Knowledge Extraction for Information Retrieval

Francesco Corcoglioniti, Mauro Dragoni(✉), Marco Rospocher, and Alessio Palmero Aprosio

Fondazione Bruno Kessler, Trento, Italy
{corcoglio,dragoni,rospocher,aprosio}@fbk.eu

Abstract. Document retrieval is the task of returning relevant textual resources for a given user query. In this paper, we investigate whether the semantic analysis of the query and the documents, obtained exploiting state-of-the-art Natural Language Processing techniques (e.g., Entity Linking, Frame Detection) and Semantic Web resources (e.g., YAGO, DBpedia), can improve the performances of the traditional term-based similarity approach. Our experiments, conducted on a recently released document collection, show that Mean Average Precision (MAP) increases of 3.5 % points when combining textual and semantic analysis, thus suggesting that semantic content can effectively improve the performances of Information Retrieval systems.

1 Introduction

Recent years have seen the growing maturity of Knowledge Extraction (KE) from natural language text. State-of-the-art KE approaches, such as FRED [1], NewsReader [2], and PIKES [3], exploit Natural Language Processing (NLP) techniques as well as Semantic Web (SW) and Linked Open Data (LOD) resources (e.g., DBpedia [4]) to extract semantic content from textual resources, linking it to well-known ontologies and to a growing body of LOD background knowledge.

In this paper we investigate the benefits of using the semantic content automatically extracted from text for Information Retrieval (IR). The goal in IR is to determine, for a given user query, the relevant documents in a text collection, ranking them according to their relevance degree for the query. Our approach aims to overcome known limitations of traditional IR approaches. Let us consider the following query example: "astronomers influenced by Gauss". Traditional IR approaches match the terms or possible term-based expansions (e.g., synonyms, related terms) of the query and the documents, but relevant documents may not necessarily contain all the query terms (e.g., the term "influenced" or "astronomers" may not be used at all in a relevant document); similarly, some relevant documents may be ranked lower than other ones containing all three terms, but in an unrelated way (e.g., documents about some astronomers born centuries before Gauss, influenced by Leonardo Da Vinci).

In our approach, both queries and documents are processed to extract semantic content pertaining to the following *semantic layers*: (i) entities, e.g., dbpedia:Carl_Friedrich_Gauss from "Gauss"; (ii) types of entities, either

H. Sack et al. (Eds.): ESWC 2016, LNCS 9678, pp. 317–333, 2016.
DOI: 10.1007/978-3-319-34129-3_20

explicitly mentioned, such as yago:Astronomer109818343 from "astronomers", or indirectly obtained from external resources for mentioned entities, such as yago:GermanMathematicians from "Gauss"; (iii) semantic frames and frame roles, such as framebase:Subjective_influence from "influenced"; and, (iv) temporal information, either explicitly mentioned in the text or indirectly obtained from external resources for mentioned entities, e.g., via DBpedia properties such as dbo:dateOfBirth (1777) and dbo:dateOfDeath (1855) for entity dbpedia:Carl_Friedrich_Gauss. We then match queries and documents considering both their textual and semantic content, according to a simple retrieval model based on the Vector Space Model (VSM) [5]. This way, we can match documents mentioning someone who is an astronomer (i.e., entities having type yago:Astronomer109818343) even if "astronomers", or one of its term-based variants, is not explicitly written in the document text. Similarly, we can exploit the entities and the temporal content to better weigh the different relevance of documents mentioning dbpedia:Carl_Friedrich_Gauss and dbpedia:GAUSS_(software), as well as to differently rank documents about Middle Age and 17th/18th centuries astronomers.

While other ontology-based IR approaches typically builds only on terminological knowledge (e.g., classes, subclasses), to the best of our knowledge our work is the first in exploiting such a variety of automatically extracted semantic content (i.e., entities, types, frames, temporal information) for IR.

We developed a first implementation of our approach, named KE4IR (read: *kee-fer*), using PIKES for KE and Apache Lucene[1] for indexing documents and evaluating IR queries. We performed a first assessment of the approach on a recently released dataset [6], showing that enriching textual information with semantic content outperforms retrieval performances over using textual data only.

The paper is structured as follows. In Sect. 2, we briefly review the state of the art in IR and KE. Section 3 presents the KE4IR approach, detailing the semantic layers and the retrieval model used for combining semantic and textual information. In Sect. 4, we describe the actual implementation of our approach, while in Sect. 5, we report a first assessment of the effectiveness of adding semantic content for IR, discussing in details some outcomes and findings. Section 6 concludes with some final remarks and future work directions.

2 State of the Art

Previous works have exploited some semantic information for IR. An early tentative in injecting domain knowledge information for improving the effectiveness of IR systems is presented in [7]. In this work, authors manually built a thesaurus supporting the expansion of terms contained in both documents and queries. Such a thesaurus modeled a set of relations between concepts including synonymy, hyponymy and instantiation, meronymy and similarity. An approach based on the same philosophy was presented in [8], where the authors propose an

[1] http://lucene.apache.org/.

indexing technique where WordNet [9] synsets, extracted from each document word, are used in place of textual terms in the indexing task.

In the last decade, semantic IR systems started to embed ontologies for addressing the task of retrieving domain-specific documents. An interesting review on IR techniques based on ontologies is presented in [10], while in [11] the author studies the application of ontologies to a large-scale IR system for Web usage. Two models for the exploitation of ontology-base knowledge bases are presented in [12,13]. The aim of these models is to improve search over large document repositories. Both models include an ontology-based scheme for the annotation of documents, and a retrieval model based on an adaptation of the classic Vector Space Model (VSM) [5]. Finally, in [14] an analysis of the usefulness on using ontologies for the retrieval task is discussed. More recently, approaches combining many different semantic resources for retrieving documents have been proposed. In [15], the authors describe an ontology-enhanced IR platform where a repository of domain-specific ontologies is exploited for addressing the challenges of IR in the massive and heterogeneous Web environment.

