Universität Leipzig
Fakultät für Mathematik und Informatik
Institut für Informatik

Leveraging the Crowdsourcing of Lexical Resources for Bootstrapping a Linguistic Data Cloud   Masterarbeit

im Studiengang Master Informatik

eingereicht von:	Jonas Brekle
eingereicht am:	1. August 2012
betreuender Professor:	Prof. Dr. Ing. habil. Klaus-Peter Fähnrich
Betreuer:	Dipl. Inf. Sebastian Hellmann

We present a declarative approach implemented in a comprehensive open-source framework (based on *DBpedia*) to extract lexical-semantic resources (an ontology about language use) from *Wiktionary*. The data currently includes language, part of speech, senses, definitions, synonyms, taxonomies (hyponyms, hyperonyms, synonyms, antonyms) and translations for each lexical word. Main focus is on flexibility to the loose schema and configurability towards differing language-editions of *Wiktionary*. This is achieved by a declarative mediator/wrapper approach. The goal is to allow the addition of languages just by configuration without the need of programming, thus enabling the swift and resource-conserving adaption of wrappers by domain experts. The extracted data is as fine granular as the source data in *Wiktionary* and additionally follows the *lemon* model. It enables use cases like disambiguation or machine translation. By offering a linked data service, we hope to extend DBpedia's central role in the LOD infrastructure to the world of Open Linguistics.

Ich danke Sebastian Hellmann für die gute Betreuung und ein professionelles Arbeitsumfeld sowie Theresa für ihre Unterstützung. Ohne sie wäre diese Arbeit so nicht möglich gewesen.

Contents

1. Introduction

1.1. Motivation

The topic of this thesis will be the flexible extraction of semantically rich data from *Wiktionary*. The resulting data set is a lexical resource for computer linguistics. But however the focus is not on linguistics but on data integration: information extraction from wikis can be conducted in two ways — in perspective of 1) text mining, where the wiki is seen as a corpus or 2) interpreting the wiki as a collection of semi-structured documents. The latter is the way we will see it and how DBpedia was designed as it enables the definition of *extractors*, that interpret wiki pages. Also opposed to text mining, DBpedia allows to tailor the data step by step closer to the intended semantic of the wiki text. Additionally weak signals (facts that don't have much support, that are only stated once) can be taken account of. DBpedia creates an ontology from Wikipedia, that is roughly said, a database of world knowledge. Opposed to Wikipedia, the DBpedia knowledge base can be queried like a database, comibining information from multiple articles. To conduct a analogous transformation on *Wiktionary*, we analysed the major differences and found that *Wiktionary* is on one hand richer in structured information-but also this structure varies widely. So we propose an declarative framework, built on top of DBpedia, to convert *Wiktionary* into a linguistic ontology about languages, about the use of words, about their properties and relations. We will show which unique properties such a knowledge base has and what possible applications are. The declarative extraction rules can be maintained by a community of domain experts, that don't necessarily need programming skills. As will be shown, this is crucial for the approach to succeed on a long term. DBpedia has proven such a approach to be working and scaling. The goal of DBpedia is to provide a tool for unsupervised but highly configurable ontology construction. By using and extending DBpedia *Wiktionary* can be automatically transformed into a machine readable dictionary — with substantial quantity and quality.

1.2. Problem

The exploitation of community-built lexical resources has been discussed repeatedly. *Wiktionary* is one of the biggest collaboratively created lexical-semantic and linguistic resources available, written in 171 languages (of which approximately 147 can be considered active[1]), containing information about hundreds of spoken and even ancient languages. For example, the English *Wiktionary* contains nearly 3 million words[2]. A *Wiktionary* page provides for a lexical word a hierarchical disambiguation to its language, part of speech, sometimes etymologies and most prominently senses. Within this tree numerous kinds of linguistic properties are given, including synonyms, hyponyms, hyperonyms, example sentences, links to Wikipedia and many more. [Meyer & Gurevych 2011] gave a comprehensive overview on why this dataset is so promising and how the ex-

[1] `http://s23.org/wikistats/wiktionaries_html.php`

[2] See `http://en.wiktionary.org/wiki/semantic` for a simple example page

tracted data can be automatically enriched and consolidated. Aside from building an upper-level ontology, one can use the data to improve NLP solutions, using it as comprehensive background knowledge. The noise should be lower when compared to other automatic generated text copora (e.g. by web crawling) as all information in *Wiktionary* is entered and curated by humans. Opposed to expert-built resources, the openness attracts a huge number of editors and thus enables a faster adaption to changes within the language.

The fast changing nature together with the fragmentation of the project into *Wiktionary* language editions (*WLE*) with independent layout rules (ELE) poses the biggest problem to the automated transformation into a structured knowledge base. We identified this as a serious problem: Although the value of *Wiktionary* is known and usage scenarios are obvious, only some rudimentary tools exist to extract data from it. Either they focus on a specific subset of the data or they only cover one or two WLE. The development of a flexible and powerful tool is challenging to be accommodated in a mature software architecture and has been neglected in the past. Existing tools can be seen as adapters to single WLE — they are hard to maintain and there are too many languages, that constantly change. Each change in the *Wiktionary* layout requires a programmer to refactor complex code. The last years showed, that only a fraction of the available data is extracted and there is no comprehensive RDF dataset available yet. The key question is: Can the lessons learned by the successful DBpedia project be applied to *Wiktionary*, although it is fundamentally different from Wikipedia? The critical difference is that only word forms are formatted in infobox-like structures (e.g. tables). Most information is formatted covering the complete page with custom headings and often lists. Even the infoboxes itself are not easily extractable by default DBpedia mechanisms, because in contrast to DBpedias *one entity per page* paradigm, *Wiktionary* pages contain information about *several* entities forming a complex graph, i.e. the pages describe the lexical word, which occurs in several languages with different senses per part of speech and most properties are defined *in context* of such child entities. Opposed to the currently employed classic and straight-forward approach (implementing software adapters for scraping), we propose a declarative mediator/wrapper pattern. The aim is to enable non-programmers (the community of adopters and domain experts) to tailor and maintain the WLE wrappers themselves. We created a simple XML dialect to encode the "entry layout explained" (ELE) guidelines and declare triple patterns, that define how the resulting RDF should be built. This configuration is interpreted and run against *Wiktionary* dumps. The resulting dataset is open in every aspect and hosted as linked data[3]. Furthermore the presented approach can be extended easily to interpret (or *triplify*) other MediaWiki installations or even general document collections, if they follow a global layout.

In section 2 I will introduce the domain of information extraction from wikis and RDF and related concepts, that form the basis of this thesis.
In section 3 and 4 I give an overview on requirements of the developed software, that arise in context of the DBpedia project and explain resulting specifications.

[3] http://wiktionary.dbpedia.org/

In the following section 5 I will present some implementation details, that turned out to be essential for the success of of the approach. Finally in section 6 the created dataset is evaluated and compared with existing datasets. The thesis is written in a top-down manner, so when ever questions seem to remain open, continue reading, as details will follow later.

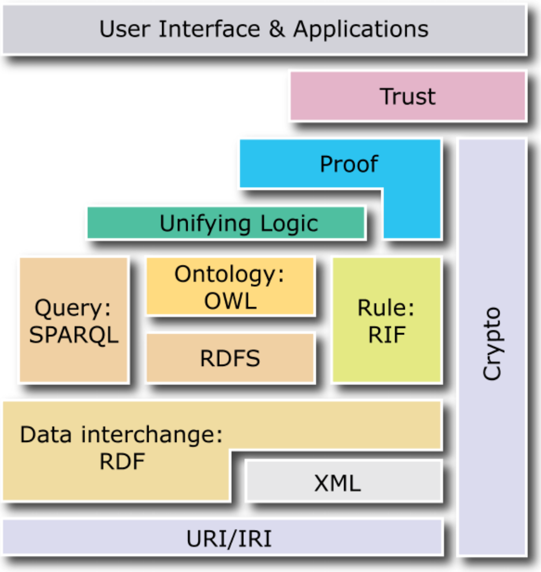


Figure 1: Semantic Web Technologien

2. Background

In the following a short overview on underlying technologies is given.

2.1. Semantic Web

The world wide web is one the most important inventions of our time. It enables the global access to documents and real time communication between individuals. This achieved by an interoperable infrastructure of independent networks, that can route any communication between two points. From our current perspective on the last three decades, this even seems technologically simple and maintainable at a reasonable cost. Furthermore the resulting benefits to our economy and society are beyond all expectations. A hole new industry was created and most industries are substantially influenced in their processes. The costs of communications dropped and public and private communications alike switched mostly to the new medium. The world wide web is enabled by a set of related technologies, which can be summarized to the following core concepts:

- TCP/IP addressing, transmission and routing
- client/server communication protocols like HTTP
- interlinked documents containing data (e.g. XML) or services (or interactive content) valuable for humans

