Chapter 6
Shared Ontology for Knowledge Management

Atanas Kiryakov, Borislav Popov, Ilian Kitchukov, and Krasimir Angelov

Abstract This chapter focuses on semantic searching at web scale. The solution presented takes advantage of the specific strengths of semantic repositories and the raw power of relational databases, the latter having been developed over decades and capable of handling efficiently large volumes of data with fixed structure (which is the case with the occurrence statistics) and the former allowing for inference and querying on top of formal knowledge. The interactive faceted search capability described is a demonstration how an approach based on these two technologies is more powerful and efficient for certain tasks as compared to traditional search engines.

The contributions of the chapter can be summarized as follows:

- novel indexing schema for semantic search, based on entity occurrence;
- description of a scalable implementation of such indexing;
- advanced faceted search interface, based on co-occurrence.

6.1 Introduction

The enormous scale of the web sets a new challenge – handling this scale, so that the users do not get lost and can efficiently perform their intended tasks. The ranking of documents and providing the most relevant information first is a step towards alleviating this information overload and has already been addressed to some extent by contemporary search engines. Another approach, which has not yet been seen in large scale applications, involves some linguistic and semantic analysis in order to better understand the user need and the data on the web and match...
them at a deeper level. This analysis may result in the production of semantic annotations, as defined in Kiryakov et al. (2005b), and represents an aspect of the annotated resources that provides machine-readable insights into the meaning of the content. Our belief is that the scalable and automatic extraction of semantic annotations for the existing web is the route towards real-world Semantic Web (SW) applications. Such metadata would allow SW applications to provide new ways of searching, navigating, summarizing, and analyzing the web resources. For several years, we have been developing the KIM Platform (Popov et al. 2004) for semantic annotation, indexing and retrieval, which basic functionality is presented in Sect. 6.1.1 below. In a nutshell, it allows for efficient semantic annotation and indexing of documents with respect to named entities (NE). The basic indexing approach is preceded by processing of the text content, which is then indexed with a standard information retrieval (IR) engine, as discussed in Sect. 6.0. Such indexing allows KIM to perform hybrid queries, combining full-text search, structured queries, and inference.

Recently we have extended KIM with a new module called CORE, focusing on co-occurrence of entities and providing for incremental searching based on this information, as well as ranking, tracking trends and popularity timelines of these entities. Having some experience in extracting predefined relations between named entities and understanding the extra investment of effort for defining extraction algorithms for new relation types, we decided to extend KIM in the direction of tracking associative (or soft) relations between entities, based on their co-occurrence in the same block of content (or context). The information gathered in the CORE module is based on the semantic annotations produced by KIM and provides another perspective to the results of the semantic analysis of the content as compared to the semantic repository where all the extracted entity instances are stored with their names, relations, and attributes. CORE focuses on tracking some statistics for the bi-directional relations between entities and documents, while leaving the semantically heavier descriptions of entities to the semantic repository.

The KIM Platform is generally independent of the domain it is applied to, but the concrete configuration context provided by default (ontology, instance base, information extraction modules) is tuned for analysis of international news. As in any domain, if one acquires large amounts of data to work on, interesting co-relations of the domain entities could be monitored, shaping trends and dependencies that are not clear without the automatic assimilation of large volumes of information. If the data is also aligned with particular points in time, timelines of these co-relations or of the popularity of entities could be obtained.

The concrete solution presented here takes advantage of the specific strengths of semantic repositories and the raw power of relational databases. The latter having been developed over decades and capable of handling efficiently large volumes of data with fixed structure (which is the case with the occurrence statistics); the former allowing for inference and querying on top of formal knowledge. The combination of the two in one system is like the unity of physical strength and prudent mind in a person. The interactive faceted search (named CORE search) is a
demonstration how an approach based on these two technologies is more powerful and more efficient for certain tasks as compared to traditional search engines.

The contributions of this paper can be summarized as follows:

- novel indexing schema, based on entity occurrence;
- description of a scalable implementation of such indexing;
- advanced faceted search interface, based on co-occurrence.

After this section, the paper continues with an overview of the related work explaining some closely or a bit more remotely relevant efforts. A short description of the environment in the third section sets the basis for the better understanding of the work. The general idea and the conceptual models behind the CORE module are explained in section four. The overall architecture of KIM and some technical details on the annotation and indexing processes are discussed in Sect. 6.5. The faceted search user interface named CORE Search is presented in Sect. 6.6; it provides a demonstration of the capabilities of the CORE module and a pictorial pathway for understanding of the featured functionality. In the last two sections we elaborate some of our intentions for the future and conclude the chapter.

