An Annotation Type System for a Data-Driven NLP Pipeline

Udo Hahn **Ekaterina Buyko** **Katrin Tomanek**
Jena University Language & Information Engineering (JULIE) Lab
Fürstengraben 30, 07743 Jena, Germany
{hahn|buyko|tomanek}@coling-uni-jena.de

Scott Piao **John McNaught** **Yoshimasa Tsuruoka** **Sophia Ananiadou**
NaCTeM and School of Computer Science
University of Manchester
{scott.piao|john.mcnaught|yoshimasa.tsuruoka|sophia.ananiadou}@manchester.ac.uk

Abstract

We introduce an annotation type system for a data-driven NLP core system. The specifications cover formal document structure and document meta information, as well as the linguistic levels of morphology, syntax and semantics. The type system is embedded in the framework of the Unstructured Information Management Architecture (UIMA).

1 Introduction

With the maturation of language technology, software engineering issues such as re-usability, interoperability, or portability are getting more and more attention. As dozens of stand-alone components such as tokenizers, stemmers, lemmatizers, chunkers, parsers, etc. are made accessible in various NLP software libraries and repositories the idea sounds intriguing to (re-)use them on an 'as is' basis and thus save expenditure and manpower when one configures a composite NLP pipeline.

As a consequence, two questions arise. First, how can we abstract away from the specific code level of those single modules which serve, by and large, the same functionality? Second, how can we build NLP systems by composing them, at the abstract level of functional specification, from these already existing component building blocks disregarding concrete implementation matters? Yet another burning issue relates to the increasing availability of multiple metadata annotations both in corpora and language processors. If alternative annotation tag sets are chosen for the same functional task a 'data conversion' problem is created which should be solved at the abstract specification level as well (Ide et al., 2003).

Software engineering methodology points out that these requirements are best met by properly identifying input/output capabilities of constituent components and by specifying a general data model (e.g., based on UML (Rumbaugh et al., 1999)) in order to get rid of the low-level implementation (i.e., coding) layer. A particularly promising proposal along this line of thought is the *Unstructured Information Management Architecture* (UIMA) (Ferrucci and Lally, 2004) originating from IBM research activities.[1] UIMA is but the latest attempt in a series of proposals concerned with more generic NLP engines such as ATLAS (Laprun et al., 2002) or GATE (Cunningham, 2002). These frameworks have in common a data-driven architecture and a data model based on annotation graphs as an adaptation of the TIPSTER architecture (Grishman, 1997). They suffer, however, from a lack of standards for data exchange and abstraction mechanisms at the level of specification languages.

This can be achieved by the definition of a common annotation scheme. We propose an UIMA schema which accounts for a significant part of the complete NLP cycle – from the collection of documents and their internal formal structure, via sentence splitting, tokenization, POS tagging, and parsing, up until the semantic layer (still excluding discourse) – and which aims at the implementation-independent specification of a core NLP system.

[1] Though designed for any sort of unstructured data (text, audio and video data), we here focus on special requirements for the analysis of written documents.

Proceedings of the Linguistic Annotation Workshop, pages 33–40,
Prague, June 2007.

2 Related work

Efforts towards the design of annotation schemata for language resources and their standardization have a long-standing tradition in the NLP community. In the very beginning, this work often focused exclusively on subdomains of text analysis such as document structure meta-information, syntactic or semantic analysis. The *Text Encoding Initiative* (TEI)[2] provided schemata for the exchange of documents of various genres. The *Dublin Core Metadata Initiative*[3] established a de facto standard for the Semantic Web.[4] For (computational) linguistics proper, syntactic annotation schemes, such as the one from the Penn Treebank (Marcus et al., 1993), or semantic annotations, such as the one underlying ACE (Doddington et al., 2004), are increasingly being used in a quasi standard way.