While these technologies are well established and scale up to a huge amount of data, the users — humans — can barely cope with this amount of data offered. If one considers the WWW a information system, it only offers basic retrieval methods and most critical, it is fragmented into heterogeneous subsystems (which however can offer very good retrieval methods). But global interoperability is not supported on content level. Data is controlled by applications, and each application keeps it to itself[4]. From a perspective of data integration, semantics are often vague or undefined. It is often unclear which entity a document refers to. Documents are only interlinked by untyped relations. These are strong limitations: if the data is too much for a human to read and machines do not have deeper insight into it, the dataset as a whole can be seen as inaccessible. Of course the way the WWW works today seems to be well suited. Programmable Webservers e.g. with PHP made the web interactive, enabling social interaction or collaborative editing. But still future development is blocked by the human-centric nature of the web. If all the knowledge humanity acquired and wrote down in e.g. Wikipedia would be also available to information systems in a structured way, even more knowledge could be inferred automatically and e.g. artificial intelligence, expert systems or search would be boosted dramatically. The key to solving this issue lies (according to [Berners-Lee *et al.* 2001]) in the establishment of *Linked Data* (which will be explained in the next section). The use of machine readable data and annotation of text with such data is crucial for information technology to enter the next level. So to conclude: the problem is the amount of knowledge, and the lack of formalization e.g. machine readability. Even data that is already structured often lacks a defined semantic or the semantic is not formalized. The web as we know it is a web of documents, the target is the web of data. Instead of just documents linking to each other, instances should be globally interlinked. Consider this example: A company stores costumer information about you in its relational database; facebook keeps record of your activities in their distributed database and you yourself have a blog on your own webserver. Why shouldn't all this personal information be linked[5]? By a global identifier for *you*. Why do we have to supply contact information over and over again, although its database 101 that redundancy is bad. The answer is incompatibility on many levels. The ideal would be that all data is interoperable by design. Shortly after the invention of concepts for the WWW, Tim Berners-Lee et al. came up with the Idea of the Semantic Web: an WWW where intelligent agents can act on behalf of users, to find information or communicate. They have a shared syntax and vocabulary, use ontologies as background knowledge and thus get deeper to the intended semantic of things. The Semantic Web (SW) is a set of complementary technologies, which are depicted in figure 1. It is a layered model to represent, query and manage information.

The effort was initiated in 2001 by Tim Berners-Lee, who defined it as

> "'an extension of the current web in which information is given well-defined

[4] `http://www.w3.org/2001/sw/`

[5] The reader may object privacy issues; but the focus of this thesis is on data integration. These two topic have to be considered separately: just because data is interoperable, it is not accessible. Even more: if you avoid redundancy, you gain control over your data. How this control can be achieved is topic to current research but already very promising. Cf. WebID

> meaning, better enabling computers and people to work in cooperation"'.
> [Berners-Lee *et al.* 2001]

Dies geschieht durch verschiedene Ansätze: zunächst durch das Anreichern von (u. U. semi-strukturierten) Dokumenten durch Metainformationen schon beim Erstellungsprozess (wenn zum Beispiel ein Name in einem Dokument vorkommt, kann er in einer Art und Weise markiert werden, die allen verarbeitenden Systemen durch ein gemeinsames Vokabular als "'Personen-Markierung"' bekannt ist). Für diese Annotation wird zum Beispiel RDFa[6] genutzt. Des Weiteren wird Wissen in Form von Ontologien aufbereitet, die keine unstrukturierten Informationen enthalten, sondern lediglich *wahre Aussagen* in einem formellen Format. Die Nutzung eines gemeinsamen dokumentierten Vokabulars (also die Einigung auf eine Sprache) sowie die Entwicklung offener Standards soll die Verständigung garantieren. Die Einbeziehung bereits existierender Webstandards (wie URIs oder das HTTP-Protokoll) soll eine schnelle Verbreitung ermöglichen. Dadurch soll Wissen für maschinelle Verarbeitung nutzbar gemacht werden[7] (vgl. [Hitzler *et al.* 2008]). Die zentrale Technologie zur Datenrepräsentation ist RDF[8]. RDF/XML[9] bzw. N3[10] sind konkrete Kodierungen von RDF, auf welche hier nicht näher eingegangen wird. SPARQL dient als Anfragesprache für RDF-Graphen (siehe Abschnitt 2.4).

2.2. RDF

RDF steht für Resource Description Framework. Es ist ein Datenmodell zur Beschreibung von Web-Ressourcen (nicht nur Webseiten usw., denn jedes Ding der Welt kann über eine URI identifiziert werden), welche über eine URI[11] identifiziert werden. Eine Ressource ist dabei eine Entität, über die Aussagen der Form *"'Subjekt Prädikat Objekt"'* — ähnlich einem natürlichsprachlichen Satz — getroffen werden können.
So kodiert zum Beispiel das Tripel

```
<http://example.com/Alice> rdf:type foaf:Person
```

die Aussage “Alice ist eine Person”. Dabei ist Folgendes zu betrachten: Auch Subjekt, Prädikat und Objekt sind URIs, werden aber optional mit Prefixen abgekürzt (deren Definition hier ausgelassen wurde); außerdem wird das FOAF Vokabular[12] genutzt. Ein Tripel wird im Kontext der Graphentheorie interpretiert: Subjekt und Objekt sind Knoten — das Prädikat ist der “Typ” der Kante.

[6] http://www.w3.org/TR/xhtml-rdfa-primer/

[7] Allerdings wird dabei ausdrücklich nicht versucht, mithilfe künstlicher Intelligenz oder heuristischer Verfahren die Semantik für die Maschine verständlich zu machen, sondern lediglich sinnvoll repräsentierbar und damit verarbeitbar.

[8] http://www.w3.org/RDF/

[9] http://www.w3.org/TR/rdf-syntax-grammar/

[10] www.w3.org/DesignIssues/Notation3

[11] http://tools.ietf.org/html/rfc3986

[12] http://xmlns.com/foaf/0.1/

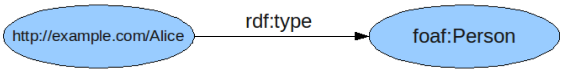


Figure 2: Das Tripel als Graph

Es entsteht (bei mehreren Tripeln) ein gerichteter Graph, der eine Ontologie repräsentiert. Graphen sind als abstraktes Konzept viel einfacher und verlässlicher zugänglich für Algorithmen als natürliche Sprache. Außerdem lassen sich implizite Informationen (zum Beispiel transitive Relationen oder geerbte Eigenschaften) ableiten oder Widersprüche finden[13].

Die Tatsache, dass hier kein hierarchisches Baummodell, sondern ein Graph zur Repräsentation gewählt wird, lässt sich auch damit begründen, dass Aussagen (bezüglich Beziehungen zweier Resourcen zueinander) oft nicht eindeutig einer von beiden zugeordnet werden können. Des Weiteren entspricht ein flaches Netz eher der dezentralen Struktur des Webs: Die Informationen sind verteilt und beim Zusammenführen zweier Informationsquellen stellt sich — zumindest auf diesem Niveau — kein Problem von Schema-Matching ein, denn das Schema ist bei allen Quellen gleichermaßen frei (vgl. [Hitzler *et al.* 2008]). Object-Matching soll von Beginn an vermieden werden, indem (wie oben erwähnt) ein gemeinsames Vokabular und somit überall eine eineindeutige URI für alle Objekte (Ressourcen) genutzt wird. In der Praxis besteht dieses Problem natürlich weiterhin.

Das Objekt eines Tripels kann in RDF auch ein Literal[14] sein. Literale sind direkte *Werte*, über die keine weiteren Aussagen getroffen werden — also zum Beispiel Zahlen oder Zeichenketten.

Man unterscheidet zwei Arten von Literalen: *plain literals* (mit optionalem *language Tag*) und *typed literals*, die einen Datentyp haben, der mit einer URI angegeben wird. Im folgenden Beispiel wird das Literal mit einem selbst definierten Datentyp typisiert:

```
ex:Alice foaf:birthday "22.09.1986"^^ex:germanDate
```

2.3. Linked Data

Linked Data is the simplest and yet most important mechanism to retrieve RDF data: as described, entities are identified URI's. When choosing the URL's one should pick a namespace under her authority, so that she can provide *some* data about that entity under that URL. As defined by W3C[15], the requirements for Linked Data are as follows:

1. Use URIs as names for things
2. Use HTTP URIs so that people can look up those names.
3. When someone looks up an URI, provide useful information, using RDF

[13] Access Control, Logic and Proof: `http://www.w3.org/2000/01/sw/#access`
[14] `http://www.w3.org/TR/rdf-concepts/#section-Graph-Literal`
[15] `http://www.w3.org/DesignIssues/LinkedData.html`

4. Include links to other URIs. so that they can discover more things.

> The first *Linked Data* principle advocates using URI references to identify, not just Web documents and digital content, but also real world objects and abstract concepts. These may include tangible things such as people, places and cars, or those that are more abstract, such as the relationship type of knowing somebody, the set of all green cars in the world, or the color green itself. This principle can be seen as extending the scope of the Web from online resources to encompass any object or concept in the world. The HTTP protocol is the Web's universal access mechanism. In the classic Web, HTTP URIs are used to combine globally unique identification with a simple, well-understood retrieval mechanism. Thus, the second Linked Data principle advocates the use of HTTP URIs to identify objects and abstract concepts, enabling these URIs to be dereferenced (i.e., looked up) over the HTTP protocol into a description of the identified object or concept [Heath & Bizer 2011].

The lookup of URI can even be enhanced with a mechanism called *Content Negotiation*: Based on the HTTP *accept* header (that is set by the application requesting), different types of formatting can be used in the response. If for example a human browses RDF data using a web browser, a HTML version of the RDF data can be generated easily. If an application requests RDF data it would set the accept header to `application/rdf+xml` and get XML, which easy to read by machines, nut not humans. The mechanism is also transparent, it happens server side at request time. Modern RDF stores like *Virtuoso* have built in support for *Linked Data* with *Content Negotiation*.

