Real-time RDF extraction from unstructured data streams

Daniel Gerber, Sebastian Hellmann, Lorenz Bühmann, Tommaso Soru, and Axel-Cyrille Ngonga Ngomo

Universität Leipzig, Institut für Informatik, AKSW,
Postfach 100920, D-04009 Leipzig, Germany,
{dgerber|hellmann|buehmann|tsoru|ngonga}@informatik.uni-leipzig.de
http://aksw.org

Abstract. The vision behind the Web of Data is to extend the current document-oriented Web with machine-readable facts and structured data, thus creating a representation of world knowledge. However, most of the Web of Data is limited to being a large compendium of encyclopedic knowledge describing entities. A huge challenge, the timely and massive extraction of RDF facts from unstructured data, has remained open so far. The availability of such knowledge on the Web of Data would provide significant benefits to manifold applications including news retrieval, sentiment analysis and business intelligence. In this paper, we address the problem of the actuality of the Web of Data by presenting an approach that allows extracting RDF triples from unstructured data streams. We employ statistical methods in combination with deduplication, disambiguation and unsupervised as well as supervised machine learning techniques to create a knowledge base that reflects the content of the input streams. We evaluate a sample of the RDF we generate against a large corpus of news streams and show that we achieve a precision of more than 85%.

1 Introduction

Implementing the original vision behind the Semantic Web requires the provision of a Web of Data which delivers timely data at all times. The foundational example presented in Berners-Lee et al's seminal paper on the Semantic Web [3] describes a software agent who is tasked to find medical doctors with a rating of excellent or very good within 20 miles of a given location at a given point in time. This requires having timely information on which doctors can be found within 20 miles of a particular location at a given time as well as having explicit data on the rating of said medical doctors. Even stronger timeliness requirement apply in decision support, where software agents help humans to decide on critical issues such as whether to buy stock or not or even how to plan their drive through urban centers. Furthermore, knowledge bases in the Linked Open Data (LOD) cloud would be unable to answer queries such as "Give me all news of the last week from the New York Times pertaining to the director of a company". Although

the current LOD cloud has tremendously grown over the last years [1], it delivers mostly encyclopedic information (such as albums, places, kings, etc.) and fails to provide up-to-date information that would allow addressing the information needs described in the examples above.

The idea which underlies our work is thus to alleviate this current drawback of the Web of Data by developing an approach that allows extracting RDF from unstructured (i.e., textual) data streams in a fashion similar to the live versions of the DBpedia[1] and LinkedGeoData[2] datasets. The main difference is yet that instead of relying exclusively on structured data like LinkedGeoData or on semi-structured data like DBpedia, we rely mostly on unstructured, textual data to generate RDF. By these means, we are able to unlock some of the potential of the documents Web, of which up to 85% is unstructured [7]. To achieve this goal, our approach, dubbed RdfLiveNews, assumes that it is given unstructured data streams as input. These are deduplicated and then used as basis to extracte patterns for relations between known resources. The patterns are then clustered to labeled relations which are finally used as basis for generating RDF triples. We evaluate our approach against a sample of the RDF triples we extracted from RSS feeds and show that we achieve a very high precision.

The remainder of this work is structured as follows: We first give an overview of our approach and give detailed insights in the different steps from unstructured data streams to RDF. Then, we evaluate our approach in several settings. We then contrast our approach with the state of the art and finally conclude.

2 Overview

We implemented the general architecture of our approach dubbed RdfLiveNews according to the pipeline depicted in Figure 1. First, we gather textual data from data streams by using RSS feeds of news articles. Our approach can yet be employed on any unstructured data published by a stream. Since input streams from the Web can be highly redundant (i.e., convey the same information), we then deduplicate the set of streams gathered by our approach. Subsequently, we apply a pattern search to find lexical patterns for relations expressed in the text. After a refinement step with background knowledge, we finally cluster the extracted patterns according to their semantic similarity and transform this information into RDF.

2.1 Data Acquisition

restructure this into a more generic notation section, SH: notation seems fine ->delete

SH: I rephrased the following sentence

Formally, our approach aims to process the output of unstructured data sources S^i by continuously gathering the data streams D^i that they generate. Each data stream consists of atomic elements d^i_j (in our case sentences), which are ordered by time and which can overlap with other data streams D^k. The first step of our

[1] `http://live.dbpedia.org/sparql`
[2] `http://live.linkedgeodata.org/sparql`

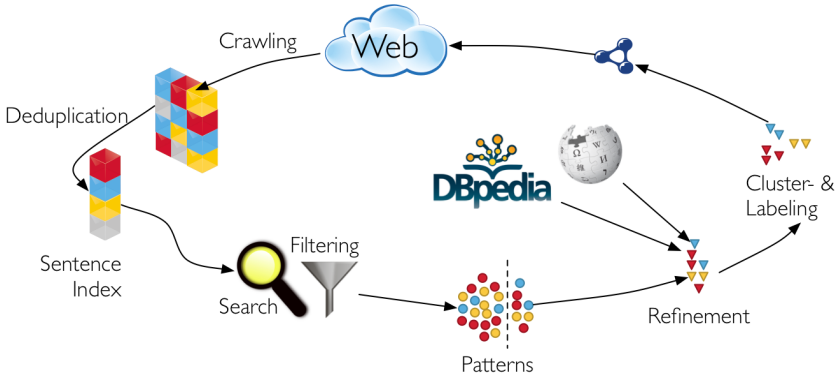

Fig. 1. Overview of the generic time slice based stream processing.

