Analyzing Middle High German Syntax with RDF and SPARQL

Christian Chiarcos, Benjamin Kosmehl, Christian Fäth, Maria Sukhareva
Goethe-Universität Frankfurt
chiarcos|kosmehl|faeth|sukhareva@informatik.uni-frankfurt.de

Abstract

The paper presents technological foundations for an empirical study of Middle High German (MHG) syntax. We aim to analyze the diachronic changes of MHG syntax on the example of direct and indirect object alterations in the middle field. In the absence of syntactically annotated corpora, we provide a rule-based shallow parser and an enrichment pipeline with the purpose of quantitative evaluation of a qualitative hypothesis. We provide a publicaly available enrichment and annotation pipeline grounded. A technologically innovative aspect is the application of CoNLL-RDF and SPARQL Update for parsing.

Keywords: Middle High German, syntactic parsing, semantic enrichment, computational philology

1. Background

Middle High German (MHG, ISO 639-3 gmh) refers to a historical stage in the development of (High) German during the middle ages (1050-1350), a period with an extensive literature and a long-time subject of philological interest. Also in terms of linguistics, the MHG period is particularly interesting, as it allows to explore the development of German syntax (whereas most Old High German prose is translated from Latin, an extant body of original MHG literature survives).

The paper presents technological foundations for the empirical study of Middle High German (MHG) syntax, with a focus on the order of postverbal arguments. German is well-known for its relatively free word order (scrambling) in the middle field, but so far, the historical dimension of word order in German has not been studied on a broad-scale quantitative basis, but only on grounds of qualitative analysis of *sample* data. As an example, (Petrova, 2009) limited their analysis of Old High German syntax to sentences where word order deviates from the Latin original, and similarly, (Speyer, 2011) manually analyzed Middle High German sentences that comprised specific verb forms that had been identified beforehand.

At the time of writing, the lack of syntactically annotated corpora prohibits quantitative studies of MHG syntax. We thus complement an existing corpus that already contains morphosyntactic annotations with a deterministic rule-based parser. It should be noted that this parser has been designed for a specific research problem, and this is reflected by its disambiguation strategies.[1]We thus refer to our implementation as a 'chunker' rather than a 'parser': Its analyses are shallow in the sense that no sophisticated disambiguation strategies are applied, but default rules, only. The chunker is implemented in SPARQL Update (Buil Aranda et al., 2013). Using the CoNLL-RDF architecture (Chiarcos and Fäth, 2017), we read a CoNLL TSV file (or, data stream) with user-defined column labels, split it into sentences, and apply graph transformation rules to each sentence. The result can be rendered in CoNLL, again, or, alternatively, evaluated directly with SPARQL.

In addition, we want to explore selected determinants of word order variation in Middle High German. We thus provide a CoNLL-TSV-based enrichment pipeline that includes the chunker in addition to other annotators in order to assess the impact of, for example, semantic factors such as animacy on word order alternation. In particular, this allows to verify existing hypotheses established by traditional qualitative research in the philologies by means of quantitative methods. In the same vein, any insight or analysis obtained by automated annotations needs to be confirmed by qualitative methods: While quantitative analyses allow to process large amounts of data and thus have a greater potential to identify statistically significant patterns than qualitative studies, their methods are incapable of reaching a comparable level of accuracy.

German word order has been a widely discussed topic in linguistics and philology. Generally, German word order is described as verb-second with a relatively free order of verb arguments. However, this order is not arbitrary and despite of the fact that no unified theory of word order in German emerged, the scholars still agree on multi-factorial nature of order variation. With respect to variation of direct and indirect objects in the middle field[2], four main groups of factors have been identified: *type of referential expressions*, *syntactic*, *semantic* and *discourse* factors, which interact flexibly in the determination of word order preferences, thereby leading to the perception of relatively free word order. An important aspect in this regard is the study of diachronic developments of word order flexibility and its possible triggers. However, syntactically annotated corpora for early stages of German are currently not available,[3] and in particular, the great wealth of Middle High German literature and functional writing has only been tackled on the morphosyntactic level, most prominently in the Referenzkorpus Mittelhochdeutsch (Klein et al., 2016, ReM, https://www.linguistics.rub.de/rem/), on which we

[1] For example, we implement low prepositional phrase (PP) attachment because we study the word order of dative and accusative objects. Knowing that no PPs may occur between dative and accusative arguments simplifies subsequent queries.

[2] According to the topological model of German syntax, the finite verb and (optional) verbal particles constitute a 'bracket' around the syntactic core arguments of a main (and similarly, relative) clause, referred to as the 'middle field'.

[3] Notable exceptions such as the Early New High German corpus (https://enhgcorpus.wikispaces.com/) only include texts from the 16th c. onwards.

also rely.

2. Addressing MHG word order

This paper aims to facilitate the development of quantitative methods for the study of diachronic order variation in the Middle High German (MHG) middle field in order to complement established qualitative methods. Empirical qualitative methods have long been one of the core strategies of humanities research. Such methods involve an in-depth thorough study of small amount of data associated with a certain phenomenon and the scholar induces conclusions about the phenomenon based on the made observations. Qualitative methods are time-consuming and require manual work, thus, it is impossible to apply them to large amount of data. Therefore, it can be argued that the made observations lack statistical significance.

As opposed to qualitative methods, quantitative methods can be applied to analyze large amount of data. While quantitative methods lack scrupulousness that is inherent to qualitative methods, they can provide statistically significant evidence to support or reject the qualitative hypothesis. Applying qualitative methods to historical languages has only recently become possible after multiple digitized historical corpora were released. However, the study of scrambling in the MHG middle field is impossible using morphosyntactically annotated corpora alone, therefore we describe the enrichment of the existing ReM annotations with a rule-based chunker.

2.1. Topological fields and word order flexibility

The theory of topological fields models a German sentence by splitting it into three major coherent chunks: prefield, middle field and postfield. The field delimiters ('brackets'), are defined by the position of (different parts of) the predicate: With a compound predicate in a main clause, the finite verb (or auxiliary) represents the left bracket (LB), and the non-finite part (verbal particle, infinite verbs) represents the right bracket (RB) of the middle field. Example (1.a,b) illustrates middle field construction in modern German.

