Workshop Program

9:00	Welcome
9:30	Integrating Linguistic Information from Multiple Sources in Lexicon Development and Spoken Language Annotation *Per Anders Jande*
10:15	Argument Structure in TimeML *James Pustejovsky, Jessica Littman, Roser Saurí*
11:00	Coffee break
11:30	Dependency conversion and parsing of the BulTreeBank *Atanas Chanev, Kiril Simov, Petya Osenova, Svetoslav Marinov*
12:15	NLP Tools Integration Using a Multi-Layered Repository *João Graça, Nuno J. Mamede, João D. Pereira*
13:00	Lunch break
14:30	Merging FrameNet and PropBank in a corpus of written Dutch *Paola Monachesi, Jantine Trapman*
15:15	XML-Based Representation of Multi-Layered Annotation in the PDT *Petr Pajas, Jan Stepánek*
16:00	DEB Tools for Merging Linguistic Resources *Ales Horák, Karel Pala*
16:30	Coffee break
17:00	Sustainability of Linguistic Resources *Stefanie Dipper, Erhard Hinrichs, Thomas Schmidt, AndreasWagner, AndreasWitt*
17:45	Merging Layered Annotations *Nancy Ide, Keith Suderman*
18:00	Demo session
18:30	Closing session

Workshop Organisers

Erhard Hinrichs, University of Tübingen, Germany
Nancy Ide, Vassar College, USA
Martha Palmer, University of Colorado-Boulder, USA
James Pustejovsky, Brandeis University, USA

Program Committee

Eneko Agirre (Basque Country University, Spain)
Collin Baker (International Computer Science Institute, USA)
Gosse Bouma (University of Groningen, The Netherlands)
Monserrat Civit (Centre de Llenguatges i Computació, University of Barcelona, Spain)
Hamish Cunningham (University of Sheffield, UK)
Bonnie Dorr (University of Maryland, USA)
Eva Ejerhed (University of Umea, Sweden)
Tomaz Erjavec (Institut Josef Stefan, Slovenia)
David Farwell (CRL New Mexico State University, USA)
Christiane Fellbaum (Princeton University, USA)
Charles J. Fillmore (International Computer Science Institute, USA)
Jan Hajic (Center for Computational Linguistics, Charles University, Czech Republic)
Eva Hajicova (Center for Computational Linguistics, Charles University, Czech Republic)
Eduard Hovy (International Sciences Institute, USA)
Sandra Kübler (University of Tübingen, Germany)
Alessandro Lenci (University of Pisa, Italy)
Lori Levin (LTI, Carnegie-Mellon University, USA)
Inderjeet Mani (MITRE, USA)
Adam Meyer (New York University, USA)
Rada Mihalcea (University of North Texas, USA)
Sergei Nirenburg (University of Maryland-Baltimore County, USA)
Joakim Nivre (Växjö University, Sweden)
Boyan A. Onyshkevych (U.S. Dept. of Defense, USA)
Karel Pala, (Masaryk University, Czech Republic)
Gerald Penn (University of Toronto, CA)
Wim Peters (University of Sheffield, UK)
Manfred Pinkal (DFKI, Saarbruecken, Germany)
Massimo Poesio (University of Essex, UK)
Adam Przepiorkowski (Polish Academy of Sciences, Poland)
Owen Rambow (Columbia University, USA)
Kiril Simov (CLPP, Sofia, Bulgaria)
Beth Sundheim (SPAWAR Systems Center, USA)
Piek Vossen (Irion Technologies, The Netherlands)
Fei Xia (IBM Research, USA)
Bert Xue (University of Pennsylvania, USA)
Dietmar Zaefferer (Ludwig-Maximilians-Universitaet, Germany)
Annie Zaenen (Palo Alto Research Center, USA)

Table of Contents

Integrating Linguistic Information from Multiple Sources in Lexicon Development and Spoken Language Annotation
Per Anders Jande 1