A further problem in IR is the ranking of retrieved results. Users typically make short queries and tend to consider only the first ten to twenty results [16]. In [17], a novel approach for determining relevance in ontology-based IR is presented, different from VSM. When IR approaches are applied in a real-world environment, the computational time needed to evaluate the match between documents and the submitted query has to be considered too. Systems using the well-known VSM have typically higher efficiency with respect to systems adopting more complex models to account for semantic information. For instance, the work presented in [18] implements a non-vectorial data structure that exhibits high computational times for both indexing and retrieving documents.

In the last few years, several approaches and tools performing comprehensive analyses to extract quality knowledge from text were presented. FRED [1] extracts Discourse Representation Structures (DRSs), mapping them to linguistic frames that in turn are transformed in RDF/OWL via Ontology Design Patterns.[2] In NewsReader [2], a comprehensive processing pipeline extracts and corefer events and entities from large (cross-lingual) news corpora. PIKES[3] [3] is an open-source frame-based knowledge extraction framework that combines the processing of various NLP tools to distill knowledge from text, aligning it to Linked Data resources such as DBpedia and FrameBase[4] [19], a recently released broad-coverage SW-ready inventory of frames based on FrameNet.[5] We exploit PIKES for the implementation of our approach, as described in Sect. 4.

3 Approach

Standard IR systems look at documents and queries as bags of terms (e.g., stemmed tokens). In KE4IR we augment textual terms with additional terms

[2] http://ontologydesignpatterns.org/.
[3] http://pikes.fbk.eu/.
[4] http://framebase.org/.
[5] http://framenet.icsi.berkeley.edu/.

coming from different semantic annotation layers produced using NLP-based KE techniques as well as Linked Open Data background knowledge (Sect. 3.1), and then propose a simple retrieval model that makes use of this additional semantic information to find and rank the documents matching a query (Sect. 3.2).

3.1 Semantic Layers

We consider four semantic layers—URI, TYPE, FRAME, TIME—that complement the base TEXTUAL layer with 'semantic terms'. These terms can be obtained using KE techniques that identify *mentions* (i.e., snippets of text) denoting entities, events and relations. From each mention, a set of semantic terms is extracted, and the number of mentions a term derives from can be used to quantify its relevance for a document. Table 1 (first three columns) reports an example of terms and associated mentions that can be extracted from the simple query of Sect. 1: "astronomers influenced by Gauss".

URI Layer. The semantic terms of this layer are the URIs of entities mentioned in the text, disambiguated against external knowledge resources such as DBpedia. Disambiguated URIs are the result of two NLP/KE tasks:[6] Named Entity Recognition and Classification (NERC), which identifies proper names of certain entity classes (e.g., persons, organizations, locations) in a text, and Entity Linking (EL), which disambiguates those names against the individuals of a knowledge base. The Coreference Resolution NLP task can be also exploited to 'propagate' the URI associated to a disambiguated named entity to its coreferring mentions in the text, so to proper count the number of entity mentions.

TYPE Layer. This layer contains as terms the URIs of the ontological types (and super-types) associated to noun phrases in the text. For disambiguated named entities (resulting from NERC and EL), associated types can be retrieved from external background knowledge resources describing those entities (e.g., DBpedia). For common nouns, disambiguation against WordNet via Word Sense Disambiguation (WSD) provides synsets, which can be mapped to types of known ontologies using existing mappings. Given these two extraction techniques, an ontology particularly suited to this layer is the YAGO taxonomy [20], due both to its WordNet origins and the availability of YAGO types for DBpedia entities.

TIME Layer. The terms of this layer are temporal values mentioned in the text, either because explicitly expressed in a time expression (e.g., "the eighteenth century") recognized via the Temporal Expression Recognition and Normalization (TERN) NLP task, or because associated via some property to a disambiguated entity in the background knowledge (e.g., the birth date associated to dbpedia:Carl_Friedrich_Gauss). We propose to represent time at different granularities—day, month, year, decade, and century—in order to support both

[6] We report in this section the main NLP/KE tasks for the extraction of semantic terms. Some of them typically build on additional NLP analyses, such as Tokenization, Part-of-Speech tagging, Dependency Parsing and Constituency Parsing.

precise and fuzzy temporal matching. Therefore, each mentioned date time value (e.g., 2015-12-18) is mapped to (max) five TIME terms, one for each granularity level (e.g., day:2015-12-18, month:2015-12, year:2015, decade:201, century:20).

FRAME Layer. A semantic frame is a star-shaped structure that represents an event or n-ary relation, having a certain frame type (e.g., framebase:frame-Subjective_influence, from "influenced") and zero or more participants (e.g., dbpedia:Carl_Friedrich_Gauss) playing a specific semantic role in the context of the frame. Semantic frames are typically extracted using NLP tools for Semantic Role Labeling (SRL) based on certain predicate models, such as FrameNet, and then mapped to an ontological representation using an RDF/OWL frame-based ontology aligned to the predicate model, such as FrameBase [19]. Semantic frames provide relational information that helps matching queries and documents more precisely. Different approaches can be used to transform a star-shaped semantic frame into a set of terms of the FRAME layer. In this work, we propose to map each ⟨frame type, participant⟩ pair whose participant is a disambiguated entity (e.g., the pair ⟨framebase:frame-Subjective_influence, dbpedia:Carl_Friedrich_Gauss⟩) to a term, including also the terms obtainable by replacing the frame type URI with the URIs of its super-classes in the ontology. We investigated also using non-disambiguated participant entities, obtaining however worse results.

3.2 Retrieval Model

We adopt a retrieval model inspired to the Vector Space Model (VSM). Given a document collection D, we represent each document $d \in D$ (resp. query q) with a vector $\mathbf{d} = (d_1 \dots d_n)$ ($\mathbf{q} = (q_1 \dots q_n)$) where each element d_i (q_i) is the weight corresponding to a term t_i and n is the number of distinct terms in collection D. Differently from text-only approaches, terms of our model come from multiple layers [21], both textual and semantic, and each document (query) vector can be thought as the concatenation of smaller, layer-specific vectors. Given a term t, we denote its layer with $\mathrm{l}(t) \in L = \{\text{TEXTUAL}, \text{URI}, \text{TYPE}, \text{FRAME}, \text{TIME}\}$.

