A 2-phase Frame-based Knowledge Extraction Framework

Francesco Corcoglioniti, Marco Rospocher, and Alessio Palmero Aprosio
Fondazione Bruno Kessler
Via Sommarive 18, 38123 Trento, Italy
{corcoglio,rospocher,aprosio@fbk.eu}

ABSTRACT

We present an approach for extracting knowledge from natural language English texts where processing is decoupled in two phases. The first phase comprises several standard NLP tasks whose results are integrated in a single RDF graph of *mentions*. The second phase processes the mention graph with SPARQL-like mapping rules to produce a knowledge graph organized around *semantic frames* (i.e., prototypical descriptions of events and situations). The decoupling allows: (i) choosing different tools for the NLP tasks without affecting the remaining computation; (ii) combining the outputs of different NLP tasks in non-trivial ways, leveraging their integrated and coherent representation in a mention graph; and (iii) relating each piece of extracted knowledge to the mention(s) it comes from, leveraging the single RDF representation. We evaluate precision and recall of our approach on a gold standard, showing its competitiveness w.r.t. the state of the art. We also evaluate execution times and (sampled) accuracy on a corpus of 110K Wikipedia pages, showing the applicability of the approach on large corpora.

CCS Concepts

•Computing methodologies → Information extraction; Knowledge representation and reasoning;

Keywords

Knowledge Extraction; Natural Language Processing; RDF Knowledge Graph; Frame Detection; SPARQL Rules

1. INTRODUCTION

Extracting knowledge from text is a challenging interdisciplinary task standing at the crossroad of Natural Language Processing (NLP), Knowledge Representation (KR), and Semantic Web (SW). Typically, comprehensive Knowledge Extraction (KE) approaches (e.g., LODifier [1], NewsReader [14], FRED [6]) build on NLP pipelines performing several subtasks such as: Named Entity Recognition and Classification (NERC), Entity Linking (EL), Temporal Expression Recognition and Normalization (TERN), and Semantic Role Labeling (SRL). The outputs of NLP tools are then combined and exposed in some KR/SW format.

In this paper we present PIKES,[1] a new frame-based knowledge extraction framework. PIKES extracts entities and complex relations between entities by identifying semantic *frames* in a text, i.e., events and situations describing n-ary relations between entities (i.e., *frame participants*). Following the neo-Davidsonian representation [12], these frames are represented as reified objects, connected to each of their participants by means of properties describing the semantic role (aka, *frame-role*) of the participants in the frame.

PIKES adopts a novel 2-phase approach for KE. In Phase 1 (*linguistic feature extraction*), by performing several standard NLP tasks, a mention-based structured representation of the input text is built, organizing all the annotations produced by NLP tools in an RDF graph of *mentions* (i.e., spans of text denoting some entities or facts), according to SW vocabularies and principles. Then, in Phase 2 (*knowledge distillation*), the mention graph is processed to distill a *knowledge graph*, where each node uniquely identifies an entity of the world, event or situation, and arcs represent relations between them (e.g., the participation and role of an entity in an event). This knowledge graph represents the knowledge conveyed by the text, in a way that abstracts from the specific occurrences of entities and relations in it.

Decoupling the linguistic feature extraction and knowledge distillation phases, and formalizing all the content produced during the extraction according to a well-defined data model, have several benefits. First, as the mention graph is organized according to the expected output of standard NLP tasks, and not the output of specific tools, it enables exploiting and experimenting with different tools that perform the same NLP tasks, without affecting the processing and techniques used in the knowledge distillation phase. Second, as the mention graph coherently organizes the linguistic information coming from different linguistic analyses, it favors

SAC 2016, April 04-08, 2016, Pisa, Italy
©2016 ACM. ISBN 978-1-4503-3739-7/16/04…$15.00
DOI: http://dx.doi.org/10.1145/2851613.2851845

[1]PIKES (http://pikes.fbk.eu/) is released as open-source software, and a fully-working online demo and a walkthrough demonstration video are available on the website.

the development of knowledge distillation techniques that combine in non-trivial ways the outputs of different NLP tasks. Specifically, PIKES implements a rule-based knowledge distillation technique, where SPARQL-like rules exploit the RDF representation of the linguistic features produced in the first phase, as well as additional RDF knowledge available in external repositories (aka *background knowledge*), to build the knowledge graph representation of the input text. Third, as the output of the linguistic features extraction and knowledge distillation phases is exposed in the same format (RDF), the mention graph and the knowledge graph content can be easily related, leveraging the *named graph* extension of RDF, enabling for instance the tracking of the mentions from which some knowledge (an entity, a fact) was extracted.

Besides offering all these advantages, PIKES quality and scalability performances are very competitive compared to state-of-the-art tools for knowledge extraction for the SW. Analogously to other approaches, we evaluated the performance of PIKES on the specific task of detecting and representing frames and frame-role relations occurring in a text. The results (precision: 0.713, recall: 0.508) show that PIKES is capable of extracting quality knowledge from text. Furthermore, we assessed the capability of PIKES in processing large corpora, by building a knowledge graph from a collection of 110K Wikipedia-like texts in about 507 core hours.

The paper is organized as follows. In Section 2 we briefly review the state of the art in KE for SW. In Section 3 we describe PIKES 2-phase knowledge extraction approach, detailing the data representation model and the processing performed in each phase. In Section 4 we present the results of an evaluation of the performance of PIKES, both in terms of quality and efficiency. Section 5 concludes.

2. STATE OF THE ART

The work presented in this paper falls in the broad area of Information Extraction (IE) [8]. IE deals with the extraction of structured knowledge from unstructured resources by means of NLP techniques. In particular, Ontology Population [2] deals with information extraction techniques to populate the ABox (i.e., instances and facts on them) of an ontology. Ontology Population has become quite popular in the last decade, thanks to the spreading of Linked Data and Semantic Web technologies (for a review of relevant state of the art up to 2011, see [13]). In particular, in the last few years (2012 onward) several contributions were presented. We briefly report the most relevant ones for this work.