6.1.1 Semantic Annotation in KIM

The main focus of KIM (before the presented extension with the CORE module) was on providing the necessary infrastructure for scalable automatic extraction of named entity (NE) references and descriptions, including attributes and relations, from text. We call both the process and the result semantic annotation. Ontologies and factual background knowledge (held in a semantic repository) are used for analysis of text, implemented as information extraction (IE) application in GATE (see Sect. 6.0). The outcome is twofold:

- extension of the semantic repository with structured data, extracted from the text;
- generation of annotations (metadata) that link the text with the repository.

Figure 6.1 presents the interlinking between the text and the data; each annotation contains an identifier of entity, described in the semantic repository. KIM recognizes both known and unknown NE, instances of predetermined classes, as well as attributes and relationships between them. In addition to the NE, KIM is also extracting and annotating key-phrases, found to be statistically characteristic for the document. Those are stored in the repository as an instance of a special class called GeneralTerm and treated as regular entities.

KIM is designed to be able to take advantage of pre-existing structured data and inference – the text-mining algorithms are initialized from a semantic repository, which can handle large amounts of instance data. A major task in the process of semantic annotation is called identity resolution: to find whether a particular NE reference denotes an already known NE in the repository and which one. In case the reference has no match in the repository, a new entity description is created and
stored in the ontology. As a result of the text analysis, KIM can also extend the descriptions of known entities by means of finding new attribute values and relationships.

Both KIM’s architecture and ontologies (Terziev et al. 2005) allow for easy domain specific specialization on the principle of modular ontological extensions and pluggable modules for IE.

### 6.1.2 Inverted Indices and Vector-Space Model for Information Retrieval

Traditional IR engines index text documents with respect to the string tokens mentioned in them. Usually, tokens are considered all the different strings which appear in the text separated by white-space or punctuation. Such are different forms of words, all the spellings of names, abbreviations, numbers, etc. The so-called full-text indexing and search is based on inverted indices which allow for retrieval of documents by tokens mentioned in them. The standard way to formulate a query, that is, to express user’s information need, is to provide several tokens, which should appear in the documents of interest. In the Boolean search model, whenever multi-token query is evaluated, the engine uses the inverted indices to check for documents where all of the query tokens appear.

A task that appears very important, when dealing with large sets of documents, is not only to retrieve the documents, but also to order (or rank)
them by relevance to the query. For this purpose a model richer than inverted
token-document index is necessary. The most popular approach for relevance
ranking in IR is the so-called Vector Space Model (VSM). The underlying
assumption is that documents are considered as objects which are characterized
by the tokens appearing in them. This way, tokens represent a set of features of
the documents. It is important that the degree of association is considered, so
that some tokens are considered more characteristic for a document than others.
The abstraction used to formally model these “weighted” characterization is a
geometrical one: documents are represented in a space, where each dimension
is associated with a token. Thus if there are 10 million different tokens appear-
ing in the indexed documents, the space used as a model for relevance ranking
will have 10 million dimensions. Each document is modeled as a vector, where
the number of occurrences of a specific token in it is taken as a coordinate value
for the corresponding dimension. The engine represents the query as a vector in
the same space and then takes the cosine of the angle between the query and the
document as a relevance measure.

In practice, IR engines implement quite a number of normalizations on top of
the basic VSM in order to handle issues like very popular terms, documents of
varying size, fields of special importance (e.g., title), etc.

### 6.1.3 Semantic Indexing and Retrieval in KIM

Once the semantic annotations are in place (as discussed in Sect. 6.0), they are
used for indexing of the documents. KIM implements a schema which forces a
standard IR engine to index the document not only by string tokens, but also by
entity identifiers. This is achieved by simple pre-processing of the queries and
the documents sent for indexing. Technically, the identifiers of the entities are
added as new unique tokens (or features), which appear as new dimensions in
the vector-space model.

The immediate benefit is that documents can be retrieved by entities, which repre-
sents a sort of conceptual search that abstracts the user from spelling (surface realiza-
tion) of the references to the entities. For example one can find documents which refer
“UK”, “United Kingdom”, and “U.K.” with one simple query, asking for document,
characterized by the identifier of UK, as it appears in the semantic repository.

This type of semantic indexing allows also hybrid queries, combining full-
text search, structured queries, and inference. Consider for example, the
following query:

“telecom company in Europe” “John Smith” director

The information need appears to be for documents concerning a telecom company
in Europe, a person called John Smith, and a management position. Note, however,
that a document containing the following sentence would not be returned using
conventional search techniques:
“the board of O2 appointed John G. Smith as CTO”

In order to be able to return this document, the search engine would need to be able to “consider” several semantic relations and apply several inference rules:

- O2 is a mobile operator in UK;
- Mobile operators are a special case of telecom companies;
- UK is part of Europe;
- CTO is a kind of director.