To compute the similarity between a document $d \in D$ and a query q, we use a similarity function $sim(d, q)$. The documents matching q are the ones with $sim(d, q) > 0$, and they are ranked according to decreasing similarity values. To derive $sim(d, q)$ we start from the cosine similarity of VSM:

$$sim_{\text{VSM}}(d, q) = \frac{\mathbf{d} \cdot \mathbf{q}}{|\mathbf{d}| \cdot |\mathbf{q}|} = \frac{\sum_{i=1}^{n} d_i \cdot q_i}{\sqrt{\sum_{i=1}^{n} d_i^2} \cdot \sqrt{\sum_{i=1}^{n} q_i^2}} \tag{1}$$

and we remove the normalization by $|\mathbf{d}|$ and $|\mathbf{q}|$, thus obtaining:

$$sim(d, q) = \mathbf{d} \cdot \mathbf{q} = \sum_{i=1}^{n} d_i \cdot q_i \tag{2}$$

Normalizing by $|\mathbf{q}|$ does not affect the ranking and only serves to compare scores of *different* queries, thus we drop it for simplicity. Normalizing by $|\mathbf{d}|$ has the

Table 1. Terms extracted from the query "astronomers influenced by Gauss", with mentions m_1 = "astronomers", m_2 = "influenced", m_3 = "Gauss"; the TEXTUAL layer is weighted 0.5; the four semantic layers are weighted 0.125 each.

Layer l	Term t_i	$M(t_i, q)$	$\mathrm{tf}_q(t_i, q)$	$\mathrm{idf}(t_i, q)$	$\mathrm{w}(l)$	q_i
TEXTUAL	astronom	m_1	1.0	2.018	0.5	1.009
TEXTUAL	influenc	m_2	1.0	3.404	0.5	1.702
TEXTUAL	gauss	m_3	1.0	1.568	0.5	0.784
URI	dbpedia:Carl_Friedrich_Gauss	m_3	1.0	3.404	0.125	0.426
TYPE	yago:GermanMathematicians	m_3	0.030	2.624	0.125	0.010
TYPE	yago:NumberTheorists	m_3	0.030	2.583	0.125	0.010
TYPE	yago:FellowsOfTheRoyalSociety	m_3	0.030	1.057	0.125	0.004
TYPE	... other 18 terms ...	m_3	0.030	...	0.125	...
TYPE	yago:Astronomer109818343	m_1, m_3	0.114	1.432	0.125	0.020
TYPE	yago:Physicist110428004	m_1, m_3	0.114	0.958	0.125	0.014
TYPE	yago:Person100007846	m_1, m_3	0.114	0.003	0.125	∼0
TYPE	... other 9 terms ...	m_1, m_3	0.114	...	0.125	...
FRAME	⟨Subjective_influence-influence.v, dbpedia:Carl_Friedrich_Gauss⟩	m_2	0.333	5.802	0.125	0.242
FRAME	⟨Subjective_influence, dbpedia:Carl_Friedrich_Gauss⟩	m_2	0.333	5.802	0.125	0.242
FRAME	⟨Frame, Carl_Friedrich_Gauss⟩	m_2	0.333	3.499	0.125	0.146
TIME	day:1777-04-30	m_3	0.1	3.404	0.125	0.043
TIME	day:1855-02-23	m_3	0.1	3.404	0.125	0.043
TIME	century:1700	m_3	0.1	0.196	0.125	0.002
TIME	... other 7 terms	m_3	0.1	...	0.125	...

effect of making the similarity score obtained by matching m terms in a small document higher than the score obtained by matching the same m terms in a longer document. This normalization is known to be problematic in some document collections, is implemented differently and optionally disabled in real systems (e.g., Lucene and derivatives), and we deem it inappropriate in our scenario, where the document vector is expanded with a large amount of semantic terms whose number is generally not proportional to the document length.

We assign the weights of document and query vectors starting from the usual product of Term Frequency (tf) and Inverse Document Frequency (idf):

$$d_i = \mathrm{tf}_d(t_i, d) \cdot \mathrm{idf}(t_i, D) \tag{3}$$

$$q_i = \mathrm{tf}_q(t_i, q) \cdot \mathrm{idf}(t_i, D) \cdot \mathrm{w}(\mathrm{l}(t_i)) \tag{4}$$

The values of tf are computed differently for documents (tf_d) and queries (tf_q), while weights $\mathrm{w}(\mathrm{l}(t_i))$ quantify the importance of each layer. Given the form of Eq. 2, it suffices to apply $\mathrm{w}(\mathrm{l}(t_i))$ only to one of $\mathbf{d}$ and $\mathbf{q}$: we chose $\mathbf{q}$ so to allow

selecting weights on a per-query basis. Table 1 reports the tf_q, idf, w, and q_i values for the terms of the example query "astronomers influenced by Gauss".

Several schemes for computing tf and idf have been proposed in the literature. Among them, we adopt the following scheme,[7] where $f(t,o)$ and $f'(t,o)$ are two measures of the frequency of a term t in a text (document or query) x:

$$\mathrm{tf}_d(t,d) = 1 + \log(f(t,d)) \tag{5}$$

$$\mathrm{tf}_q(t,q) = f'(t,q) \tag{6}$$

$$\mathrm{idf}(t,D) = \log \frac{|D|}{|\{d \in D | f(t,d) > 0\}|} \tag{7}$$

The *raw frequency* $f(t,x)$ is defined as usual as the number of occurrences of term t in x. To account also for semantic terms, we denote with $M(t,x)$ the set of mentions in text x from where term t has been extracted, valid also for textual terms whose mentions are simply their occurrences in the text, and let $f(t,x) = |M(t,x)|$. The *normalized frequency* $f'(t,x)$ is newly introduced to account for the fact that multiple terms can be extracted from a single mention for the same semantic layer, differently from the textual case. It is defined as:

$$f'(t,x) = \sum_{m \in M(t,x)} \frac{1}{|T(m, \mathrm{l}(t))|} \tag{8}$$

where $T(m,l)$ is the set of terms of layer l extracted from mention m. Since $|T(m, \textsc{textual}))|$ is always 1, $f(t,x) = f'(t,x)$ for TEXTUAL terms. Note that Eq. 7 can indifferently use $f(t,x)$ or $f'(t,x)$.