LODifier [1] extracts Discourse Representation Structures (DRSs) from a text by exploiting Deep Semantic Analysis, mapping them to RDF triples using transformation rules. FRED [6] also builds on DRSs, but differently from LODifier, DRSs are mapped to linguistic frames in VerbNet[2] and FrameNet,[3] which in turn are transformed in RDF/OWL via Ontology Design Patterns.[4] Graphia [5] is an open IE pipeline producing an entity-centric RDF representation of text in the form of Structured Discourse Graphs. In NewsReader [14], a comprehensive processing pipeline was developed for extracting and coreferring events and entities from large (cross-lingual) news corpora. To the best of our knowledge, none of these tools adopt a 2-phase approach for KE where all extracted content, including the intermediate linguistic information, is exposed in RDF according to a comprehensive data model.

A recent survey of KE tools for the SW [7] showed that FRED stands out on various KE tasks, including frame and frame-role detection. Therefore, in Section 4 we compare the performance of PIKES and FRED[5] on the same text corpus used for the evaluation conducted in [7], highlighting further differences of the content they extract.

3. THE APPROACH

PIKES frame-based KE approach works in two main phases, exemplified in Figure 1 for a simple running example document comprising two sentences: "*G. W. Bush and Bono are very strong supporters of the fight of HIV in Africa. Their March 2002 meeting resulted in a 5 billion dollar aid.*".

In the first phase, called *linguistic feature extraction* (detailed in Section 3.2), PIKES performs several standard NLP tasks – POS tagging, Word Sense Disambiguation, NERC, TERN, EL, SRL, Dependency Parsing, Coreference Resolution – to build a linguistic-oriented structured representation of the input text, centered around the notion of 'mentions'. A *mention* is a piece of text denoting some entity, event, temporal expression, or even fact. Mentions are distilled from the linguistic annotations obtained performing NLP tasks, and are characterized by *attributes*, that are specific to a particular type of mention. Mentions aim at representing, in a structured form, all the linguistic information needed to extract the knowledge conveyed by the text.

The notion of mention generalizes the distinct annotations produced by tools performing typical NLP tasks and usually coming in different formats, thus enabling to handle uniformly the different types of linguistic annotations produced. This aspect is exploited in the second phase of the approach, called *knowledge distillation* (detailed in Section 3.3), where knowledge-oriented techniques, potentially combining structured information spread over different mentions, extract instances of events and entities, as well as facts about them, abstracting from their actual occurrences in text.

3.1 Data Model for Information Extraction

In order to represent the mention and instance contents, as well as the links to the pieces of text where they come from, we encode all the information in an RDF model organized in three distinct yet interlinked representation layers: *Text*, *Mention*, and *Instance* layers. This model is inspired to (and is compliant with) the model presented in [3], where it serves as data model for a framework – the KnowledgeStore – supporting the interlinking of unstructured and structured content. That is, the input and output consumed and produced by PIKES can be used to populate a KnowledgeStore framework, where all the content processed and produced can be accessed, navigated, and queried in an in-

[2]http://verbs.colorado.edu/

[3]http://framenet.icsi.berkeley.edu/

[4]http://ontologydesignpatterns.org/

[5]Note that FRED also aims at extracting TBox content from text, something not considered here.

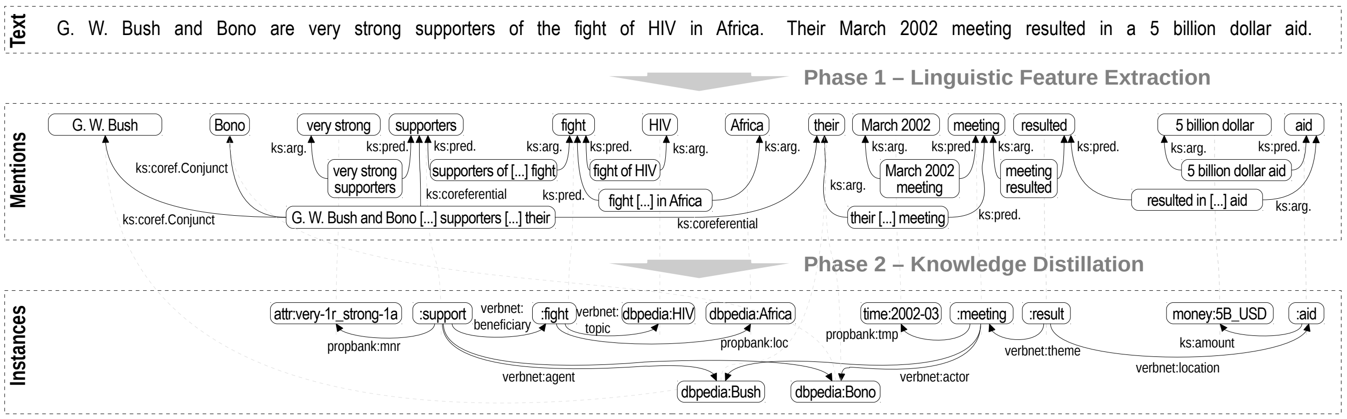

Figure 1: (excerpt) Representation of knowledge extracted from the running example. Rounded boxes are mentions (actually identified by NIF URIs) in the Mention layer, and instances in the Instance layer.