algorithm thus consists of a *data acquisition step* in which we access and preprocess such data streams. In the following, we instantiate this workflow by using the processing of RSS feeds against DBpedia. The workflow is yet generic and can be applied to any source of unstructured data. Data gathering from RSS is carried out by creating an *index* of the sentences contained in the data streams. This index is generated by crawling a set of RSS feeds for a given duration, which we call the *time slice duration*. The data gathered from all sources during a time slice duration with given start and end time is called a *time slice*. The main content of each data stream D^i for a time slice t^i is extracted and cleaned from HTML markup. We apply sentence splitting on the resulting text and store the extracted sentences in an index. This workflow gets repeated until we reach the end of the time slice. The resulting data is forwarded to the data analysis components of the approach while a new time slice is begun in parallel. These components rely on two main resources for the disambiguation and linking of resources: (1) The first is created from a Wikipedia dump. Again, we apply a sentence boundary disambiguation on all Wikipedia articles and store the resulting sentences in a Lucene index. Furthermore we apply the named entity recognition tool and POS tagger from Stanford Core NLP framework[3] on all sentences and store all occurring named entities alongside the sentences and their article URL. (2) The second resource is an index of DBpedia. We store all URIs from DBpedia with *rdfs:label* and *rdf:type*[4] information, a precomputed disambiguation score (see Section 2.4) and a list of surface forms ("U.S." or "USA" for the *United States of America*) of each resource. This list is compiled by analyzing the *redirect* and *disambiguation* links from Wikipedia as presented in [12]. Note that the extraction of all resources required here is fully automated.

[3] `http://www-nlp.stanford.edu/software/corenlp.shtml`
[4] `http://www.w3.org/TR/rdf-schema/`

Table 1. Number of non-duplicate sentences in 1% of the data extracted from 1457 RSS feeds within a window of 10 time slices (2h each). The second column shows the original number of sentences without duplicate removal.

Iteration	No deduplication	$\theta = 1.0$	$\theta = 0.95$	$\theta = 0.9$	$\theta = 0.85$	$\theta = 0.8$
1	2997	2764	2764	2759	2754	2753
5	3047	2335	2334	2327	2320	2315
10	3113	2033	2040	2022	2015	2010
15	2927	1873	1868	1866	1864	1855
20	3134	1967	1966	1949	1945	1941
25	3065	1936	1932	1924	1919	1915
30	3046	1941	1940	1933	1928	1924

2.2 Deduplication

The aim of the deduplication step is to remove very similar elements from data streams before the RDF extraction. This removal accounts for some Web data streams simply repeating the content of one of several other streams. Our deduplication approach is based on measuring the similarity of single elements s_i and s_j found in unstructured streams. Elements of streams are considered to be different iff $qgrams(s_i, s_j) < \theta$, where $\theta \in [0, 1]$ is a similarity threshold and $qgrams(s_i, s_j)$ measures the similarity of two strings by computing the Jaccard similarity of the trigrams they contain. Given that the number of stream items to deduplicate can be very large, we implemented the following two-step approach: For each stream S^i, we first deduplicate the elements s^i_j within S^i. This results in a duplicate-free data stream $D^i = \{d^i_j : (d^i_j \in S^i) \wedge (\forall s^i_k \in S^i \ \exists d^i_j \in D^i \ qgrams(s^i_k, d^i_j) \geq \theta) \wedge (\forall d^i_j, d^i_k \in D^i \ qgrams(d^i_k, d^i_j) < \theta)\}$. The elements of D^i are then compared to all other elements of the w previous streams D^{i-1} to D^{i-w}, where w is the size of the deduplication window. Only the non-duplicate elements of D^i are stored for further processing. To ensure the scalability of the deduplication step, we are using deduplication algorithms implemented in the LIMES framework [16]. Table 1 gives an overview of the number of unique data stream items in our dataset when using different deduplication thresholds.

SH: I still can not understand, what an iteration in table 1 is?

2.3 Pattern Search and Filtering

In order to find patterns we first apply Named Entity Recognition (NER) and Part of Speech (POS) tagging on the deduplicated sentences. RdfLiveNews can use two different ways to extract patterns from annotated text. The POS tag method uses *NNP* and *NNPS*[5] tagged tokens to identify a relation's subject and object, whereas the Named Entity Tag method relies on *Person*, *Location*, *Organization* and *Miscellaneous* tagged tokens. In an intermediate step all consecutive POS and NER tags are merged. An unrefined RdfLiveNews pattern p is

[5] All POS tags can be found in the Penn Treebank Tagset

now defined as a pair $p = (\theta, \mathcal{S}_\theta)$, where θ is the natural language representation (NLR) of p and $\mathcal{S}_\theta = \{(s_i, o_i) : i \in \mathbb{N}; 1 \leq i \leq n\}$ is the support set of θ, a set of the subject and object pairs. For example the sentence:

David/NNP hired/VBD John/NNP ,/, former/JJ manager/NN of/IN ABC/NNP ./.

would result in the patterns:

$p_1 = (\ [hired], \{(David,\ John)\}$ and
$p_2 = ([,\ former\ manager\ of], \{(John,\ ABC)\})$.

After the initial pattern acquisition step, we filter all patterns to improve their quality. We discarded all patterns, that did not match these criteria: The pattern should (1) contain at least a verb or a noun, (2) contain at least one salient word (i.e. a word that is not a stop word), (3) not contain more than one non-alpha-numerical character (except ", ' ‘") and (4) be shorter than 50 characters. Since the resulting list still contains patterns of low quality, we first sort it by the number of elements of the support set $\mathcal{S}_\theta$ and solely select the top 1% for pattern refinement to ensure high quality.

2.4 Pattern Refinement

The goal of this step is to find a suitable *rdfs:range* and *rdfs:domain* as well as to disambiguate the support set of a given pattern. To achieve this goal we first try to find an URI for the subjects and objects in the support set of p by matching the pairs to entries in a knowledge base. With the help of those URIs we can query the knowledge base for the classes (*rdf:type*) of the given resources and compute a common *rdfs:domain* for the subjects of p and *rdfs:range* for the objects respectively. A refined RdfLiveNews pattern p_r is now defined as a quadruple $p_r = (\theta, \mathcal{S}_\theta{}', \delta, \rho)$, where θ is the natural language representation, $\mathcal{S}_\theta{}'$ the disambiguated support set, δ the *rdfs:domain* and ρ the *rdfs:range* of p_r.