In recent years, however, the NLP community is trying to combine and merge different kinds of annotations for single linguistic layers. XML formats play a central role here. An XML-based encoding standard for linguistic corpora XCES (Ide et al., 2000) is based on CES (Corpus Encoding Standard) as part of the EAGLES Guidelines.[5] Work on TIGER (Brants and Hansen, 2002) is an example for the liaison of dependency- and constituent-based syntactic annotations. New standardization efforts such as the *Syntactic Annotation Framework* (SYNAF) (Declerck, 2006) aim to combine different proposals and create standards for syntactic annotation.

We also encounter a tendency towards multiple annotations for a single corpus. Major bio-medical corpora, such as GENIA (Ohta et al., 2002) or PennBioIE,[6] combine several layers of linguistic information in terms of morpho-syntactic, syntactic and semantic annotations (named entities and events). In the meantime, the *Annotation Compatibility Working Group* (Meyers, 2006) began to concentrate its activities on the mutual compatibility of annotation schemata for, e.g., POS tagging, treebanking, role labeling, time annotation, etc.

The goal of these initiatives, however, has never been to design an annotation scheme for a complete NLP pipeline as needed, e.g., for information extraction or text mining tasks (Hahn and Wermter, 2006). This lack is mainly due to missing standards for specifying comprehensive NLP software architectures. The MEANING format (Pianta et al., 2006) is designed to integrate different levels of morpho-syntactic annotations. The HEART OF GOLD middleware (Schäfer, 2006) combines multidimensional mark-up produced by several NLP components. An XML-based NLP tool suite for analyzing and annotating medical language in an NLP pipeline was also proposed by (Grover et al., 2002). All these proposals share their explicit linkage to a specific NLP tool suite or NLP system and thus lack a generic annotation framework that can be re-used in other developmental environments.

Buitelaar et al. developed in the context of an information extraction project an XML-based multi-layered annotation scheme that covers morpho-syntactic, shallow parsing and semantic annotation (Buitelaar et al., 2003). Their scheme borrows concepts from object-oriented programming (e.g., abstract types, polymorphism). The object-oriented perspective already allows the development of a domain-independent schema and extensions of core types without affecting the base schema. This schema is comprehensive indeed and covers a significant part of advanced NLP pipelines but it is also not connected to a generic framework.

It is our intention to come full circle within a general annotation framework. Accordingly, we cover a significant part of the NLP pipeline from document meta information and formal document structure, morpho-syntactic and syntactic analysis up to semantic processing. The scheme we propose is intended to be compatible with on-going work in standardization efforts from task-specific annotations and to adhere to object-oriented principles.

3 Data-Driven NLP Architecture

As the framework for our specification efforts, we adopted the *Unstructured Information Management Architecture* (UIMA) (Ferrucci and Lally, 2004). It provides a formal specification layer based on UML, as well as a run-time environment for the interpretation and use of these specifications. This dualism is going to attract more and more researchers as a basis

[2] `http://www.tei-c.org`
[3] `http://dublincore.org`
[4] `http://www.w3.org/2001/sw`
[5] `http://www.ilc.cnr.it/EAGLES96/`
[6] `http://bioie.ldc.upenn.edu`

for proper NLP system engineering.

3.1 UIMA-based Tool Suite

UIMA provides a platfrom for the integration of NLP components (ANALYSIS ENGINES in the UIMA jargon) and the deployment of complex NLP pipelines. It is more powerful than other prominent software systems for language engineering (e.g., GATE, ATLAS) as far as its pre- and post-processing facilities are concerned — so-called COLLECTION READERS can be developed to handle any kind of input format (e.g., WWW documents, conference proceedings), while CONSUMERS, on other hand, deal with the subsequent manipulation of the NLP core results (e.g., automatic indexing). Therefore, UIMA is a particularly suitable architecture for advanced text analysis applications such as text mining or information extraction.