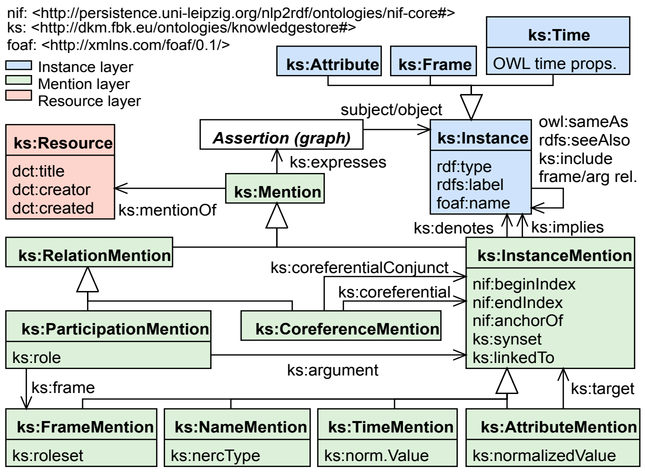

Figure 2: Overview of the representation model.

tegrated fashion. The three representation layers and their relations are described next. Figure 2 provides an overview of the main model elements, while Figure 1 summarizes the instantiation of the model for the running example.

Text layer This is the textual content from which we extract knowledge. It consists of text resources (i.e., documents) identified by URIs. Each resource has a raw text and an accompanying RDF description rooted at the resource URI, consisting of metadata triples (e.g., the resource `dct:title` or the document creation time `dct:created`) that are expressed with standard vocabularies (e.g., Dublin Core[6]) and may be exploited while processing the text.

Mention layer This layer consists of mentions (class `ks:Mention`), i.e., pieces of text denoting something of interest, such as an entity or relation. As shown in Figure 2, we identified different types of mentions, according to the kind of knowledge that can be extracted from them. More in detail, we have two disjoint top-level classes: class `ks:InstanceMention`, which corresponds to mentions of instances of the domain of discourse, such as entities, events, and so on; and, class `ks:RelationMention`, which covers the possible relations grounded in the text between `ks:InstanceMentions`, such as coreference and frame-role relations. These classes are further specialized in several subclasses, not necessarily disjoint, that identify special typologies of mentions:

- `ks:NameMentions` denote named instances (e.g., mention 'Bono' in Figure 1) and result from tasks such as NERC that identifies spans of text denoting entities of predefined categories (e.g., persons or organizations);
- `ks:FrameMentions` and `ks:ParticipationMentions` denote respectively frame instances (e.g., mention 'fight' in Figure 1) and the links between argument instances and participated frame instances (e.g., mention 'fight of HIV' linking argument 'HIV' to frame 'fight'); they are the result of the SRL task that identifies frames with their predicates and arguments in a text;
- `ks:TimeMentions` denote temporal intervals (such as 'March 2002') resulting from the TERN task that identifies and normalizes (i.e., maps to a standard representation) temporal expressions in a text; we encode the normalized value using the OWL Time ontology;[7]
- `ks:AttributeMentions` denote instances in the value space of lexical attributes (e.g., mention 'very strong') and are extracted via NERC and Dependency Parsing (DP), i.e., a kind of syntactic parsing identifying dependencies between tokens of a sentence;
- `ks:CoreferenceMentions` denote spans of text with the same referent (e.g., 'G. W. Bush and Bono', 'supporters', 'they'), result of Coreference Resolution.

Mentions are characterized by attributes whose values are extracted from the annotations produced by the aforementioned NLP tasks (e.g., attribute `ks:nercType` from NERC, `ks:normalizedValue` from TERN, `ks:roleset` and `ks:role` from SRL frame type and roles), as well as other processing tasks such as: Tokenization, i.e., splitting a text into tokens; Part-of-Speech (POS) tagging, i.e., assigning each token to a POS (e.g. proper noun); Word Sense Disam-

[6]http://dublincore.org/

[7]http://www.w3.org/TR/owl-time/

biguation (WSD), i.e., assigning each token to a WordNet synset (attribute `ks:synset`); Entity Linking (EL), i.e., linking a span of text to the corresponding entity in DBpedia or other resource (attribute `ks:linkedTo`); and so on. Whenever possible, resources from standard RDF/OWL vocabularies (e.g., NIF[8]) were reused to define mention attributes.

Instance layer The instance layer describes the things of interest contained in a textual resource, abstracting from the actual ways they are expressed in the text. The main objects are instances (class `ks:Instance` in Figure 2) of persons, organizations, locations, frames, dates and other entities of the domain of discourse. Instances are typed w.r.t. various taxonomies, are enriched with textual properties (`rdfs:label` and `foaf:name`), and are linked by a number of relations, including `owl:sameAs` triples triggered by `ks:CoreferenceMention`s and frame-argument participation triples triggered by `ks:ParticipationMention`s (e.g., `:fight verbnet:topic :HIV`), where the property conveys the role played by the argument. In this representation, frame instances play a crucial role in relating instances, basically reifying complex relationships and overcoming the limits of the standard binary relation representation of RDF.

Inter-layer relations Mention and Text layers are related by property `ks:mentionOf` that links a `ks:Mention` to the `ks:Resource` it belongs to. Mention and Instance layers are related by three properties: `ks:denotes`, `ks:implies`, and `ks:expresses`. `ks:denotes` is a functional property linking a `ks:InstanceMention` to the `ks:Instance` it denotes (e.g., it links mention 'March 2002' to time instance `time:2002-03`). `ks:implies` is associated to `ks:FrameMention`s and identifies another instance, besides the denoted one, whose existence is implied by the mention. This situation occurs when the mention refers to a noun predicate being an *argument nominalization* (e.g., 'supporter' in Figure 1), which denotes the argument (the person acting as supporter) but also implies the existence of the frame (the support event). The third property, `ks:expresses`, links a `ks:Mention` to the Instance layer triples it expresses (i.e., that can be derived from it). Instead of reifying the triple we use named graphs to keep the RDF representation more compact. In details, each triple of the Instance layer is placed in a named graph that represents the set of mentions (possibly one) that express that particular triple; `ks:expresses` is then asserted between each mention URI and the graph URI. E.g.:

```
:gXYZ { :fight rdf:type framenet:Intentionally_act . }
mention:fight ks:expresses :gXYZ .
```