These are precisely the kinds of relations which can be present in the semantic repository — either as pre-loaded background knowledge or as facts extracted from documents. Finally, to match the query, the semantic repository should be able to perform simple inference like: if O2 is located in UK and UK is part of Europe, then it is also located in Europe.\(^1\)

6.2 Related Work

Very interesting and relevant work has been done at the Joint Research Center of the EC in Italy by the group of Ralf Steinberger and Clive Best. One of their products is called EMM NewsExplorer and analyses news articles in several languages. The extracted information is presented in a knowledge base and static web pages are produced to ensure stable and fast access for the users. Besides extracting key-phrases and named entities, NewsExplorer calculates clusters of news articles where each cluster represents a single event covered in the news. This analysis is also cross-lingual and each cluster is formed of articles in different languages referring to the same event. It is obvious that searching technology helps users find their way in the contemporary information overflow, but the identification of a single event out of many news articles that discuss the same topic brings the usability of NewsExplorer to a new level. JRC also provides timelines of major news on the basis of the sizes of the event clusters. Another interesting aspect is the front-end provided to users to help them navigate through the news. For each news article there are people and organizations that are either related to or associated with the content. Similarly for each entity one can see the articles that refer to it. Generally their ideas, efforts and results are remarkable in several aspects: multilinguality, IE, clustering of news to find out events, and presentation of the news to the end-user.

Rocha et al. (2004) describe a search architecture that applies a combination of spread activation and conventional information retrieval to a domain-specific semantic model in order to find concepts relevant to a keyword based query. The

\(^1\)With thanks to John Davies, who first proposed this example.
essence of spread activation, as applied in conventional textual searching, is that a
document may be returned by a query, even if it contains none of the query key-
words. This happens if the document is linked to by many other documents which
do contain the keywords.

The query being a set of keywords is forwarded to a conventional search
engine which assigns a score to each document in its index in the usual way. In
addition to a conventional index, the system contains a domain-specific KB,
which includes instance nodes pointing to web resources. As usual in RDF,
each instance is described in terms of links labeled with properties in compliance
with the ontology. The basic assumption is that weightings that express the
strength of each instance relation can be derived.² Thus the resulting network
has, for each relationship, a semantic label and an associated numerical weight.
The intuition behind this approach is that better search results can be obtained
by exploiting not only the relationships within the ontology, but also the
strength of those relationships.

The QuizRDF search engine (Davies et al. 2003) combines FTS with a  capability
to exploit RDF annotation querying. QuizRDF combines search and browsing
capabilities into a single tool and allows document-level RDF annotations to be
exploited (searched over) where they exist, but will still function as a conventional
search engine in the absence of those annotations.

A document can be described with properties appropriate for the class of its
subject. For example a document associated with subject Employee can be
given attributes firstName, lastName (inherited from the Person super-class)
and position. Full-text indexing is applied not only to the content of the
documents, but also the literal values of any relations. This way a keyword
search for “CTO” will return this document, although this string may not
appear in the document.

This datamodel would have been clearer if the documents and the objects they
are about were defined as separate RDF resources, for example, <cv,isAbout,emp1>,
<emp1,type,Employee>, <emp1,firstName,“George”>. On the other hand, this
model has all the advantages of being more simple and efficient.

TAP (Guha and McCool 2003) is a Semantic Web architecture, which allows
RDF(S)-compliant consolidation and querying of structured information. In
Guha et al. (2003), two Semantic Web-based search engines are described: ABS
– activity-based search and W3C Semantic Search. In both cases TAP is
employed to improve traditional search results (obtained from Google) when
seeking information in relation to people, places, events, news items, etc. TAP
is used for two tasks:

Result augmentation: the list of documents returned by the IR system is
complemented by text and links generated from the available background
knowledge;

² A number of different approaches to this derivation are taken and the authors state that no single
weight derivation formula is optimal for all application areas.
Query term disambiguation: the user is given the opportunity to choose the concrete entity she is searching for, than the system attempts to filter the results of the IR system to those referring only this entity. An approach using several statistics for this purpose is sketched in Guha et al. (2003) without details of the implementation.

A system called SemTag is presented in Dill et al. (2004), which performs automatic semantic annotation of texts with respect to large scale knowledge bases available through TAP, solving a task similar to the one solved by the KIM Platform (Popov et al. 2004).