We currently provide ANALYSIS ENGINES for sentence splitting, tokenization, POS tagging, shallow and full parsing, acronym detection, named entity recognition, and mapping from named entities to database term identifiers (the latter is motivated by our biological application context). As we mainly deal with documents taken from the biomedical domain, our collection readers process documents from PUBMED,[7] the most important literature resource for researchers in the life sciences. PUBMED currently provides more than 16 million bibliographic references to bio-medical articles. The outcomes of ANALYSIS ENGINES are input for various CONSUMERS such as semantic search engines or text mining tools.

3.2 Common Analysis System

UIMA is based on a data-driven architecture. This means that UIMA components do not exchange or share code, they rather exchange data only. The components operate on common data referred to as COMMON ANALYSIS SYSTEM (CAS)(Götz and Suhre, 2004). The CAS contains the subject of analysis (document) and provides meta data in the form of annotations. Analysis engines receive annotations through a CAS and add new annotations to the CAS. An annotation in the CAS then associates meta data with a region the subject of the analysis occupies (e.g., the start and end positions in a document).

UIMA defines CAS interfaces for indexing, accessing and updating the CAS. CASes are modelled independently from particular programming languages. However, JCAS, an object-oriented interface to the CAS, was developed for JAVA. CASes are crucial for the development and deployment of complex NLP pipelines. All components to be integrated in UIMA are characterized by abstract input/output specifications, so-called *capabilities*. These specifications are declared in terms of *descriptors*. The components can be integrated by wrappers conforming with the descriptors. For the integration task, we define in advance what kind of data each component may manipulate. This is achieved via the UIMA *annotation type system*. This type system follows the object-oriented paradigm. There are only two kinds of data, *viz.* types and features. *Features* specify slots within a type, which either have primitive values such as integers or strings, or have references to instances of types in the CAS. *Types*, often called feature structures, are arranged in an inheritance hierarchy.

In the following section, we propose an ANNOTATION TYPE SYSTEM designed and implemented for an UIMA Tool Suite that will become the backbone for our text mining applications. We distinguish between the design and implementation levels, talking about the ANNOTATION SCHEME and the TYPE SYSTEM, respectively.

4 Annotation Type System

The ANNOTATION SCHEME we propose currently consists of five layers: *Document Meta, Document Structure & Style, Morpho-Syntax, Syntax* and *Semantics*. Accordingly, annotation types fall into five corresponding categories. *Document Meta* and *Document Structure & Style* contain annotations about each document's bibliography, organisation and layout. *Morpho-Syntax* and *Syntax* describe the results of morpho-syntactic and syntactic analysis of texts. The results of lemmatisation, stemming and decomposition of words can be represented at this layer, as well. The annotations from shallow and full parsing are represented at the *Syntax* layer. The appropriate types permit the representation of dependency- and constituency-based parsing results. *Semantics*