(where `mention:fight` corresponds to mention 'fight' in Figure 1) states that the fact '`:fight` instantiates the Intentionally_act frame from FrameNet' was distilled from mention `mention:fight`. Clearly, a named graph may contain many triples, meaning that they were all extracted from the same mention (e.g., different type triples of an instance), and a named graph may be related to multiple mentions via multiple `ks:expresses` triples, meaning that all the triples in the named graph were extracted from each one of these mentions. Put together, properties `ks:denotes`, `ks:implies` and `ks:expresses` allow any instance and triple in the Instance

Table 1: NLP tasks backing each mention class.

Mention class	POS	NERC	TERN	EL	WSD	SRL	COREF	DP
`ks:NameMention`	✓	✓		✓				
`ks:TimeMention`		✓	✓					
`ks:AttributeMention`	✓				✓			✓
`ks:FrameMention`						✓		
`ks:InstanceMention` (plain)	✓			✓	✓			
`ks:ParticipationMention`						✓		✓
`ks:CoreferenceMention`							✓	✓

layer to be referred to the mention(s) from where it was derived, thus enabling a fine grained tracking of the specific piece of text from where a bit of knowledge was extracted.

3.2 Linguistic Feature Extraction

Given an input text, several NLP tasks are applied to produce the necessary NLP annotations from which a set of interlinked mentions is extracted. Table 1 reports the NLP tasks (columns) and the mention classes (rows) considered, showing which tasks contribute to each mention class. Mapping of NLP annotations to mentions is performed as follows:

- `ks:NameMention`s are mainly derived from NERC (except for time and date categories) but also from tokens POS-tagged as proper nouns and not marked by NERC; they are enriched with DBpedia links from EL.
- `ks:TimeMention`s are derived from TERN and NERC (time and date entity categories). For the normalized time value we map from the standard NLP ISO-TimeML[9] representation to the OWL Time ontology.
- `ks:AttributeMention`s are derived from tokens POS-tagged as adjective or adverb (e.g., 'strong' in Figure 1) augmented with adjectival or adverbial modifiers based on DP (e.g., 'very' for 'very strong'); attribute `ks:normalizedValue` is computed from WSD synsets (e.g., `attr:very-1r_strong-1a` derives from adjective synset strong#1 and adverb synset very#1).
- `ks:FrameMention`s (with attribute `ks:roleset`) are created for verb and noun predicates recognized by SRL.
- Plain `ks:InstanceMention`s are created for each token POS-tagged as pronoun or common noun and not covered by another mention, and are enriched with DBpedia links from EL and WordNet synsets from WSD.
- `ks:ParticipationMention`s are extracted from each text span marked as predicate argument by SRL (e.g., 'of HIV' for predicate 'fight'). We locate the argument `ks:InstanceMention`s within the span (an arbitrary constituent) by matching a regular expression against the DP path from the span head (e.g., 'of' for 'of HIV') to the mention head ('HIV'). In case of coordinated conjuncts (i.e., 'of A, B, and C'), this method selects the mentions corresponding to each of them (A, B, C).
- `ks:CoreferenceMention`s derive from coreference resolution, where each coreferential text span (e.g., 'G. W. Bush and Bono', 'supporters', 'they') is mapped either to a `ks:InstanceMention`s (e.g., 'supporters') or to a conjunction of `ks:InstanceMention`s ('G. W. Bush' and 'Bono') using DP data and regular expressions. Properties `ks:coreferential` and `ks:coreferentialConjunct` are respectively used to link these

[8]http://persistence.uni-leipzig.org/nlp2rdf/

[9]http://www.timeml.org/

mentions to the `ks:CoreferenceMention`.

3.3 Knowledge Distillation

The first step for distilling the 'knowledge graph' consists in the evaluation of a set of *mapping rules* that match certain patterns in the Mention layer and create consequent facts in the Instance layer. Mapping rules are formulated as SPARQL Update **INSERT**... **WHERE**... statements that are repeatedly executed until a fixed-point is reached. Rules can create new individuals, can invoke external code by means of custom SPARQL functions and can access and match also data in auxiliary resources (e.g., for mapping purposes) as well as the instance data created so far. Current rules can be organized in six categories based on their function:

- *Instance creation* rules map `ks:InstanceMention`s to instances, using either a fresh identifier or a function of mention attributes (for `ks:AttributeMention`s and `ks:TimeMention`s) as the instance URI, and linking it to the mention via a `ks:denotes` or `ks:implies` triple.
- *Typing* rules enrich instances with `rdf:type` triples based on several mention attributes, including WordNet synset, NERC class, and PropBank, NomBank, VerbNet, and FrameNet frame types. Emitted types include SUMO [11] and YAGO2 [9] classes selected based on existing mappings from WordNet synsets.
- *Naming* rules assert `rdfs:label` or `foaf:name` triples based on the textual extent of `ks:InstanceMention`s; `foaf:name` is used for `ks:NameMention`s (proper names).
- *DBpedia linking* rules link instances to DBpedia resources (mention attribute `ks:linkedTo`). `owl:sameAs` is used for `ks:NameMention`s; `rdfs:seeAlso` is used for other mentions whose links are usually less accurate.
- *Frame-argument linking* rules link frame to argument instances based on `ks:ParticipationMention`s, mapping their PropBank, NomBank, VerbNet and FrameNet `ks:role` attributes to linking properties.
- *Coreference assertion* rules create `owl:sameAs` links between instances denoted by mentions coreferred via a `ks:CoreferenceMention`; if a mention (e.g., 'supporters' in Figure 1) corefers with a conjunction of mentions (e.g., 'G.W. Bush' and 'Bono'), the *group* instance denoted by the first and each *member* instance for the latter are linked by `ks:include` triples. Coreference of subject and complement mentions implied by *copular* verbs (e.g., between 'Bush' and 'president of US' in 'Bush became president of US') is also handled.

