LODifier: Generating Linked Data from Unstructured Text

Isabelle Augenstein, Sebastian Padó, and Sebastian Rudolph

Department of Computational Linguistics, Universität Heidelberg, DE
Institute AIFB, Karlsruhe Institute of Technology, DE
{augenste,pado}@cl.uni-heidelberg.de, rudolph@kit.edu

Abstract. The automated extraction of information from text and its transformation into a formal description is an important goal in both Semantic Web research and computational linguistics. The extracted information can be used for a variety of tasks such as ontology generation, question answering and information retrieval. LODifier is an approach that combines deep semantic analysis with named entity recognition, word sense disambiguation and controlled Semantic Web vocabularies in order to extract named entities and relations between them from text and to convert them into an RDF representation which is linked to DBpedia and WordNet. We present the architecture of our tool and discuss design decisions made. An evaluation of the tool on a story link detection task gives clear evidence of its practical potential.

1 Introduction

The term Linked Data (*LD*) stands for a new paradigm of representing information on the Web in a way that enables the global integration of data and information in order to achieve unprecedented search and querying capabilities. This represents an important step towards the realization of the Semantic Web vision. At the core of the LD methodology is a set of principles and best practices describing how to publish structured information on the Web. In recent years these recommendations have been adopted by an increasing number of data providers ranging from public institutions to commercial entities, thereby creating a distributed yet interlinked global information repository.

The formalism underlying this "Web of Linked Data" is the Resource Description Framework (*RDF*) which encodes structured information as a directed labelled graph. Hence, in order to publish information as Linked Data, an appropriate graph-based representation of it has to be defined and created. While this task is of minor difficulty and can be easily automatized if the original information is already structured (as, e.g., in databases), the creation of an adequate RDF representation for unstructured sources, particularly textual input, constitutes a challenging task and has not yet been solved to a satisfactory degree.

Most current approaches [7,19,16,6] that deal with the creation of RDF from plain text fall into the categories of relation extraction or ontology learning. Typically, these approaches process textual input very selectively, that is, they scan the text for linguistic

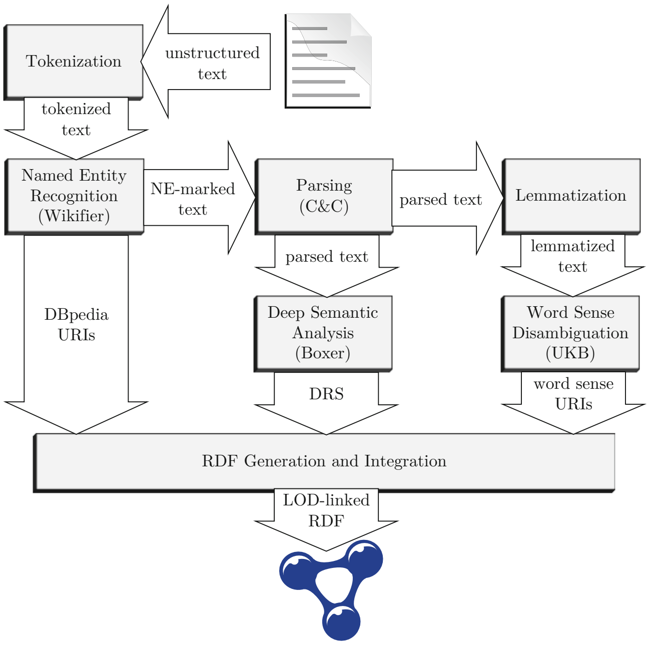

Fig. 1. The architecture of LODifier

patterns that realize a small number of pre-specified types of information (e.g., *is-CEO-of* relations). This strategy is oriented toward a high precision of the extracted structured information and certainly adequate if the result of the extraction process is meant to be used for accumulation of large sets of factual knowledge of a predefined form.

In contrast to these approaches, we propose a strategy which aims at translating the textual input *in its entirety* into a structural RDF representation. We aim at open-domain scenarios in which no a-priori schema for the information to be extracted is available. Applying our method to a document similarity task we demonstrate that it is indeed both practical and beneficial to retain the richness of the full text as long as possible.

Our system, *LODifier*, employs robust techniques from natural language processing (NLP) including named entity recognition (*NER*), word sense disambiguation (*WSD*) and deep semantic analysis. The RDF output is embedded in the Linked Open Data (*LOD*) cloud by using vocabulary from DBpedia and WordNet 3.0.

Plan of the Paper. Section 2 begins by sketching the architecture of the system. Section 3 presents an evaluation of LODifier on a document similarity task. After discussing related work in Section 4, we conclude in Section 5.

2 The System

This section describes the resources and algorithms used to build LODifier. Figure 1 shows the architecture of the system. After tokenization, mentions of entities in the input text are recognized using the NER system *Wikifier* [18] and mapped onto DBpedia URIs. Relations between these entities are detected using the statistical parser *C&C* and the semantics construction toolkit *Boxer* [8], which generates discourse representation structures (*DRSs*) [14]. Thereafter, the text is lemmatized and words are disambiguated with the WSD tool *UKB* [1] to get WordNet mappings. The RDF graph is then created by further processing the Boxer DRS output, transforming it into triples. Finally, it is enriched with the DBpedia URIs (to link its entities to the LOD cloud) and the WordNet sense URIs (to do the same for the relations). The following subsections provide details on the individual processing steps.