[7] http://www.pubmed.gov

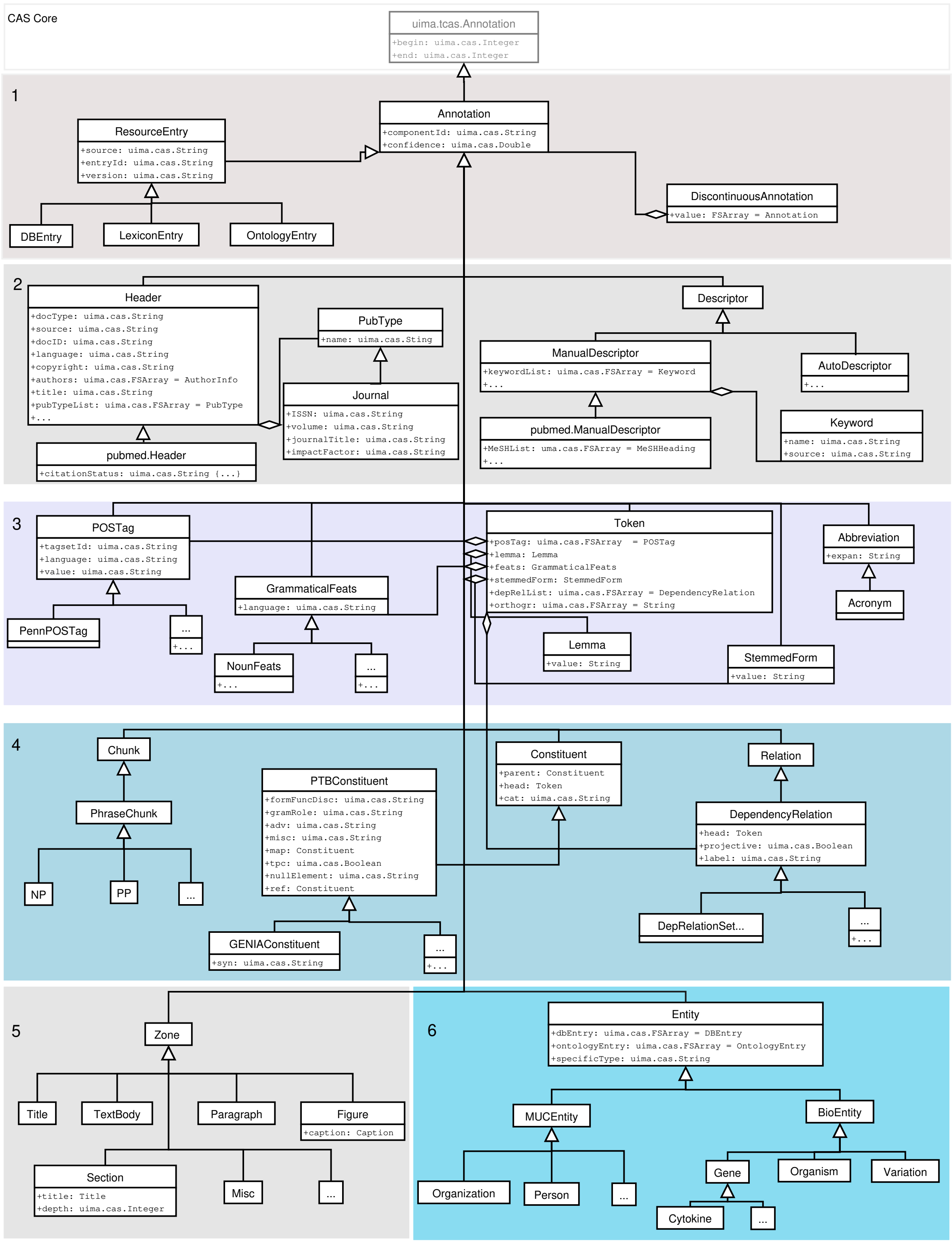

Figure 1: Multi-Layered UIMA Annotation Scheme in UML Representation. 1: Basic Feature Structure and Resource Linking. 2: Document Meta Information. 3: Morpho-Syntax. 4: Syntax. 5: Document Structure & Style. 6: Semantics.

currently covers information about named entities, events and relations between named entities.

4.1 Basic Feature Structure

All types referring to different linguistic layers derive from the basic type `Annotation`, the root type in the scheme (cf. Figure 1-1). The `Annotation` type itself derives information from the default UIMA annotation type `uima.tcas.Annotation` and, thus, inherits the basic annotation features, *viz. begin* and *end* (marking spans of annotations in the subject of analysis). `Annotation` extends this default feature structure with additional features. The *componentId* marks which NLP component actually computed this annotation. This attribute allows to manage multiple annotations of the same type The unique linkage between an analysis component and an annotation item is particularly relevant in cases of parallel annotations. The component from which the annotation originated also assigns a specific confidence score to its *confidence* feature. Each type in the scheme is at least supplied with these four slots inherited from their common root type.