We report two concrete rule examples. The first is the instance creation rule that introduces distinct frame and argument instances in case of argument nominalization:

```
INSERT { ?m ks:denotes ?i ; ks:implies ?if ; ks:expresses ?g .
         GRAPH ?g { ?i a ks:Instance . ?if a ks:Instance , ks:Frame } }
WHERE { ?m a ks:FrameMention ; nif:anchorOf ?a ; ks:roleset ?s .
         ?s a ks:ArgumentNominalization . BIND (ks:mint(?m) AS ?g)
         BIND (ks:mint(concat(?a, "_pred"), ?m) AS ?if)
         BIND (ks:mint(?a, ?m) AS ?i) }
```

In the rule, custom function `ks:mint(args...)` maps its arguments to a human-readable URI whose local name is based on the first argument plus a disambiguating hash of all its arguments. Argument nominalization is detected based on the SRL frame type (attribute `ks:roleset`), marked in an auxiliary resource as being of type `ks:ArgumentNominalization`. The second example is the core frame-argument linking rule:

```
INSERT { ?m ks:expresses ?g . GRAPH ?g { ?if ?p ?ia } }
WHERE { ?m a ks:ParticipationMention ; ks:frame ?mf ; ks:argument ?ma .
         ?mf a ks:FrameMention ; ks:denotes|ks:implies ?if .
         ?ma a ks:InstanceMention ; ks:denotes ?ia .
         ?if a ks:Frame . ?m ks:role ?r . ?r ks:mappedTo ?p .
         BIND (ks:mint(?m) AS ?g) }
```

In the rule, the mapping between the SRL role `?r` and the Instance layer property `?p` is given by `ks:mappedTo` triples coming from an external, customizable mapping resource.

The knowledge graph resulting from the application of mapping rules is refined in four steps:

1. *Inference.* We materialize all the inferences computed applying OWL 2 RL[10] rules and a custom rule establishing that whenever an instance participates to a frame, also the instances it `ks:includes` participate to the frame. Inference rules are augmented so to assert `ks:expresses` links between inferred triples and all the mentions expressing antecedent triples.
2. *Smushing.*[11] We keep only a canonical URI for each instance, possibly from DBpedia, and discard the other `owl:sameAs` aliases (`:HIV`) generated by mapping rules.
3. *Redundancy elimination.* We discard every unnamed instance that `ks:includes` some member instance (e.g., the instance derivable from mention 'supporters' of Figure 1), as the custom inference rule already propagated participation information to its members.
4. *Compaction.* We optimize the RDF representation of `ks:expresses` information by ensuring that whenever a maximal set of triples $\{\langle s_i, p_i, o_i\rangle\}, i = 1..n$ is expressed by the same set of mentions $\{m_j\}, j = 1..m$, then all the triples are placed in a unique graph g associated to all the mentions, resulting in a maximal reduction of the total number of triples from $2 \cdot n \cdot m$ (worst-case before compaction) to $n + m$.

3.4 Implementation Details

We implemented PIKES as an open-source Java application. For the linguistic feature extraction phase we wrapped several state-of-the-art NLP tools: Stanford CoreNLP[12] for POS, NERC, TERN, and coreference resolution; mate-tools[13] for DP and SRL, with mappings between frame resources from the Predicate Matrix [10]; DBpedia Spotlight[14] for EL; and, UKB[15] for WSD. A dedicated module was developed to map NLP annotations to mentions. For the knowledge distillation phase we extended RDFpro [4], an RDF manipulation tool supporting SPARQL-like rule evaluation, smushing, filtering and other RDF processing tasks.

PIKES is accessible through an HTTP ReST API and a web interface (publicly available on PIKES website) that allows users to freely test our approach on sentences of their choice.

[10]http://www.w3.org/TR/owl2-profiles/#OWL_2_RL
[11]http://patterns.dataincubator.org/book/smushing.html
[12]http://nlp.stanford.edu/software/corenlp.shtml
[13]https://code.google.com/p/mate-tools/
[14]http://spotlight.dbpedia.org/
[15]http://ixa2.si.ehu.es/ukb/

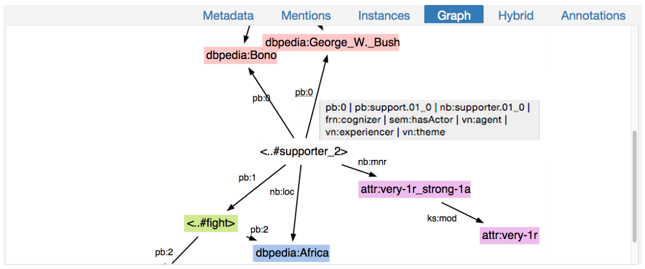

Figure 3: Graphical rendering (excerpt) of knowledge extracted from the sentence of Section 3.1

Several tabs become available once PIKES finished processing the input text: The *Metadata* tab reports the document metadata and a summary of the NLP tools applied. The *Mentions* tab lists the mentions identified in the text with their attributes. The *Instances* tab shows the extracted Instance layer triples, including `ks:expresses` links to mentions. The *Graph* tab, exemplified in Figure 3, shows a graphical rendering of the Instance layer where nodes are instances and arcs are triples between them; additional triples (e.g., types, labels, etc) are shown by tooltips (e.g., the gray box in Figure 3). The *Hybrid* tab highlights, sentence by sentence, mentions and their attributes, as well as the fragment of knowledge graph extracted from each sentence. The *Annotations* tab shows the raw output of NLP tools.