2.1 Recognizing Named Entities

The first step is to identify mentioned individuals. They are recognized using the NER system Wikifier [18] that enriches English plain text with Wikipedia links. If Wikifier finds a named entity, it is substituted by the name of the corresponding English Wikipedia page. Applied to the test sentence

> *The New York Times reported that John McCarthy died. He invented the programming language LISP.*

Wikifier recognizes the named entities and generates the output

```
[[The New York Times]] reported that [[John McCarthy (computer scientist)|
John McCarthy]] died. He invented the [[Programming language|programming
language]] [[Lisp (programming language)|Lisp]].
```

To disambiguate the Wikipedia links, Wikifier employs a machine learning approach that uses the links between Wikipedia articles as training data. Since the links between Wikipedia articles are manually created by Wikipedia editors, the training data consists of highly reliable disambiguation choices.

Note that the Boxer system itself also performs a NER. We employ Wikifier to increase NER coverage and, most notably, to obtain links to the LOD cloud via DBpedia URIs.

2.2 Linking DBpedia URIs to Recognized Named Entities

The next step is to generate DBpedia URIs out of the Wikifier output and link those DBpedia URIs to previously introduced Boxer classes.

DBpedia [4] is a large, freely available domain-independent multilingual ontology extracted from Wikipedia, comprising Wikipedia page names, infobox templates, categorization information, images, geo-coordinates and links to external webpages. DBpedia contains links to various data sets including *FOAF*, *Geonames* and *WordNet*.

We exploit the fact that every Wikipedia page has a corresponding DBpedia page, which allows for a straightforward conversion of Wikipedia URLs to DBpedia URIs.

2.3 Recognizing Relations

Next, relations between the entities are determined. This is done by the parser C&C and the Boxer system developed by Curran, Clark and Bos [8].

The C&C parser first tags input words with parts of speech from the Penn Treebank tagset. It then constructs parse trees in the combinatorial categorial grammar (*CCG*) paradigm. In addition, C&C contains a named entity recognizer that distinguishes between ten different named entity types: organization (*org*), person (*per*), title (*ttl*), quotation (*quo*), location (*loc*), first name (*fst*), surname (*sur*), URL (*url*), e-mail (*ema*) and unknown name (*nam*). The parser is rather robust for a "deep" natural language processing tool, with precision and recall scores well above 80%. The C&C output for our example is displayed in Fig. 2. It forms a derivation tree in which each non-terminal is labelled with the CCG rule that was used to construct it (e.g., `fa` for 'forward application') as well as its CCG category. The terminals, labelled `t`, provide information about the words' CCG categories, forms and lemmas and parts of speech (in this order). The two last elements of each terminal specify information on shallow chunks and Named Entities, using IOB (inside-outside-begin) notation, a common format for representing non-hierarchical chunks. An IOB label starting with `I-`, like `I-NP`, indicates that a given word is *inside* an NP chunk. The label `O` means that the word is not part of any chunk. For example, the only Named Entity recognized in the first sentence in `John_McCarthy` (a `PER`).[1]

Boxer builds on the output of the statistical parser *C&C* and produces *discourse representation structures* (DRSs, cf. [14]). DRSs model the meaning of texts in terms of the relevant entities (*discourse referents*) and the relations between them (*conditions*). Figure 3 shows the DRSs for our example. Discourse referents are shown above the dotted lines and conditions below.

Discourse referents are introduced by new noun phrases or events and are, from a logical standpoint, essentially variables. For previously introduced discourse referents, Boxer attempts to resolve anaphora by either binding them to previously introduced discourse referents or accommodating them. A condition, which is described by a unary or binary predicate, is created for every relation found between discourse referents. Unary relations (also referred to as *classes*) are introduced by nouns, verbs, adverbs and adjectives, these are e.g., *person*, *event* or *topic*. Binary relations are introduced by prepositions and verb roles, e.g., *agent*, *patient* or *theme*. As the example shows, conditions can embed DRSs themselves. Such conditions are called complex conditions and are used to specify logical dependencies between partial propositions: *disjunction*, *implication*, *negation*, *necessity*, *possibility*.

Note that the DRS conditions only use unary and binary relations. Therefore, DRSs are structurally very similar to RDF, and can hence serve as a convenient intermediate data structure for converting text into RDF.