4.2 Document Meta Information

The *Document Meta* layer (cf. Figure 1-2) describes the bibliographical and content information of a document. The bibliographical information, often retrieved from the header of the analyzed document, is represented in the type `Header`. The *source* and *docID* attributes yield a unique identifier for each document. We then adopted some Dublin Core elements, e.g., *language, title, docType*. We distinguish between domain-independent information such as language, title, document type and domain-dependent information as relevant for text mining in the bio-medical domain. Accordingly, the type `pubmed.Header` was especially created for the representation of PUBMED document information. A more detailed description of the document's publication data is available from types which specialize `PubType` such as `Journal`. The latter contains standard journal-specific attributes, e.g., *ISSN*, *volume*, *journalTitle*.

The description of the document's content often comes with a list of keywords, information assigned to the `Descriptor` type. We clearly distinguish between content descriptors manually provided by an author, indexer or curator, and items automatically generated by text analysis components after document processing. While the first kind of information will be stored in the `ManualDescriptor`, the second one will be represented in the `AutoDescriptor`. The generation of domain-dependent descriptors is also possible; currently the scheme contains the `pubmed.ManualDescriptor` which allows to assign attributes such as chemicals and genes.

4.3 Document Structure & Style

The *Document Structure* & *Style* layer (cf. Figure 1-5) contains information about the organization and layout of the analyzed documents. This layer enables the marking-up of document structures such as paragraphs, rhetorical zones, figures and tables, as well as typographical information, such as italics and special fonts. The focus of modeling this layer is on the annotation of scientific documents, especially in the life sciences. We adopted here the SCIXML[8] annotation schema, which was especially developed for marking-up scientific publications. The `Zone` type refers to a distinct division of text and is the parent type for various subtypes such as `TextBody`, `Title` etc. While it seems impossible to predict all of the potential formal text segments, we first looked at types of text zones frequently occurring in scientific documents. The type `Section`, e.g., represents a straightforward and fairly standard division of scientific texts into introduction, methods and results sections. The divisions not covered by current types can be annotated with `Misc`. The annotation of tables and figures with corresponding types enables to link text and additional non-textual information, an issue which is gaining more and more attention in the text mining field.

4.4 Morpho-Syntax

The *Morpho-Syntax* layer (cf. Figure 1-3) represents the results of morpho-syntactic analysis such as tokenization, stemming, POS tagging. The smallest annotation unit is `Token` which consists of five attributes, including its part-of-speech information

[8] `http://www.cl.cam.ac.uk/~aac10/escience/sciborg.html`

(*posTag*), *stemmedForm*, *lemma*, grammatical features (*feats*), and orthographical information (*orthogr*).

With respect to already available POS tagsets, the scheme allows corresponding extensions of the supertype `POSTag` to, e.g., `PennPOSTag` (for the Penn Tag Set (Marcus et al., 1993)) or `GeniaPOSTag` (for the GENIA Tag Set (Ohta et al., 2002)). The attribute *tagsetId* serves as a unique identifier of the corresponding tagset. The value of the POS tag (e.g., NN, VVD, CC) can be stored in the attribute *value*. The potential values for the instantiation of this attribute are always restricted to the tags of the associated tagset. These constraints enforce formal control on annotation processes.

As for morphologically normalized lexical items, the `Lemma` type stores the canonical form of a lexical token which can be retrieved from a lexicon once it is computed by a lemmatizer. The lemma *value*, e.g., for the verb *'activates'* would be *'activate'*. The `StemmedForm` represents a base form of a text token as produced by stemmers (e.g., *'activat-*' for the noun *'activation'*).

Due to their excessive use in life science documents, abbreviations, acronyms and their expanded forms have to be considered in terms of appropriate types, as well. Accordingly, `Abbreviation` and `Acronym` are defined, the latter one being a child type of the first one. The expanded form of a short one can easily be accessed from the attribute *expan*.

