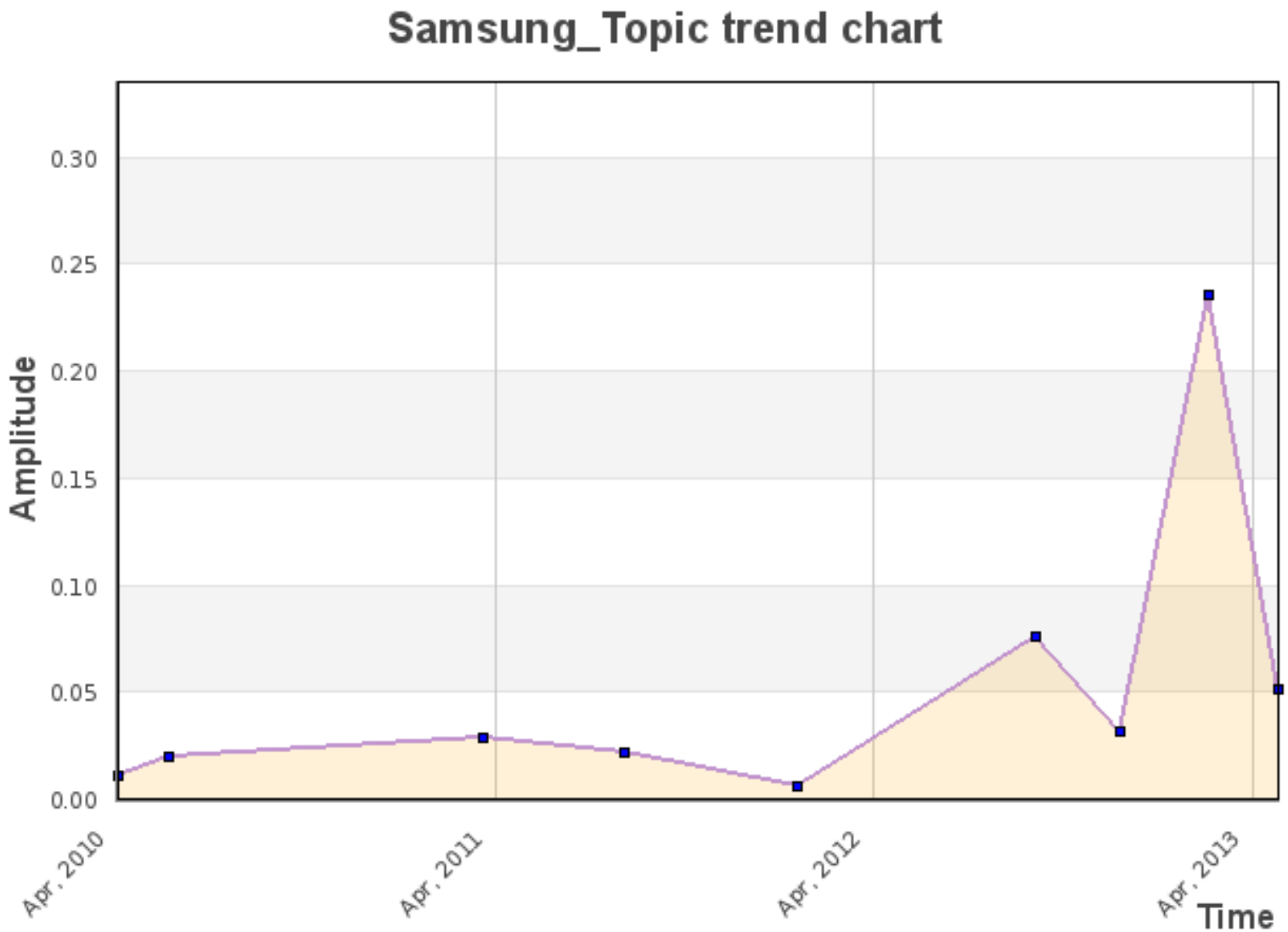


Chart generated using the text based model

Future developements

- Further tests using a more significant corpus.
- Keep developing the web application.
- Implement a "total" knowledge based algorithm: contrast the ambivalence of natural languages by using pre-assigned ontological entities.