These four basic principles together make a fundamental difference regarding the architecture of Linked Data seen as a database. Instead of custom communication protocols, well established web standards are used. This makes it interoperable at a technical level and easy to adopt. And it is backward compatible to very simple solutions: If one wants to publish *Linked Data*, no RDF store is required at all, one could as well use a file server, with the documents available materialized. Also *Linked Data* implicitly is a *basic distributed database*: Even though the query language is limited to a simple GET (for now), one can easily access data on different physical servers at no cost. Also load balancing is easy by deploying any default web proxy. Of course there if no free lunch, many topics that have been long solved on traditional relational databases, have still to be solved for RDF stores. And this little collection of advantages is not meant to compare RDF stores to relational databases, but it should show how the use of open web standards can help solving hard problems *by design*. Other non-relational systems (key-value stores like Apache Cassandra or document stores like CouchDB) have shown that it is possible to make distributed databases scale and simultaneously simplify access mechanisms. RDF stores should analogously extend the expressiveness of the data model, but too be more open and designed to integrate easy.

Table 1: five star rating of Linked Data

★	Available on the web (whatever format) (optionally with an open licence, to be *Open Data*)
★★	Available as machine-readable structured data (e.g. excel instead of image scan of a table)
★★★	two stars plus use of a non-proprietary format (e.g. CSV instead of excel)
★★★★	All the above plus, Use open standards from W3C (RDF and SPARQL) to identify things, so that people can point at your stuff
★★★★★	All the above, plus: Link your data to other people's data to provide context

2.4. SPARQL

SPARQL[16] steht für *SPARQL Protocol and RDF Query Language* und ist eine Anfragesprache für RDF-Graphen sowie ein Protokoll für deren Nutzung über einen Webservice. Mit ihr können Literale, Ressourcen oder ganze Teilgraphen extrahiert werden. Sie hat sich aus anderen Sprachen, wie SquishQL[17], RDQL[18], RQL[19] oder SeRQL[20] (vgl. [Haase *et al.* 2004]), entwickelt und stellt nun den offiziellen Standard des W3C dar.
Eine kurze Übersicht[21] über die Features der Sprache werde ich im Folgenden geben.

Es gibt 4 Typen von SPARQL Querys:

- SELECT um Daten, die auf ein angegebenes Muster passen (*matchen*), zu extrahieren
- ASK um zu prüfen, *ob* ein Muster im Graph existiert
- CONSTRUCT gibt einen Graph zurück, der unter Umständen durch ein Such-Muster erzeugt wurde oder explizit angegeben wird
- DESCRIBE liefert Informationen über gematchte Ressourcen (ist nicht standardisiert, abhängig von der Konfiguration der Datenbank, oft beschreibende Eigenschaften, wenn diese eingetragen wurden)

Eine SPARQL-Anfrage (Query) lässt sich in folgende Teile aufgliedern:

- Prolog
  um Prefixe und Base-URI zu deklarieren. Relative URIs, die innerhalb des Querys

[16] http://www.w3.org/TR/rdf-sparql-query/
bzw. http://www.dajobe.org/2005/04-sparql/SPARQLreference-1.8-us.pdf

[17] siehe [Guha *et al.* 1998]

[18] http://www.w3.org/Submission/RDQL/

[19] http://139.91.183.30:9090/RDF/RQL/

[20] siehe [Broekstra & Kampman 2004]

[21] Hier werden lediglich für das weitere Verständnis zentrale Ideen erläutert — für eine umfassende und detaillierte Beschreibung ist der Standard zu konsultieren.

vorkommen, werden auf die Base-URI bezogen. Prefixe sind Abkürzungen für häufig verwendete URIs

- Projektionsanweisung
  ähnlich SQL: Variablen (bei SQL Spalten), welche im Ergebnis sichtbar sein sollen oder "'*"' für alle verwendeten Variablen

- Ergebnis-Modifikatoren
  - DISTINCT zur Duplikatentfernung
  - REDUCED "'kann"' Duplikate entfernen, wenn dies für die Laufzeit sinnvoll ist.
  - ORDER BY sorgt für Sortierung nach einem Ausdruck (oft einer Variablen)
  - LIMIT und OFFSET um einen gewisses Intervall von Lösungen auszuwählen

- GraphPattern (und Construct Pattern bei CONSTRUCT Querys)
  geben einen Teilgraphen an und bestehen aus Tripeln oder weiteren GraphPattern, wobei allerdings Variablen genutzt werden können — daher der Name Pattern. Nach diesem Muster wird im Graph gesucht. Mögliche Belegungen für Variablen sind das Ergebnis. Es gibt folgende GraphPattern-Typen:
  - GRAPH: ein elementares Pattern, bestehend aus einer beliebigen Folge von Tripeln, weiteren GraphPattern und Filtern. Ein Tripel ist eine Aussage aus dem angefragten RDF-Graph. Zwei aufeinanderfolgende Tripel werden durch einen Punkt getrennt und bilden eine Konjenktion dieser Aussagen. Die Komponenten Subjekt, Prädikat und Objekt können durch Variablen ersetzt werden.
  - OPTIONAL: ein GroupGraphPattern, das nicht notwendig ist. Das heißt: Wenn dieses Teilmuster nicht gematcht werden kann, beinflusst dies nicht das matching des gesamten Musters — allerdings sind enthaltene Variablen dann natürlich ungebunden.
  - UNION: verknüpft mehrere GraphPattern disjunktiv. So können Alternativen ausgedrückt werden.

- Filter sind logische Ausdrücke, die für jede mögliche Lösung[22] evaluiert werden und (wenn sie zu *false* ausgewertet werden) Ergebnisse löschen können. Für die Ausdrücke steht eine logische, numerische und relative Algebra zur Verfügung, die über eingebaute und entfernte Funktionsaufrufe erweitert wurde.

Ein Beispiel Query:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?subj ?name
WHERE {
   ?subj rdf:type foaf:Person .
```

[22] Eine Lösung ist eine Kombination von Variablen-Bindings, für die das Pattern *matcht*.