Grammatical features of tokens are represented in those types which specialize the supertype `GrammaticalFeats`. Its child types, *viz.* `NounFeats`, `VerbFeats`, `AdjectiveFeats`, `PronounFeats` (omitted from Figure 1-3) cover the most important word categories. Attributes of these types obviously reflect the properties of particular grammatical categories. While `NounFeats` comes with *gender*, *case* and *number* only, `PronounFeats` must be enhanced with *person*. A more complex feature structure is associated with `VerbFeats` which requires attributes such as *tense*, *person*, *number*, *voice* and *aspect*. We adapted here specifications from the TEI to allow compatibility with other annotation schemata.

The type `LexiconEntry` (cf. Figure 1-1) enables a link to the lexicon of choice. By designing this type we achieve much needed flexibility in linking text snaps (e.g., tokens, simplex forms, multiword terms) to external resources. The attributes *entryId* and *source* yield, in combination, a unique identifier of the current lexicon entry. Resource version control is enabled through an attribute *version*.

Text annotations often mark disrupted text spans, so-called *discontinuous annotations*. In coordinated structures such as *'T and B cell'*, the annotator should mark two named entities, *viz. 'T cell'* and *'B cell'*, where the first one results from the combination of the disjoint parts *'T'* and *'cell'*. In order to represent such discontinous annotations, we introduced the type `DiscontinuousAnnotation` (cf. Figure 1-1) which links through its attribute *value* spans of annotations to an annotation unit.

4.5 Syntax

This layer of the scheme provides the types and attributes for the representation of syntactic structures of sentences (cf. Figure 1-4). The results from shallow and full parsing can be stored here.

Shallow parsing (chunking) aims at dividing the flow of text into phrases (chunks) in a non-overlapping and non-recursive manner. The type `Chunk` accounts for different chunk tag sets by subtyping. Currently, the scheme supports `Phrase-Chunks` with subtypes such as NP, VP, PP, or ADJP (Marcus et al., 1993).

The scheme also reflects the most popular full parsing approaches in NLP, *viz.* constituent-based and dependency-based approaches. The results from constituent-based parsing are represented in a parse tree and can be stored as single nodes in the `Constituent` type. The tree structure can be reconstructed through links in the attribute *parent* which stores the *id* of the parent constituent. Besides the attribute *parent*, `Constituent` holds the attributes *cat* which stores the complex syntactic category of the current constituent (e.g., NP, VP), and *head* which links to the head word of the constituent. In order to account for multiple annotations in the constituent-based approach, we introduced corresponding constituent types which specialize `Constituent`. This parallels our approach which we advocate for alternatives in POS tagging and the management of alternative chunking results.

Currently, the scheme supports three different constituent types, *viz.* `PTBConstituent`, GENIAConstituent (Miyao and Tsujii, 2005) and PennBIoIEConstituent. The attributes of the type PTBConstituent cover the complete repertoire of annotation items contained in the Penn Treebank, such as functional tags for form/function dicrepancies (*formFuncDisc*), grammatical role (*gramRole*), adverbials (*adv*) and miscellaneous tags (*misc*). The representation of null elements, topicalized elements and gaps with corresponding references to the lexicalized elements in a tree is reflected in attributes *nullElement*, *tpc*, *map* and *ref*, respectively. GENIAConstituent and PennBIoIEConstituent inherit from PTB-Constituent all listed attributes and provide, in the case of GENIAConstituent , an additional attribute *syn* to specify the syntactic idiosyncrasy (coordination) of constituents.

Dependency parsing results are directly linked to the token level and are thus referenced in the Token type. The DependencyRelation type inherits from the general Relation type and introduces additional features which are necessary for describing a syntactic dependency. The attribute *label* characterizes the type of the analyzed dependency relation. The attribute *head* indicates the head of the dependency relation attributed to the analyzed token. The attribute *projective* relates to the property of the dependency relation whether it is projective or not. As different dependency relation sets can be used for parsing, we propose subtyping similar to the constituency-based parsing approaches. In order to account for alternative dependency relation sets, we aggregate all possible annotations in the Token type as a list (*depRelList*).