(1)

a. *Peter* *hat* ***das Buch*** ***dem Freund*** *gegeben*
Peter has.LB **the book.ACC** **the friend.DAT** gave.RB

b. *Peter* *hat* ***dem Freund*** ***das Buch*** *gegeben*
Peter has.LB **the friend.DAT** **the book.ACC** gave.RB

"Peter has given the book to the friend"

c. *si* *zeigete* *dem küninge* *den mandel*
she.NOM showed.LB the king.DAT the coat.ACC

"she showed the coat to the king"

Figure 1: Example Middle Field

In (1.a), the middle field brackets are formed by the auxiliary verb and the present participle which enclose the direct object (DO) and indirect object (IO) of the verb. Similarly, other types of compound predicates such as phrasal verbs, compound nominal and adjective predicates etc. can serve as middle field brackets.

The middle field in German would include all the non-clausal verbal arguments and adjuncts as well as all the non-clausal arguments and adjuncts of the elements in the middle field. The clausal arguments and adjuncts are placed in the postfield. While the German language has constraints on the placement of the topological field, the word order within the fields is relatively free. As shown in (1.a,b), direct object can precede or succeed the indirect object without affecting the meaning.

According to (Paul, 1918) the theory of a topological fields model can be applied to the analysis of Middle High German sentences as well, within certain limitations. An example taken directly form the ReM corpus is shown in (1.c) which will be used further on. It should be noted that in this example there is no right bracket as it is regarded as optional both in modern German as well as in Middle High German.

Traditionally, it is assumed that this level of word order flexibility (scrambling in the middle field) is a relatively archaic feature of German, and that only the relatively recent loss of morphological case contributed to its disappearance in English. By analogy, one would assume that word order flexbility in German either remained stable since its separation from Old English, or that it decreased since then (as case morphology was simplified since Old High German). However, (Speyer, 2011) reported an unexpected *increase* of word order flexibility for direct and indirect NP arguments since the middle ages. While the numbers he reported were not statistically significant, this is an observation that calls for verification. Speyer employed a data-driven, but qualitative methodology: Using a list of ditransitive verbs, he extracted attestations of these verbs from prose texts and manually analyzed the structure of the sentences that these verbs occurred in. In order to achieve statistically significant numbers, in order to identify factors that contribute to reordering preferences, but also in order to identify factors that may possibly have been confounded with Speyer's classification, we develop a rule-based shallow parser for Middle High German topological fields, building on a morphosyntactically annotated corpus.

2.2. Reference corpus Middle High German (ReM)

The ReM corpus used in the context of this paper was first published in December 2016 and consists of roughly 2.5 million tokens. It includes about 397 texts and text fragments of various genres (e.g. administrative texts, prose, poems, letters etc.) and covers several dialects of MHG. The corpus is annotated with MHG lemmata. For lemmata annotations the authors relied on the Lexer dictionary (Lexer, 1872 1878). The corpus also has morphological (gender, case etc.) and morphosyntactic annotations, i.e. Parts-Of-Speech (POS) tags. The morphosyntactic annotations are done in accordance with HiTS tagset (Dipper et

al., 2013) which was developed on the basis STTS tagset (Schiller et al., 1999) and adjusted towards the diachronic periods of German.
The ReM corpus does not provide annotations for topological fields nor syntactic arguments, but their respective building stones in parts-of-speech, morphosyntactic features and clause separators. We thus take these as a basis for our chunker, guided by grammars of Middle High German, native competence on (Modern) German and the expertise of language experts.

2.3. RDF-based parsing of topological fields

On a technical level, chunking is implemented as a rule-based graph transformation. Building on recent developments at the intersection of NLP and the Semantic Web community, we employ the recently proposed CoNLL-RDF framework (Chiarcos and Fäth, 2017).[4] CoNLL-RDF provides an isomorphic reconstruction of CoNLL data structures in RDF, and enables the application of SPARQL Update for the flexible querying and transformation of this data. The updated CoNLL-RDF data can then be accessed via a SPARQL end point or transformed back into a CoNLL format, fed into an RDF triple store or visualized in a compact, human-readable fashion. In a mode operating sentence-by-sentence, this also allows manipulations on data streams. Supported are all tab-separated-value (TSV) corpus formats following the one-word-per-line (OWPL) principle, with configurable labels and order of columns. This includes CoNLL-U, CoNLL-X, all other CoNLL TSV dialects, the CWB format, the Sketch Engine format, etc. Using Semantic Web formalisms allows us to develop a modular and slim architecture with highly reusable and portable components (SPARQL) embedded in a thriving technological ecosystem that provides a rich off-the-shelf technology stack that can subsequently be applied to the data.
CoNLL-RDF employs RDF (multi-)graphs, for which various serializations exist. In the following, we use Turtle (Beckett et al., 2014) which fundamentally builds on the notion of triples, i.e., the segmentation of a graph into pairs of nodes and their connecting edge: a subject URI ('node'), a property (relation, 'edge') URI, and an object URI (or value). Triples are separated by (`.`). Abbreviations include the prefix notation for URIs as well as the use of triple separators that allow to skip (keep) the subject (`;`) or subject and property (`,`). RDF data can be queried with SPARQL (Buil Aranda et al., 2013) which extends Turtle with query and update operators (`SELECT`, `INSERT`, `DELETE`), the introduction of variables (marked by `?`), filters, etc. As data representation and query language are closely tied to each other, it is relatively easy to develop parsing rules for existing data samples.
The chunking rules are applied deterministically and create structures in a bottom-up fashion, identifying NPs, PPs, verbal chunks, topological fields, clauses, and clausal juncture, respectively. As the resulting parse contains nonterminal nodes, we also provide a transformation of those nodes into newly established CoNLL "words". This functionality is useful for *visualizing parses* on grounds of an existing visualization for dependency syntax. For operating and querying parse trees, however, we recommend to operate diretly on the RDF, as this allows to address elements and paths directly with SPARQL without creating artificial 'word's and without compressing its information into a less interpretable string representation (such as the original rendering of PTB syntax since CoNLL 2005).