4. EVALUATION

To assess the performances of PIKES we perform two complementary evaluations. First (Section 4.1), we compute PIKES precision and recall in extracting entities, frames, attributes, and their relations from a text for which a gold standard knowledge graph was manually built; we also compare FRED and PIKES performances on the same text. Then (Section 4.2), we assess PIKES capability of extracting accurate knowledge from large text corpora. All data for both evaluations are available on PIKES website.

4.1 Assessing PIKES Precision and Recall

Gold standard The gold standard consists of a text T and a corresponding RDF knowledge graph G. Text T, shown in Figure 4a, consists of the same 8 sentences used in [7], except for sentence S7 being slightly shortened due the unprocessability of its full version with FRED online demo (tested Sept. 2015). Graph G, collaboratively built by two annotators, consists of the *relevant* RDF triples that should be included in the output of a frame-oriented KE system, which may include additional (irrelevant) triples provided that they do not conflict with the meaning of T. The nodes of G are the instances mentioned in T (entities, frames, attributes). Each instance is anchored to exactly one mention, with coreferring mentions giving rise to distinct instances. Instances are linked by `owl:sameAs` triples to matching entities in DBpedia, and typed with respect to classes encoding VerbNet (VN), FrameNet (FN), PropBank (PB) and NomBank (NB) frame types (most specific types represented). The edges of G are given by triples connecting different instances. They express `owl:sameAs` equivalence relations (to explicitly represent and evaluate coreference resolution), instance-attribute association relations, and frame-argument participation relations whose RDF properties encode VN/FN/PB/NB thematic roles.

Evaluation methodology To evaluate a KE system S (PIKES, FRED), we process the gold text T with S and collect the resulting RDF knowledge graph. We then replace the vocabulary terms (properties, types) and instance identifiers in this graph with the corresponding ones in gold graph G (where possible), obtaining a graph $\hat{G}_S$ that is now comparable with G. Corresponding instances are identified by leveraging their grounding to mentions in both graphs and, in case of multiple mappings, by selecting the one that maximizes the number of triples shared by $\hat{G}_S$ and G.
We evaluate the performances of S in extracting the following knowledge graph *components*: instances, edges, and triples, considered either globally and divided by category (links to DBpedia, VN/FN/PB/NB types, VN/FN/PB/NB participation relations, `owl:sameAs` equivalence relations). For each component, the associated set of *elements* (instances, edges, triples) in $\hat{G}_S$ is compared to the corresponding gold set in G, computing true positives (TP), false positives (FP), and false negatives (FN) and from them precision (P), recall (R), and F_1-measure (F_1). While TP and FN are computed as usual, for FP we discriminate between wrong and irrelevant elements in $\hat{G}_S$: the first are marked as FP, the latter are ignored, not affecting evaluation results. Irrelevant instances are classified manually first, with edges and triples involving them marked automatically as irrelevant; the remaining edges and triples are judged manually.

Separate evaluation We first evaluate PIKES alone on the full gold standard $\langle T, G\rangle$, applying the methodology above. Table 4b reports the results obtained, with the number N_G of gold elements for each evaluated component. Precision is generally higher than recall, which is common in KE systems where the lack of recall can be compensated by information redundancy in text. Instances and edges of the knowledge graph are extracted with high precision ($> 89\%$), while for extraction of triples the performances depend on their category. In particular, precision of frame type and participation triples is good for PB and low for FN, which reflects the fact that the SRL tool used is trained on PB and NB data with FN and VN triples obtained via mapping; using a FN-trained SRL tool will likely reverse the situation.

Comparative evaluation A fair comparison of PIKES and FRED[16] is possible only on a simpler gold graph G' against which both systems are comparable. G' is derived automatically from G by considering two characteristics of FRED: (i) FRED does not return PB/NB frame types and PB/NB/FN frame roles,[17] so we ignore them; and (ii) FRED does not support nominal predicates and argument nominalization, although it often represents the associated participation relations with arbitrary triples. To exemplify, the spans (a) 'Their success' in sentence S4, and (b) 'Iraqi officials' in sentence S5, are represented by FRED as

```
(a) :success_1 a :Success ; :successOf :thing_1 .
(b) :official_1 a :Official ; dul:associatedWith :Iraqi .
```

[16]We refer to FRED on-line version available in Sept. 2015.
[17]FRED marks frame arguments with generic predicate `:fe`.

S1 *The lone Syrian rebel group with an explicit stamp of approval from Al Qaeda has become one of the uprising most effective fighting forces, posing a stark challenge to the United States and other countries that want to support the rebels but not Islamic extremists.* **S2** *Money flows to the group, the Nusra Front, from like-minded donors abroad.* **S3** *Its fighters, a small minority of the rebels, have the boldness and skill to storm fortified positions and lead other battalions to capture military bases and oil fields.* **S4** *As their successes mount, they gather more weapons and attract more fighters.* **S5** *The group is a direct offshoot of Al Qaeda in Iraq, Iraqi officials and former Iraqi insurgents say, which has contributed veteran fighters and weapons.* **S6** *"This is just a simple way of returning the favor to our Syrian brothers that fought with us on the lands of Iraq," said a veteran of Al Qaeda in Iraq, who said he helped lead the Nusra Front's efforts in Syria.* **S7** *The United States, sensing that time may be running out for Syria president Bashar al-Assad, hopes to isolate the group to prevent it from inheriting Syria.* **S8** *As the United States pushes the Syrian opposition to organize a viable alternative government, it plans to blacklist the Nusra Front as a terrorist organization, making it illegal for Americans to have financial dealings with the group and prompting similar sanctions from Europe.*