Recently Google released Google Trends, which allows the user to choose keywords and see the search frequency for these in a timeline chart. Based on this timeline, one can see the change in frequency and sometimes news articles associated with a point on that timeline, but without a particular relation to the extremes in that timeline. An interesting feature is that the statistics can be restricted by region and thus track trends specific only for this geographic area or location.

6.3 Description of the Environment

The CORE module is a part of the KIM Platform which consists of an ontology, a world knowledge base, a KIM Server providing API for remote access, embedding, and integration and some front end applications. It is based on robust open-source platforms specialized in three different domains:

- **Ontology management**: the knowledge resources are kept in the SESAME\(^3\)-based OWLIM\(^4\) semantic repository, which provides storage, inference, and query infrastructure on top of RDF(S) or OWL;
- **Text mining**: the GATE\(^5\) platform has been used as a basis for the information extraction (IE) processing and also as framework for the management of content and annotations. It provides the fundamental text analysis technologies, on top of which we have built the semantically aware extensions, specific for the IE of KIM.
- **Information retrieval (IR)**: the Lucene\(^6\) IR engine has been adopted to perform indexing, retrieval and evaluation of content relevance with respect to named entities.

The CORE module is extending the combination of the mentioned technologies towards a relational database management systems (RDBMS), taking benefits of

---

\(^3\)http://www.openrdf.org – one of the most popular RDF repository frameworks.

\(^4\)http://www.ontotext.com/owlim – the fastest and most scalable RDF(S)/OWL semantic repository.

\(^5\)http://www.gate.ac.uk – the “General Architecture for Text Engineering” – the most popular NLP and text-mining platform developed in University of Sheffield.

\(^6\)http://lucene.apache.org/ – the most popular open-source IR engine.
the specific strengths of these engines. The actual implementation featured here is based on Oracle, although the design of the CORE module allows for implementations based on other RDBMS. It takes over some of the functionality present in the semantic repository and the IR engine, enriching it with co-occurrence and ranking of entities.

The corpus used for populating CORE with data exceeds 1 million news articles in English. It includes news from numerous sites like BBC, CNN, AP, AFP, Reuters, etc. The corpus currently covers all major news for the period from 2002 till now. Each news article is associated with a point in time when it has been published. This allows statistics over time to be calculated for the contained entities that were recognized, as well as to track the popularity of these entities.

### 6.4 Co-Occurrence and Ranking Model

The model of KIM’s CORE module differs from the traditional IR indexing approach (see Sect. 6.0) in two ways:

- It considers as features of the documents not all the tokens appearing in them, but rather only the named entities and the key-phrases. This leads to reduced-dimensionality vector-space model.
- While the inverted indices support only retrieval of documents by features, CORE maintains bi-directional indices, which allow also for retrieval of features by documents. This way CORE allows for efficient retrieval of entities, which are mentioned in documents.

The reduction of the dimensionality of the space is a technique used in other advanced IR approaches, for example, LSA (Landauer and Dumais 1997). When implemented “properly” it allows for indexing and retrieval by concepts, a sort of improved feature space, which is a better representation of the meaning of the documents, as compared to the strings of characters appearing in the text. The schema selected by us has the following advantages:

- The selection of the dimensions is cheaper and can be performed in a more dynamic fashion, as compared to statistical methods like LSA. This is because, we do not need heavy computational effort to derive the features – we take the named entities and the key-phrases which come as output of the semantic annotation;
- The reduced dimensionality allows efficient management and scaling of the bi-directional weighted index of CORE. As it is more expensive to maintain than a simple inverted index, such an index is impractical if applied over the entire original feature space;

---

7http://www.oracle.com/ – one of the leading RDBMS.
• It can always be complemented by a standard full-text search index, maintained in parallel. This is the case implemented in KIM.

This new indexing schema allows for efficient handling of several new types of queries. One of them is CORE’s capability to track co-occurrences and answer queries focused on entities as an end result like give me all entities that co-occur with Europe and Iran. Although we have shown ways of altering a full-text search (FTS) engine to work with respect to named entities (Popov et al. 2004), such queries cannot be processed by these engines for the sole reason of the organization of their indices which point from the indexed terms to the documents but not vice versa. In CORE one could ask for co-occurring entities, their rank or the deviations in their occurrence in time (timelines) conforming to restrictions like time period or a sub-set of the corpus. Reducing the feature space to entities allows for effective tracking of these correlations between entities, time points and documents for large sets of data (millions of documents) in a relational database. Modeling of occurrence tracking, FTS indexing over the content, class hierarchy, and a significant part of the entity description in the database allows for expressive and efficient querying on a combination of these features without the need to intersect result sets from different data sources like an IR engine, a semantic repository, and a database.