[4] `https://github.com/acoli-repo/conll-rdf`

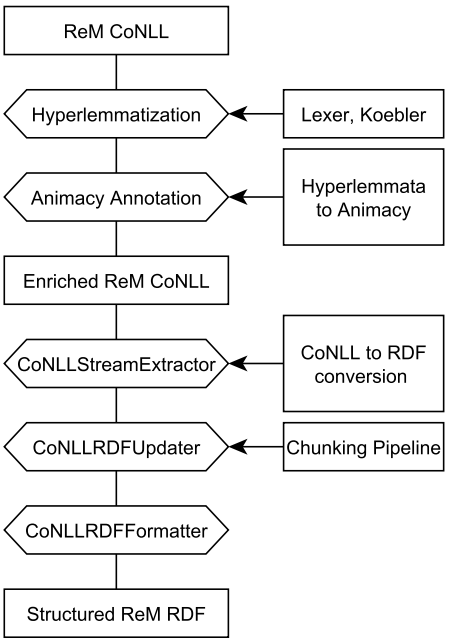


Figure 2: Enrichment pipeline

3. Annotation pipeline

This pipeline processes the files of the ReM corpus step-by-step from the raw data to a linguistically structured and annotated RDF format that can be easily queried. We ground our pipeline in the wide-used CoNLL format(s), i.e., as tab-separated values with one word per line, annotations separated by tabulators, and sentences separated by empty lines. We consider this a minimal and portable setting, as CoNLL is a relatively minimalistic and well-understood annotation format, but also, it can be easily customized for novel applications (by adding or dropping columns) and remains processable by both low-level shell commands and specialized tools. We provide an implementation of the pipeline as a simple Bash script which components are sketched out in Fig. 2. The code of the pipeline and its modules are available from `https://github.com/acoli-repo/germhist` under the Apache 2.0 license. An experimental UiMA implementation has been developed in addition but is currently not included in the release.

3.1. Corpus preprocessing

The ReM corpus is available in several formats, including CorA-XML (Bollmann et al., 2014).[5] We convert CorA-

[5] `https://cora.readthedocs.io/en/latest/coraxml/`

XML files into CoNLL with a focus on subsequent syntactic analysis: As such, we use the canonical ('modernized') version of a word as the value for the token (Klein and Dipper, 2016) instead of the original orthographical value. Additionally a number of other token-level features are pruned, as well. This includes, for example, morphological information such as the inflectional class. The resulting CoNLL files contain one token per line with the following columns: ID (token id), WORD (the canonical form of the written word), LEMMA, POS (parts-of-speech annotation), INFL (morphosyntactic features) and BOUNDARY (the clausal or sentence boundary marking). In accordance with the original annotation, sentences are separated by an empty line.

3.2. Enriching with hyperlemmas

Using CoNLL formats with user-defined column labels, it is possible to add novel components into the pipeline. At the moment, we support two types of enrichment modules, hyperlemmatization and animacy annotation.

By hyperlemmatization (or, more precisely, hyperlemmatization/translation), we mean to assign a historical word its modern counterpart. Ideally, this is a word that corresponds to the historical original both in meaning and etymology. However, depending on the strategy employed by the respective module, this may also be a translation, if a corresponding word cannot be found or if its meaning has changed.

We support two types of hyperlemmatization strategies: A wordlist-based approach and a transliteration-based approach. The wordlist-based approach is extensible to dictionaries, and thus capable to produce translations rather than hyperlemmas, it is, however, limited in coverage. The hyperlemmatization module thus uses its word list to train an internal transliterator, which is applied only if the lookup failed. Within the pipeline, the hyperlemmatizer is called several times for several word lists. We compiled the initial word list from Lexer (1872 1878), a 19th-century dictionary, by returning the head word together with the Levenshtein-closest word per gloss. However, this approach gave imprecise results: Frequently, glosses circumscribe a word rather than to provide its German equivalent. For multiword glosses, we thus added a similarity threshold for extraction. Moreover, we found that many glosses are in Latin rather than Modern German, and often not in line with modern orthography (for example, *wonung* for Modern *Wohnung*). Therefore, we consulted Köbler (2014) as a source of secondary evidence.

Hyperlemmatization modules can be run multiple times, with different word lists and in different configurations (e.g., for fuzzy search), each adding a column with hyperlemma candidates.

3.3. Enriching with animacy

Animacy is considered to be a major factor of word order, with the assumption that animate referents tend to precede inanimate referents (Jacobs, 1988) , and it can be relatively easily derived from lexical resources such as the Princeton WordNet (Fellbaum, 1998), resp., GermaNet (Hamp et al., 1997; Henrich and Hinrichs, 2010).

As WordNet is hierarchically organized, we retrieved the top-level synsets from WordNet 3.1 and (where appropriate) classified them for their animacy. We employ three primary classes: Human, Animate (non-human), Inanimate. Human includes persons, but also groups and organizations; Animate includes animals, plants and bacteria, but no plant or animal products; Inanimate includes substances and objects, but also abstractions.

For non-classified synsets, we increased the search depth and iterated the procedure. For verification, we ran animacy annotation against the entire WordNet and inspected 100 random samples for possible errors. After 5 iterations of the procedure, no more errors could be found. Animacy extraction from WordNet was implemented with a SPARQL SELECT statement against a local instance of WordNet 3.1. Animacy extraction for German was derived from WordNet using the existing GermaNet-WordNet linking.

In preparation for annotating textual data, we compiled a word list, where every German *lexeme* is described with its animacy classification. Note that we do not perform word sense disambiguation, so that a lexeme may have more than one animacy feature. As such, *Schwein* is animate ('pig'), inanimate ('pork') and human ('a person regarded as greedy and pig-like').

The actual animacy annotation reads hyperlemmatized ReM data from stdin, it takes a word list (TSV file) and a column number (the column of the hyperlemma) as parameters, and adds this information as another column. Again, animacy annotation can be run multiple times for different hyperlemma columns. If a TSV file for another feature is provided, this annotator can also be used for other kinds of lexical annotation.