2.4 Assigning RDF WordNet URIs to Boxer Relations

Our first candidate for a target vocabulary for linking Boxer relations onto Linked Open Data entities was DBPedia. DBpedia contains about 44.000 different property types

[1] IOB supports labels of type `B-` to mark the first word in a chunk, but this is not used by Boxer.

```
ccg(1,
 rp(s:dcl,
  ba(s:dcl,
   lx(np, n,
    t(n, 'The_New_York_Times', 'The_New_York_Times', 'NNS', 'I-NP', 'O')),
   fa(s:dcl\np,
    t((s:dcl\np)/s:em, 'reported', 'report', 'VBD', 'I-VP', 'O'),
    fa(s:em,
     t(s:em/s:dcl, 'that', 'that', 'IN', 'I-SBAR', 'O'),
     ba(s:dcl,
      lx(np, n,
       t(n, 'John_McCarthy', 'John_McCarthy', 'NNP', 'I-NP', 'I-PER')),
      t(s:dcl\np, 'died', 'die', 'VBD', 'I-VP', 'O'))))),
  t(period, '.', '.', '.', 'O', 'O'))).

ccg(2,
 rp(s:dcl,
  ba(s:dcl,
   t(np, 'He', 'he', 'PRP', 'I-NP', 'O'),
   fa(s:dcl\np,
    t((s:dcl\np)/np, 'invented', 'invent', 'VBD', 'I-VP', 'O'),
    fa(np:nb,
     t(np:nb/n, 'the', 'the', 'DT', 'I-NP', 'O'),
     fa(n,
      t(n/n, 'programming_language', 'programming_language', 'NN', 'I-NP', 'O'),
      t(n, 'LISP', 'LISP', 'NNP', 'I-NP', 'O'))))),
  t(period, '.', '.', '.', 'O', 'O'))).
```

Fig. 2. C&C output for the example sentences

created by extracting properties from infoboxes and templates within Wikipedia articles. The *Raw Infobox Property Definition Set* consists of a URI definition for each property as well as a label. However, this property set turned out to be much too restricted to cover all the relations identified by Boxer.

Therefore, we decided to map Boxer relations onto RDF WordNet class types instead.

WordNet [10] is a large-scale lexical database for English. Its current version contains more than 155.000 words (nouns, verbs, adjectives and adverbs), grouped into sets of synonyms, which are called *synsets*. Ambiguous word belong to several synsets (one per *word sense*). The synsets are linked to other synsets by *conceptual relations*. Synsets contain *glosses* (short definitions) and short example sentences. *RDF WordNet* [3] is a Linked Data version of WordNet. For each word it provides one URI for each word sense. To map instances of words onto URIs the words have to be disambiguated.

For word sense disambiguation (WSD), we apply *UKB* [1], an unsupervised graph-based WSD tool, to all our input words, but focus on the results for words which have given rise to relations in the Boxer output. We use the disambiguated RDF WordNet URIs as the Linked Data hooks for these relations.

```
  ______________________________   _______________________
 | x0 x1 x2 x3                  | | x4 x5 x6              |
 |..............................| |.......................|
(| male(x0)                     |+| event(x4)             |)
 | named(x0,john_mccarthy,per)  | | invent(x4)            |
 | programming_language(x1)     | | agent(x4,x0)          |
 | nn(x1,x2)                    | | patient(x4,x2)        |
 | named(x2,lisp,nam)           | | event(x5)             |
 | named(x3,new_york_times,org) | | report(x5)            |
 |______________________________| | agent(x5,x3)          |
                                  | theme(x5,x6)          |
                                  | proposition(x6)       |
                                  |       ______________  |
                                  |      | x7           | |
                                  |   x6:|..............| |
                                  |      | event(x7)    | |
                                  |      | die(x7)      | |
                                  |      | agent(x7,x0) | |
                                  |      |______________| |
                                  |_______________________|
```

Fig. 3. Discourse representation structure generated for the example sentences

2.5 Generating an RDF Graph

Finally, we construct an RDF graph. Our first step is to define URIs for the predicate and relation types provided by Boxer. In this manner, we distinguish between predicate and relation types which come from a *closed* class (`event`, `agent`, etc.), and the *open* classes of predicate and relation types that represent words (`programming_language`, `die`, etc.). The second step is a translation of discourse referents and DRS conditions (unary and binary relations) into RDF triples according to the following strategy:

– For each discourse referent, a blank node (bnode) is introduced. If it has been recognized as a NE by Boxer, we assign an URI from the `ne:` namespace to it via the property `drsclass:named`. If an according DBpedia URI could be identified via Wikifier, we link the blank node to the according DBpedia URI via `owl:sameAs`.
– The assignment of a Boxer class (that is, a unary predicate) to a discourse referent is expressed by an RDF typing statement which associates a class URI to the discourse referent's bnode. For closed-class relations (like `event`), the class URI comes from the predefined vocabulary (using the namespace `drsclass:`), for relations from the open class we use the appropriate word sense URI extracted from WordNet via UKB (in the namespace `wn30:`) or create a URL (in the namespace `class:`).
– A closed-class binary relation between two discourse referents (e.g., `agent`) is expressed by an "ordinary" RDF triple with the referents' bnodes as subject and object, and using the corresponding URI from the closed-class Boxer vocabulary namespace `drsrel:`. For open-class relations, the namespace `rel:` is used instead.
– Finally, we may encounter embedded DRSs, possibly related by complex conditions expressing logical (*disjunction*,*implication*, *negation*) or modal (*necessity*,