The formulation of $f'(t,x)$ and its use in Eq. 6 aim at giving each mention the same importance when matching the query against the document collection. To explain, let's consider a query containing two mentions m_1 and m_2, with respectively n_1 and n_2 disjoint terms of a certain semantic layer (e.g., TYPE) extracted from each mention, $n_1 > n_2$; also assume that these terms have equal idf and tf_d values in the document collection. If we give these terms equal tf_q values, then a document matching the n_1 terms of m_1 (and nothing else) will be scored and ranked higher than a document matching the n_2 terms of m_2 (and nothing else). However, the fact that $n_1 > n_2$ does not reflect a preference of m_1 by the user; rather, it may merely reflect the fact that m_1 is described more richly than m_2 in the background knowledge. Our definition of normalized frequency corrects for this bias by assigning each mention a total weight of 1, which is equally distributed among the terms extracted from it for each semantic layer (e.g., weight $1/n_1$ assigned to terms of m_1, $1/n_2$ to terms of m_2).

For similar reasons, the use of $f'(t,x)$ in place of $f(t,x)$ in Eq. 5 would be inappropriate. Consider a query whose vector has a single TYPE term t (similar considerations apply to other semantic layers). Everything else being equal (e.g., idf values), two documents mentioning two entities of type t the same number

[7] Given the lack of normalization in $sim(d,q)$, our scheme can be roughly classified as ltn.ntn using the SMART notation; see http://bit.ly/weighting_schemes [22].

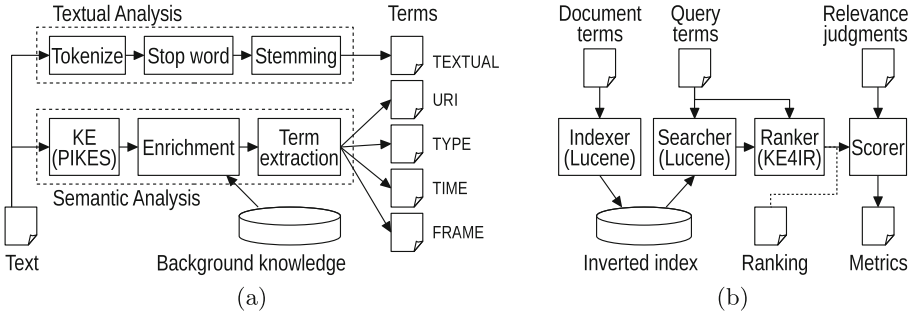

Fig. 1. Implementation: (a) term extraction; (b) indexing, searching and scoring.

of times should receive the same score. While this happen with $f(t, x)$, using $f'(t, x)$ the document mentioning the entity with fewest TYPE terms (beyond t) will be scored higher, although this clearly does not reflect a user preference.

4 Implementation

We built an evaluation infrastructure that implements the KE4IR approach presented in Sect. 3 and allows applying it on arbitrary documents and queries, measuring retrieval performances against gold relevance judgments. All the code is available for download on KE4IR website.[8]

Figure 1a shows the pipeline used to extract terms (with raw frequencies) from documents and queries, combining both textual and semantic analysis of input texts. Textual analysis generates the TEXTUAL layer employing standard components for text tokenization, stop word filtering, and stemming from Apache Lucene. Semantic analysis makes use of a KE tool for transforming the input text into an RDF knowledge graph where each instance is grounded to one or more mentions. This graph is then enriched with triples about selected URIs (DBpedia entities, YAGO types) retrieved from a persistent key-value store previously populated with the required LOD background knowledge;[9] the enrichment is done recursively and RDFS reasoning is done at the end to materialize inferences. The enriched graph is finally queried to extract semantic terms.

As KE extraction tool we use PIKES [3], a frame-based KE framework adopting a 2-phase approach. First—phase 1: *linguistic feature extraction*—an RDF graph of mentions is built by distilling the output of several state-of-the-art NLP tools, including Stanford CoreNLP[10] (tokenization, POS-tagging, lemmatization, NERC, TERN, parsing and coreference resolution), UKB[11] (WSD),

[8] http://pikes.fbk.eu/ke4ir.html.

[9] Subset of FrameBase ontology used in PIKES. Mapping-based properties with xsd:date, xsd:dateTime, xsd:gYear, and xsd:gYearMonth objects, YAGO types and type hierarchy from DBpedia 2015-04. All data available on KE4IR website.

[10] http://nlp.stanford.edu/software/corenlp.shtml.

[11] http://ixa2.si.ehu.es/ukb/.

DBpedia Spotlight[12] (EL), Mate-tools[13] and Semafor[14] (SRL). Then—phase 2: *knowledge distillation*—the mention graph is processed to distill the knowledge graph using SPARQL-like mapping rules, which are evaluated using RDFpro[15] [23], an RDF manipulation tool used also for RDFS reasoning. KE and the NLP tasks it relies on are computationally expensive. Using PIKES on a server with 24 cores (12 physical) and 192 GB RAM we obtained a throughput of ~700K tokens/h (~30K tokens/h core), corresponding to ~1200 documents/h for the document collection of Sect. 5 (570 tokens/document). While potentially inappropriate for a Web-scale deployment, this throughput is however adequate for small to medium-sized document collections (e.g., as encountered in corporate environments). Furthermore, larger collections can also be processed, with some loss in retrieval performances, by disabling the extraction of some of the layers.[16]

Figure 1b shows the workflow implemented for indexing extracted terms, executing queries and computing evaluation metrics. Document terms are directly indexed in a Lucene inverted index with their raw frequencies. At search time, query terms are OR-ed in a Lucene query that locates the documents containing at least one term (for which $sim(d, q) > 0$). Matched documents are scored and ranked externally to Lucene (for ease of testing) according to the KE4IR retrieval model of Sect. 3.2, starting from the term vectors of the query and the matched documents, and computing the necessary tf_d, tf_q, and idf values based on raw and normalized term frequencies and some statistics produced by Lucene (number of documents and document frequencies). The resulting ranking is compared with the gold relevance judgments to compute a comprehensive set of evaluation metrics, which are averaged along different queries.