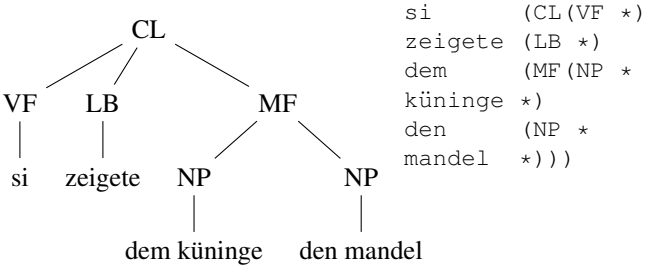


Figure 3: MHG sample parse (ReM, M403-G1, simplified) in tree view and conventional CoNLL

3.4. Annotation with CoNLL-RDF

CoNLL is an established exchange format in NLP, and enjoys high popularity as a representation formalism for dependency syntax, e.g., in the context of the Universal Dependencies (Marneffe et al., 2014) . For representing topological fields as part of a syntactic analysis, however, it is necessary to establish nonterminal nodes that span multiple words, and that are combined to form clauses and sentences. The conventional representation of CFG parses introduced with CoNLL 2005, however, requires to represent nonterminal nodes implicitly by pairs of matching brackets in different words: The word *si* in Fig. 3 thus carries the annotations of the prefield (VF) node as well as those of

its dominating clause (CL) node, whose right bracket only closes with the word *mandel*.

Processing phrasal annotations in CoNLL thus requires an internal mapping into a structured representation, e.g., a graph. For CoNLL data in general, such a mapping is provided by **CoNLL-RDF**, a data format and a library that provides the isomorphic and lossless reconstruction of CoNLL data as an RDF graph (Chiarcos and Fäth, 2017).[6]

This graph can then be extended with novel elements, but also, it is possible to perform rule-based graph transformation. Using the RDF query language SPARQL 1.1 (Buil Aranda et al., 2013), and in particular its update functions, CoNLL data can be enriched with additional structures and these can subsequently be transformed into CoNLL annotations.

Relevant features of CoNLL-RDF include:

- Assign every non-empty row a unique URI ('primary key') based on a user-provided base URI for the document, the sentence number and the word ID (or position): In the resource `file:M403-G1.conll`, the 26th word in the 59th first sentence will receive the URI `file:M403-G1.conll#s59.26`, resp., `:s59.26` in short.
- Define every row as a word, and connect it to its successor using the NIF vocabulary (Hellmann et al., 2013):[7] `:s59.26 a nif:Word; nif:nextWord :s59.27 .`
- Given a **user-provided list of column labels** (as an example: `LEMMA`), we create datatype properties in the `conll:` namespace, and assign the word its corresponding annotation as a literal value, e.g., `:s59.26 conll:LEMMA ''ër''.`

In consequence, we obtain an isomorphic representation of the original CoNLL data structure in RDF which is semantically shallow,[8] but can be effectively queried, manipulated and serialized back into CoNLL using off-the-shelf RDF technology. In particular, this includes a rich infrastructure of databases, webservices, APIs, models for resource publication and linking (Chiarcos et al., 2013). Even though it lacks formal semantics (by design), the CoNLL-RDF model can also serve as a basis to transform CoNLL data into semantically well-defined formalisms such as POWLA (Chiarcos, 2012) or NIF (Hellmann et al., 2013).

CoNLL-RDF comes with a Java API that allows to parse CoNLL data into CoNLL-RDF, to apply and to iterate SPARQL update transformations on this data, and to serialize `conll:` graphs in a lossless fashion as TSV (e.g., CoNLL-U or CoNLL-X), a human-readable dependency view or as a compact RDF/TTL representation that uses one word per line, ;-separated annotations and attribute-value pairs for different annotations. The latter serialization is also referred to as canonical CoNLL-RDF.

[6] `https://github.com/acoli-repo/conll-rdf`

[7] `http://persistence.uni-leipzig.org/nlp2rdf/ontologies/nif-core`

[8] The `conll:` namespace used here is not connected with any ontology, but populated by properties as defined by the user (column labels).

3.5. Querying with POWLA

POWLA (Chiarcos, 2012) is a small vocabulary that provides an OWL2/DL rendering of the Linguistic Annotation Framework (Ide and Suderman, 2014) . Here, we use the following vocabulary elements:

- `powla:Node` represents anything that can carry a linguistic annotation
- `powla:hasParent` points from a node to its parent node, thereby establishing a hierarchical structure
- children of the same `powla:Node` that are not interrupted by other siblings should be connected by a `powla:nextNode` property.

These data structures complement the original CoNLL-RDF rendering of (enriched) ReM CoNLL: The data structures that result from the chunking process (`conll:SHIFT` and `conll:REDUCE`, see below) are transformed with SPARQL Update into POWLA representations. Relations between siblings, ancestors, and descendants can thus be effectively queried and represent a basis for the quantitative evaluation with SPARQL: Using SPARQL SELECT, we can easily retrieve or count attestations of arbitrary graph patterns, including both CoNLL and POWLA data structures.

4. Shallow parsing

In this part we describe principles of the parsing process, we illustrate sample rules for a given sentence, and we provide a code example for the verbal chunking.

4.1. Parsing principles

Our chunker is designed to be a shallow parser in the sense that it does not attempt to disambiguate critical attachment decisions. Instead, deterministic default rules are applied that yield an analysis that is particularly convenient (but not limited to) the study of the order of arguments in the MHG middle field. One major limitation includes the treatment of PP attachment: As we are interested in the order of argument NPs, only, we perform low attachment. In this way, a PP positioned between a dative and an accusative NP will be attached to the preceding NP and both arguments will be adjacent siblings in the parse tree. Likewise, we regard genitive NPs (which can – rarely – have argument status in MHG) as nominal modifiers and treat them accordingly.