```
6      ?subj foaf:age ?age
7      OPTIONAL { ?subj foaf:name ?name }
8      FILTER( ?age > 23 )
9   }
```

Diese Anfrage findet alle Personen, die älter als 23 sind, und, wenn möglich, deren Namen. Genauer wird folgendes ausgedrückt: Zuerst wird ein Prefix deklariert, das das RDF und FOAF Vokabular abkürzt. Im WHERE-Teil wird ein GraphPattern angegeben, mit dem die Variable `?subj` mit allen Ressourcen belegt wird, für die die Aussage gilt, dass sie vom Typ `Person` aus dem FOAF-Vokabular sind. In der siebenten Zeile wird optional der zugehörige Name an die Variable `?name` gebunden. In der sechsten und achten Zeile wird abschließend noch eine Einschränkung auf das Alter deklariert, indem zunächst der Wert an eine Variable gebunden wird und diese dann mithilfe eines Filter-Ausdrucks eingeschränkt. Bei der Auswertung dieser Anfrage wird der sogenannte Triple-Store (also eine auf RDF-Daten spezialisierte Datenbank) versuchen, dieses Muster im angefragten Graph zu finden und alle möglichen Variablen-Belegungen als Ergebnis zurückliefern — also beispielsweise eine Tabelle oder XML mittels eines speziellen Result-Set-Formats[23].

2.5. Information Extraction

2.6. DBpedia

2.7. Related Work

In the last five years, the importance of *Wiktionary* as a lexical-semantic resource has been examined by multiple studies. Meyer et al. ([Meyer & Gurevych 2010b, Meyer & Gurevych 2010a]) presented an impressive overview on the importance and richness of *Wiktionary*. In [Zesch *et al.* 2008a] the authors presented the *JWKTL* framework to access *Wiktionary* dumps via a Java API. In [Meyer & Gurevych 2011] this JWKTL framework was used to construct an upper ontology called *OntoWiktionary*. The framework is reused within the *UBY project* [Gurevych *et al.* 2012], an effort to integrate multiple lexical resources (besides *Wiktionary* also *WordNet*, *GermaNet*, *OmegaWiki*, *FrameNet*, *VerbNet* and *Wikipedia*). The resulting dataset is modelled according to the *LMF ISO standard*[ISO 24613:2008]. [Moerth *et al.* 2011] and [T. Declerck & Váradi] discussed the use of *Wiktionary* to canonicalize annotations on cultural heritage texts (namely the Thompson Motif-index). Zesch et. al. also showed, that *Wiktionary* is suitable for calculating semantic relatedness and synonym detection; and it outperforms classic approaches [Zesch *et al.* 2008b, Weale *et al.* 2009]. Furthermore, other NLP tasks such as sentiment analysis have been conducted with the help of *Wiktionary* [Chesley *et al.* 2006].
Several questions arise, when evaluating the above approaches: Why are there not more NLP tools reusing the free *Wiktionary* data? Why are there no web mashups of

[23] http://www.w3.org/TR/rdf-sparql-XMLres/

name	active	available	RDF	#triples	ld	languages
JWKTL	✓	dumps	✗	-	✗	en, de
wikokit	✓	source + dumps	✓	n/a	✗	en, ru
texai	✗	dumps	✓	∼ 2.7 million	✗	en
lemon scraper	✓	dumps	✓	∼16k per lang	✗	6
blexisma	✗	source	✗	-	✗	en
WISIGOTH	✗	dumps	✗	-	✗	en, fr
lexvo.org	✓	dumps	✓	∼353k	✓	en

Table 2: Comparison of existing Wiktionary approaches (ld = linked data hosting). None of the above include any crowd-sourcing approaches for data extraction. The wikokit dump is not in RDF.

the data[24]? Why has *Wiktionary* not become the central linking hub of lexical-semantic resources, yet?

From our point of view, the answer lies in the fact, that although the above papers presented various desirable properties and many use cases, they did not solve the underlying knowledge extraction and data integration task sufficiently in terms of coverage, precision and flexibility. Each of the approaches presented in Table 2 relies on tools to extract machine-readable data in the first place. In our opinion these tools should be seen independent from their respective usage and it is not our intention to comment on the scientific projects built upon them in any way here. We will show the state of the art and which open questions they raise.

JWKTL is used as data backend of *OntoWiktionary* as well as UBY[25] and features a modular architecture, which allows the easy addition of new extractors (for example *wikokit* [Krizhanovsky 2010] is incorporated). The Java binaries and the data dumps in LMF are publicly available. Among other things, the dump also contains a mapping from concepts to lexicalizations as well as properties for part of speech, definitions, synonyms and subsumption relations. The available languages are English, German (both natively) and Russian (through *wikokit*). According to our judgement, *JWKTL* can be considered the most mature approach regarding software architecture and coverage and is the current state of the art. *Texai*[26] and *Blexisma*[27] are also Java based APIs, but are not maintained anymore and were most probably made obsolete by changes to the *Wiktionary* layout since 2009. There is no documentation available regarding scope or intended granularity. A very fine grained extraction was conducted using WISIGOTH [Sajous *et al.* 2010], but unfortunately there are no sources available and the project is unmaintained since 2010. Two newer approaches are the *lexvo.org* service and the algorithm presented in [McCrae *et al.* 2012]. The *lexvo.org* service offers a linked

[24] For example in an online dictionary from `http://en.wikipedia.org/wiki/List_of_online_dictionaries`

[25] `http://www.ukp.tu-darmstadt.de/data/lexical-resources/uby/`, `http://www.ukp.tu-darmstadt.de/data/lexical-resources/uby/`

[26] `http://sourceforge.net/projects/texai/`

[27] `http://blexisma.ligforge.imag.fr/index.html`

data representation of *Wiktionary* with a limited granularity, namely it does not disambiguate on sense level. The source code is not available and only the English *Wiktionary* is parsed. As part of the Monnet project[28], McCrae et al. [McCrae *et al.* 2012] presented a simple scraper to transform *Wiktionary* to the *lemon* RDF model [McCrae *et al.* 2011]. The algorithm (like many others) makes assumptions about the used page schema and omits details about solving common difficulties (as shown in the next section). At the point of writing, the sources are not available, but they are expected to be published in the future. Although this approach appears to be the state of the art regarding RDF modelling and linking, the described algorithm will *not scale to the community-driven heterogeneity* as to be defined in Section **??**. All in all, there exist various tools that implement extraction approaches at various levels of granularity or output format. In the next section, we will show several challenges that in our opinion are insufficiently tackled by the presented approaches. Note that this claim is not meant to diminish the contribution of the other approaches as they were mostly created for solving a single research challenge instead of aiming to establish *Wiktionary* as a stable point of reference in computational linguistics using linked data.

[28] See `http://www.monnet-project.eu/`. A list of the adopted languages and dump files can be found at `http://monnetproject.deri.ie/lemonsource/Special:PublicLexica`

3. Requirements

Als Anforderung wird eine Bedingung oder Eigenschaft bezeichnet, die ein System benötigt, um ein Problem zu lösen oder um einem Standard zu genügen (vgl. IEEE Std 610.12-1990). Drei Arten von Anforderungen sind funktionale Anforderungen, Qualitätsanforderungen und Rahmenbedingungen (vgl. [Young 2004]). In diesem Kapitel werde ich zunächst auf bestehende und neu entstandene Anwendungsfälle eingehen und die zugrundeliegenden Anforderungen aufzeigen. Danach möchte ich im Abschnitt **??** einige Qualitätsanforderungen erläutern.

3.1. Problem Description

In order to conceive a flexible, effective and efficient solution, we survey in this section the challenges associated with Wiki syntax, *Wiktionary* and large-scale extraction.

3.2. Processing Wiki Syntax

Pages in *Wiktionary* are formatted using the *wikitext* markup language[29]. Operating on the parsed HTML pages, rendered by the *MediaWiki engine*, does not provide any significant benefit, because the rendered HTML does not add any valuable information for extraction. Processing the database backup XML dumps[30] instead, is convenient as we could reuse the DBpedia extraction framework[31] in our implementation. The framework mainly provides input and output handling and also has built-in multi-threading by design. Actual features of the wikitext syntax are not notably relevant for the extraction approach, but we will give a brief introduction to the reader, to get familiar with the topic. A wiki page is formatted using the lightweight (easy to learn, quick to write) markup language *wikitext*. Upon request of a page, the MediaWiki engine renders this to an HTML page and sends it to the user's browser. An excerpt of the *Wiktionary* page *house* and the resulting rendered page are shown in Figure 3.

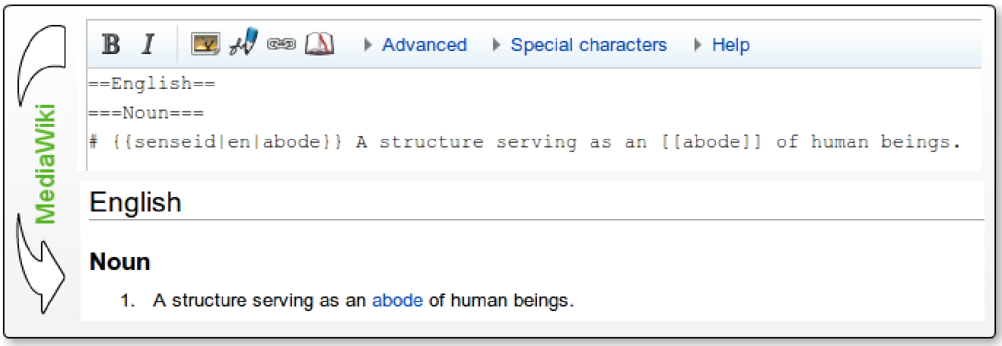
The markup == is used to denote headings, # denotes a numbered list (* for bullets), `[[link label]]` denotes links and `{{}}` calls a template. Templates are user-defined rendering functions that provide shortcuts aiming to simplify manual editing and ensuring consistency among similarly structured content elements. In MediaWiki, they are defined on special pages in the `Template:` namespace. Templates can contain any wikitext expansion, HTML rendering instructions and placeholders for arguments. In the example page in Figure 3, the `senseid` template[32] is used, which does nothing being visible on the rendered page, but adds an id attribute to the HTML `li`-tag (which is created by using `#`). If the English *Wiktionary* community decides to change the layout of senseid definitions at some point in the future , only a single change to the template definition is required. Templates are used heavily throughout *Wiktionary*, because they

[29] http://www.mediawiki.org/wiki/Markup_spec

[30] http://dumps.wikimedia.org/backup-index.html

[31] http://wiki.dbpedia.org/Documentation

[32] http://en.wiktionary.org/wiki/Template:senseid