5 Evaluation

In this section, we present an evaluation of KE4IR and discuss some insights emerged from it. All the evaluation materials are available on KE4IR website.

5.1 Evaluation Setup

KE4IR has been validated on the *ad-hoc* IR task, consisting in performing a set of queries over a document collection for which the list of relevance judgments is available. For the presented evaluation, we adopted the document collection created in [6], composed by a set of 331 documents and 35 queries. The relevance

[12] http://spotlight.dbpedia.org/.
[13] http://code.google.com/p/mate-tools/.
[14] http://www.cs.cmu.edu/~ark/SEMAFOR/.
[15] http://rdfpro.fbk.eu/.
[16] To give an idea, the impact of each semantic layer on the whole processing time for the document collection of Sect. 5 is: URI (3.5 %), TYPE (16.3 %), TIME (2.9 %), FRAME (77.3 %). Note also that substantial improvements of KE4IR indexing throughput can be achieved with further engineering and optimization, out-of-scope here.

of each document is expressed in a multi-value scale with scores going from 5 (the document contains exact information with respect to what the user is looking for) to 1 (the document is of no interest for the query). The peculiarity of this collection is the underlying semantic purpose with which it has been built. Indeed, the set of queries has been selected by varying from queries very close to keyword-based search (i.e., the query "Romanticism") to queries requiring semantic capabilities for retrieving relevant documents (i.e. "Aviation pioneers' publications"). In that work, the authors discuss some techniques exploiting manual annotations for semantic IR purposes. Unfortunately, their results and our results described next are not directly comparable, as the semantic techniques described in [6] are evaluated over annotations manually validated by experts, whereas we rely on totally automatic (and thus inevitably noisy) annotations.

Thus, we compared our approach against the two baselines introduced below:

- *Google baseline*: we exploited the Google custom search API for indexing pages containing our documents. The rationale behind this choice is to assess the performances of a commercial search engine, having as main challenge the "scalability" of indexing and retrieving documents, when a more custom document analysis is required. Google can be considered the same way as a black box, and we were not able to customize the way it analyzes text and computes document scores with respect to performed queries;
- *Textual baseline*: we indexed the raw text by adopting the standard Lucene library customized with the scoring formula described in Sect. 4. In our experiments, this customization provides the same (actually, slightly better) performances of a standard Lucene configuration,[17] and it also allows properly assessing the impact of semantic layers by excluding any interference related to slight differences in the definition of the scoring formula.

The protocol we used has been inspired by TREC [24]; however, due to the small size of the collection, we had to carry out some changes. Instead of drawing the precision/recall curve, we computed the precision values after the first (Prec@1), fifth (Prec@5), and tenth (Prec@10) document, respectively. The rationale behind this decision is the fact that the majority of search result click activity (89.8 %) happens on the first page of search results [16] corresponding to a set varying from 10 to 20 documents. Then, we provided two further metrics: (i) the Mean Average Precision (MAP) and (ii) the Normalized Discounted Cumulated Gain (NDCG) [25], computed both on the entire rank and after the first ten documents retrieved (resp., MAP@10 and NDCG@10). Validation on the NDCG metric is necessary in scenarios where multi-value relevance is used.

[17] For comparison, on KE4IR website we make available for download an instance of SOLR (a popular search engine based on Lucene) indexing the same document collection used in our evaluation, and we report on its performances on the test queries.

Table 2. Comparison of KE4IR against the two baselines.

Approach/System	Prec@1	Prec@5	Prec@10	NDCG	NDCG@10	MAP	MAP@10
Google	0.543	0.411	0.343	0.434	0.405	0.255	0.219
Textual	0.943	0.669	0.453	0.832	0.782	0.733	0.681
KE4IR	**0.971**	**0.680**	**0.474**	**0.854**	**0.806**	**0.758**	**0.713**
KE4IR vs. Textual	3.03 %	1.71 %	4.55 %	2.64 %	2.99 %	3.50 %	4.74 %
p-value (paired t-test)	0.324	0.160	0.070	**0.003**	**0.015**	**0.024**	**0.029**
p-value (approx. random.)	1.000	0.496	0.111	**0.003**	**0.020**	**0.020**	**0.030**

Table 3. KE4IR results using different layer combinations.

Layers	Prec@1	Prec@5	Prec@10	NDCG	NDCG@10	MAP	MAP@10
TEXTUAL,URI,TYPE, FRAME,TIME	**0.971**	**0.680**	**0.474**	**0.854**	**0.806**	**0.758**	**0.713**
TEXTUAL,URI,TYPE,FRAME	**0.971**	**0.680**	**0.474**	0.853	0.804	0.757	0.712
TEXTUAL,URI,TYPE,TIME	**0.971**	**0.680**	**0.474**	0.851	0.802	0.757	0.712
TEXTUAL,URI,TYPE	**0.971**	**0.680**	**0.474**	0.849	0.801	0.755	0.710
TEXTUAL,URI,FRAME,TIME	**0.971**	0.674	0.465	0.844	0.796	0.750	0.702
TEXTUAL,URI,FRAME	**0.971**	0.674	0.465	0.842	0.795	0.749	0.702
TEXTUAL,URI,TIME	**0.971**	0.674	0.465	0.840	0.791	0.747	0.700
TEXTUAL,URI	**0.971**	0.674	0.465	0.837	0.791	0.747	0.700
TEXTUAL,TYPE,FRAME,TIME	0.943	0.674	0.471	0.848	0.799	0.745	0.700
TEXTUAL,TYPE,TIME	0.943	0.674	0.471	0.843	0.794	0.743	0.697
TEXTUAL,TYPE,FRAME	0.943	0.674	0.468	0.847	0.797	0.743	0.695
TEXTUAL,FRAME,TIME	0.943	0.674	0.462	0.842	0.793	0.741	0.693
TEXTUAL,TYPE	0.943	0.674	0.468	0.842	0.792	0.740	0.693
TEXTUAL,TIME	0.943	0.669	0.462	0.836	0.786	0.737	0.689
TEXTUAL,FRAME	0.943	0.674	0.453	0.839	0.789	0.737	0.686