SPARQL by itself does support unrestricted graph transformation, albeit grounded in URI-defined properties and RDF resources. For implementing syntactic parsers, it is thus advisable to establish a designated vocabulary to represent data structures required during the parsing process. As a rule of thumb, however, 'data structures' refers to relations (object properties) between annotation elements, not to collections of partial parses. While this approach is qualitatively different from conventional parsing, we adopt the terminology of classical Shift-Reduce parsing (Nivre et al., 2007, 100-104): We introduce the properties `conll:SHIFT` to connect (the root nodes of) adjacent partial parses, and `conll:REDUCE` to represent attachment within a (partial) parse.[9]

[9] Originally, 'shift' and 'reduce' refer to parsing operations.

```
#ID WORD     LEMMA     POS    INFL
26  si       ër        PPER   Fem.Nom.Sg.3
27  zeigete  zèigen    VVFIN  *.Past.Sg.3
28  dem      dër       DDART  Masc.Dat.Sg
29  küninge  küni(n)g  NA     Dat.Sg
30  den      dër       DDART  Masc.Akk.Sg
31  mandel   mantel    NA     Akk.Sg
```

Figure 4: ReM CoNLL sample (ReM, M403-G1, slightly simplified)

Parsing is initialized by adding a `conll:SHIFT` relation for every `nif:nextWord` property in the graph, i.e., the 'queue' of unparsed words corresponds to the sequence of words. Parsing rules modify the existing graph, and if an attachment rule applies for a word/partial parse X, it is removed from the 'queue' of words (which is no longer distinguished from the 'stack' of partial parses) by dropping its `conll:SHIFT` relations. Instead, a `conll:REDUCE` relation with its head is established, and the sequence of `conll:SHIFT`s is restored by connecting the root of the partial parse with the root of the preceding partial parse. After initialization, `SHIFT`s over clause boundaries are removed (and restored later on from `nif:nextWord` for clausal juncture).

In the implementation here, parsing is deterministic and greedy. As RDF graphs are unordered, parsing is not conducted in a sequential fashion, but bottom-up and simultaneously for all matching graph patterns. All manipulations are expressed as SPARQL Update statements, which are applied and iterated in a predefined order until no more transformations occur, i.e., because a single root for the sentence has been established.

In the following chapters we describe selected parsing steps and rules using the example sentence (1.c) given above. Fig. 4 shows the resp. CoNLL representation. We represent parsing rules in SPARQL, but omit `SHIFT` updates. In the tree visualizations, vertical edges represent `REDUCE`, neighboring root nodes of partial parses (or words) are connected by `SHIFT`.

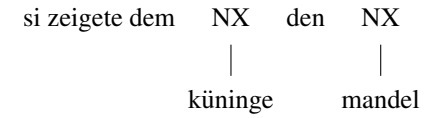
Furthermore, annotations are restructured: `conll:UPOS` provides the UD part-of-speech for the ReM part-of-speech, and separate properties `conll:CASE`, `conll:NUMBER` and `conll:FIN`(iteness) are extrapolated from `conll:INFL`.

4.2. Nominal, prepositional and verbal chunks

In a first step, noun chunks (NX) are formed for every token marked as nominal (noun or nominalization) if

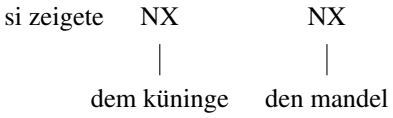
- it is the last word in a sentence,
- the next word is not a nominal, or
- the next word differs in case or number

The second case is illustrated in Fig. 5. In the example, the noun *küninge* projects an NX because the next word is a demonstrative with a different case marking, *mandel*




```
INSERT { _:nx conll:CAT    "NX";
              conll:CASE   ?case;
              conll:NUM    ?num.
          ?n  conll:REDUCE _:nx. }
WHERE {   ?n  conll:UPOS   "NOUN";
              conll:SHIFT/conll:UPOS ?p
              FILTER(?p!="NOUN").
          ?n  conll:CASE   ?case;
              conll:NUMBER ?num.}
```

Figure 5: Chunking: NX creation




```
INSERT { ?x conll:REDUCE ?nx. }
WHERE {  ?x conll:SHIFT  ?nx;
            conll:CASE   ?case;
            conll:NUMBER ?num.
         ?nx conll:CASE   ?case;
            conll:NUMBER ?num. }
```

Figure 6: Chunking: NX expansion

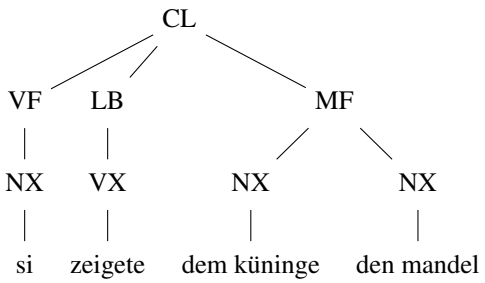


Figure 7: Topological fields and CL node

projects an NX because it precedes a punctuation sign. After the sequence of `SHIFT`s is restored, the next chunker rule does apply.

Subsequently, NX nodes are extended to the left by preceding words that match in case and number:[10] In Fig. 6, both determiners have the same case (`Dat`, `Akk`) and number (`Sg`) as the chunk they precede (resp. its head, cf. Fig. 4), and the NX is thus extended. This rule is iterated.

Furthermore, another rule creates NX elements for (unattached) pronouns.

Building on noun chunks, prepositional chunks are created when a preposition stands directly in front of an NX. The preposition and the noun chunk are joined into the new prepositional chunk (PX). We implement low attachment: The resulting PX nodes are attached to any immediately

Here, both terms refer to the data structures consulted/generated during these operations.

[10] Additional rules do exist for other (less frequent) constructions, e.g., attributive adjectives that follow their head noun.

preceding NX node.

Similar to nominal chunks, verb chunks (VX) are created from consecutive sequences of verbs and selected particles (e.g., ReM POS PTKVZ). If a verb carries the finiteness feature (conll:FIN ``true'', inferred from the ReM POS tag, e.g., from VV**FIN**), this is propagated to the VX. In Fig. 7, the verb *zeigete* is identified as a verbal chunk.