To find the URIs of each subject-object pair $(s, o) \in \mathcal{S}_\theta$ we first try to complete the entity name. This step is necessary and beneficial because entities usually get only written once in full per article. For example the newly elected president of the United States of America might be referenced as “President Barack Obama” in the first sentence of a news entry and subsequently be referred to as “Obama”. In order to find the subjects’ or objects’ full name, we first select all named entities $e \in \mathcal{E}_a$ of the article the pair (s, o) was found in. We then use the longest matching substring between s (or o) and all elements of $\mathcal{E}_a$ as the name of s or o respectively. Additionally we can filter the elements of $\mathcal{E}_a$ to contain only certain NER types. Once the complete names of the entities are found, we can use them to generate a list of URI candidates $\mathcal{C}_{uri}$. This list is generated with the help of a Lucene query for the given entity name on the list of surface forms as described in Section 2.1. Each URI candidate $c \in \mathcal{C}_{uri}$ is now evaluated on four different features and the combined score of those features is used to rank the candidates and chose the most probable URI for an entity. The first feature is the *Apriori*-score $a(c)$ of the URI candidate c, which is calculated beforehand for all URIs in the knowledge base by analyzing the number of inbound links of

c by the following formula: $a(c) = \log(inbound(c) + 1)$. The second and third features are based on the context information found in the Wikipedia article of c and the news article text (s, o) was found in. For the *global context*-score c_g we apply a co-occurrence analysis of the entities $\mathcal{E}_a$ found in the news article and the entities $\mathcal{E}_w$ found in the Wikipedia article of c. The *global context*-score is now computed as $c_g(\mathcal{E}_a, \mathcal{E}_w) = |\mathcal{E}_a \cap \mathcal{E}_w| \,/\, |\mathcal{E}_a \cup \mathcal{E}_w|$. The *local context*-score c_l is the number of mentions of the second element of the pair (s, o), o in the case of s and vice versa, in $\mathcal{E}_w$. The last feature to determine a URI for an entity is the maximum string similarity sts between s (or o) and the elements of the list of surface forms of c. We used the qgram distance[6] as the string similarity metric. We normalize all non-$[0, 1]$ features (c_g, c_l, a) by applying a minimum-maximum normalization of the corresponding scores for $\mathcal{C}_{uri}$ and multiply it with a weight parameter which leads to the overall URI score:

$$c(s, o, uri) = \frac{\frac{\alpha a}{a_{max}} + \frac{\beta c_g}{c_{g_{max}}} + \frac{\gamma c_l}{c_{l_{max}}} + \delta sts}{4}$$

If the URI's score is above a certain threshold $\lambda \in [0, 1]$ we use it as the URI for s, otherwise we create a new URI. Once we have computed the URIs for all pairs $(s, o) \in \mathcal{S}_\theta$ we determine the most likely *domain* and *range* for p_r. This is done by analyzing the *rdf:type* statements returned for each subject or object in $\mathcal{S}_\theta$ from a background knowledge base. Since the DBpedia ontology is designed in such a way, that classes do only have one super-class, we can easily analyze its hierarchy. We implemented two different determination strategies for analyzing the class hierarchy. The first strategy, dubbed "most general", selects the highest class in the hierarchy for each subject (or object) and uses the most occurring class as *domain* or *range* of p_r. The second strategy, dubbed "most specific", works similar to the "most general" strategy with the difference that it uses the most descriptive class to select the *domain* and *range* of p_r.

2.5 Pattern Similarity and Clustering

In order to cluster patterns according to their meaning, we created a set of similarity measures. A similarity measure takes two patterns p_1 and p_2 as input and outputs the similarity value $s(p_1, p_2) \in [0, 1]$. As a baseline we implemented a qgram measure, which calculates the string similarity between all non stop words of two patterns. Since this baseline measure fails to return a high similarity for semantically related, but not textually similar patterns like "'s attorney ," and "'s lawyer ," we also implemented a Wordnet measure. As a first step the Wordnet similarity measure filters out the stop words of p_1 and p_2 and applies the Stanford lemmatizer on the remaining tokens. Subsequently, for all token combinations of p_1 and p_2, we apply a Wordnet Similarity metric (Path [18], Lin [11] and Wu & Palmer [23]) and select the maximum of all comparisons as the similarity value $s(p_1, p_2)$. As a final similarity measure we created a Wordnet

[6] `http://sourceforge.net/projects/simmetrics/`

and string similarity measure with the help of a linear combination from the before-mentioned metrics. In this step we also utilize the *domain* and *range* of p_r. If this feature is enabled, a similarity value between two patterns p_1 and p_2 can only be above 0, iff $\{\delta_{p_1}, \rho_{p_1}\} \setminus \{\delta_{p_2}, \rho_{p_2}\} = \emptyset$.

The result of the similarity computation can be regarded as a similarity graph $G = (V, E, \omega)$, where the vertices are patterns and the weight $\omega(p_1, p_2)$ of the edge between two patterns is the similarity of these patterns. Consequently, unsupervised machine learning and in particular graph clustering is a viable way of finding groups of patterns that convey similar meaning. We opted for using the BorderFlow clustering algorithm [17] as it is parameter-free and has already been used successfully in diverse applications including clustering protein-protein interaction data and queries for SPARQL benchmark creation [13]. For each node $v \in V$, the algorithm begins with an initial cluster X containing only v. Then, it expands X iteratively by adding nodes from the direct neighborhood of X to X until X is node-maximal with respect to the border flow ratio described in [13]. The same procedure is repeated over all nodes. As different nodes can lead to the same cluster, identical clusters (i.e., clusters containing exactly the same nodes) that resulted from different nodes are subsequently collapsed to one cluster. The set of collapsed clusters and the mapping between each cluster and the nodes that led to it are returned as result.