(a)

Component	N_G	P	R	F$_1$
Instances	153	.943	.967	.955
Triples	596	.713	.508	.594
DBpedia links	18	.700	.778	.737
types (VN)	44	.706	.545	.615
types (FN)	53	.731	.358	.481
types (PB)	53	.844	.717	.776
types (NB)	37	.690	.784	.734
roles (VN)	94	.742	.489	.590
roles (FN)	108	.500	.259	.341
roles (PB)	119	.829	.571	.677
roles (NB)	55	.627	.582	.604
`owl:sameAs`	15	.714	.333	.455
Edges	171	.891	.766	.824

(b)

Component	$N_{G'}$	FRED vs G'			PIKES vs G'			$N_{G''}$	FRED vs G''			PIKES vs G''		
		P	R	F$_1$	P	R	F$_1$		P	R	F$_1$	P	R	F$_1$
Instances	137	.930	.869	.898	.937	.978	.957	136	.930	.875	.902	.937	.985	.961
Triples	166	.543	.416	.471	.713	.554	.624	115	.543	.600	.570	.713	.800	.754
DBpedia links	18	.615	.444	.516	.700	.778	.737	14	.615	.571	.593	.700	1.000	.824
types (VN)	31	.593	.516	.552	.667	.581	.621	27	.593	.593	.593	.667	.667	.667
types (FN)	26	.550	.423	.478	.762	.615	.681	17	.550	.647	.595	.762	.941	.842
roles (VN)	76	.547	.382	.450	.722	.513	.600	51	.547	.569	.558	.722	.765	.743
`owl:sameAs`	15	.357	.333	.345	.714	.333	.455	6	.357	.833	.500	.714	.833	.769
Edges	155	.869	.555	.677	.937	.768	.844	134	.869	.642	.738	.937	.888	.912

(c)

Figure 4: Precision/recall evaluation: (a) gold standard; (b) separate evaluation of PIKES on gold graph G; (c) comparative evaluation with FRED on simplified gold graph G' and union of correct tools answers G''.

where `:success_1`, `:thing_1`, `:official_1`, and `:Iraqi` are the instances denoted by mentions 'success', 'Their', 'officials', and 'Iraqi', whereas the gold standard and PIKES employ the nominal frames

```
(a) :success_1 a fn:Success_or_failure ; fn:agent :they .
(b) :official_frame a nb:official.01 ;
                    nb:official.01_0 :official_1 ;
                    nb:official.01_2 :Iraqi .
```

where `:official_frame` is a frame instance also denoted by ':officials'. Thus, we automatically transform the latter representation (in G' and in PIKES output) into FRED one.

In line with the approach of [7], we also compare PIKES and FRED against an additional gold graph G'' obtained by merging the outputs of both tools, cleaned up of incorrect triples. By definition, $G'' \subseteq G'$. The goal of this additional evaluation, as noted in [7], is to comparatively evaluate each tool within KE tool space (i.e., considering only correct triples that can be extracted by at least one tool).

The results of the comparison against G' and G'' are reported in Table 4c, together with the numbers $N_{G'}$ and $N_{G''}$ of gold elements for each component. PIKES exhibits better precision and recall than FRED for all the considered components and gold graphs: the difference in terms of F$_1$ between PIKES and FRED ranges, for the various components, from .059 to .221 on G', and from .059 to .269 on G''. Overall, the results show that splitting the KE process in two-phases, decoupled by the mention graph, allows reaching performances that are very competitive with the state of the art while providing all the benefits listed in Section 1.

4.2 Evaluating PIKES on Large Text Corpora

To asses its capability of processing large corpora, we applied PIKES to a general domain corpus, namely a dump of the Simple English Wikipedia (SEW).[18] SEW is a variant of Wikipedia, where each page is written using basic English words. The corpus consists of 109,242 text document containing a total of 1,584,406 sentences and 23,877,597 tokens.

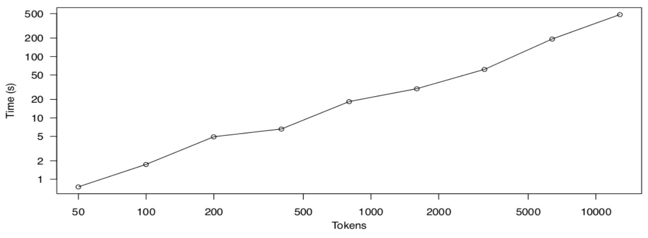

Figure 5: PIKES processing time linearly grows w.r.t. the number of tokens in a text.

PIKES processed the whole SEW corpus in ~507 core hours, with an average of 1.2s per sentence and 16.7s per document.[19] We used 16 parallel instances of PIKES on the same machine,[20] ending the whole processing in less than 32 hours. Processing time grows linearly with the number of tokens in the input text, as shown in Figure 5. The resulting dataset is available on PIKES website. A total of 357,853,792 triples were produced (~2M triples Text layer,

[18]http://simple.wikipedia.org/ – Dump date: April 6, 2015.

[19]A proper baseline (i.e., same hardware and test conditions) to put the reported figures into perspective is not available, as the closest tool to PIKES, FRED, is not available for download (only a demo service is available).

[20]Dell PowerEdge M520 Server, 2CPUs Intel Xeon E5-2430 2.50GHz, 192GB RAM, 480GB SSD HD. Each PIKES instance was allocated 1 core and 7GB of RAM, used mainly for NLP models. We are working on a multithreaded setup where models are shared by threads, reducing RAM usage.

Table 2: Evaluation of PIKES accuracy on SEW.