Argument Structure in TimeML
James Pustejovsky, Jessica Littman, Roser Saurí 8

Dependency conversion and parsing of the BulTreeBank
Atanas Chanev, Kiril Simov, Petya Osenova, Svetoslav Marinov 16

NLP Tools Integration Using a Multi-Layered Repository
João Graça, Nuno J. Mamede, João D. Pereira 24

Merging FrameNet and PropBank in a corpus of written Dutch
Paola Monachesi, Jantine Trapman 31

XML-Based Representation of Multi-Layered Annotation in the PDT 2.0
Petr Pajas, Jan Stepánek 40

Sustainability of Linguistic Resources
Stefanie Dipper, Erhard Hinrichs, Thomas Schmidt, AndreasWagner, AndreasWitt 48

DEB Tools for Merging Linguistic Resources
Ales Horák, Karel Pala 55

Merging Layered Annotations
Nancy Ide, Keith Suderman 61

Integrating Linguistic Information from Multiple Sources in Lexicon Development and Spoken Language Annotation

Per Anders Jande

Dept. of Speech, Music and Hearing, School of Computer Science and Communication, KTH
Lindstedtsvägen 24, SE-100 44 Stockholm, Sweden

Abstract

In this paper, two related spoken language-oriented projects are presented. Both projects deal with integrating linguistic information from multiple sources. The first project described is the development of a multi-purpose central lexicon database including phonemic representations. Special emphasis is put on central availability and facilitating incremental development. The second project described is a spoken langue annotation project aimed at creating data for data-driven pronunciation modelling. The annotation is designed to form a general description of discourse context, including variables from the discourse level down to the articulatory feature level. A multi-layer annotation scheme for spoken language is described and the information included in the annotation is presented. Models of pronunciation variation induced from the annotation are evaluated in a tenfold cross validation experiment. On average, the models produce 8.1% errors on the phone level. Models trained on phoneme level information only produce an average error of 14.2%. This means that including information above the phoneme level in the context description can improve model performance by 42.6%.

1. Introduction

Studies of spoken language commonly involve various types of linguistic information. For example, in data-driven modelling of various spoken language phenomena, it is often necessary to annotate spoken language data with information on the phoneme and/or phone level as well as information on the word level, such as part of speech and morphology. At the development of speech synthesis systems and automatic speech recognition systems, pronunciation lexica are important.

In this paper, two spoken language-oriented projects are presented. A common denominator of the projects is that they deal with integrating linguistic information from multiple sources. The first project discussed is the development of a multi-purpose central lexicon database, which is used for the annotation of spoken language and in various other contexts. The second project, which is the main focus of this paper, is a data-driven approach to modelling phone-level pronunciation variation, involving the annotation of spoken language with various kinds of linguistic information in multiple layers.

2. A Central Lexicon Database

A multi-purpose central lexicon database called CENTLEX is being developed at the department of Speech, Music and Hearing (TMH) and the Centre for Speech Technology (CTT) at KTH. The lexicon is based on lexical resources of different types and on different formats, developed for various research projects at TMH/CTT over the years. The information is stored in a relational database with separate tables for different types of information.

2.1. Information Included in the Lexicon

CENTLEX is a full-form lexicon, with each entry minimally containing an orthographic word form and a grammatical analysis (part of speech and morphology). An entry can also have an arbitrary number of phonemic representations, ordered by their probability of use. Each phonemic representation can be enriched with information about the intended context of the representation (e.g. *reduced form* or *foreign language*). Such information is added e.g. for proper names, since orthographically identical names may be pronounced differently depending on the native language environment of the person bearing the name. An entry also contains information about the probability of the particular grammatical analysis given the orthographic word (estimated from a large automatically tagged text corpus). Presently, the database contains about 410,000 entries with 330,000 unique orthographic word forms.

2.2. Availability

One of the main ideas with the CENTLEX database is that all lexical data used in projects at TMH and within CTT is stored centrally, so that the data is immediately and easily available for all researchers at the department and for all partners involved in the Centre. Lexicon-related work conducted in different projects can be easily integrated with the central lexical resource, and the results immediately available for all users. Standards for mapping between the CENTLEX format and several commonly used formats have been developed to facilitate information sharing.