2.6 Cluster Labeling and Merging

Based on the clusters $\mathcal{C}$ obtained through the clustering algorithm, this step selects descriptive labels for each cluster $c_i \in \mathcal{C}$, which can afterwards be used to merge the clusters. In the current version, we apply a straightforward majority voting algorithm, i.e. for each cluster c_i, we select the most frequent natural language representation θ (stop words removed) occurring in the patterns of c_i. Finally, we use the representative label of the clusters to merge them using a string similarity and WordNet based similarity measure. Taking into account that this kind of similarity measures are not transitive, with focus on accuracy. We are currently only applying this merging procedure once, although it could indeed by applied iteratively to further reduce the number of clusters.

2.7 Mapping to RDF and Publication on the Web of Data

To close the circle of the round-trip pipeline of RdfLiveNews, the following pre-requisite steps are required to re-publish the extraction results in a sensible way:

1. The facts and properties contained in the internal data structure of our tool have to be mapped to OWL.
2. Besides the extracted factual information several other aspects and meta data are interesting as well, such as extraction and publication data and provenance links to the text the facts were extracted from
3. URIs need to be minted to provide the extracted triples as linked data.

Mapping to OWL. Each cluster $c_i \in \mathcal{C}$ represents an *owl:ObjectProperty* $prop_{c_i}$. The *rdfs:domain* and *rdfs:range* of $prop_{c_i}$ is determined by a majority voting algorithm with respect to δ and ρ of all $p_r \in \mathcal{C}$. The *skos:prefLabel*[7] of $prop_{c_i}$ is the label determined by the cluster labeling step and all other NLRs of the patterns in c_i get associated with $prop_{c_i}$ as *skos:altLabels*. For each subject-object pair in $\mathcal{S}_\theta{}'$ we produce a triple by using $prop_{c_i}$ as predicate and by assigning learned entity types from DBpedia or *owl:Thing*.

Provenance tracking with NIF. Besides converting the extracted facts from the text, we are using the current draft of the NLP Interchange Format (NIF) Core ontology[8] to serialize the following information in RDF: the sentence the triple was extracted from, the extraction date of the triple, the link to the source URL of the data stream item, the publication date of the item on the stream. Furthermore, NIF allows us to link each element of the extracted triple to their origin in the text for further reference and querying.

NIF is an RDF/OWL based format to achieve interoperability between language tools, annotation and resources. NIF offers several URI schemes to create URIs for strings, which can then be used as subjects for annotation. We employ the NIF URI scheme, which is grounded on URI fragment identifiers for text (RFC 5147[9]). NIF was previously used by NERD [19] to link entities to text. For our use case, we extended NIF in two ways: (1) we added the ability to represent extracted triples via the ITS 2.0 / RDF Ontology[10]. *itsrdf:taPropref* is an *owl:AnnotationProperty* that links the NIF String URI to the *owl:ObjectProperty* by RdfLiveNews. The three links from the NIF String URIs (str_1, str_2,str_3) to the extracted triple (s, p,o) itself make it well traceable and queryable: $str_1 \mapsto s$, $str_2 \mapsto p$, $str_3 \mapsto o$, $s \mapsto p \mapsto o$ An example of NIF RDF serialization is shown in Listing 1. (2) Although the NERD paper [19] already suggested the minting of new URIs, a concrete method for doing so was not yet researched. In RdfLiveNews we use the source URL of the data stream item to re-publish the facts for individual sentences as linked data. We strip the scheme component (http://) of the source URL and percent encode the ultimate part of the path and the query component[11] and add the md5 encoded sentence to produce the following URI:

```
http://rdflivenews.aksw.org/extraction/ + example.com:8042/over/ +
    urlencode(there?name=ferret) + / +md5(‘sentence‘)}
```

```
@base <http://rdflivenews.aksw.org/extraction/www.necn.com/07/04/12/
    Scientists-discover-new-subatomic-partic/landing.html\%3FblockID
    \%3D735470\%26feedID\%3D4213/8a1e5928f6815c99b9d2ce613cf24198#>.
## for prefixes please use http://prefix.cc, e.g. http://prefix.cc/
    rlno
## extracted property + result of linking
rlno:directorOf a owl:ObjectProperty ;
  skos:prefLabel "director of" , skos:altLabel ", director of " ;
  owl:equivalentProperty dbp:director .
```

[7] http://www.w3.org/2004/02/skos/

[8] http://persistence.uni-leipzig.org/nlp2rdf/ontologies/nif-core#

[9] http://tools.ietf.org/html/rfc5147

[10] http://www.w3.org/2005/11/its/rdf#

[11] http://tools.ietf.org/html/rfc3986#section-3

```
## extracted facts:
rlnr:Rolf_Heuer a dbo:Person ;
  rdfs:label "Rolf Heuer"@en ;
  rlno:directorOf dbpedia:CERN .
dbpedia:CERN a owl:Thing ;
  rdfs:label "CERN"@en .
## provenance tracking with NIF:
<char=0,10>   itsrdf:taClassRef dbo:Person ;
  itsrdf:taIdentRef rlnr:Rolf_Heuer .
<char=14,18>  itsrdf:taIdentRef dbpedia:CERN .
<char=11,24>  nif:anchorOf      ", director of"^^xsd:string ;
  itsrdf:taPropRef rlno:directorOf .
## detailed NIF output with context, indices and anchorOf
<char=0,> a nif:String, nif:Context, nif:RFC5147String ;
  nif:isString "Rolf Heuer , director of CERN , said the newly
      discovered particle is a boson , but he stopped just shy of
      claiming outright that it is the Higgs boson itself - an
      extremely fine distinction." ;
  nif:sourceUrl <http://www.necn.com/07/04/12/Scientists-discover-
      new-subatomic-partic/landing.html?blockID=735470&feedID=4213>;
## extraction date:
  dcterms:created "2013-05-09T18:27:08+02:00"^^xsd:dateTime .
## publishing date:
<http://www.necn.com/07/04/12/Scientists-discover-new-subatomic-
    partic/landing.html?blockID=735470&feedID=4213>
  dcterms:created "2012-08-15T14:48:47+02:00"^^xsd:dateTime .
<char=0,10> a nif:String, nif:RFC5147String ;
  nif:referenceContext <char=0,>; nif:anchorOf "Rolf Heuer" ;
  nif:beginIndex "0"^^xsd:long ; nif:endIndex "10"^^xsd:long ;
```