As presented in Sect. 6.0, the CORE module allows incremental search and timelines based on extracted semantic information and on entities rank and co-occurrence. It is essential that this is a mixture of both semantic and statistical information. The semantic information is about the entities relations and classification, while the entity rank is purely statistical information. The co-occurrence information has both statistical and hidden semantic meaning. We will show that this mixture allows for a new way of searching and analysis that makes it easier for the user to find a set of documents relevant to her need.

When two entities co-occur in the same context usually this is a clue for a kind of semantic relation between them which may or may not be explicitly expressed. The IE of KIM extracts relations like Organization located in Location; Person having a Position within an Organization or associated with a Location (e.g., a mayor), but the set of possible relations is unlimited. Even if the user interest is in relations that are not recognized explicitly, still, one can easily find related entities and respective documents based on the associative relations found on the basis of co-occurrence in the same context.

The nature of ranking is statistical, but from that we also can derive semantic information. Let us suppose that we are interested in the relations of John with other people. CORE allows asking about the most popular people that are mentioned together with John. The term “most popular” means “with the highest rank in the set of documents that also mentions John”. The co-occurrence of the not so popular people will be considered as a less relevant part of the result set.

Since each document in the corpus has explicit date, we can track trends in time (timelines) for a set of entities based on their occurrence or co-occurrence frequency. Having these timelines, one can find out the important events in the life of a given entity following the extreme points of this function.
6.5 System Architecture

We have to manage both semantic and statistical information and this is inefficient or impossible with the usage of either semantic repositories or contemporary IR engines. For storage of semantic information the usage of semantic repository is the natural decision but for processing (grouping, sorting and aggregation) of large volumes of numeric data the modern database systems are much more efficient. KIM employs a semantic repository [based on SESAME and OWLIM (Kiryakov et al. 2005a)] and a relational database (based on Oracle). OWLIM is the fastest and most scalable semantic repository currently known, while Oracle has been the leading database system for a long time. Scalability is a key design goal because we intended to build a system that could handle millions of documents and hundreds of millions of entity co-occurrences.

The KIM Server automatically extracts semantic annotations from unstructured text and associates these with the processed document (Fig. 6.2). Each annotation keeps URI references to the extracted entity. The annotated documents with the entity occurrence information are stored in a RDBMS, while the semantic description of entities is stored in the semantic repository. The usage of two heterogeneous systems inherently causes duplication of some pieces of information. To keep the two in synchrony we have to keep the correspondence between the entity description in the semantic repository and the database.

There are three major types of information stored in the database (Fig. 6.3): entity information, document information and the relation between them (information for the occurrence of entities in the documents).

The entity information is the mapping between its identity in the database and in the semantic repository. Since the database indices are optimized to work with integers while the semantic repository is based on URIs, we have to generate one
unique integer ID for each entity URI. This is the most important part of the entity information since it links its representation in the database to the full entity description in the semantic repository. In some kinds of queries we are interested only in entities of a given class or with a given alias. The entity class and aliases are stored in the semantic repository but for performance reasons we have to keep them in the database too. This is not strictly necessary but it is the only way to make the queries efficient without the need to aggregate query results from different sources.

All documents, from which the semantic information was extracted, are stored in the database. The important features of a document are the content and the document-level metadata, among which - its date. The text is required if you want to combine FTS queries with purely semantic queries. The document date is necessary for the modeling of entity occurrences with respect to time points.

While the above two pieces were related to two disjoint aspects of the data (i.e., documents and entities) the third one is the connection between them – a many-to-many relation between entity and document. Each entity can be referred to in many documents and each document refers to many entities. For each unique pair entity/document the number of occurrences is kept so that we can compute the entity rank. The actual computation of ranking in a given sub-corpus requires aggregation and sorting of the occurrence information in each individual document. The presentation of entity occurrence in the database also allows for querying the co-occurrence of entities in the same context using efficient relational operations.

The current scale of the CORE module has been examined with about a million documents and all the respective entity occurrences. For such amounts of potential results the first step towards successful information retrieval is to limit the results either by FTS keywords, time period, or selecting an entity and incrementally restricting the set of results until a comprehensible scale is achieved. By selecting an entity, one restricts the set of results to the ones that refer to this entity, and the options for further restrictions are based on further selecting one of the entities that co-occur with the selected one.

All further queries are performed only in the scope of the selected sub-corpus. Currently CORE can retrieve the following types of results: a list of all documents contained; count of the documents in the sub-corpus; a list of all or the first N most popular entities referred from a given class; the number of entities referred; rank or popularity timeline for a given entity (or set of entities) within the sub-corpus in focus; ranking timeline for the first N most popular entities in the

![Diagram](image_url)

**Fig. 6.3** Part of the information stored in the database
(sub-)corpus; time distribution of the documents in the corpus based on daily, monthly or yearly units.