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix reify: <http://www.aifb.kit.edu/web/LODifier/reify#> .
@prefix ne: <http://www.aifb.kit.edu/web/LODifier/ne#> .
@prefix drsclass: <http://www.aifb.kit.edu/web/LODifier/drsclass#> .
@prefix class: <http://www.aifb.kit.edu/web/LODifier/class#> .
@prefix drsrel: <http://www.aifb.kit.edu/web/LODifier/drsrel#> .
@prefix dbpedia: <http://dbpedia.org/resource/> .
@prefix wn30: <http://purl.org/vocabularies/princeton/wn30/> .
_:var0x0 drsclass:named ne:john_mccarthy ;
         rdf:type drsclass:male , foaf:Person ;
         owl:sameAs dbpedia:John_McCarthy_(computer_scientist) .
_:var0x1 rdf:type class:programming_language ;
         owl:sameAs dbpedia:Programming_language .
_:var0x2 drsrel:nn _:var0x1 .
_:var0x2 drsclass:named ne:lisp ;
         owl:sameAs     dbpedia:Lisp_(programming_language) .
_:var0x3 drsclass:named ne:the_new_york_times ;
         owl:sameAs     dbpedia:The_New_York_Times .
_:var0x4 rdf:type drsclass:event ,  wn30:wordsense-invent-verb-2 .
         drsrel:agent _:var0x0 ;  drsrel:patient _:var0x2 .
_:var0x5 rdf:type drsclass:event ,  wn30:wordsense-report-verb-3 ;
         drsrel:agent _:var0x3 ;  drsrel:theme _:var0x6 .
_:var0x6 rdf:type drsclass:proposition ,  reify:proposition ,  reify:conjunction ;
         reify:conjunct [  rdf:subject   _:var0x7 ;
                           rdf:predicate rdf:type ;
                           rdf:object    drsclass:event .  ]
         reify:conjunct [  rdf:subject   _:var0x7 ;
                           rdf:predicate rdf:type ;
                           rdf:object    wn30:wordsense-die-verb-1 . ]
         reify:conjunct [  rdf:subject   _:var0x7 ;
                           rdf:predicate drsrel:agent ;
                           rdf:object    _:var0x0 .  ]
```

Fig. 4. LODifier output for the test sentences

possibility) operators. We recursively reify the RDF representations of these subordinate DRSs by using the predefined RDF reification vocabulary (consisting of the property URIs `rdf:subject`, `rdf:predicate`, and `rdf:object`, for an introduction to reification in RDF see e.g. [13], Section 2.5.2).[2] The logical or modal dependencies between these sub-DRSs are then expressed by means of additional fixed vocabulary in the namespace `reify:`. This results in flat RDF output for nested DRSs.

The result of applying this strategy to our example text is shown in Figure 4.

[2] To obtain output that adheres to the current W3C RDF specification and is entirely supported by standard-compliant tools, we refrain from using named graphs or quads to encode nested DRS.

3 Automatic Evaluation: Story Link Detection

3.1 Task and Setup

In our evaluation we use LODifier to assess document similarity. More specifically, we consider a *story link detection* task, part of the topic detection and tracking (TDT) family of tasks. It is defined as *"[...] the problem of deciding whether two randomly selected stories discuss the same news topic"* [2].

We use a subset of the TDT-2 benchmark dataset. TDT-2 consists of a total of almost 84.000 documents from the year 1998, drawn from newspapers, radio news, and television news in English, Arabic and Mandarin. Each document is manually assigned to one of a set of 100 topics. Each topic also comes with one "seed" story that is presumed to be representative for the topic.

We follow the general lead of the original TDT-2 benchmark evaluation schema. We focus on English as a language and newspapers as the source since LODifier can currently only deal with English text and presumably degrades on potentially noisy automatic radio and TV transcripts. We therefore restrict our attention to the 19 topics whose seed story was an English newspaper article. For each topic, we pair the seed story with all other articles of this topic that met our constraints. However, since the distribution of topics over documents is very skewed and we want to avoid undue influence of very large topics, we restrict the number of document pairs for each topic to 50. This results in a total of 183 *positive* document pairs, an average of 11.2 document pairs per topic. We then sample the same number of *negative* document pairs from the dataset by pairing each document with the seed document from a different topic. The total number of document pairs is 366 with an equal positive/negative distribution.

We approach the task by defining various document similarity measures *sim*. We assume that this similarity is a direct indicator of relatedness, which leads to a very simple classification procedure for document pairs *dp*, given a threshold θ:

$$\text{class}(dp, \theta) = \begin{cases} \text{positive} & \text{if } sim(dp) \geq \theta \\ \text{negative} & \text{if } sim(dp) < \theta \end{cases}$$

Thus, document pairs are predicted to describe the same topic exactly if they have a similarity of θ or more.

The parameter θ is usually determined with supervised learning. We randomly split our dataset k times (we use k=1000) into equally-sized training and testing sets. For each split, we compute an optimal decision boundary $\hat{\theta}$ as the threshold which predicts the training set as well as possible. More precisely, we choose $\hat{\theta}$ so that its distance to wrongly classified training document pairs is minimized. Formally, $\hat{\theta}$ is defined as follows: Let pos_{train} and neg_{train} be the positive and negative partitions of the training set respectively. Then:

$$\hat{\theta} = \arg\min_{\theta} \left[\sum_{dp \in pos_{train}} \min(0, sim(dp) - \theta)^2 + \sum_{dp \in neg_{train}} \min(0, \theta - sim(dp))^2 \right]$$

We can then compute the accuracy of $\hat{\theta}$ on the current split's test set, consisting of the positive and negative partitions pos_{test} and neg_{test}, as follows:

$$acc_{\hat{\theta}} = \frac{||\{dp \in pos_{test} \mid sim(dp) \geq \hat{\theta}\}|| + ||\{dp \in neg_{test} \mid sim(dp) < \hat{\theta}\}||}{||pos_{test}|| + ||neg_{test}||}$$

After repeating this procedure for all k splits, the final accuracy is computed by averaging the k accuracies.

3.2 Similarity Computation without Structure

As baselines, we consider a number of measures that do not take structural information into account. The first one is a *random baseline*, which performs at 50% in our setup, since the two classes (positive and negative) are balanced. Second, we consider a simple *bag-of-words baseline* which measures the word overlap of two documents in a document pair without any preprocessing. Third, we experiment with a family of *bag-of-URI baselines* that measure the URI overlap of two RDF documents. We experiment with three variants which correspond to different assumptions about the relative importance of various URI classes:

Variant 1 considers all NEs identified by Wikifier and all words successfully disambiguated by UKB (namespaces `dbpedia:` or `wn30:`).

Variant 2 adds all NEs recognized by Boxer (namespace `ne:`).

Variant 3 further adds the URIs all words that were not recognized by either Wikifier, UKB or Boxer (namespace `class:`).

Extended setting. For each of the variants we also construct an extended setting, where the generated RDF graph was enriched by information from DBpedia and WordNet, namely DBpedia categories, WordNet synsets and WordNet senses related to the respective URIs in the generated graph. This setting aims at drawing more information from Linked Open Data into the similarity computation.

3.3 Structurally Informed Similarity Computation

Recall our motivation for using LODifier, namely the intuition that structural RDF information can provide an improvement over the simple comparison of words or URIs. This requires the formulation of structure-aware similarity measures between documents (i.e., RDF graphs). First attempts showed that full-fledged graph similarity measures based on homomorphic or isomorphic subgraphs of arbitrary size or common cliques [21], which are generally NP-complete, are infeasible due to the size of the RDF graphs we consider (up to 19.000 nodes).

We decided to perform a more relaxed structural comparison based on the shortest paths between relevant nodes. This mirrors our intuition that a short path in a RDF graph between two URIs denotes a salient semantic relation between those entities. For such URI pairs, we would expect that they are related by a short path in other documents (graphs) on the same topic as well.

Formally, let G_1 and G_2 be two RDF graphs. We write $\ell(a, b)$ to denote the length of the shortest path between two nodes a and b in a graph G, and let $C_k(G)$ denote the set of all paths of length $\leq k$ in G, that is, the set of salient relations in G.[3] Furthermore, we write $Rel(G)$ for the set of relevant nodes in an RDF graph G. As motivated in Section 3.2, not all URIs are equally relevant and we experiment with different choices. We can now define a family of similarity measures called *path relevance overlap similarity (*proSim*)*:

$$\text{proSim}_{k,Rel,f}(G_1, G_2) = \frac{\sum\limits_{\substack{a,b \in Rel(G_1)\\ \langle a,b\rangle \in C_k(G_1)\cap C_k(G_2)}} f(\ell(a,b))}{\sum\limits_{\substack{a,b \in Rel(G_1)\\ \langle a,b\rangle \in C_k(G_1)}} f(\ell(a,b))}$$

In words, the denominator of proSim determines the set of relevant semantic relations *Rel* in G_1 – modelled as the set of pairs of relevant URIs that are linked by a path of length $\leq k$ – and quantifies them as the sum over a function applied to their path lengths. The numerator does the same for the intersection of the relevant nodes from G_1 and G_2.

We experiment with three instantiations for the function f. The first one, $\text{proSim}_{\text{cnt}}$, uses $f(\ell) = 1$, that is, just counts the number of paths irrespective of their length. The second one, $\text{proSim}_{\text{len}}$, uses $f(\ell) = 1/\ell$, giving less weight to longer paths. The third one, $\text{proSim}_{\text{sqlen}}$, uses $f(\ell) = 1/\sqrt{\ell}$, discounting long paths less aggressively than $\text{proSim}_{\text{len}}$.

All measures of the proSim family have the range [0;1], where 0 indicates no overlap and 1 perfect overlap. It is deliberately asymmetric: the overlap is determined relative to the paths of G_1. This reflects our intuitions about the task at hand. For a document to be similar to a seed story, it needs to subsume the seed story but can provide additional, new information on the topic. Thus, the similarity should be maximal whenever $G_1 \subseteq G_2$, which holds for proSim.

3.4 Results

As described in Section 3.2, we experiment with several variants for defining the set of relevant URIs (Variants 1 to 3, both normal and extended). These conditions apply to all bag-of-URI and proSim models.