Listing 1. Example RDF extraction of RdfLiveNews

Republication of RDF. The extracted triples are hosted on: `http://rdflivenews.aksw.org`. The data for individual sentences is crawlable via the file system of the Apache2 web server. We assume that source URLs only occur once in a stream when the document is published and the files will not be overwritten. Furthermore, the extracted properties and entities are available as linked data at *http://rdflivenews.aksw.org/{ontology|resource}/$name* and they can be queried via SPARQL at `http://rdflivenews.aksw.org/sparql`.

2.8 Linking

The approach described above generates a set of properties with several labels. In our effort to integrated this data source into the Linked Open Data Cloud, we use the deduplication approach proposed in Section 2.2 to link our set of properties to existing knowledge bases (e.g., DBpedia). To achieve this goal, we consider the set of properties we generated as set of source instances S while the properties of the knowledge base to which we link are considered to be a set of target T. Two properties $s \in S$ and $t \in T$ are linked iff $trigrams(s,t) \geq \theta_p$, where $\theta_p \in [0,1]$ is the property similarity threshold.

3 Evaluation

The aim of our evaluation was three-fold. First, we aimed at testing how well RdfLiveNews is able to disambiguate found entities. Our second goal was to

it's more like four :)

determine if the proposed similarity measures can be used to cluster patterns with respect to their semantic similarity. Finally, we wanted to measure if all computational heavy tasks can be applied in real-time, meaning the processing of one iteration takes less time than its compilation.

something is missing after wolrd,

For this evaluation we used a list of 1457 RSS feeds as compiled in [9]. The list includes all major worldwide newspapers and a wide range of topics, e.g. *World*, We crawled this list for 76 hours, which resulted in a corpus, dubbed *100%* of 38 time slices of 2 hours and 11.7 million sentences. Additionally we created two subsets of this corpus by randomly selecting 1% and 10% of the contained sentences. All evaluations were carried out on a MacBook Pro with a quad-core Intel Core i7 (2GHz), a solid state drive and 16 GB of RAM.

3.1 URI Disambiguation

To evaluate the URI disambiguation we created a gold standard manually. We took the 1% corpus, applied deduplication with a window size of 40 (contains all time slices) and a threshold of 1 (identical sentences), which resulted in a set of 69884 unique sentences. On those sentences we performed the pattern extraction with part of speech tagging as well as filtering. In total we found 16886 patterns and selected the Top 1%, which have been found by 1729 entity pairs. For 473 of those entity pairs we manually selected a URI for subject and object. This resulted in an almost equally distributed gold standard with 456 DBpedia and 478 RdfLiveNews URIs. We implemented a hill climbing approach with random initialization to optimize the parameters (see Section 2.4). The precision of our approach is the ratio between correctly found URIs for subject and object to the number of URIs above the threshold λ as shown in Equation 1. The recall, shown in Equation 2, is determined by the ratio between the number of correct subject and object URIs and the total number of subjects and objects in the gold standard. The F_1 measure is determined as usual by: $F_1 = 2 \cdot \frac{P \cdot R}{P+R}$. We optimized our approach for precision since we can compensate a lower recall and could achieve a precision of **85.01%** where the recall is **40.69%** and the resulting F_1 is **55.03%**. The parameters obtained through the hill-climbing search indicate that the *Apriori*-score is the most influential parameter (1.0), followed by *string-similarity* (0.78), *local-context* (0.6), global context (0.45) and a URI score threshold of 0.61. If we optimize for F_1, we were able to achieve a F_1 measure of **66.49%** with a precision of **67.03%** and a recall of **65.95%**.

For 487 out of the 934 URI in the gold standard no confident enough URI could be found. The most problems occured for DBpedia URIs which could not be determined in 305 cases, in comparison to 182 URIs for newly created resources. Additionally, for 30 resources RdfLiveNews created new URIs where DBpedia URIs should be used and in 0 cases a DBpedia URI was used where a new resource should be created. The reason for those mistakes are tagging errors, erroneous spellings and missing context information. For example Wikipedia has 97 disambiguations for “John Smith” which can not be disambiguated without prior knowledge.

$$P = \frac{|s_{uri_c}| + |o_{uri_c}|}{|s_{uri}| + |o_{uri}|} \quad (1) \qquad R = \frac{|s_{uri_c}| + |o_{uri_c}|}{2 \cdot |GS|} \quad (2)$$

3.2 Pattern Clustering

To evaluate the similarity generation as well as the clustering algorithm we relied on the measures Sensitivity, Positive Predictive Value (PPV) and Accuracy. We used the adaptation of those measures as presented in [4] to measure the match between a set of pattern mappings[12] from the gold standard and a clustering result. The gold standard was created by clustering the patterns as presented in the previous section manually. This resulted in a list of 25 clusters with more than 1 pattern and 54 clusters with 1 pattern. Since cluster with a size of 1 would skew our evaluation into unjustified good results, we excluded them from this evaluation.