The KIM Platform is a distributed system with Java RMI interface between the server and client sides (Popov et al. 2004). The CORE module API is only a part of the interface provided by the KIM Server. The CORE API is divided into two parts: one for general operations over documents and entities and the other for timeline queries. There is a query object that represents the possible restrictions over the data and it is passed to the different methods of this API which return a list of entities, list of documents or a timeline. The API for timelines requires extra parameters like the time unit and the entity set in focus. All methods in the timelines API return an object that represents the time distribution of the featured documents or entities.

6.6 User Interface

The user interface that gives access to the CORE resources and presents a demonstration of the server functionality allows the user to perform different requests and is visually divided into two parts. The first one is called CORE Search and provides the faceted search functionality, while the second one called Timelines is intended to give the user an easy way to rank entities by popularity and see occurrence trends for specific entities of interest. There is still another piece of functionality that is not separated from these, but is very useful along with CORE Search and Timelines. This is the option to see documents’ distribution over time when searching for documents matching certain criteria and given a time period.

CORE Search utilizes most of the functionality CORE was designed for. It also presents some nice features that make it easier and more convenient to narrow down the documents to the desired set. There are generally two ways the user can restrict documents of interest (Fig. 6.4):

- words or phrases can be typed in the Document Keyword Filter field to limit documents to those that contain them. This is the standard token/term/key-word-based full text search restriction;
- entities can be filtered by name and added to the Selected Items list to restrict the documents to only those in which the selected items appear.

The entities could be selected on the right side of the interface shown in Fig. 6.4, where they are listed in columns by class. To find and add a specific entity to the filter the user can type in a (part of) the entity name in the alias filter above the list for the desired class. This will narrow down the entities in the list and highlight will appear for the filter matches. Afterwards, the user can browse the full list of entities by clicking on the number representing the entity result count if the searched one is not displayed in the top ranked entities list. Another option is to click on the magnifying glass icon beside an entity and see the full entity description which will be requested from the semantic repository (Fig. 6.5). The entity description explorer
also allows further navigation through the instance base by following the links to the proper class or the related entities.

By selecting an entity from one of the lists it is being transferred to the Select Items List and the document set and displayed entities are being refreshed. The new result displays the updated count of documents containing the selected entities. The entity columns are refreshed to show only the entities co-occurring with the selected ones, thus ensuring that the document result can be further narrowed down. As the example in Fig. 6.4 shows, there are 1351 people that appear in the same 465 documents, that were restricted by a keyword and two selected entities.

The list of displayed entities for each class is limited by the obvious space restrictions of the single page layout of the user interface. The visible entities are the top ranking ones given the current co-occurrence context, entity name and FTS restrictions. If the entity of interest is not there, either a more restrictive filter can be entered or the full entity list can be browsed. In Fig. 6.4 it is seen that only the 25 most popular people for the given set are shown out of more than thousand that meet the restrictions.

Selecting too many entities would lead to a situation where the document or entity set has been over-restricted and too small. At this point entities could be removed from the Selected Items List and will appear as history but can easily be moved back to the selected list later.

Another characteristic of the CORE Search UI is that it is highly interactive. For instance, the count of matching documents is updated on the run as the user types in tokens in the keyword filter field. Entities are filtered in the same way: as one enters a part of the name of the needed entity, the list of the matching entities is being updated dynamically. This is another demonstration of the extreme efficiency

Fig. 6.4 CORE search user interface
of the CORE module – effectively, each character entered by the user causes numerous non-trivial queries, which are handled in real time, usually, matching the speed of typing.

The user can also change the number of entity columns to be shown and assign a custom class to each of them from the Options menu (top right of Fig. 6.4).

Finally, when narrowing down the document set to a browsable scale, one can view the document result matching the current restrictions. Along with the resulting documents list, one also sees a chart of the distribution of these documents over time (bottom of Fig. 6.8).

If not interested in documents but in the timelines of entity popularity, one can choose to see these using the restricted document set as a basis for their calculation. Timelines interface allows trends to be calculated and conveniently viewed and navigated through a chart. The calculation can be performed over all the documents in store or just over some limited document set that came from CORE Search.

**Fig. 6.5** Semantic description of an entity
Selecting a time period for the calculation will also limit those documents. Depending on the length of that period a granularity can be set that ensures the desired level of details for the resulting chart.

There are two different types of timelines that can be calculated: one can ask for the timeline of the most popular entities of a given class (Fig. 6.6) or view the distribution over time of the occurrences of specific entities chosen by the user.