The results of our evaluation are shown in Table 1. The upper half shows results for similarity measures without structural knowledge. At 63%, the bag-of-words baseline clearly outperforms the random baseline (50%). It is in turn outperformed by almost all bag-of-URI baselines, which yield accuracies of up to 76.4%. Regarding parameter choice, we see the best results for Variant 3, the most inclusive definition of relevant URIs (cf. Section 3.2). The URI baseline also gains substantially from the extended setting, which takes further Linked Open Data relations into account.

Moving over to the structural measures, proSim, we see that all parametrizations of proSim perform consistently above the baselines. Regarding parameters, we again see a consistent improvement for Variant 3 over Variants 1 and 2. In contrast, the performance is relatively robust with respect to the path length cutoff k or the inclusion of further Linked Open Data (extended setting).

[3] Shortest paths can be computed efficiently using Dijkstra's algorithm [9]. We exclude paths across "typing" relations such as `event` which would establish short paths between every pair of event nodes (cf. Figure 3) and drown out meaningful paths.

Table 1. Accuracy on Story Link Detection Task

Model	normal	extended
Similarity measures without structural knowledge		
Random Baseline	50.0	–
Bag of Words	63.0	–
Bag of URIs (Variant 1)	61.6	75.1
Bag of URIs (Variant 2)	70.6	76.0
Bag of URIs (Variant 3)	**73.4**	**76.4**
Similarity measures with structural knowledge		
proSim$_{\text{cnt}}$ (k=6, Variant 1)	79.0	78.9
proSim$_{\text{cnt}}$ (k=6, Variant 2)	80.3	80.3
proSim$_{\text{cnt}}$ (k=6, Variant 3)	81.6	81.6
proSim$_{\text{cnt}}$ (k=8, Variant 1)	77.7	77.6
proSim$_{\text{cnt}}$ (k=8, Variant 2)	79.2	79.0
proSim$_{\text{cnt}}$ (k=8, Variant 3)	**82.1**	**81.9**
proSim$_{\text{len}}$ (k=6, Variant 3)	81.5	81.4
proSim$_{\text{len}}$ (k=8, Variant 3)	80.3	80.1
proSim$_{\text{len}}$ (k=10, Variant 3)	80.0	79.8
proSim$_{\text{sqlen}}$ (k=6, Variant 3)	80.4	80.4
proSim$_{\text{sqlen}}$ (k=8, Variant 3)	81.1	80.9
proSim$_{\text{sqlen}}$ (k=10, Variant 3)	80.5	80.4

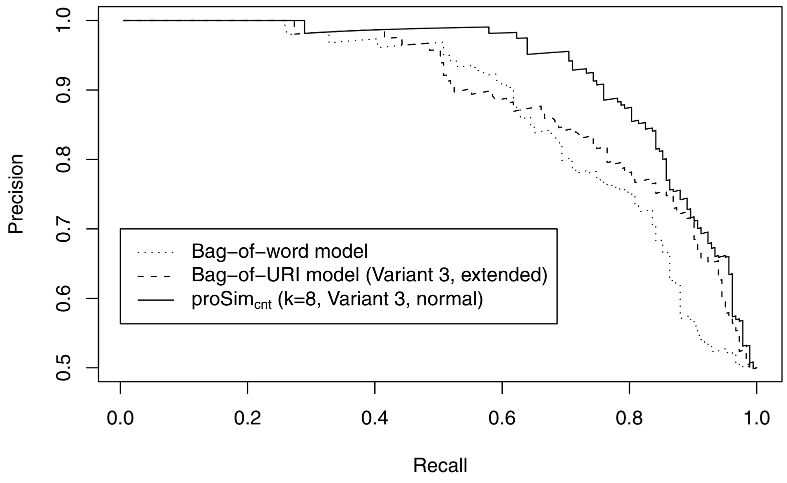

Fig. 5. Precision-Recall-plot for best Story Link Detection models

Moving from proSim$_{\text{cnt}}$ to more involved similarity functions, we decided to concentrate on Variant 3 which showed the best results. However, neither proSim$_{\text{len}}$ nor proSim$_{\text{sqlen}}$ yielded considerably different results: On this dataset, performance plateaued between 80% and 82% accuracy. The numerically best result was 82.1%, obtained for proSim$_{\text{cnt}}$, Variant 3, with k=8 in the normal setting. The difference to the best bag-of-URI model is more than 5% (absolute) accuracy. We tested the difference for statistical significance with bootstrap resampling [15] and obtained a negative result. We believe, however, that this outcome is mainly due to the small size of our current test set.

To illustrate the behavior of proSim family in more detail, Figure 5 shows a precision-recall evaluation for the most promising models for each class. This mode of evaluation still assumes the same decision rule (cf. Section 3.1) but does not optimize the threshold on training data: Rather, it varies the threshold between 0 and 1 and computes at each point the precision and recall for the positive ("same topic") class. At θ=0, the recall is 1, but the precision only 0.5, due to the design of the dataset. At θ=1, the precision is (close to) 1, but the recall (close to) 0. The closer the curve to the top right corner (high precision and recall) the better. This evaluation provides more detail on the performance of the similarity measures across the range of document similarities.

Figure 5 demonstrates that the precision-recall evaluation corresponds well to the accuracy-based results reported above. The bag-of-word model is outperformed by the bag-of-URI model for almost all values of θ, which is in turn outperformed by proSim$_{\text{cnt}}$. The plot shows that the particular benefit provided by the structural models is the ability to retain a high precision for much higher recall rates compared to the other models. For example, they show an almost perfect precision for a recall of up to 0.6, where the shallower models have dropped to a precision below 0.90; for recall values between 0.6 and 0.8, the difference in precision remains at about 10% (absolute) before all curves converge for very high recall values.

In sum, we found a consistent numerical improvement for the structural measures compared to the URI measures. More generally, our results indicate that structurally informed measures of graph similarity can deliver practical benefits for applications, even for document-level tasks. Currently, however, our proSim measures do not profit either from more involved weighting schemes or from the inclusion of further Linked Open Data. Progress in this direction will require more research on possible weighting schemes and strategies to select informative features from the range of information present in LOD.

The raw data of all evaluations performed here including the generated RDF graphs is available via `http://www.aifb.kit.edu/web/LODifier`

4 Related Work