4.6 Semantics

The *Semantics* layer comprises currently the representation of named entities, particularly for the biomedical domain. The entity types are hierarchically organized. The supertype Entity (cf. Figure 1-6) links annotated (named) entities to the ontologies and databases through appropriate attributes, *viz. ontologyEntry* and s*dbEntry*. The attribute *specificType* specifies the analyzed entity in a more detailed way (e.g., Organism can be specified through the species values 'human', 'mouse', 'rat', etc.) The subtypes are currently being developed in the bio-medical domain and cover, e.g., genes, proteins, organisms, diseases, variations. This hierarchy can easily be extended or supplemented with entities from other domains. For illustration purposes, we extended it here by MUC (Grishman and Sundheim, 1996) entity types such as Person, Organization, etc.

This scheme is still under construction and will soon also incorporate the representation of relationships between entities and domain-specific events. The general type Relation will then be extended with specific conceptual relations such as location, part-of, etc. The representation of events will be covered by a type which aggregates pre-defined relations between entities and the event mention. An event type such as InhibitionEvent would link the text spans in the sentence *'protein A inhibits protein B'* in attributes *agent* (*'protein A'*), *patient* (*'protein B'*), *mention* (*'inhibits'*).

5 Conclusion and Future work

In this paper, we introduced an UIMA annotation type system which covers the core functionality of morphological, syntactic and semantic analysis components of a generic NLP system. It also includes type specifications which relate to the formal document format and document style. Hence, the design of this scheme allows the annotation of the entire cycle of (sentence-level) NLP analysis (discourse phenomena still have to be covered).

The annotation scheme consists mostly of core types which are designed in a domain-independent way. Nevertheless, it can easily be extended with types which fit other needs. The current scheme supplies an extension for the bio-medical domain at the document meta and structure level, as well as on the semantic level. The morpho-syntactic and syntactic levels provide types needed for the analysis of the English language. Changes of attributes or attribute value sets will lead to adaptations to other natural languages.

We implemented the scheme as an UIMA type system. The formal specifications are implemented using the UIMA run-time environment. This direct link of formal and implementational issues is a major asset using UIMA unmatched by any previous specification approach. Furthermore, all annotation results can be converted to the XMI format within the UIMA framework. XMI, the XML Metadata Interchange format, is an OMG[9] standard for the XML representation of object graphs.

The scheme also eases the representation of annotation results for the same task with alternative and often competitive components. The identification of the component which provided specific annotations can be retrieved from the attribute *componentId*. Furthermore, the annotation with alternative and multiple tag sets is supported as well. We have designed for each tag set a type representing the corresponding annotation parameters. The inheritance trees at almost all annotation layers support the parallelism in annotation process (e.g., tagging may proceed with different POS tagsets).

The user of the scheme can restrict the potential values of the types or attributes. The current scheme makes use of the customization capability for POS tagsets, for all attributes of constituents and chunks. This yields additional flexibility in the design and, once specified, an increased potential for automatic control for annotations.

The scheme also enables a straightforward connection to external resources such as ontologies, lexicons, and databases as evidenced by the corresponding subtypes of `ResourceEntry` (cf. Figure 1-1). These types support the specification of a relation between a concrete text span and the unique item addressed in any of these resources.

With these considerations in mind, we strive for the elaboration of a common standard UIMA type system for NLP engines. The advantages of such a standard include an easy exchange and integration of different NLP analysis engines, the facilitation of sophisticated evaluation studies (where, e.g., alternative components for NLP tasks can be plugged in and out at the spec level), and the reusability of single NLP components developed in various labs.

Acknowledgments. This research was funded by the EC´s 6th Framework Programme (4th call) within the BOOTStrep project under grant FP6-028099.

References

S. Brants and S. Hansen. 2002. Developments in the TIGER annotation scheme and their realization in the corpus. In *Proc. of the 3rd LREC Conference*, pages 1643–1649.