5.2 Overall Evaluation Results

We report here an overview of the results obtained, using equal weights for textual and semantic information in KE4IR, i.e., w(TEXTUAL) = w(SEMANTICS) = 0.5, with w(SEMANTICS) divided equally among semantic layers. We also provide a first analysis of KE4IR behavior using different layer combinations.

Comparison with the Baselines. Table 2 shows the comparison between the results achieved by KE4IR exploiting all the semantic layers, and the ones obtained by the proposed baselines. It is possible to see that KE4IR matches or outperforms the baselines for all the considered metrics. With respect to the textual baseline, the higher improvements are registered on the MAP, MAP@10, and Prec@10 values that quantify the capability of the proposed approach of producing an effective documents ranking. While the gains on the MAP and MAP@10

metrics assess only the retrieval of relevant documents without considering their relevance scores, the improvements obtained on the NDCG and NDCG@10 metrics highlight that produced rankings are effective also from a quality point of view. The improvements over the textual baseline are statistically significant for MAP, MAP@10, NDCG, and NDCG@10 (significance threshold 0.05), based on the p-values computed with the paired t-test (claimed as one of the best tests for IR in [26]) and the approximate randomization test [27]. With respect to the Google baseline, the marked difference of performances derives from Google returning far less results than KE4IR for the evaluation queries. Indeed, large-scale (web-scale) IR systems such as Google are heavily tuned for precision, as any query usually matches a large number of documents and the problem is to discard the irrelevant ones. In our context, small-scale IR, systems as our tool deal with fewer documents and hence they must be tuned also for recall.

Impact of Various Layer Combinations. A detailed analysis of the results obtained using different layer combinations in KE4IR is shown in Table 3. Combining all the semantic layers produces the best performances for all the considered metrics. In particular, the URI layer seems to be the most effective, as it is always included in the top settings for MAP. These results show that the integration of different semantic information leads to a general improvement of the effectiveness of the IR task, in line with the purpose of the proposed approach.

5.3 Query-by-Query Analysis

To complete the analysis of the overall results, we investigate the performances of the system query-by-query, discussing four representative queries more in-depth.

Impact of Each Single Layer. Table 4 shows, query by query, the impact of each semantic layer on system effectiveness. The first column contains the query identifier; from the second to the fifth columns, we show the comparison between the NDCG@10 computed using textual and single-layer semantic information, and the NDCG@10 computed by using only the textual information. From the sixth to the ninth columns, the same values are shown for the MAP metric. We selected only the MAP and the NDCG@10 metrics because those are the most indicative metrics for evaluating the performances of IR systems in general (MAP), and for deployment in a real-world environment (NDCG@10).

The TYPE layer affects the highest number of queries, but for some of them (e.g., "q28") its contribution is negative. This issue is likely a consequence of the large quantity of information inserted when the query is expanded with TYPE terms, especially the ones corresponding to super-types of entities and concepts mentioned in the query, which may lead documents scarcely related to the query to be matched by the system. Indeed, by injecting too much information in queries, it is possible to obtain a detrimental effect as shown in [28], where the authors discuss query expansion trade-offs and impact on IR effectiveness.

The URI layer also impacts on many queries with both positive and negative effects. Differences on NDCG@10 and MAP scores are larger than the ones resulting from the TYPE layer (see, e.g., queries "q16" and "q28"), reflecting the

Table 4. Differences on NDCG@10 and MAP obtained using TEXTUAL and a single semantic layer with respect to TEXTUAL only. Queries with no performance change are omitted ("q01", "q04", "q06", "q12", "q26", "q32", "q34" and "q46"). Note that semantic information may be available even if no difference is observed.

Query	Δ NDCG@10				Δ MAP			
	URI	TYPE	FRAME	TIME	URI	TYPE	FRAME	TIME
q02						0.001		
q03	0.002							
q07						0.005		
q08		0.049				0.012		
q09	0.029							
q10		0.093				0.061		
q13	0.066	0.066						
q14		0.015				0.005		
q16	0.018				0.283			
q17						0.026		0.015
q18		0.002				0.011		
q19						0.014		0.007
q22						0.021		−0.001
q23						0.012		0.003
q24					−0.012	−0.009		
q25	0.007	0.002			0.004	0.004		
q27	−0.007	0.011	0.218		0.011	0.020	0.135	
q28	−0.117	−0.016			−0.090	−0.042		
q29	0.002							
q36		−0.016						
q37						0.013		
q38		0.028				−0.003		
q40	0.054	0.007	0.017	0.005	0.030	0.021	0.021	0.021
q41	0.104	−0.004			0.140	0.010		0.032
q42		0.011				0.023		
q44	0.149	0.091		0.141	0.131	0.049		0.088
q45	−0.002	−0.002		−0.002		−0.007		−0.007

fact that URI terms impact more on document scores (with respect to the textual layer) since they are generally more selective (high idf) and often correspond to entities mentioned multiple times in a document (high tf).

The FRAME and TIME layers, where available, have almost always a positive impact on the performances (esp. for "q27" and "q44"). The FRAME layer affects the smallest number of queries. As described in Sect. 3, this layer describes relations between entities detected in the text, and thus requires to have a query structure that is more complex with respect to a simple keyword-based one.