Sensitivity. With respect to the clustering gold standard, we define sensitivity as the fraction of patterns of pattern mapping i which are found in cluster j. In $Sn_{i,j} = T_{i,j}/N_i$, N_i is the number of patterns belonging to pattern mapping i. We also calculate a pattern mapping-wise sensitivity Sn_{pm_i} as the maximal fraction of patterns of pattern mapping i assigned to the same cluster. $Sn_{pm_i} = max_{j=1}^{m} Sn_{i,j}$ reflects the coverage of pattern mapping i by its best-matching cluster. To characterize the general sensitivity of a clustering result, we compute a clustering-wise sensitivity as the weighted average of Sn_{pm_i} over all pattern mappings: $Sn = \frac{\sum_{i=1}^{n} N_i Sn_{pm_i}}{\sum_{i=1}^{n} N_i}$.

Positive Predictive Value. The positive predictive value is the proportion of members of cluster j which belong to pattern mapping i, relative to the total number of members of this cluster assigned to all pattern mappings. $PPV_{i,j} = T_{i,j}/\sum_{i=1}^{n} T_{i,j} = T_{i,j}/T_{.j}$
$T_{.j}$ is the sum of column j. We also calculate a cluster-wise positive predictive value PPV_{cl_j}, which represents the maximal fraction of patterns of cluster j found in the same annotated pattern mapping. $PPV_{cl_j} = max_{i=1}^{n} PPV_{i,j}$ reflects the reliability with which cluster j predicts that a pattern belongs to its best-matching pattern mapping. To characterize the general PPV of a clustering result as a whole, we compute a clustering-wise PPV as the weighted average of PPV_{cl_j} over all clusters: $PPV = \frac{\sum_{j=1}^{m} T_{.j} PPV_{cl_j}}{\sum_{j=1}^{m} T_{.j}}$.

Accuracy. The geometric accuracy (Acc) indicates the tradeoff between sensitivity and positive predictive value. It is obtained by computing the geometrical mean of the Sn and the PPV: $Acc = \sqrt{Sn \cdot PPV}$.

We evaluated the three similarity measures with respect to the underlying WordNet similarity metric (see Section 2.5). Furthermore we varied the clustering similarity threshold between 0.1 and 1 with a 0.1 step size. In case of the qgram and WordNet similarity metric we performed a grid search on the WordNet and qgram parameter in $[0, 1]$ with a step size of 0.05. We achieved the best configuration with the qgram and WordNet similarity metric with an accuracy

[12] A pattern mapping maps NLRs to RDF properties.

Table 2. Accuracy of RDF Extraction on 1% dataset with varying cluster sizes.

Cluster size	1	2	3	4	5
Subject Accuracy	81.0%	88.0%	86.0%	85.7%	80.4%
Predicate Accuracy	86.0%	89.0%	90.0%	93.5%	100.0%
Object Accuracy	93.0%	91.0%	90.0%	94.8%	94.1%
Total Accuracy	86.5%	89.2%	88.5%	91.1%	90.6%
# of sample facts	100	100	100	77	51
# of distinct properties	28	22	12	6	1

Table 3. Example for linking between RdfLiveNews and DBpedia.

RdfLiveNews-URI	DBpedia-URI	Sample of cluster
rlno:directorOf	dbo:director	[manager], [, director of], [, the director of]
rlno:spokesperson	dbo:spokesperson	[, a spokeswoman for], [spokesperson], [, a spokesman for]
rlno:attorney	—	['s attorney ,], ['s lawyer ,], [attorney]

of **82.45%**, a sensitivity of **71.17%** and a positive predictive value of **95.51%**. The best WordNet metric is Lin, the clustering threshold 0.3 and the qgram parameter is with 0.45 significantly less influential than the WordNet parameter with 0.75. As a reference value, the plain WordNet similarity metric achieved an accuracy of 78.86% and the qgram similarity metric an accuracy of 69.1% in their best configuration.

3.3 RDF Extraction and Linking

To assess the quality of the RDF data extracted by RdfLiveNews, we sampled the output of our approach and evaluated it manually. We generated five different evaluation sets. Each set may only contain triples with properties of clusters having at least $i = 1 \ldots 5$ patterns. We selected 100 triples (if available) randomly for each test set. As the results in Table 2 show, we achieve high accuracy on subject and object disambiguation. As expected, the precision of our approach grows with the threshold for the minimal size of clusters. This is simply due to the smaller clusters having a higher probability of containing outliers and thus noise.

The results of the linking with DBpedia (see Table 3) showed the mismatch between the relations that occur in news and the relations designed to model encyclopedic knowledge. While some relations such as `dbo:director` are used commonly in news streams and in the Linked Data Cloud, relations with a more volatile character such as `rlno:attorney` which appear frequently in news text are not mentioned in DBpedia.

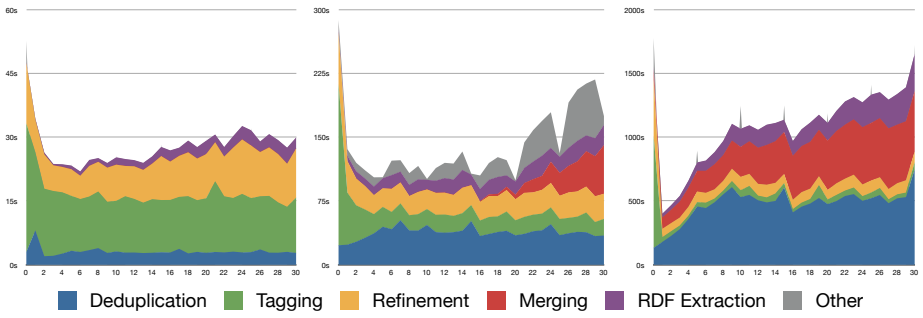

Fig. 2. Runtimes for different components and corpora (1% left, 10% middle, 100% right) per iteration.