There are various approaches to extracting relationships from text. These approaches usually include the annotation of text with named entities and relations and the extraction of those relations. Two approaches that are very similar to LODifier are of [5] and [19]. They both use NER, POS-tagging and parsing to discover named entities and relations between them. The resulting relations are converted to RDF. The disadvantage of these methods is however that they use labelled data as a base for extracting relations, which is not flexible, as labelled data requires manual annotation.

In terms of the pursued goal, that is, processing natural language, converting the result into RDF and possibly exhibiting it as linked (open) data, LODifier shares the underlying motivation with the NLP2RDF framework.[4] The latter provides a generic and flexible framework of how to represent any kind of NLP processing result in RDF. While the current version of LODifier is a stand-alone tool not resting on this framework, a future integration might improve its interoperability and reusability further.

The Pythia system [20] which is targeted at natural language question answering in information systems also employs deep semantic analysis on posed questions in order to come up with a translation into SPARQL queries which are then posed against RDF stores. Pythia does, however, presume the existence of a lexicon specifying how lexical expressions are to be mapped to RDF entities of the queried data source. Thereby, the approach is inherently domain-specific, whereas we aim at an open domain setting where no a-priori lexical mappings or specific schematic information is available.

The *AsKNet* system [12] is aimed at automatically creating a semantic network. Thereby, the processing strategy is similar to ours: the system uses C&C and Boxer to extract semantic relations. To decide which nodes refer to the same entities, similarity scores are computed based on spreading activation and nodes are then mapped together. An approach building on AsKNet comes from [22]. They use AsKNet to build a semantic network based on relations between concepts instead of relations between named entities as already present in AsKNet. The resulting graph is then converted to RDF. AsKNet and LODifier differ in the way they disambiguate named entities. LODifier uses NER and WSD methods before generating RDF triples and describes the entities and relations using DBpedia and WordNet URIs whereas AsKNet first generates semantic structures from text and then tries to map nodes and edges together based on similarity. Moreover, the graph output of the latter is not interlinked with other data sources. This is one of the key features of LODifier, and we feel that we have only scratched the surface regarding the benefit of interlinking.

5 Conclusion and Outlook

Much current work in text processing makes exclusive use of shallow methods, either statistical or pattern-based, and makes up for their limitations by the redundancy in large data collections. Their main criticism towards deeper processing are the lack of robustness and efficiency.

In this paper, we have argued that (lack of) robustness is not a knock-out argument. We have presented LODifier, a proof-of-concept implementation which converts open-domain unstructured natural language into Linked Data. The system incorporates several well-established NLP methods: named entities are recognized using NER; normalized relations are extracted by parsing the text and performing deep semantic analysis; WSD helps identifying the proper senses for event types. The meaning of a document is finally consolidated into an RDF graph whose nodes are connected to the broad-coverage Linked Data vocabularies DBpedia and WordNet.

The central benefits that LODifier provides are (a) abstracting away from linguistic surface variation such as lexical or syntactic choice; (b) making explicit structural

[4] `http://nlp2rdf.org/about`

information as a semantic graph; and (c) linking up the concepts and relations in the input to the LOD cloud. These benefits provide types of additional information for subsequent processing steps, which are generally not provided by "shallow" approaches. Concretely, we have demonstrated that the LODifier representations improve topic detection over competitive shallow models by using a document similarity measure that takes semantic structure into account. More generally, we believe that methods like ours are suitable whenever there is little data, for example, in domain-specific settings.

A few of the design decisions made for the RDF output may not be adequate for all conceivable applications of LODifier. The use of blank nodes is known to bring about computational complications, and in certain cases it is beneficial to Skolemize them by URIs e.g. using MD5 hashes. Employing `owl:sameAs` to link discourse referents to their DBpedia counterparts might lead to unwanted logical ramifications in case the RDF output is flawed. Hence, we will provide a way to configure LODifier to produce RDF in an encoding that meets the specific requirements of the use case.