Analysis of Selected Queries. While a comprehensive analysis of the performances of each query is not doable due to lack of space, we select four queries giving hints about pros and cons of using semantic information in IR. Table 5 shows these queries and their performances when using all the semantic layers.

Table 5. Results for selected queries using all the semantic layers.

Query	Query text	Δ NDCG@10	Δ MAP
q27	Nazis confiscate or destroy art and literature	0.154	0.099
q28	Modern age in English literature	−0.117	−0.095
q44	Napoleon's Russian campaign	0.151	0.147
q46	First woman who won a nobel prize	0	0

Query "q46" is an example where semantic information has no effects. This is because entities at different granularities are injected in the URI layers of query and documents. Specifically, the query is annotated with dbpedia:Nobel_Prize, while relevant documents have annotations like dbpedia:Nobel_Prize_in_X, where X is one of the disciplines for which Nobel Prizes are assigned. Unfortunately, these entities are not related in DBpedia (also in terms of types), thus it is not possible to expand the query in order to find matches with relevant documents.

Query "q28" is an example where worse performances are achieved by using semantic information, due to Entity Linking errors. From the query, two URI terms (and related TYPE terms) are correctly extracted: dbpedia:Modern_history, with no matches in the document collection, and dbpedia:English_literature, with 12 matches. Of these matches, 11 are incorrect and refer to irrelevant documents where dbpedia:English_literature is wrongly linked to mentions of other "English" things (e.g. "English scholar", "English society", "English medical herbs").

Queries "q27" and "q44" are examples where semantic information significantly boost performances. In "q44", the correct link to dbpedia:Napoleon and the type and time information associated to that entity in DBpedia allow extracting URI, TYPE and TIME terms that greatly help ranking relevant documents higher. In "q27", the major improvement derives from the extraction and matching of FRAME term ⟨framebase:frame-Destroying, dbpedia:Nazism⟩; while TIME information is also available (as dbpedia:Nazism is linked to category dbc:20th_century in DBpedia), our KE4IR implementation is not sophisticated enough to exploit it.

5.4 Balancing Semantic and Textual Content

In our work, we combine both textual and semantic content to improve the performances of IR. While in previous analyses we assigned equal weights to

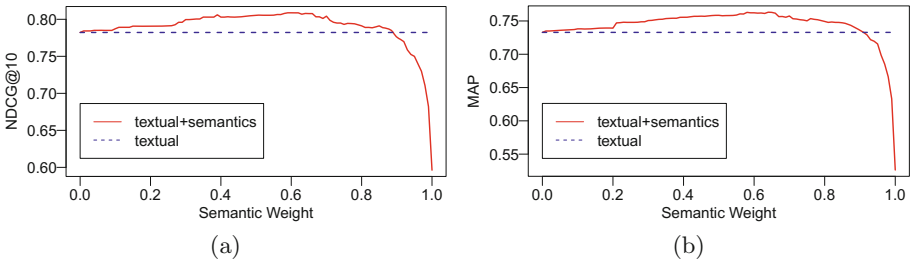

Fig. 2. Graphs showing the trend of (a) NDCG@10 and (b) MAP based on the quantity of semantic information considered with respect to the textual one.

semantic and textual information, here we experiment with different balances. Figure 2 shows how the NDCG@10 and MAP metrics change when the importance given to the semantic information changes as well. On the y-axis, we have the NDCG@10 (Fig. 2a) and MAP (Fig. 2b) values, while on the x-axis we have the weight w(SEMANTICS) assigned to all the semantic information and divided equally among semantic layers, with w(TEXTUAL) $= 1 -$ w(SEMANTICS); a w(SEMANTICS) value of 0.0 means that only textual information is used (and no semantic content), while a value of 1.0 means that only semantic information is used (and no textual content). Results show that semantic information impacts positively on system performances up to w(SEMANTICS) ≤ 0.89 for NDGC@10 and w(SEMANTICS) ≤ 0.92 for MAP, reaching the highest scores around 0.61 and 0.65, respectively. Similar behaviors can be observed for NDCG and MAP@10. The highest scores obtained (NDCG@10 $= 0.809$, MAP $= 0.763$) are better than the scores reported where equal textual and semantic weights were intuitively used, suggesting the importance of a methodology for optimal weight tuning.

6 Concluding Remarks and Future Work

In this paper we investigated the benefits of using semantic content automatically extracted from text for Information Retrieval. Building on the Vector Space Model, we designed and implemented an approach, KE4IR, where both queries and documents are processed to extract semantic content such as entities, types, semantic frames, and temporal information. By evaluating our approach on a state-of-the-art document collection, we showed that complementing the textual information of queries and documents with the content resulting by processing them with typical knowledge extraction tools, enables to outperform document retrieval performances when only textual information is exploited.

Performance measured with different layer combinations shows that the aggregation of different semantic layers leads to effective rankings of relevant documents even in a multi-value relevance setting. The analysis of the NDCG and MAP values, representing the most meaningful metrics for evaluating an IR system, both in general and with respect to common user behaviors, validated the possibility of deploying KE4IR in a real-world environment.

Starting from the results obtained in this first experience, future work on the platform will touch different aspects. The evaluation here reported may be considered a first step for observing the behavior of the approach under different configurations and for enabling an analysis of the impact of each semantic layer. Extending the evaluation campaign to additional, larger document collections (e.g., TREC WT10g, ClueWeb) will be the next step for comparing the presented platform in different environments where further issues have also to be addressed as, for example, the scalability of the entire pipeline.

Results gave interesting insights about the components that should be improved for augmenting the effectiveness of the retrieval system. As shown in Table 5, concerning queries "q28" and "q46", single issues in the linking phase may lead to poor results. Thus, instead of trying to enrich as much text as possible with linked information coming from different knowledge bases, the use of approaches favoring precision of suggested links instead of recall may be the best strategy for obtaining a better average improvement of system effectiveness. Similar considerations can be done about all the other semantic layers.