When requesting the most popular entities the user may choose their class, the number of entities to be covered by the timeline charts, and the time period granularity. One can also specify if the entities in interest are the ones that have overall top rank across the period or the ones that have a top rank only in some of the granularity time periods. The latter option would include in the timeline entities that are having their “fifteen minutes of fame”, while the former would cover only such that are with more stable occurrence frequency throughout the featured period.

There is an option to see a timeline for a set of selected entities, which can be selected from columns like the ones in the CORE Search interface. The difference is that the selected entities do not restrict the document set but are only used as input parameters for the generation of the timeline chart.

In Fig. 6.6, the top 5 people for a given period are being requested, and at the same time their popularity is to be calculated only based on their appearance in the document set limited through the CORE Search interface (Fig. 6.4). The chart in Fig. 6.7 is the resulting timeline for these entities.

Navigation on the timeline chart itself is simple and intuitive. Positioning over a point from the chart shows information about the entity, the time period and the number of occurrences of the entity for that period and the given set of documents. Clicking on that point would redirect to the list of the documents that forms the respective part of the timeline along with a document distribution chart for the period of calculation (Fig. 6.8).
Fig. 6.7 Timelines result chart

Fig. 6.8 Document result
The document distribution is displayed in an active chart which can be clicked to navigate deeper into smaller and smaller periods until the desired documents are found.

A screen similar to Fig. 6.8 would also appear if documents were required directly through the CORE Search interface. The documents content and associated document-level metadata and contained entities could also be seen if the user navigates further through clicking on a document title in the list.

### 6.6.1 CORE Search Versus Standard Faceted Search

The standard notion for faceted search considers searching objects in multi-dimensional space, where the objects are characterized with respect to several independent aspects (facets of criteria). Each facet can be regarded as an attribute, which can be given a value for each object. For each facet (some of) the values are presented to the user, so, that she can further specify her needs (search query) by means of simple selection of one of the values.

Such example is the search for hardware items in the e-commerce portal http://www.newegg.com; the examples in Fig. 6.9 reflect its state on 3rd of Dec, 2007. When the user starts searching for a CPU, she is initially presented with the following facets Price, Manufacturer, Processor Type, Series, and few others (on the left hand side of Fig. 6.8). The number of matching items is presented in brackets after each of the values. If the user selects “Server” from the Processor type list, she gets an updated faceted search pane, where the values for each of the facets are updated, “Processor Type” facet disappears and a new facet “Core” is presented (on the right hand side of Fig. 6.9).

The philosophy of faceted search is that the most relevant facets and values are presented, to help the user further constrain her search. This paradigm is similar to the Navigational searches, as the ones offered by taxonomy-based directories (e.g., http://www.dmoz.org). The difference is that the objects are classified against several independent criteria, rather than a single classification hierarchy. Further, the categories under each of these facets are automatically collected, rather than pre-determined.

CORE search is a special case of faceted search. In the standard faceted search a single object (in our case document) can have a single value for each of the facets. The model of CORE Search is much different than this. As presented on schema (1) of Fig. 6.10, in CORE we start with documents that are annotated with references to entities. Within single document there could be multiple annotations referring one and the same entity; each of them bears information irrelevant for CORE search (e.g., offsets of the references in the document). Schema (2) presents the structure of the CORE index – single Occurrence consolidates the information about the number of annotations, referring a concrete entity in a specific document. Finally, schema (3) represents the virtual facets, which are underlying CORE Search, as opposed to the standard facets, presented on schema (4) in Fig. 6.10. The main characteristics of CORE facets are as follows:
Single facet is created for each class of entities (e.g., CF_C1 and CF_C2);
Each of these facets can have multiple values, namely entities of different classes, that were referred in the document (e.g., Entity2 and Entity3 are both values of CF_C2);
Each facet value is actually a pair of a value and a number, which in our case represents the number of occurrences of the entity in the document.

In a sense, CORE facets represent “weighted” relationships between documents (or any other type of objects) and entities (or values).

We can represent each CORE facet and each standard facet as mathematical functions with the following description:

\[
\text{CORE Facet} : O \rightarrow \{V, n\} * \quad \text{Standard Facet} : O \rightarrow V
\]

It becomes obvious that the data-model underlying CORE search is richer than the one behind the standard faceted search. Still, the philosophy for search and navigation remains the same:
Facet values are aggregated, across the dataset (the search space);
The highest scoring values are presented to the user, as those have the highest chance to be subject of her interest;
When the user puts some constraints, those affect the suggested values, so that the most informative ones are offered.