```
==English==
===Noun===
# {{senseid|en|abode}} A structure serving as an [[abode]] of human beings.
```

English

Noun

1. A structure serving as an abode of human beings.

Figure 3: An excerpt of the *Wiktionary* page *house* with the rendered HTML.

substantially increase maintainability and consistency. But they also pose a problem to extraction: on the unparsed page only the template name and its arguments are available. Mostly this is sufficient, but if the template adds static information or conducts complex operations on the arguments (which is fortunately rare), the template result can only be obtained by a running MediaWiki installation hosting the pages. The resolution of template calls at extraction time slows the process down notably and adds additional uncertainty.

3.3. Wiktionary

Wiktionary has some unique and valuable properties:

- **Crowd-sourced**
  Wiktionary is community edited, instead of expert-built or automatically generated from text corpora. Depending on the activeness of its community, it is up-to-date to recent changes in the language, changing perspectives or new research. The editors are mostly semi-professionals (or guided by one) and enforce a strict editing policy. Vandalism is reverted quickly and bots support editors by fixing simple mistakes and adding automatically generated content. The community is smaller than Wikipedia's but still quite vital (between 50 and 80 very active editors with more than 100 edits per month for the English *Wiktionary* in 2012[33]).
- **Multilingual**
  The data is split into different Wiktionary Language Editions (WLE, one for each language). This enables the independent administration by communities and leaves the possibility to have different perspectives, focus and localization. Simultaneously one WLE describes multiple languages; only the representation language is restricted. For example, the German *Wiktionary* contains German description of German words **as well as** German descriptions for English, Spanish or Chinese words. Particularly the linking across languages shapes the unique value of *Wiktionary* as a rich multi-lingual linguistic resource. Especially the WLE

[33] http://stats.wikimedia.org/wiktionary/EN/TablesWikipediaEN.htm

for not widely spread languages are valuable, as corpora might be rare and experts are hard to find.

- **Feature rich**
  As stated before, *Wiktionary* contains for each lexical word (A lexical word is just a string of characters and has no disambiguated meaning yet) a disambiguation regarding language, part of speech, etymology and senses. Numerous additional linguistic properties exist normally for each part of speech. Such properties include word forms, taxonomies (hyponyms, hyperonyms, synonyms, antonyms) and translations. Well maintained pages (e.g. frequent words) often have more sophisticated properties such as derived terms, related terms and anagrams.
- **Open license**
  All the content is dual-licensed under both the *Creative Commons CC-BY-SA 3.0 Unported License*[34] as well as the *GNU Free Documentation License (GFDL)*.[35] All the data extracted by our approach falls under the same licences.
- **Big and growing**
  English contains 2,9M pages, French 2,1M, Chinese 1,2M, German 0,2 M. The overall size (12M pages) of *Wiktionary* is in the same order of magnitude as Wikipedia's size (20M pages)[36]. The number of edits per month in the English *Wiktionary* varies between 100k and 1M — with an average of 200k for 2012 so far. The number of pages grows — in the English *Wiktionary* with approx. 1k per day in 2012.[37]

The most important resource to understand how *Wiktionary* is organized are the *Entry Layout Explained* (ELE) help pages. As described above, a page is divided into sections that separate languages, part of speech etc. The table of content on the top of each page also gives an overview of the hierarchical structure. This hierarchy is already very valuable as it can be used to disambiguate a lexical word. The schema for this tree is restricted by the ELE guidelines[38]. The entities illustrated in Figure 4 of the ER diagram will be called *block* from now on. The schema can differ between WLEs and normally evolves over time.

3.4. Wiki-scale Data Extraction

The above listed properties that make *Wiktionary* so valuable, unfortunately pose a serious challenge to extraction and data integration efforts. Conducting an extraction for specific languages at a fixed point in time is indeed easy, but it eliminates some of the main features of the source. To fully synchronize a knowledge base with a community-driven source, one needs to make distinct design choices to fully capture all desired benefits. MediaWiki was designed to appeal to non-technical editors and abstains from intensive error checking as well as formally following a grammar — the community

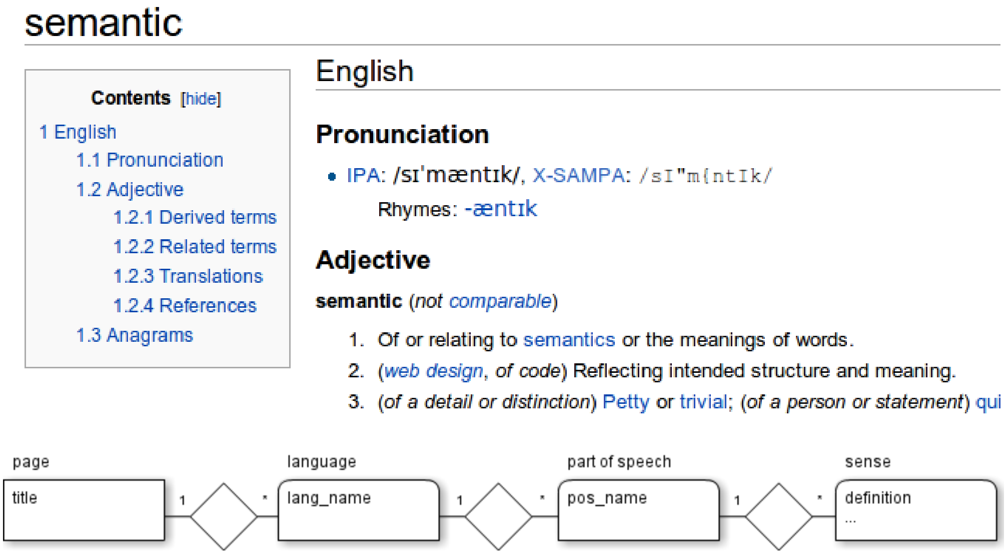
[34] `http://en.wiktionary.org/wiki/Wiktionary:Text_of_Creative_Commons_Attribution-ShareAlike_3.0_Unported_License`

[35] `http://en.wiktionary.org/wiki/Wiktionary:GNU_Free_Documentation_License`

[36] `http://meta.wikimedia.org/wiki/Template:Wikimedia_Growth`

[37] `http://stats.wikimedia.org/wiktionary/EN/TablesWikipediaEN.htm`

[38] For English see `http://en.wiktionary.org/wiki/Wiktionary:ELE`



semantic

Contents [hide]

1 English
1.1 Pronunciation
1.2 Adjective
1.2.1 Derived terms
1.2.2 Related terms
1.2.3 Translations
1.2.4 References
1.3 Anagrams

English

Pronunciation

- IPA: /sɪˈmæntɪk/, X-SAMPA: /sI"m{ntIk/
  Rhymes: -æntɪk

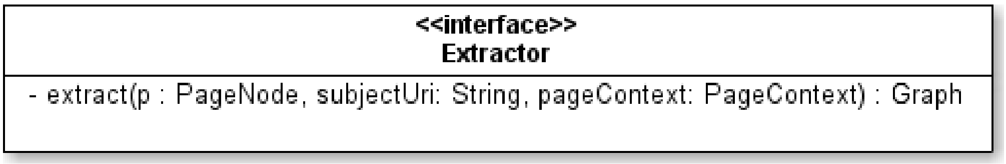
Adjective

semantic (*not comparable*)

1. Of or relating to semantics or the meanings of words.
2. (*web design, of code*) Reflecting intended structure and meaning.
3. (*of a detail or distinction*) Petty or trivial; (*of a person or statement*) qui

Figure 4: Example page `http://en.wiktionary.org/wiki/semantic` and underlying schema (only valid for the English *Wiktionary*, other WLE might look very different.)

gives itself just layout guidelines. One will encounter fuzzy modelling and unexpected information. Editors often see no problem with such "noise" as long as the page's visual rendering is acceptable. Overall, the main challenges can be summed up as (1) the constant and frequent changes to data *and schema*, (2) the heterogeneity in WLE schemas and (3) the human-centric nature of a wiki.



<<interface>>
Extractor

```
- extract(p : PageNode, subjectUri: String, pageContext: PageContext) : Graph
```

Figure 5: The extractor interface.

4. Specification

4.1. Overview

In the following I will describe the used architecture.

Existing extractors as presented in Section 2.7 mostly suffer from their *inflexible* nature resulting from their narrow use cases at development time. Very often approaches were only implemented to accomplish a short term goal (e.g. prove a scientific claim) and only the needed data was extracted in an *ad-hoc* manner. Such evolutionary development generally makes it difficult to generalize the implementation to heterogeneous schemas of different WLE. Most importantly, however, they ignore the community nature of a *Wiktionary*. Fast changes of the data require ongoing maintenance, ideally by the wiki editors from the community itself or at least in tight collaboration with them. These circumstances pose serious requirements to software design choices and should not be neglected. All existing tools are rather monolithic, hard-coded black boxes. Implementing a new WLE or making a major change in the WLE's ELE guidelines will require a programmer to refactor most of its application logic. Even small changes like new properties or naming conventions will require software engineers to align settings. The amount of maintenance work necessary for the extraction correlates with change frequency in the source. Following this argumentation, a community-built resource can only be efficiently extracted by a community-configured extractor. This argument is supported by the successful crowd-sourcing of DBpedia's internationalization [Kontokostas *et al.* 2012] and the non-existence of *open* alternatives with equal extensiveness.

Given these findings, we can now conclude four high-level design goals:

- declarative description of the page schema;
- declarative information/token extraction, using a terse syntax, maintainable by non-programmers;
- configurable mapping from language-specific tokens to a global vocabulary;
- fault tolerance (uninterpretable data is skipped).

The extractor is built on top of the the DBpedia framework, and thus it is required to conform to a simple interface:

Extractors are registered with the framework via a configuration file, and instantiated with a requested context. The context can be the DBpedia ontology or the selected language etc. Extractors are then subsequently invoked for every page of the MediaWiki XML dump. They are given the page in parsed form of an AST, subjectURI the URI of

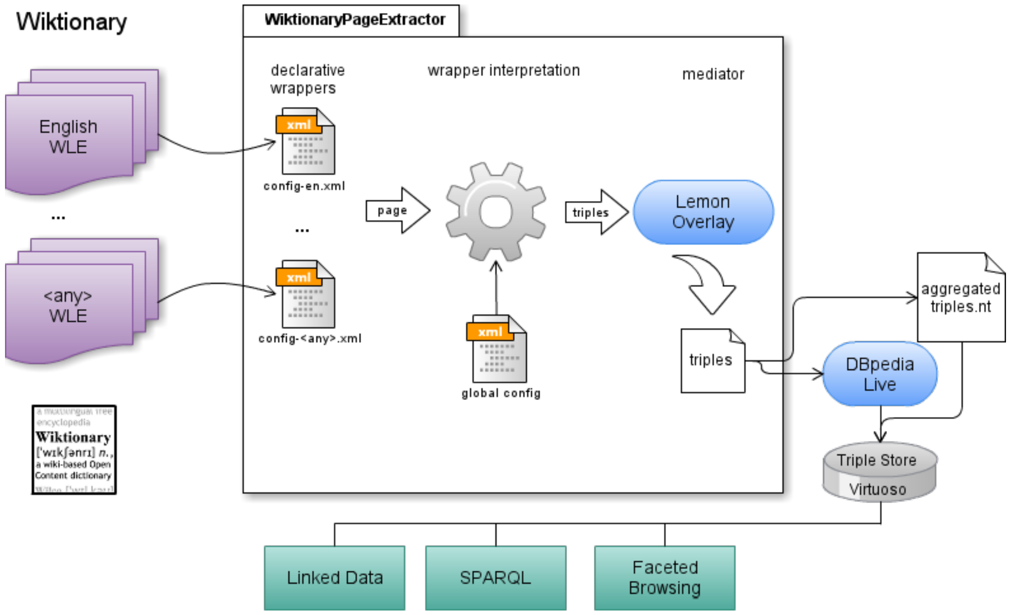


Figure 6: Architecture for extracting semantics from Wiktionary leveraging the DBpedia framework.

the resource this page should be referring to (`http://dbpedia.org/resource/$PAGENAME`) and pageContext — a helper for URI generation. The interface defines the extractor to return a Graph in turn, which is basically a set of triples (or quads in this case). Internally the extractor will inspect the AST and generate triples, when he finds information he interprets as relevant. This straightforward interface makes the DBpedia framework so modular. Input and output handling (parsing and serialization) is left to the framework, and the resulting RDF data can be directly inserting into an triple store.

We solve the above requirements with an additional extractor, which internally follows a rather sophisticated workflow, shown in Figure 6.

The *Wiktionary* extractor is invoked by the DBpedia framework to handle a page. It uses a language-specific configuration file, that has to be tailored to match the WLE's ELE guidelines to interpret the page, to extract the desired information. At first, the resulting triples still adhere to a language-specific schema, that directly reflects the configured layout of the WLE. A generic lossless transformation and annotation using the *lemon* vocabulary is then applied to enforce a global schema and reduce semantic heterogeneity. Afterwards the triples are returned to the DBpedia frameworks, which takes care of the serialization and (optionally) the synchronization with a triple store via DBpedia Live[39]. The process of interpreting the declarative wrapper is explained in more detailed in Figure 7.

[39] `http://live.dbpedia.org/live`

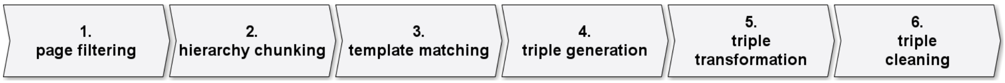


Figure 7: Overview of the extractor workflow.

5. Design and Implementation

In the following I will present a bunch of noteworthy implementation details.

5.1. Extraction Templates

As mentioned in Section 3.3, we define *block* as the part of the hierarchical page that is responsible for a certain entity in the extracted RDF graph. For each *block*, there can be declarations on how to process the page on that level. This is done by so called *extraction templates*(ET) (not to be confused with the templates of *wikitext* . Each possible section in the *Wiktionary* page layout (i.e. each linguistic property) has an ET configured (explained in detail below). The idea is to provide a declarative and intuitive way to encode *what to extract*. For example consider the following page snippet:

```
===Synonyms===
* [[building]]
* [[company]]
```

Since the goal is to emit a link to each resource per line, we can write the ET in the following style (using the popular scraping paradigms such as regular expressions):

```
===Synonyms===
(* [[\$target]]
)+
```

Some simple constructs for variables (“`$target`”) and loops (“`(*`”, “`)+`”) are defined for the ET syntax. If they are *matched against* an actual wiki page, *bindings* are extracted by a matching algorithm. We omit a low-level, technical description of the algorithm — one can think of it like a Regular Expression *Named Capturing Group*. The found *variable bindings* for the above example are `{(target->building), (target->company)}`. The triple generation rule encoded in XML looks like:

```
<triple s="http://some.ns/$entityId" p="http://some.ns/hasSynonym" o="http://some.ns/
    $target" />
```

Notice the reuse of the `$target` variable: The data extracted from the page is inserted into a triple. The variable `$entityId` is a reserved global variable, that holds the page name e.g. the word. The created triples in N-Triples syntax are:

```
<http://some.ns/house-1> <http://some.ns/hasSynonym> <http://some.ns/building> .
<http://some.ns/house-1> <http://some.ns/hasSynonym> <http://some.ns/company> .
```

The actual patterns are more complex (as explained below), but the mechanism is consistently used throughout the system.

5.2. Algorithm

The algorithm of processing a page works as follows:

Input: Parsed page obtained from the DBpedia Framework (essentially a lexer is used to split the Wiki Syntax into tokens)

1. Filter irrelevant pages (user/admin pages, statistics, list of things, files, templates, etc.) by applying string comparisons on the page title. Return an empty result on that condition.

2. Build a finite state automaton[40] from the page layout encoded in the WLE specific XML configuration. This schema also contains so called *indicator templates* for each *block*, that — if they match at the current page token — indicate that their respective block starts. So they trigger state transitions. In this respect the mechanism is similar to [McCrae *et al.* 2012], but in contrast our approach is declarative — the automaton is constructed *on-the-fly* and not hard-coded. The current state represents the current position in the disambiguation tree.

3. The page is processed token by token:
   a) Check if *indicator templates* match. If yes, the corresponding block is entered. The *indicator templates* also emit triples like in the *extraction template* step below. These triples represent the block in RDF – for example the resource `http://wiktionary.dbpedia.org/resource/semantic-English` represents the English block of the page "semantic".
   b) Check if any *extraction template* of the current block match.

      If yes, transform the variable bindings to triples.[41] Localization specific tokens are replaced as configured in the so called *language mapping* (explained in detail in section 5.3).

4. The triples are then *transformed*. In our implementation *transformation* means, that all triples are handed to a static function, which return a set of triples again. One could easily load the triples into a triple store like JENA and apply arbitrary SPARQL Construct and Update transformations. This step basically allows post-processing, e.g. consolidation, enrichment or annotation. In our case, we apply the schema transformation (by the mediator) explained in detail in Section 5.6).

5. The triples are sorted and de-duplicated to remove redundancy in the RDF dumps.

Output: Set of triples (handed back to the DBpedia Framework).

[40] Actually a finite state transducer, most similar to the Mealy-Model.

[41] In our implementation: Either declarative rules are given in the XML config or alternatively static methods are invoked on user-defined classes (implementing a special interface) for an imperative transformation. This can greatly simplify the writing of complex transformation.

5.3. Language Mapping

The language mappings are a very simple way to translate and normalize tokens, that appear in a WLE. In the German WLE, for example, a noun is described with the German word "*Substantiv*". Those tokens are translated to a shared vocabulary, before emitting them (as URIs for example). The configuration is also done within the language specific XML configuration:

```
<mapping from="Substantiv" to="Noun">
<mapping from="Deutsch" to="German">
...
```

The mapping consists currently of mappings for part of speech types and languages. But arbitrary usage is possible. Section 5.5 shows how this mapping is used.

5.4. Reference Matching

A *Wiktionary* specific requirement is the resolution of intra-page references: All Wiktionaries use *some* way to refer to parts of the traits of the word. For example on the page *house* at first senses are defined:

```
# A structure serving as an [[abode]] of human beings.
# {{politics}} A deliberative assembly forming a component of a legislature, or, more rarely, the room or building in which such an assembly normally meets.
# [[house music|House music]].
```

Later on relations to other words are noted — but *in context* of a sense. For example house *in context* of *abode* has the translation *Haus* in German. So the following notion is used:

```
====Translations====
{{trans-top|abode}}
...
* German: {{t+|de|Haus|n}}, {{t+|de|Häuser|p}}
...
```

The problem is to match the gloss that is given in the `trans-top` template argument against the available senses. There is no simple way to determine that. As described in [Meyer & Gurevych 2011] as *relation anchoring*, a string based measure is used[42]. There is a simple data structure that is initialized per page, to this matching mechanism. Which measure is used can be configured in the global configuration. Available measures are: Levenshtein and trigram set similarity with dice, jaccard or overlap coefficient. A sense can be registered with the matcher by passing its definition sentence. An *id* is generated for that sense. Later on, glosses (short forms of the definition) that refer to a sense can be passed to lookup which sense matches best, and the corresponding *id* is returned. Section 5.5 shows how this mechanism is used.

[42] Opposed to the approach described in [Meyer & Gurevych 2011], we only rely on information actually encoded on the page, the source of the link is determined by the position in the disambiguation tree and sense definitions. But a *target anchoring* — is not performed. *Target anchoring* refers to the disambiguation of the target word (or entity): this target of course has also a disambiguation tree, and it is possible to *bend* the link to point to a node deeper in that tree instead of just the root node. We consider this highly assumptious and it introduces noise. We leave that to postprocessing. Also it is not implementable easily within the DBpedia framework, because data extracted on other pages is not available to a extractor at runtime.

5.5. Formatting functions in Result Templates

The question arises, how this mapping is then used within the application. It is certainly not reasonable to replace *all* occurrences of those `from` tokens. This would lead to a number of false positive matches and screwed output. It is crucial to offer a possibility to configure in which context output should be handled so. I therefore introduced formatting functions in result templates: when you note the triples that are generated you can apply functions to e.g. variables. An example:

```
<triple s="http://some.ns/uri($entityId)" p="http://some.ns/hasLanguage" o="http://some
    .ns/map($target)" />
```

In this RT, two functions are used: `uri` and `map`. They are wrapped around variable parts, and on rendering they are resolved. The following functions are available:
The functions with *Id* in their name relate to the matching introduced in section 5.4.

`uri(str)`	URL encode
`map(str)`	replace if mapping found
`assertMapped(str)`	dont emit triple if not in mapping vocabulary
`assertNumeric(str)`	dont emit triple if argument is not numeric
`getId(str)`	lookup a gloss and return the id of the best matching sense
`getOrMakeId(str)`	lookup a id or generate one if below a similarity threshold
`makeId(str)`	save a sense and generate an id
`saveId(str, str)`	save a sense with a given id

Continuing the example of senses and translations, one would write the RT in a way to save definition sentences when generating them as triples and later matching the gloss, when generating triples about the translation section: For the defintions

```
<triple s="http://some.ns/$entityId-makeId($definition)" p="http://some.ns/
    hasDefinition" o="$definition" oType="literal" />
```

and for the translations

```
<triple s="http://some.ns/$entityId-getId($gloss)" p="http://some.ns/hasTranslation" o=
    "http://some.ns/uri($target)" />
```

The idea is that (if the matching is correct), the subject URIs are equal (e.g. `http://some.ns/house-1`) in both triples — there are two triples about one resource — information successfully merged.