P. Buitelaar, T. Declerck, B. Sacaleanu, Š. Vintar, D. Raileanu, and C. Crispi. 2003. A multi-layered, XML-based approach to the integration of linguistic and semantic annotations. In *Proc. of EACL 2003 Workshop NLPXML-03*.

H. Cunningham. 2002. GATE, a general architecture for text engineering. *Computers and the Humanities*, 36:223–254.

T. Declerck. 2006. SYNAF: Towards a standard for syntactic annotation. In *Proc. of the 5th LREC Conference*.

G. Doddington, A. Mitchell, M. Przybocki, L. Ramshaw, S. Strassel, and R. Weischedel. 2004. The Automatic Content Extraction (ACE) Program. In *Proc. of the 4th LREC Conference*, pages 837–840.

D. Ferrucci and A. Lally. 2004. UIMA: an architectural approach to unstructured information processing in the corporate research environment. *Natural Language Engineering*, 10(3-4):327–348.

T. Götz and O. Suhre. 2004. Design and implementation of the UIMA Common Analysis System. *IBM Systems Journal*, 43(3):476–489.

R. Grishman and B. Sundheim. 1996. Message Understanding Conference – 6: A brief history. In *Proc. of the 16th COLING*, pages 466–471.

R. Grishman. 1997. Tipster architecture design document, version 2.3. Technical report, Defense Advanced Research Projects Agency (DARPA), U.S. Departement of Defense.

C. Grover, E. Klein, M. Lapata, and A. Lascarides. 2002. XML-based NLP tools for analysing and annotating medical language. In *Proc. of the 2nd Workshop NLPXML-2002*, pages 1–8.

U. Hahn and J. Wermter. 2006. Levels of natural language processing for text mining. In S. Ananiadou and J. McNaught, editors, *Text Mining for Biology and Biomedicine*, pages 13–41. Artech House.

N. Ide, P. Bonhomme, and L. Romary. 2000. XCES: An XML-based standard for linguistic corpora. In *Proc. of the 2nd LREC Conference*, pages 825–830.

N. Ide, L. Romary, and E. de la Clergerie. 2003. International standard for a linguistic annotation framework. In *Proc. of the HLT-NAACL 2003 SEALTS Workshop*, pages 25–30.

C. Laprun, J. Fiscus, J. Garofolo, and S. Pajot. 2002. A practical introduction to ATLAS. In *Proc. of the 3rd LREC Conference*, pages 1928–1932.

M. Marcus, B. Santorini, and M.A. Marcinkiewicz. 1993. Building a large annotated corpus of English: The PENN TREEBANK. *Computational Linguistics*, 19(2):313–330.

A. Meyers. 2006. Annotation compatibility working group report. In *Proc. of the COLING-ACL 2006 Workshop FLAC 2006'*, pages 38–53.

Y. Miyao and J. Tsujii. 2005. Probabilistic disambiguation models for wide-coverage HPSG parsing. In *Proc. of the ACL 2005*, pages 83 – 90.

T. Ohta, Y. Tateisi, and J.-D. Kim. 2002. The GENIA corpus: An annotated research abstract corpus in molecular biology domain. In *Proc. of the 2nd HLT*, pages 82–86.

E. Pianta, L. Bentivogli, C. Girardi, and B. Magnini. 2006. Representing and accessing multilevel linguistic annotation using the MEANING format. In *Proc. of the 5th EACL-2006 Workshop NLPXML-2006*, pages 77–80.

J. Rumbaugh, I. Jacobson, and G. Booch. 1999. *The Unified Modeling Language Reference Manual*. Addison-Wesley.

U. Schäfer. 2006. Middleware for creating and combining multi-dimensional NLP markup. In *Proc. of the 5th EACL-2006 Workshop NLPXML-2006*, pages 81–84.

[9] `http://www.omg.org`