There are also some differences between CORE Search and the NewEgg’s Guided search, taken as representative of the standard faceted search paradigm. At present, CORE search does not support automatic selection of the most relevant facets. This is partly because of the nature of facets in CORE – as those are virtually created on the basis of the classes of entities and can be customized by the user, so,
it is not obvious that automatic modification of the sets of presented facets will be
intuitive. The second difference is that in CORE Search we do not present the
number of objects matching each facet value. The reason here is that we believe that
ordering by popularity already provides to the user sufficient evidence regarding
the selectivity of the different values. Thus, we prefer not to show this figures on
the screen in order to display the optimal amount of information for the sake of
better ergonomics.

6.7 Ongoing Work

Semantic Web applications, although gifted with obvious advantages, have a
long way to go to match the scale introduced by years of development of rela-
tional database systems or the mass analysis of web content performed by
contemporary search engines. The KIM Platform and its CORE module cur-
rently achieve real-time retrieval from about a million documents and a million
of entity descriptions which should be enough in an intranet environment.
However, the objectives are to reach levels of efficiency allowing us to cover a
significant amount of the currently available web resources, and we will
definitely continue in this direction.

In this regard, KIM already has a cluster architecture where its semantic analysis
functionality, being the main bottleneck, is distributed over independent annotator
components running (optionally) on different machines. Despite the obvious advan-
tages of such architecture there are still some centralized synchronization points for
our semantic repository, indexes and database. A work in distributing the semantic
repository done by us or another group, allowing for effective intersections of query
results across repositories, would bring the scale of such applications as KIM to a
completely different level.

Regarding the CORE module it would be interesting to introduce means for
defining the scope of the context of co-occurrence of entities to paragraphs, sets of
documents or others.

The semantic repository of KIM and CORE DB are not tightly integrated at
present. CORE DB “knows” the class hierarchy and very little about the semantics
of the entities – their aliases and class memberships. This allows CORE Search to
involve look up for organizations (i.e., entities of a sub-class of Organization) with
specific strings in their aliases. However, formal typed relationships between
entities cannot be involved, even when those are “known” in the semantic
repository; such example would a statement that <Org1, tradedOn, Exchange2>. An
team we are currently considering is whether to introduce more of the semantics
of the entities to the underlying database representation of the CORE module or to
introduce a low-level integration of the RDBMS with the semantic repository.

At present CORE DB is using the semantic annotation, that is, the metadata
generation and ontology population services provided by other modules of KIM. In the
opposite direction, CORE DB can facilitate the metadata generation in two ways:
Both class- and instance-level disambiguation of entity occurrences can be performed via comparison on the CORE profiles of the candidates and the context;

Relation extraction is a tough IE task, so, when the confidence threshold is high, there are a relatively small number of extracted relations. This makes the descriptions of the new entities, extracted by KIM, to be relatively flat – there are not many attributes and relationships to other entities; the degree of connectivity in the semantic repository is getting lower as it grows through extracted knowledge. Associative relationships, extracted from CORE DB, can be added to the semantic repository. Although the types of the relations may not be easily determined, such extension would be useful, as the confidence in the existence of a relation could be high enough. A possible implementation is to add to the semantic repository the relationship between each entity and, say, the ten others it is most strongly associated with in CORE DB.

The current ranking schema in CORE is very simple – numbers of occurrences are counted without any normalization with respect to the size of the document or frequency of the entity (the TF part of TF/IDF). We are working on various statistical measurements and metrics for ranking, popularity and timelines calculation, which can be adjusted and changed dynamically.

Another interesting potential option that the future holds for us might be the cooperation with information extraction groups that would like to run their own domain specific extensions on our platform and specifically employing the CORE module, a thing we have already done in the past with KIM.

6.8 Conclusion

This paper presented a novel search paradigm based on popularity-ordered faceted search on top of semantic annotations. The technology used to demonstrate this was the CORE module of the KIM semantic annotation, indexing, and retrieval platform. The ideas behind it and their advantages to the traditional keyword search have been demonstrated on the conceptual level and through the CORE Search user interface. The essence of the approach is a specific indexing, performed on the basis of semantic annotation of text with respect to named entities and key-phrases allowing for semantic search through adaptation of standard probabilistic IR models.

The frequency of occurrence of an entity indicates its level of association with the given context and can also be considered as “popularity”. Various parts of documents or collections of such can be considered contexts or compound contexts. The co-occurring entities can be used to form a “statistical profile” of an entity, a document, or a group of such objects.

The paper also presented the beneficial symbiosis of a relational database and a semantic repository in the CORE module of KIM for the purpose of bi-directional indexing of the association of semantic annotations and contexts.
Acknowledgment This work was supported by the IST Programme of the European Community under SEKT (IST-IP-2003-506826).

References