Finally, in the presented version of KE4IR, we considered only general-purpose knowledge bases for enriching documents. However, the deployment in more domain-specific contexts would require the use of domain-specific resources able to provide more effective annotations. For instance, domain-specific KBs can be used with KB-agnostic entity linking tools to extract domain-specific URI and TYPE terms. For FRAME terms, domain-specific frames can be defined and annotated in a corpus to retrain the SRL tools used. We plan to validate these strategies to assess the usability of KE4IR with domain-specific document collection.

References

1. Gangemi, A., Draicchio, F., Presutti, V., Nuzzolese, A.G., Recupero, D.R.: A machine reader for the semantic web. In: Demos of ISWC, pp. 149–152 (2013)
2. Rospocher, M., van Erp, M., Vossen, P., Fokkens, A., Aldabe, I., Rigau, G., Soroa, A., Ploeger, T., Bogaard, T.: Building event-centric knowledge graphs from news. J. Web Semant. (to appear)
3. Corcoglioniti, F., Rospocher, M., Palmero Aprosio, A.: A 2-phase frame-based knowledge extraction framework. In: Proceedings of ACM Symposium on Applied Computing (SAC 2016) (2016, to appear)
4. Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P.N., Hellmann, S., Morsey, M., van Kleef, P., Auer, S., Bizer, C.: DBpedia - a large-scale, multilingual knowledge base extracted from Wikipedia. Semantic Web **6**(2), 167–195 (2015)
5. Salton, G., Wong, A., Yang, C.S.: A vector space model for automatic indexing. Commun. ACM **18**(11), 613–620 (1975)
6. Waitelonis, J., Exeler, C., Sack, H.: Linked data enabled generalized vector space model to improve document retrieval. In: Proceedings of NLP & DBpedia 2015 Workshop in Conjunction with 14th International Semantic Web Conference (ISWC 2015). CEUR Workshop Proceedings (2015)
7. Croft, W.B.: User-specified domain knowledge for document retrieval. In: Bernardi, L.R., Rabitti, F. (eds.) SIGIR, pp. 201–206. ACM (1986)

8. Gonzalo, J., Verdejo, F., Chugur, I., Cigarrán, J.: Indexing with WordNet synsets can improve text retrieval. CoRR (1998)
9. Fellbaum, C. (ed.): WordNet: An Electonic Lexical Database. MIT Press, Cambridge (1998)
10. Dridi, O.: Ontology-based information retrieval: overview and new proposition. In: RCIS, pp. 421–426 (2008)
11. Tomassen, S.L.: Research on ontology-driven information retrieval. In: Meersman, R., Tari, Z., Herrero, P. (eds.) OTM 2006 Workshops. LNCS, vol. 4278, pp. 1460–1468. Springer, Heidelberg (2006)
12. Castells, P., Fernández, M., Vallet, D.: An adaptation of the vector-space model for ontology-based information retrieval. IEEE Trans. Knowl. Data Eng. **19**(2), 261–272 (2007)
13. Vallet, D., Fernández, M., Castells, P.: An ontology-based information retrieval model. In: Gómez-Pérez, A., Euzenat, J. (eds.) ESWC 2005. LNCS, vol. 3532, pp. 455–470. Springer, Heidelberg (2005)
14. Jimeno-Yepes, A., Llavori, R.B., Rebholz-Schuhmann, D.: Ontology refinement for improved information retrieval. Inf. Process. Manage. **46**(4), 426–435 (2010)
15. Fernández, M., Cantador, I., Lopez, V., Vallet, D., Castells, P., Motta, E.: Semantically enhanced information retrieval: an ontology-based approach. J. Web Sem. **9**(4), 434–452 (2011)
16. Spink, A., Jansen, B., Blakely, C., Koshman, S.: A study of results overlap and uniqueness among major web search engines. Inf. Process. Manage. **42**(5), 1379–1391 (2006)
17. Stojanovic, N.: An approach for defining relevance in the ontology-based information retrieval. In: Web Intelligence, pp. 359–365 (2005)
18. Baziz, M., Boughanem, M., Pasi, G., Prade, H.: An information retrieval driven by ontology: from query to document expansion. In: RIAO (2007)
19. Rouces, J., de Melo, G., Hose, K.: FrameBase: representing n-ary relations using semantic frames. In: Gandon, F., Sabou, M., Sack, H., d'Amato, C., Cudré-Mauroux, P., Zimmermann, A. (eds.) ESWC 2015. LNCS, vol. 9088, pp. 505–521. Springer, Heidelberg (2015)
20. Hoffart, J., Suchanek, F.M., Berberich, K., Weikum, G.: YAGO2: a spatially and temporally enhanced knowledge base from Wikipedia. Artif. Intell. **194**, 28–61 (2013)
21. da Costa Pereira, C., Dragoni, M., Pasi, G.: Multidimensional relevance: prioritized aggregation in a personalized information retrieval setting. Inf. Process. Manage. **48**(2), 340–357 (2012)
22. Manning, C.D., Raghavan, P., Schütze, H., et al.: Introduction to Information Retrieval, vol. 1. Cambridge University Press, Cambridge (2008)
23. Corcoglioniti, F., Rospocher, M., Mostarda, M., Amadori, M.: Processing billions of RDF triples on a single machine using streaming and sorting. In: ACM SAC, pp. 368–375 (2015)
24. Voorhees, E., Harman, D.: Overview of the sixth text retrieval conference (trec-6). In: TREC, pp. 1–24 (1997)
25. Järvelin, K., Kekäläinen, J.: Cumulated gain-based evaluation of IR techniques. ACM Trans. Inf. Syst. **20**(4), 422–446 (2002)
26. Sanderson, M., Zobel, J.: Information retrieval system evaluation: effort, sensitivity, and reliability. In: SIGIR, pp. 162–169. ACM (2005)
27. Noreen, E.W.: Computer-Intensive Methods for Testing Hypotheses: An Introduction. Wiley, New York (1989)
28. Abdelali, A., Cowie, J., Soliman, H.: Improving query precision using semantic expansion. Inf. Process. Manage. **43**(3), 705–716 (2007)