4.3. Topological fields and clausal junction

Middle field detection is relatively complicated and requires specialized rules for main and relative clauses as well as a special handling of discontinuous clauses. The most essential rules are the following: As a first step, left and right bracket (LB and RB) are detected. For main clauses, LB is a finite VX, and RB is a verbal particle or the next non-finite VX. We define the middle field (MF) as the sequence of chunks between a LB and an RB, resp., the end of the clause. The prefield (VF, for German *Vorfeld*) is a NX, PX or adverb immediately preceding the LB, the pre-prefield (VVF) is created for a conjunction preceding VF or LB, the postfield (NF, for German *Nachfeld*) is the span of chunks between RB and the end of the clause. Figure 7 illustrates the application of these rules to the example clause. For every LB, the preceding (optional) VVF and VF, and the following (optional) MF, RB and NF are then conjoined in a clause (CL) node, yielding a tree structure akin to Fig. 3. These rules (and a similar rule set for relative clauses) have been developed and tested on sample sentences from the ReM corpus.

It should be noted that these rules, as formulated so far, succeed only for continuous clauses. In order to handle clause fragments separated by a dependent relative clause, conll:SHIFT transitions over clause boundaries are restored from nif:nextWord, and a rule is applied that attaches the relative clause to the last NX. It should be noted that this rule exceeds the shift-reduce approach described so far in that we dive into the structure of the preceding clause: It does not attach to the root of the last parse fragment, but to its last NX descendant (possibly across a RB node, thereby producing a non-projective tree). In SPARQL, this is possible because the data structure can be traversed in the same way as SHIFT and REDUCE relations. All NX (?nx) chunks from the parse fragment that precedes a relative clause ?relCL can be retrieved by the first 3 lines in Fig. 8, and ?nx is the last NX chunk if (any nif:Word in) ?nx is not (MINUS) followed by any (nif:Word in another) NX chunk ?nx2 in ?last.

After attachment, clause fragments formerly separated by ?last can be conjoined and processed as above. Again, this rule does not perform disambiguation but it implements a low attachment strategy. The example illustrates how SPARQL can elegantly exceed beyond the local context, as well as some more advanced SPARQL expressions, like iterated (*, +) and concantenated (/) properties, as well as an example for FILTERs and set operators (MINUS).

Given this degree of expressivity, it is not surprising to find that SPARQL can be successfully employed to perform parsing using off-the-shelf Semantic Web technologies.

tokens (nif:Word)	2,514,585
sentences (conll:CAT "S")	147,398
middle fields (clauses) (conll:CAT "MF")	224,820
Acc before Dat	3,498
Dat before Acc	8,197

Table 1: Parsing statistics for the ReM corpus

5. Application and evaluation

After parsing, the result is transformed via SPARQL Update into POWLA data structures as illustrated in Fig. 10. Instances of dative and accusative arguments in the middle field can now easily be retrieved (and likewise, counted) by a query such as

```
SELECT ?acc ?dat
WHERE {
 ?acc conll:CASE "Akk".  # (an NX with) Akk
 ?acc powla:hasParent/conll:CAT "MF".
                       # in the middle-field
 ?acc powla:nextNode+ ?dat. # that precedes
 ?dat conll:CASE "Dat".  # (an NX with) Dat
}
```

Similarly, animacy features of nouns covered by the NX nodes can be counted, as well.

Without gold annotated data, we evaluate the parser only with respect to coverage as summarized in Tab. 1. A qualitative evaluation assessing the precision of middle field argument extraction is currently being conducted. In total, we obtained frequencies as shown in Fig. 9.

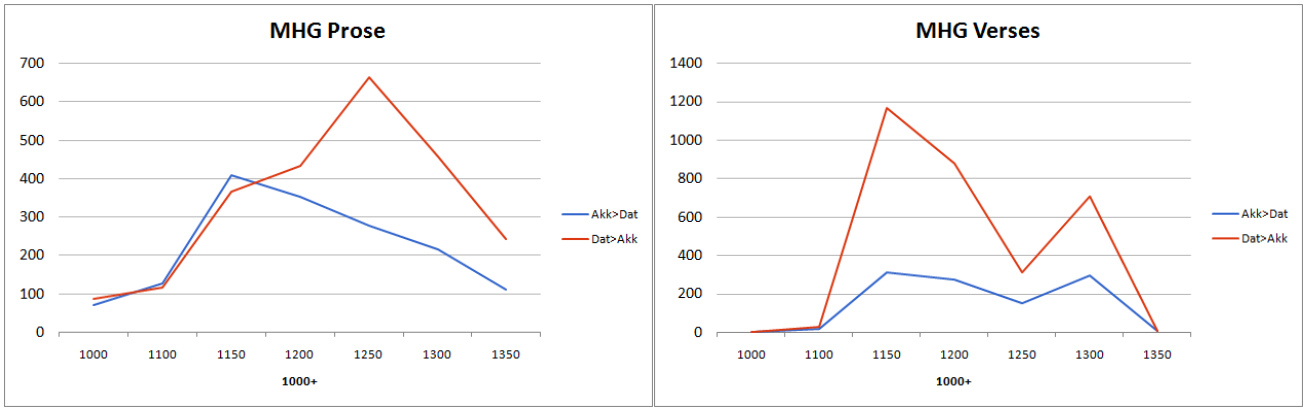
Above, we mentioned Speyer's qualitative experiments (Speyer, 2011) which indicated an increase of word order flexibility since the middle ages therefore contradicting the general scientific consensus. As a quantitative control experiment we thus developed a MHG shallow parser, and calculated the relative frequency of accusative and dative arguments in their relative order for all 50-year periods in the Middle High German era (1050 - 1350). A detailed linguistic analysis of these results, as well as a qualitative evaluation of the accuracy of argument and middle field detection is currently being conducted. In this paper, we focus on providing the technical pre-requisits for such a study, i.e., syntactic annotations, a convenient query language, as well as a workflow for enrichment with lexical (hyperlemmas) and semantic (animacy) features whose impact on diachronic word order variation is to be studied along with other shallow semantic annotations.

At a first glance, the results on prose text as shown in Fig. 9 do indeed seem to conform with the scientific consensus, i.e., that we see a decrease of word order flexibility during the middle ages: Until 1150, we find DO (direct object, accusative) > IO (indirect object, dative) about as often as IO > DO. After 1200, IO > DO is relatively more frequent, with a peak around 1250. The apparent decrease afterwards is due to the number of total attestations (i.e., texts from these periods). In verse, we always see a dominance of IO > DO, the reasons are not well understood, but we can expect interference with rhyme and meter.