A current shortcoming of LODifier is its pipeline architecture which treats the modules as independent so that errors cannot be recovered. We will consider joint inference methods to find a globally most coherent analysis [11]. Regarding structural similarity measures, we have only scratched the surface of the possibilities. More involved graph matching procedures remain a challenge due to efficiency reasons; however, this is an area of active research [17].

Possible applications of LODifier are manifold. It could be used to extract DBpedia relation instances from textual resources, provided a mapping from WordNet entities to DBpedia relations is given. Moreover, our system could also be applied for hybrid search, that is, integrated search over structured and non-structured sources. In such a setting, the role of LODifier would be to "pre-process" unstructured information source (off-line) into a representation that matches the structured information sources. This would reduce on-line search to the application of structured query processing techniques to a unified dataset. Presuming that a good accuracy at the semantic micro-level can be achieved, our method could also prove valuable in the domain of question answering. In that case, LODifier could be used to transform structured resources into RDF against which structured (SPARQL) queries generated by question answering systems such as Pythia [20] could be posed.

Acknowledgements. We thank Pia Wojtinnek, Johan Bos and Stephen Clark for stimulating discussions and technical advice about the Boxer system as well as Christina Unger and Philipp Cimiano for pointers to related work. We acknowledge the funding of the European Union (project XLike in FP7).

References

1. Agirre, E., de Lacalle, O.L., Soroa, A.: Knowledge-based WSD and specific domains: performing over supervised WSD. In: Proceedings of the International Joint Conferences on Artificial Intelligence, Pasadena, CA (2009)
2. Allan, J.: Introduction to topic detection and tracking, pp. 1–16. Kluwer Academic Publishers, Norwell (2002)

3. van Assem, M., van Ossenbruggen, J.: WordNet 3.0 in RDF (2011), `http://semanticweb.cs.vu.nl/lod/wn30/` (Online; accessed July 12, 2011)
4. Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., Hellmann, S.: DBpedia - A crystallization point for the Web of Data. Web Semant. 7, 154–165 (2009)
5. Byrne, K., Klein, E.: Automatic extraction of archaeological events from text. In: Proceedings of Computer Applications and Quantitative Methods in Archaeology, Williamsburg, VA (2010)
6. Cafarella, M.J., Ré, C., Suciu, D., Etzioni, O., Banko, M.: Structured querying of web text: A technical challenge. In: Proceedings of the Conference on Innovative Data Systems Research, Asilomar, CA (2007)
7. Cimiano, P., Völker, J.: Text2Onto. In: Montoyo, A., Muńoz, R., Métais, E. (eds.) NLDB 2005. LNCS, vol. 3513, pp. 227–238. Springer, Heidelberg (2005)
8. Curran, J.R., Clark, S., Bos, J.: Linguistically Motivated Large-Scale NLP with C&C and Boxer. In: Proceedings of the ACL 2007 Demo Session, pp. 33–36 (2007)
9. Dijkstra, E.W.: A note on two problems in connexion with graphs. Numerische Mathematik 1, 269–271 (1959)
10. Fellbaum, C. (ed.): WordNet: An Electronic Lexical Database. MIT Press (1998)
11. Finkel, J.R., Manning, C.D.: Hierarchical Joint Learning: Improving Joint Parsing and Named Entity Recognition with Non-Jointly Labeled Data. In: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pp. 720–728 (2010)
12. Harrington, B., Clark, S.: Asknet: Automated semantic knowledge network. In: Proceedings of the American Association for Artificial Intelligence, Vancouver, BC, pp. 889–894 (2007)
13. Hitzler, P., Krötzsch, M., Rudolph, S.: Foundations of Semantic Web Technologies. Chapman & Hall/CRC (2009)
14. Kamp, H., Reyle, U.: From Discourse to Logic: Introduction to Model-theoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory. Studies in Linguistics and Philosophy, vol. 42. Kluwer, Dordrecht (1993)
15. Koehn, P.: Statistical Significance Tests for Machine Translation Evaluation. In: Proceedings of Empirical Methods in Natural Language Processing, Barcelona, Spain, pp. 388–395 (2004)
16. Li, Y., Chu, V., Blohm, S., Zhu, H., Ho, H.: Facilitating pattern discovery for relation extraction with semantic-signature-based clustering. In: Proceedings of the ACM Conference on Information and Knowledge Management, pp. 1415–1424 (2011)
17. Lösch, U., Bloehdorn, S., Rettinger, A.: Graph Kernels for RDF Data. In: Simperl, E., et al. (eds.) ESWC 2012. LNCS, pp. 134–148. Springer, Heidelberg (2012)
18. Milne, D., Witten, I.H.: Learning to link with Wikipedia. In: Proceedings of the ACM Conference on Information and Knowledge Management (2008)
19. Ramakrishnan, C., Kochut, K.J., Sheth, A.P.: A Framework for Schema-Driven Relationship Discovery from Unstructured Text. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L.M. (eds.) ISWC 2006. LNCS, vol. 4273, pp. 583–596. Springer, Heidelberg (2006)
20. Unger, C., Cimiano, P.: Pythia: Compositional Meaning Construction for Ontology-Based Question Answering on the Semantic Web. In: Muñoz, R., Montoyo, A., Métais, E. (eds.) NLDB 2011. LNCS, vol. 6716, pp. 153–160. Springer, Heidelberg (2011)
21. Valiente, G.: Algorithms on Trees and Graphs. Springer, Berlin (2002)
22. Wojtinnek, P.-R., Harrington, B., Rudolph, S., Pulman, S.: Conceptual Knowledge Acquisition Using Automatically Generated Large-Scale Semantic Networks. In: Croitoru, M., Ferré, S., Lukose, D. (eds.) ICCS 2010. LNCS, vol. 6208, pp. 203–206. Springer, Heidelberg (2010)