3.4 Scalability

In order to perform real-time RDF extraction, the processing of the proposed pipeline needs to be done in less time than its acquisition requires. This also needs to be true for a growing list of RSS feeds. Therefore, we analyzed the time each module needed in each iteration and compared these values between the three test corpora. An early approximation of this evaluation implied that the pipeline indeed was not fast enough, which led to the parallelization of the pattern refinement and similarity generation. The results of this evaluation can be seen in Figure 2. With an average time slice processing time of about 20 minutes for the 100% corpus (2.2 minutes for 10% and 30s for 1%), our approach is clearly fit to handle up to 1500 RSS and more. The spike in the first iteration results out of the fact that RSS feeds contain the last n previous entries, which leeds to an disproportional large first time slice. The most time consuming modules are the deduplication, tagging and cluster merging. To tackle these bottlenecks we can for example parallelize sentence tagging and the deduplication.

The results of the growth evaluation for patterns until iteration 30 can be seen in Figure 3. The number of patterns grows with the factor of 3 from 1% to 10% and 10% to 100% corpora. Also, the number of patterns found by more than one subject-object pair increases approximately by factor 2. Additionally we observed a linear growth for all patterns (also for patterns with $|\mathcal{S}'_\theta| > 1$) and 100% showing the highest growth rate with a factor 2.5 over 10% and 4.8 over 10%.

The results of the growth evaluation for clusters can be seen in Figure 4. The evaluation shows that the number of clusters increases by a factor of 2.5 from 1% to 10% and 10% to 100%. Moreover, approximately 25% of all cluster have more than 1 pattern and the number of clusters grows linear for 1% and 10% but for the 100% corpus it seems to coverage to 800. The same holds true for clusters with more then one pattern, as they stop to grow at around 225 clusters.

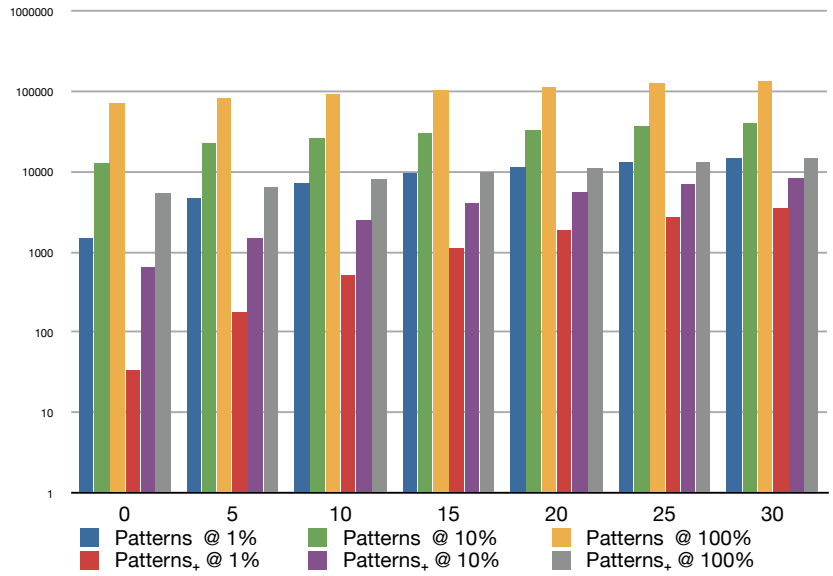

Fig. 3. Number of patterns (log scale) and patterns with $|\mathcal{S}'_\theta| > 1$ (Patterns$_+$) for iterations and test corpus.

4 Related Work

While Semantic Web applications rely on formal, machine understandable languages such as RDF and OWL, enabling powerful features such as reasoning and expressive querying, humans use Natural Language (NL) to express semantics. This gap between the two different languages has been filled by Information Extraction (IE) approaches, developed by the Natural Language Processing (NLP) research community [21], whose goal is to find desired pieces of information, such as concepts (hierarchy of terms which are used to point to shared definitions), entities (name, numeric expression, date) and facts in natural language texts and print them in a form that is suitable for automatic querying and processing. Ever since the advent of the Linked Open Data initiative[13], IE is also an important key enabler for the Semantic Web. For example, LODifier [2, **?**] combines deep semantic analysis with named entity recognition, word-sense disambiguation and controlled Semantic Web vocabularies. FOX [15] uses ensemble learning to improve the F-score of IE tools. The BOA framework [8] uses structured data as background knowledge for the extraction of natural language patterns, which are subsequently employed to extract additional RDF data from natural language text.The authors of [14] propose a simple model for fact extraction in real-time taking into account the difficult challenges that timely fact extraction on frequently updated data entails. A specific application for the news domain

[13] `http://linkeddata.org/`

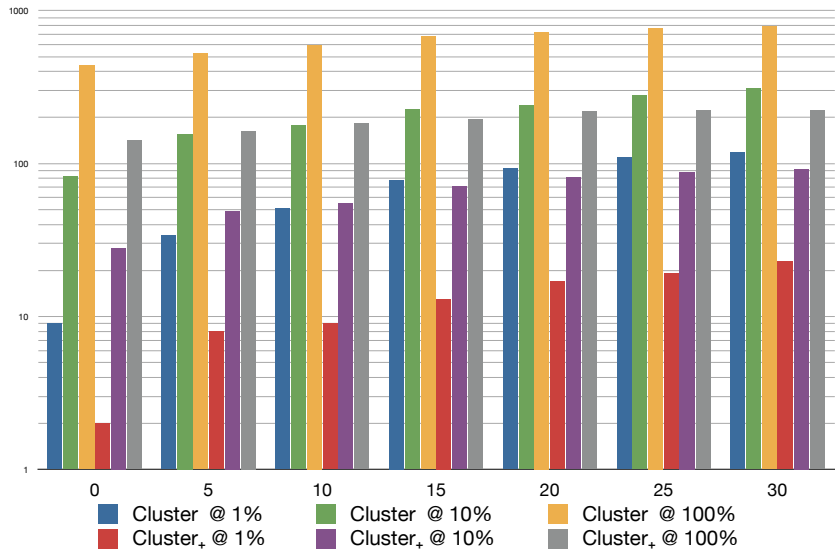

Fig. 4. Number of clusters (log scale) and clusters with $|\mathcal{C}| > 1$ (Cluster_+) for iterations and test corpus.

is described in [22], wherein a knowledge base of entities for the French news agency AFP is populated.