This analysis is yet to be extended to Early Modern High German in order to verify (or refute) Speyer's thesis.

```
?last conll:SHIFT ?relCL.      # ?last directly precedes ?relCL
?nx conll:REDUCE* ?last.       # ?nx is a descendant of ?last
?nx conll:CAT "NX".            # and it is a NX
MINUS {                              # exclude all matches with
  ?nxWord conll:REDUCE+ ?nx.                 #  any word in ?nx
  ?nxWord nif:nextWord+/conll:REDUCE+ ?nx2. # followed by ?nx2
  ?nx2 conll:CAT "NX". FILTER(?nx2 != ?nx)  # i.e., another NX
  ?nx2 conll:REDUCE* ?last. }           # from the same clause
```

Figure 8: Advanced SPARQL: Identifying the last NX in the preceding partial parse



(a) Distribution of direct and indirect objects in MHG prose

(b) Distribution of direct and indirect objects in MHG verses

Figure 9: Diachronic quantitative analysis of the word order of direct and indirect objects in MHG

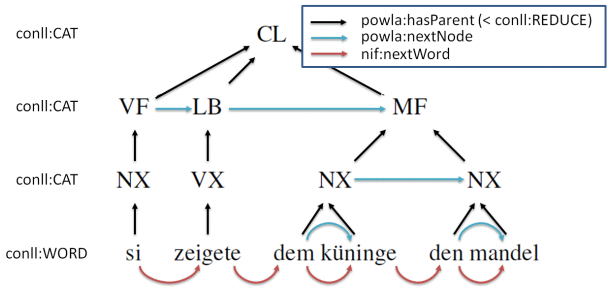


Figure 10: Resulting POWLA RDF graph

6. Summary and conclusion

We describe a pipeline for the syntactic annotation and the semantic enrichment of Middle High German. To our best knowledge, NLP for Middle High German consists of early prototypes towards morphosyntactic annotation (Hinrichs and Zastrow, 2012; Schulz and Kuhn, 2016). For more abstract levels, however, we are not aware of any attempts to conduct automated syntactic or semantic annotation on Middle High German.

Our approach builds on two core formalisms, the CoNLL format (resp., a specific dialect), and RDF. In general, pipeline modules communicate via CoNLL, resp. a TSV format, however, this seamlessly integrates with the SPARQL-based extraction of semantic features from WordNet 3.1 (i.e., a SPARQL SELECT query which produces TSV data) and with the SPARQL-based syntactic annotation (building on CoNLL-RDF). The resulting POWLA RDF data structure can be conveniently queried using SPARQL SELECT.

With respect to syntactic parsing we provide – to our best knowledge – the first publicly available implementation of a parser which solely relies on off-the-shelf Semantic Web technology. Related research includes the application of RDF and OWL for corpus representation and querying (Burchardt et al., 2008; Chiarcos, 2012) as well as a backend formalism for manual dependency annotation (Mazziotta, 2010). The only experiment on automated natural language parsing we are aware of (Wilcock, 2007), differs greatly by design from our implementation. Unfortunately, this implementation never left an experimental stage (p.c. G. Wilcock, Sep 2015). This experiment heavily relied on OWL/DL reasoning, resp., the use of rule languages building on top of OWL (Wilcock, 2006), and was thus relatively resource-intense. In comparison, our approach is designed to perform shallow, fast and transparent graph transformations using a formalism (CoNLL-RDF) that allows its integration in existing NLP pipelines. Its modular structure allows simple and comfortable integration of additional rules implemented as SPARQL updates.

In summary, we report the development of a shell-based enrichment pipeline for Middle High German including a CoNLL-RDF-based chunker for the analysis of Middle High German syntax and its semantic determinants. Both efforts improve the state of the art in natural language processing on Middle High German, and in terms of the technology applied, also for the processing of historical and low resource languages in general.

Acknowledgments

The research described in this paper was conducted at the Goethe Universität Frankfurt, Germany, within a project on Quantitative and Qualitative Methods in German Historical Philology (QuantQual@CEDIFOR), at the Centre for the Digital Foundation of Research in the Humanities, Social, and Educational Sciences (CEDIFOR) [11], funded by the German Ministry of Research and Education (BMBF, first phase 2014-2017). We would like to thank Ralf Plate, Arbeitsstelle Mittelhochdeutsches Wörterbuch, Trier / Institut für Empirische Sprachwissenschaft, Goethe-Universität Frankfurt, for the fruitful collaboration within this project. Furthermore, we would like to thank Thomas Klein and Claudia Wich-Reif for providing us with an access to the ReM corpus even before its ultimate publication, as well as Thomas Burch and the Trier Center for Digital Humanities [12], for the access to the digital Lexer[13] dictionary data. We would like to thank Margarete Springeth for access to the Middle High German Conceptual Database (MHDBDB)[14] at the Universität Salzburg. While not reported here, our hyperlemmatization routine was also applied to produce an annotation with MHDBDB concepts. Finally, we thank the anonymous reviewers for helpful comments and feedback.

7. References

Beckett, D., Berners-Lee, T., Prud'hommeaux, E., and Carothers, G. (2014). RDF 1.1 turtle. Technical report, W3C Recommendation.

Bollmann, M., Petran, F., Dipper, S., and Krasselt, J. (2014). Cora: A web-based annotation tool for historical and other non-standard language data. In *Proceedings of the 8th Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities (LaTeCH)*, pages 86–90, Gothenburg, Sweden.

Buil Aranda, C., Corby, O., Das, S., Feigenbaum, L., Gearon, P., Glimm, B., Harris, S., Hawke, S., Herman, I., Humfrey, N., Michaelis, N., Ogbuji, C., Perry, M., Passant, A., Polleres, A., Prud'hommeaux, E., Seaborne, A., and Williams, G. (2013). SPARQL 1.1 overview. Technical report, W3C Recommendation.

Burchardt, A., Padó, S., Spohr, D., Frank, A., and Heid, U. (2008). Formalising multi-layer corpora in OWL/DL - lexicon modelling, querying and consistency control. In *Proceedings of the Third International Joint Conference on Natural Language Processing (IJCNLP-2008)*, pages 389–396, Hyderabad, India, January.