```
<http://some.ns/house-1> <http://some.ns/hasDefinition> "A structure serving as an
    abode of human beings." .
<http://some.ns/house-1> <http://some.ns/hasTranslation> <http://some.ns/Haus> .
```

5.6. Schema Mediation by Annotation with *lemon*

The last step of the data integration process is the schema normalization. The global schema of all WLE is not constructed in a centralized fashion — instead we found a way to both making the data globally navigable and keeping the heterogeneous schema

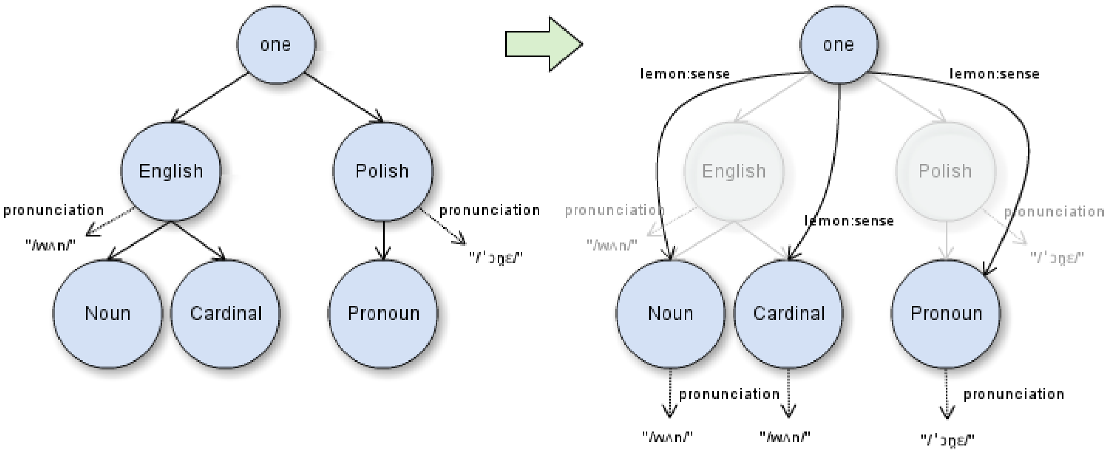


Figure 8: Schema normalization.

without loosing information. *lemon* [McCrae *et al.* 2011] is an RDF model for representing lexical information (with links to ontologies — possibly DBpedia). We use part of that model to encode the relation between *lexical entries* and *lexical senses*. *lemon* has great potential of becoming the *de facto* standard for representing dictionaries and lexica in RDF and is currently the topic of the OntoLex W3C Community group[43]. The rationale is to add *shortcuts* from *lexical entities* to *senses* and propagate properties that are along the intermediate nodes down to the senses. This can be accomplished with a generic algorithm (a generic tree transformation, regardless of the depth of the tree and used links). Applications assuming only a *lemon* model, can operate on the shortcuts and (if applied as an overlay — leaving the original tree intact) this still allows applications, to also operate on the actual tree layout. The (simplified) procedure is presented in Figure 8[44]. The use of the *lemon* vocabulary and model as an additional schema layer can be seen as our mediator. This approach is both lightweight and effective as it takes advantage of *multi-schema modelling*.

5.7. Configuration

As presented in section 3, the most important requirement of the approach is configurability. The extractor itself is as generic as possible, it is not tailored to linguistics or even *Wiktionary*. It has commitment to wiki syntax, but is also able to process plain text as it can be interpreted as wikitext without markup, thus the extractor may be suitable for most flat file formats. However the configuration makes up the heart of the extractor: it is a big XML file interpreted at runtime and describes how to interpret the page. I will go through the available options and show their relevance to the given requirements.

[43] `http://www.w3.org/community/ontolex/`

[44] Note, that in the illustration it could seem like the information about part-of-speech would be missing in the *lemon* model. This in not the case. Actually from the part-of-speech nodes, there is a link to corresponding language nodes. These links are also propagated down the tree.

At first the configuration is splitted into a generic part and a language specific part. The generic part is always loaded and does not need to be localized to a WLE. It contains options like the namespace, in which all URIs are created and an option that specifies which language configuration should be used. The language specific configuration is loaded based on that option at runtime. It has to be tailored to a WLE by the maintainer of the dataset.

Both configuration types are stored in the `config` folder. The naming convention restricts the folder to like the following:

```
config
├─config.xml
├─config-de.xml
├─config-en.xml
└─...
```

The generic configuration has two parts: a properties list and a mapping. The properties are the mentioned namespace, the language, the `loglevel` (to configure debug verbosity) and options to configure the matcher. The mapping is — as explained in section 5.3 — a way to replace tokens found on the page to a global vocabulary. But opposed to language specific tokens, in the generic configuration, globally used tokens are configured. It is used to provide a mapping from ISO 639-1 and -2 codes to the *Wiktionary* vocabulary.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<config>
   <properties>
      <property name="logLevel" value="0"/>
      <property name="language" value="ru"/>
      <property name="ns" value="http://wiktionary.dbpedia.org/"/>
      <property name="matchingStrategy" value="levenshtein"/>
      <property name="matchingThreshold" value="0.5"/>
   </properties>
   <mappings>
      <!-- ISO 639-1 -->
      <mapping from="aa" to="Afar" />
      <mapping from="ab" to="Abkhazian" />
      <mapping from="ae" to="Avestan" />
      ...
```

6. Evaluation

6.1. Resulting Data

The extraction has been conducted as a proof-of-concept on three major WLE: The English, French and German *Wiktionary*. The datasets combined contain more than 100 million facts. The data is available as N-Triples dumps[45], Linked Data[46], via the *Virtuoso Faceted Browser*[47] or a SPARQL endpoint[48].

lang	*#words*	*#triples*	*#resources*	*t/w*[a]	*#predicates*	*#senses*	*#wws*[b]	*s/wws*
en	2,903,933	71,230,704	33,428,598	24.52	26	966673	708644	1.36
fr	2,093,017	32,530,177	20,241,644	15.54	21	793,640	628,299	1.26
de	204,045	6,677,192	3,448,052	32.72	23	170762	116622	1.46

[a] *Triples per word.* A good measure of information density.
[b] *Words with senses.* The number of words, that have at least one sense extracted.

Table 3: Statistical comparison of three *Wiktionary* extraction result datasets.

The statistics show, that the extraction produces a vast amount of data with broad coverage, thus resulting in the largest lexical linked data resource. There might be partially data quality issues with regard to missing information (for example the number of *words with senses* seems to be relatively low intuitively), but detailed quality analysis was out-of-scope for this article as we focused on describing the procedure.

6.2. Lessons Learned

Making unstructured sources machine-readable creates feedback loops Although this is not yet proven by empirical data, the argument that extracting structured data from an open data source and making it freely available in turn encourages users of the extracted data to contribute to the source, seems reasonable. The clear incentive is to *get the data out again*. This increase in participation besides improving the source, also illustrates the advantages of machine readable data to common Wiktionarians. Such a positive effect from DBpedia supported the current *Wikidata*[49] project.

6.3. Suggested changes to Wiktionary

Although it's hard to persuade the community of far-reaching changes, we want to conclude how *Wiktionary* can increase its data quality and enable better extraction.

- **Homogenize Entry Layout across all WLE's.**
- **Use anchors to markup senses:** This implies creating URIs for senses. These can then be used to be more specific when referencing a *word* from another article.

[45] `http://downloads.dbpedia.org/wiktionary`
[46] for example `http://wiktionary.dbpedia.org/resource/dog`
[47] `http://wiktionary.dbpedia.org/fct`
[48] `http://wiktionary.dbpedia.org/sparql`
[49] `http://meta.wikimedia.org/wiki/Wikidata`

This would greatly benefit the evaluation of automatic anchoring approaches like in [Meyer & Gurevych 2011].

- **Word forms:** The notion of word forms (e.g. declensions or conjugations) is not consistent across articles. They are hard to extract and often not given.

7. Conclusion

7.1. Discussion and Future Work

Our main contributions are (1) an extremely flexible extraction from *Wiktionary*, with simple adaption to new Wiktionaries and changes via a declarative configuration. By doing so, we are (2) provisioning a linguistic knowledge base with unprecedented detail and coverage. The DBpedia project provides (3) a mature, reusable infrastructure including a public Linked Data service and SPARQL endpoint. All resources related to our *Wiktionary* extraction, such as source-code, extraction results, pointers to applications etc. are available from our project page[50]. As a result, we hope it will evolve into a central resource and interlinking hub on the currently emerging Web of Linguistic Data.

7.2. Next Steps

Wiktionary Live: Users constantly revise articles. Hence, data can quickly become outdated, and articles need to be re-extracted. DBpedia-Live enables such a continuous synchronization between DBpedia and Wikipedia. The WikiMedia foundation kindly provided us access to their update stream, the Wikipedia OAI-PMH[51] live feed. The approach is equally applicable to *Wiktionary*. The *Wiktionary* Live extraction will enable users for the first time ever to query *Wiktionary* like a database in real-time and receive up-to-date data in a machine-readable format. This will strengthen *Wiktionary* as a central resource and allow it to extend its coverage and quality even more.
Wiki based UI for the WLE configurations: To enable the crowd-sourcing of the extractor configuration, an intuitive web interface is desirable. Analogue to the mappings wiki[52] of DBpedia, a wiki could help to hide the technical details of the configuration even more. Therefore a JavaScript based WYSIWYG XML editor seems useful. There are various implementations, which can be easily adapted.
Linking: Finally, an alignment with existing linguistic resources like WordNet and general ontologies like YAGO or DBpedia is essential. That way *Wiktionary* will allow for the interoperability across a multilingual semantic web.