State-of-the-art open-IE system such as ReVerb automatically identifies and extracts relationships from text, relying on (in the case of ReVerb) simple syntactic constraints expressed by verbs [6]. The authors of [5] present a novel pattern clusters method for nominal relationship classification using an unsupervised learning environment, which makes the system domain and language-independent. [20] shows how lexical patterns and semantic relationships can be learned from concepts in Wikipedia.

5 Conclusion and Future Work

In this paper, we presented RdfLiveNews, a framework for the extraction of RDF from unstructured data streams. We presented the components of the RdfLiveNews framework and evaluated its disambiguation, clustering, linking and scalability capabilities as well as its extraction quality. We are able to disambiguate resources with a precision of 85%, cluster patterns with an accuracy of 82.5% and extract RDF with an total accuracy of around 90% and handle two hour time slices with around 300.000 sentences within 20 min on a small server. In future work, we will extend our approach to also cover datatype properties. For example from the sentence "..., Google said Motorola Mobility contributed revenue of US$ 1.25 billion for the second quarter." the triple *dbpedia:Google*

rlno:says "Motorola Mobility contributed revenue of US$ 1.25 billion for the second quarter" can be extracted. Additionally we plan to integrate DeFacto [10], which is able to verify or falsify a triple extracted by RdfLiveNews. Finally, we will extend our approach with temporal logics to explicate the temporal scope of the triples included in our knowledge base.

References

1. S. Auer, J. Lehmann, and A.-C. N. Ngomo. Introduction to linked data and its lifecycle on the web. In *Reasoning Web*, pages 1–75, 2011.
2. I. Augenstein, S. Padó, and S. Rudolph. Lodifier: Generating linked data from unstructured text. In *ESWC*, 2012.
3. T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 284(5):34–43, May 2001.
4. S. Brohée and J. van Helden. Evaluation of clustering algorithms for protein-protein interaction networks. *BMC Bioinformatics*, 2006.
5. D. Davidov and A. Rappoport. Classification of semantic relationships between nominals using pattern clusters. ACL, 2008.
6. A. Fader, S. Soderland, and O. Etzioni. Identifying relations for open information extraction. In *EMNLP*, pages 1535–1545. ACL, 2011.
7. A. Gaag, A. Kohn, and U. Lindemann. Function-based solution retrieval and semantic search in mechanical engineering. In *IDEC '09*, pages 147–158, 2009.
8. D. Gerber and A.-C. Ngonga Ngomo. Bootstrapping the linked data web. In *1st Workshop on Web Scale Knowledge Extraction @ ISWC 2011*, 2011.
9. D. Goldhahn, T. Eckart, and U. Quasthoff. Building large monolingual dictionaries at the leipzig corpora collection: From 100 to 200 languages. In *LREC*, 2012.
10. J. Lehmann, D. Gerber, M. Morsey, and A.-C. Ngonga Ngomo. DeFacto - Deep Fact Validation. In *ISWC*, 2012.
11. D. Lin. An Information-Theoretic Definition of Similarity. In J. W. Shavlik and J. W. Shavlik, editors, *ICML*, pages 296–304. Morgan Kaufmann, 1998.
12. P. N. Mendes, M. Jakob, A. Garcia-Silva, and C. Bizer. DBpedia Spotlight: Shedding Light on the Web of Documents. In *I-SEMANTICS*, ACM International Conference Proceeding Series, pages 1–8. ACM, 2011.
13. M. Morsey, J. Lehmann, S. Auer, and A.-C. Ngonga Ngomo. Dbpedia sparql benchmark - performance assessment with real queries on real data. In *International Semantic Web Conference (1)*, pages 454–469, 2011.
14. N. Nakashole and G. Weikum. Real-time population of knowledge bases: opportunities and challenges. In *Proceedings of AKBC-WEKEX*, 2012.
15. A.-C. N. Ngomo, N. Heino, K. Lyko, R. Speck, and M. Kaltenböck. SCMS - Semantifying Content Management Systems. In *ISWC*, 2011.
16. A.-C. Ngonga Ngomo. On link discovery using a hybrid approach. *J. Data Semantics*, 1(4):203–217, 2012.
17. A.-C. Ngonga Ngomo and F. Schumacher. Borderflow: A local graph clustering algorithm for natural language processing. In *CICLing*, pages 547–558, 2009.
18. T. Pedersen, S. Patwardhan, and J. Michelizzi. Wordnet: : Similarity - measuring the relatedness of concepts. In *AAAI*, 2004.
19. G. Rizzo, R. Troncy, S. Hellmann, and M. Brümmer. NERD meets NIF: Lifting NLP extraction results to the linked data cloud. In *LDOW, 2012, France*.

20. M. Ruiz-Casado, E. Alfonseca, and P. Castells. Automatising the learning of lexical patterns: An application to the enrichment of wordnet by extracting semantic relationships from wikipedia. 2007.
21. S. Sarawagi. Information extraction. *Found. Trends databases*, 2008.
22. R. Stern and B. Sagot. Population of a knowledge base for news metadata from unstructured text and web data. In *Proceedings of the AKBC-WEKEX*, 2012.
23. Z. Wu and M. S. Palmer. Verb semantics and lexical selection. In J. Pustejovsky, editor, *ACL*, pages 133–138. Morgan Kaufmann Publishers / ACL, 1994.