Chiarcos, C. and Fäth, C. (2017). Conll-rdf: Linked corpora done in an nlp-friendly way. In Jorge Gracia, et al., editors, *Language, Data, and Knowledge: First International Conference, LDK 2017, Galway, Ireland, June 19-20, 2017, Proceedings*, pages 74–88. Springer International Publishing, Cham.

Chiarcos, C., McCrae, J., Cimiano, P., and Fellbaum, C. (2013). Towards open data for linguistics: Linguistic linked data. In Alessandro Oltramari, et al., editors, *New Trends of Research in Ontologies and Lexical Resources: Ideas, Projects, Systems*, pages 7–25. Springer Berlin Heidelberg, Berlin, Heidelberg.

Chiarcos, C. (2012). A generic formalism to represent linguistic corpora in rdf and owl/dl. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey, may. European Language Resources Association (ELRA).

Dipper, S., Donhauser, K., Klein, T., Linde, S., Müller, S., and Wegera, K.-P. (2013). Hits: ein tagset für historische sprachstufen des deutschen. *Journal for Language Technology and Computational Linguistics*, 28(1):85–137. Special Issue.

Fellbaum, C. (1998). *WordNet: An Electronic Lexical Database*. Bradford Books.

Hamp, B., Feldweg, H., et al. (1997). Germanet-a lexical-semantic net for german. In *Proceedings of ACL workshop Automatic Information Extraction and Building of Lexical Semantic Resources for NLP Applications*, pages 9–15.

Hellmann, S., Lehmann, J., Auer, S., and Brümmer, M. (2013). Integrating NLP using linked data. In *Proc. of the 12th International Semantic Web Conference (ISWC-2013)*, pages 98–113, Sydney, Australia, Oct. Springer.

Henrich, V. and Hinrichs, E. (2010). GernEdiT - The GermaNet Editing Tool. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC-2010)*, pages 2228–2235, Valletta, Malta, May.

Hinrichs, E. and Zastrow, T. (2012). Automatic annotation and manual evaluation of the diachronic german corpus tüba-d/dc. In *Proceedings of the Eigth International Conference on Language Resources and Evaluation (LREC-2012)*, pages 1622–1627, Istanbul, Turkey, May.

Ide, N. and Suderman, K. (2014). The linguistic annotation framework: a standard for annotation interchange and merging. *Language Resources and Evaluation*, 48(3):395–418, Sep.

Jacobs, J. (1988). Probleme der freien wortstellung im deutschen. *Arbeitsbericht Nr. 5 des Forschungsprogramms Sprache und Pragmatik*, pages 8–37.

Klein, T. and Dipper, S. (2016). *Handbuch zum Referenzkorpus Mittelhochdeutsch*. Ruhr-Universität Bochum. Bochumer Linguistische Arbeitsberichte; 19.

Klein, T., Wegera, K.-P., Dipper, S., and Wich-Reif, C. (2016). Referenzkorpus Mittelhochdeutsch (1050–1350), Version 1.0. `https://www.linguistics.ruhr-uni-bochum.de/rem/`. ISLRN 332-536-136-099-5.

Köbler, G. (2014). Mittelhochdeutsches Wörterbuch. `http://www.koeblergerhard.de/mhd/2A/mhd.html`, last visited on 07/02/2018. 3rd ed.

Lexer, M. (1872-1878). *Mittelhochdeutsches Handwörterbuch*. S. Hirzel, Leipzig.

Marneffe, M.-C. D., Dozat, T., Silveira, N., Haverinen, K., Ginter, F., Nivre, J., and Manning, C. D. (2014). Universal Stanford dependencies: a cross-linguistic typology. In *Proceedings of the Ninth International Conference*

[11] `https://www.cedifor.de/en/`
[12] `http://kompetenzzentrum.uni-trier.de/de/`
[13] `http://woerterbuchnetz.de/Lexer/`
[14] `http://mhdbdb.sbg.ac.at/`

on Language Resources and Evaluation (LREC-2014), Reykjavik, Iceland, May. European Language Resources Association (ELRA).

Mazziotta, N. (2010). Building the syntactic reference corpus of medieval french using notabene rdf annotation tool. In *Proceedings of the Fourth Linguistic Annotation Workshop (LAW-2010)*, pages 142–146. Association for Computational Linguistics.

Nivre, J., Hall, J., Nilsson, J., Chanev, A., Eryiğit, G., Kübler, S., Marinov, S., and Marsi, E. (2007). Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13:95–135, 06.

Paul, H. (1918). *Mittelhochdeutsche Grammatik*, volume 2 of *Sammlung kurzer Grammatiken germanischer Dialekte. A, Hauptreihe*. Niemeyer Tübingen. reprint 2007, 25. edition.

Petrova, S. (2009). Information structure and word order variation in the Old High German Tatian. *Information Structure and Language Change : New Approaches to Word Order Variation in Germanic*, pages 251–280.

Schiller, A., Teufel, S., Stöckert, C., and Thielen, C. (1999). Guidelines für das Tagging deutscher Textcorpora mit STTS (kleines und großes Tagset). Technical report, Universität Stuttgart, Universität Tübingen, Stuttgart, Germany.

Schulz, S. and Kuhn, J. (2016). Learning from within? comparing pos tagging approaches for historical text. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Slovenia, May. European Language Resources Association (ELRA).

Speyer, A. (2011). Die Freiheit der Mittelfeldabfolge im Deutschen. Ein modernes Phänomen. *Beiträge zur Geschichte der deutschen Sprache und Literatur (PBB)*, 133(1):14–31.

Wilcock, G. (2006). Natural language parsing with GOLD and SWRL. In *Proceedings of the Second International Conference on Rules and Rule Markup Languages for the Semantic Web (RuleML 2007)*, Athens, Georgia, November. `http://2006.ruleml.org/group3.html#3`.

Wilcock, G. (2007). An OWL ontology for HPSG. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (ACL-2007)*, pages 169–172, Prague, Czech Republic. Association for Computational Linguistics.