7.3. Open Research Questions

7.3.1. Publishing Lexica as Linked Data

The need to publish lexical resources as linked data has been recognized recently [Nuzzolese *et al.* 2011]. Although principles for publishing RDF as Linked Data are already well established [Auer & Lehmann 201 Heath & Bizer 2011], the choice of identifiers and first-class objects is crucial for any linking approach. A number of questions need to be clarified, such as which entities in

[50] `http://wiktionary.dbpedia.org`

[51] Open Archives Initiative Protocol for Metadata Harvesting, cf. `http://www.mediawiki.org/wiki/Extension:OAIRepository`

[52] `http://mappings.dbpedia.org/`

the lexicon can be linked to others. Obvious candidates are entries, senses, synsets, lexical forms, languages, ontology instances and classes, but different levels of granularity have to be considered and a standard linking relation such as `owl:sameAs` will not be sufficient. Linking across data sources is at the heart of linked data. An open question is how lexical resources with differing schemata can be linked and how are linguistic entities to be linked with ontological ones. There is most certainly an impedance mismatch to bridge.

The success of DBpedia as a "crystallization point for the Web of Data" is predicated on the stable identifiers provided by Wikipedia and are an obvious prerequisite for any data authority. Our approach has the potential to drive this process by providing best practices and live showcases and data in the same way DBpedia has provided it for the LOD cloud. Especially, our work has to be seen in the context of the recently published Linguistic Linked Data Cloud[Chiarcos *et al.* 2012] and the community effort around the Open Linguistics Working Group (OWLG)[53] and NIF [Hellmann *et al.* 2012]. Our Wiktionary conversion project provides valuable data dumps and linked data services to further fuel development in this area.

7.3.2. Algorithms and methods to bootstrap and maintain a Lexical Linked Data Web

State-of-the-art approaches for interlinking instances in RDF knowledge bases are mainly build upon similarity metrics [Ngonga Ngomo & Auer 2011, Volz *et al.* 2009] to find duplicates in the data, linkable via `owl:sameAs`. Such approaches are not directly applicable to lexical data. Existing linking properties either carry strong formal implications (e.g. `owl:sameAs`) or do not carry sufficient domain-specific information for modelling semantic relations between lexical knowledge bases.

[53] `http://linguistics.okfn.org`

A. Literature

References

[Auer & Lehmann 2010] Sören Auer and Jens Lehmann. *Making the Web a Data Washing Machine - Creating Knowledge out of Interlinked Data*. Semantic Web Journal, 2010.

[Auer *et al.* 2006] Sören Auer, Sebastian Dietzold and Thomas Riechert. *OntoWiki - A Tool for Social, Semantic Collaboration.* In Isabel F. Cruz, Stefan Decker, Dean Allemang, Chris Preist, Daniel Schwabe, Peter Mika, Michael Uschold and Lora Aroyo, editeurs, Proceedings of 5th International Semantic Web Conference 2006, volume 4273 of *Lecture Notes in Computer Science*, pages 736–749. Springer, 2006.

[Berners-Lee *et al.* 2001] Tim Berners-Lee, James Hendler and Ora Lassila. *The Semantic Web*. The Scientific American, 2001.

[Breitman *et al.* 2007] Karin Koogan Breitman, Marco Antonio Casanova and Walter Truszkowski. Semantic Web - Concepts, Technologies and Applications. Springer, 2007.

[Broekstra & Kampman 2004] Jeen Broekstra and Arjohn Kampman. *SeRQL. An RDF Query and Transformation Language*. ISWC, 2004.

[Chesley *et al.* 2006] P. Chesley, B. Vincent, L. Xu and R. K. Srihari. *Using Verbs and Adjectives to Automatically Classify Blog Sentiment.* In AAAI Spring Symposium: Computational Approaches to Analyzing Weblogs, 2006.

[Chiarcos *et al.* 2012] C. Chiarcos, S. Hellmann, S. Nordhoff, S. Moran, R. Littauer, J. Eckle-Kohler, I. Gurevych, S. Hartmann, M. Matuschek and C. M. Meyer. *The Open Linguistics Working Group*. In LREC, 2012.

[Guha *et al.* 1998] R.V. Guha, Ora Lassila, Eric Miller and Dan Brickley. *Enabling Inference*. W3C Query Language meeting, 1998.

[Gurevych *et al.* 2012] I. Gurevych, J. Eckle-Kohler, S. Hartmann, M. Matuschek, C. M. Meyer and C. Wirth. *Uby - A Large-Scale Unified Lexical-Semantic Resource Based on LMF*. In EACL 2012, 2012.

[Haase *et al.* 2004] P. Haase, J. Broekstra, A. Eberhart and R. Volz. *A comparison of RDF query languages*. ISWC, 2004.

[Heath & Bizer 2011] T. Heath and C. Bizer. Linked Data: Evolving the Web into a Global Data Space. Synthesis Lectures on the Semantic Web: Theory and Technology. Morgan and Claypool, 2011.

[Hellmann *et al.* 2012] S. Hellmann, J. Lehmann and S. Auer. *Towards an Ontology for Representing Strings for the NLP Interchange Format (NIF)*. In EKAW, 2012.

[Hitzler *et al.* 2008] Pascal Hitzler, Markus Krötzsch, Sebastian Rudolph and York Sure. Semantic Web. Springer, 2008.

[ISO 24613:2008] ISO 24613:2008. Language resource management — lexical markup framework. ISO, Geneva, Switzerland.

[Kontokostas *et al.* 2012] D. Kontokostas, C. Bratsas, S. Auer, S. Hellmann, I. Antoniou and G. Metakides. *Internationalization of Linked Data: The case of the Greek DBpedia edition*. Journal of Web Semantics, 2012.

[Krizhanovsky 2010] A. A. Krizhanovsky. *Transformation of Wiktionary entry structure into tables and relations in a relational database schema*. CoRR, 2010. http://arxiv.org/abs/1011.1368.

[McCrae *et al.* 2011] J. McCrae, D. Spohr and P. Cimiano. *Linking Lexical Resources and Ontologies on the Semantic Web with lemon*. In ESWC, 2011.

[McCrae *et al.* 2012] J. McCrae, P. Cimiano and E. Montiel-Ponsoda. *Integrating WordNet and Wiktionary with lemon*. In C. Chiarcos, S. Nordhoff and S. Hellmann, editeurs, Linked Data in Linguistics. Springer, 2012.

[Meyer & Gurevych 2010a] C. M. Meyer and I. Gurevych. *How Web Communities Analyze Human Language: Word Senses in Wiktionary*. In Second Web Science Conference, 2010.

[Meyer & Gurevych 2010b] C. M. Meyer and I. Gurevych. *Worth its Weight in Gold or Yet Another Resource – A Comparative Study of Wiktionary, OpenThesaurus and GermaNet*. In CICLing. 2010.

[Meyer & Gurevych 2011] C. M. Meyer and I. Gurevych. *OntoWiktionary – Constructing an Ontology from the Collaborative Online Dictionary Wiktionary*. In M.T. Pazienza and A. Stellato, editeurs, Semi-Automatic Ontology Development: Processes and Resources. IGI Global, 2011.

[Moerth *et al.* 2011] K. Moerth, T. Declerck, P. Lendvai and T. Váradi. *Accessing Multilingual Data on the Web for the Semantic Annotation of Cultural Heritage Texts*. In 2nd Workshop on the Multilingual Semantic Web, ISWC, 2011.

[Ngonga Ngomo & Auer 2011] Axel-Cyrille Ngonga Ngomo and Sören Auer. *LIMES - A Time-Efficient Approach for Large-Scale Link Discovery on the Web of Data*. In Proceedings of IJCAI, 2011.

[Nuzzolese *et al.* 2011] A. G. Nuzzolese, A. Gangemi and V. Presutti. *Gathering lexical linked data and knowledge patterns from FrameNet*. In K-CAP, 2011.

[Sajous *et al.* 2010] Franck Sajous, Emmanuel Navarro, Bruno Gaume, Laurent Prévot and Yannick Chudy. *Semi-automatic Endogenous Enrichment of Collaboratively Constructed Lexical Resources: Piggybacking onto Wiktionary*. In Advances in Natural

Language Processing, volume 6233 of *Lecture Notes in Computer Science*, pages 332–344. 2010.

[T. Declerck & Váradi] K. Mörth G. Budin T. Declerck P. Lendvai and T. Váradi. *Towards Linked Language Data for Digital Humanities*.

[Volz *et al.* 2009] J. Volz, C. Bizer, M. Gaedke and G. Kobilarov. *Discovering and Maintaining Links on the Web of Data.* In ISWC, 2009.

[Weale *et al.* 2009] T. Weale, C. Brew and E. Fosler-Lussier. *Using the Wiktionary graph structure for synonym detection*. In Proc. of the Workshop on The People's Web Meets NLP, ACL-IJCNLP, 2009.

[Young 2004] R. R. Young. The Requirements Engineering Handbook. Artech House, Boston, 2004.

[Zesch *et al.* 2008a] T. Zesch, C. Müller and Iryna Gurevych. *Extracting Lexical Semantic Knowledge from Wikipedia and Wiktionary*. In LREC, 2008.

[Zesch *et al.* 2008b] Torsten Zesch, C. Müller and I. Gurevych. *Using Wiktionary for Computing Semantic Relatedness*. In AAAI, 2008.

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig angefertigt habe. Die aus fremden Quellen direkt oder indirekt übernommenen Inhalte sind als solche kenntlich gemacht.
Diese Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.
Ich bin mir bewusst, dass nicht wahrheitsgemäße Angaben rechtlich verfolgt werden können.

____

Jonas Brekle