	Total Triples	Eval. Triples	Accuracy			
			Ev 1	Ev 2	Ev 3	Avg.
Annotation	1,617	35	.900	.886	.857	.881
Type	1,699	35	.943	.771	.857	.857
Participation	10,805	130	.904	.785	.850	.846
Total	14,121	200	.910	.800	.853	.854

~283M Mention layer, ~72M Instance layer). More than 4M frame instances were created (most frequent: use.01, play.01, know.01), involving over 72K DBpedia persons (most frequent: Pope, Jesus, Napoleon), 19K DBpedia organizations, and 49K DBpedia places. Additionally, 470K persons, 173K organizations, and 18K locations not linked to DBpedia URIs were created. In total, we extracted over 26M triples about DBpedia entities (1.7M annotations, 2.6M types, 21M participations in different frame vocabularies for 7M frame-argument pairs), which might be used to extend DBpedia.

To evaluate the quality of the produced knowledge graph, given the absence of a gold standard and the impossibility to properly appraise recall due to the size of the corpus, we opted to assess the accuracy of a sampled subset of the extracted triples involving DBpedia entities. A similar strategy was applied to evaluate a large knowledge graph such as YAGO2 [9]. We built the evaluation dataset as follows. We randomly sampled 200 triples from the graph involving DBpedia entities, focusing in particular on annotation and name assertions (35 triples), type assertions (35), and PB/NB frame participations (130). For each triple, we randomly selected one of the mentions from where the triple was extracted, and a context of three sentences in the text centered around it. The evaluation dataset was provided to three evaluators, with skills in knowledge engineering, which were asked to judge if each triple produced by PIKES is correct for the given mention (i.e., if the knowledge encoded in the triple is compatible with the knowledge conveyed by the mention). Evaluators were allowed to use three values: 1 – correct, 0.5 – partly correct,[21] and 0 – not correct. Table 2 summarizes the results obtained. Accuracy was computed for each assertion type, and for each evaluator (Ev 1-3). The average accuracy over the whole evaluation dataset is slightly above 0.85. The computed Fleiss' kappa coefficient ($\kappa = 0.372$) shows a fair agreement between the evaluators.

5. CONCLUSIONS

We presented PIKES, an approach for extracting frame-oriented knowledge from text that divides processing into two phases (linguistic feature extraction and knowledge distillation) decoupled through an intermediate RDF mention graph. PIKES decoupling provides several benefits: (i) easy replacement of NLP tools; (ii) easy (rule-based) combination of the outputs of different NLP tasks during knowledge distillation to support non-trivial linguistic phenomena (e.g., argument nominalization); (iii) single RDF model enabling the representation of all the information involved at various levels (resource, mention, instance) as well as provenance tracking for each piece of extracted knowledge. At the same time, PIKES approach results competitive w.r.t. FRED – a current state-of-the-art tool – and supports the processing of large text corpora. PIKES code is released open source with a publicly accessible online demo service.

Acknowledgements The research leading to this paper was supported by the European Union's 7th Framework Programme via the NewsReader Project (ICT-316404).

[21]Used when the tool correctly identified subject and object of a participation triple, but failed identifying the predicate.

6. REFERENCES

[1] I. Augenstein, S. Padó, and S. Rudolph. LODifier: Generating Linked Data from unstructured text. In *Proc. of ESWC'12*, pages 210–224. Springer, 2012.

[2] P. Cimiano. *Ontology learning and population from text*. Springer, 2006.

[3] F. Corcoglioniti, M. Rospocher, R. Cattoni, B. Magnini, and L. Serafini. The KnowledgeStore: a storage framework for interlinking unstructured and structured knowledge. *Int. J. Semantic Web Inf. Syst.*, 11(2):1–35, 2015.

[4] F. Corcoglioniti, M. Rospocher, M. Mostarda, and M. Amadori. Processing billions of RDF triples on a single machine using streaming and sorting. In *Proc. of SAC'15*, pages 368–375. ACM, 2015.

[5] A. Freitas, J. P. da Silva, D. S. Carvalho, S. O'Riain, and E. Curry. Representing texts as contextualized entity-centric Linked Data graphs. In *Proc. of 12th WebS Workshop at DEXA'13*, pages 133–137, 2013.

[6] A. Gangemi, F. Draicchio, V. Presutti, A. G. Nuzzolese, and D. R. Recupero. A machine reader for the semantic web. In *Demos of ISWC*, 2013.

[7] A. Gangerni. A comparison of knowledge extraction tools for the Semantic Web. In *Estended Semantic Web Conference*. Springer, 2013.

[8] R. Grishman and B. Sundheim. Message Understanding Conference-6: A brief history. In *Proc. of COLING'96*, pages 466–471, 1996.

[9] J. Hoffart, F. M. Suchanek, K. Berberich, and G. Weikum. YAGO2: A spatially and temporally enhanced knowledge base from Wikipedia. *Artif. Intell.*, 194:28–61, 2013.

[10] M. Lopez De Lacalle, E. Laparra, and G. Rigau. Predicate Matrix: extending SemLink through WordNet mappings. In *Proc. of LREC'14*, pages 903–909. ELRA, 2014.

[11] I. Niles and A. Pease. Towards a standard upper ontology. In *Proc. of FOIS'01*, pages 2–9. ACM, 2001.

[12] T. Parsons. *Events in the Semantics of English: A study in subatomic semantics*. MIT Press, 1990.

[13] G. Petasis, V. Karkaletsis, G. Paliouras, A. Krithara, and E. Zavitsanos. Ontology population and enrichment: State of the art. In *Knowledge-driven Multimedia Information Extraction and Ontology Evolution*, pages 134–166. Springer-Verlag, 2011.

[14] M. Rospocher, A.-L. Minard, P. Mirza, P. Vossen, T. Caselli, A. Cybulska, R. Morante, and I. Aldabe. Event Narrative Module, version 2 - NewsReader FP7 EU Project Deliverable D5.1.2.