An interface to the database on the TMH internal web makes it possible to search the lexicon and to check out purpose-specific lexica with the set of information requested on several different output formats. Selected users also have the possibility to edit the lexicon via the web interface, to stimulate continuous lexicon expansion and improvement of existing data. The web interface is not suited for large-scale changes of the database, so a stand-alone annotation/correction tool has been developed for lexicon development on a larger scale. This tool stores information on a CENTLEX import format, so that it can be easily incorporated with the database.

The lexicon is thus incrementally built and the latest version is always available at a central location. Some of the information first included in the database has been automatically generated and the initial information merger was done with automatic methods. The data thus has to be checked with respect to quality, which is done continuously. Subsequently added information is, however, mostly information which is manually obtained or checked. Each lexicon entry is annotated with information about whether it has been manually checked/corrected, by whom and when, to separate information of different quality.

2.3. Applications

Thus far, the CENTLEX database has been used as a lexicon in an experimental speech synthesis system (used in various research-oriented applications at the department of Speech, Music and Hearing at KTH) and in a large vocabulary speech recognition system. CENTLEX has also been used for training grapheme-to-phoneme conversion rules for commercial speech synthesis and as a lexicon for commercial speech synthesis applications. It has further been used as a reference in the development of a system for production of talking books with synthetic speech for visually impaired and dyslectic university students. Finally, CENTLEX has been used for annotation in research projects aimed at context-sensitive prosody prediction and phone-level pronunciation prediction.

3. Pronunciation Variation Modelling

Although there is a certain degree of individual and random variation in the pronunciation of words in context, the variation is largely systematic within a restricted, relatively homogeneous group of language users. This agreement on systematic variation strategies can be seen as a property of the language variety (e.g. dialect) spoken by the group. The aim in the pronunciation variation modelling project described here is to model this systematic variation inherent to a language variety, with the focus on variation in phone level realisation. The target language variety used in the work presented in this paper is central standard Swedish.

3.1. Annotating Spoken Language Data

The methods used for pronunciation variation modelling are data-driven. Spoken language is annotated with various kinds of linguistic and related information, which is used by machine learning algorithms to create pronunciation models. The phoneme is the central unit in the approach and the annotation is aimed at describing the discourse context of a phoneme from high-level linguistic variables such as speaking style, down to the articulatory feature level. This multi-variable linguistic context description is then used to predict the context-sensitive realisation of the phoneme.
The results reported in this paper are based on recent additions to the annotated data. The effect of making information on different linguistic levels available as predictors of phone level pronunciation is investigated and the predictive power of specific linguistic variables is discussed.

3.2. Background

Phonological work on pronunciation variation in Swedish has been reported by several authors, e.g. Gårding (1974), Bruce (1986), Bannert and Czigler (1999), Jande (2003) and Jande (2005). There is an extensive corpus of research on the influence of various context variables on the pronunciation of words. Variables that have been found to influence the segmental realisation of words in context are foremost speech rate, word predictability (often estimated by global word frequency) and speaking style, cf. e.g. Fosler-Lussier and Morgan (1999), Finke and Waibel (1997), Jurafsky et al. (2001a) and Van Bael et al. (2004).
The influence of various other variables on the pronunciation of words has also been studied, but these have mostly been studied in isolation. When more variables are taken into account, the number of variables simultaneously under study is in most cases limited to less than a handful. Describing the discourse context more generally, including a large variety of linguistic and related variables, enables studies of the interplay between various information sources on e.g. phone-level pronunciation.
Machine learning methods can be used for such studies. A model of pronunciation variation created through machine learning can be useful in speech technology applications, e.g. for creating more dynamic and natural-sounding speech synthesis. In addition to models which can predict the pronunciation of words in context, it is possible to create models which are descriptive and to some degree explains the interplay between different types of variables involved in the predictions.

3.3. Speech Data

The speech data used for pronunciation variation modelling is the VAKOS database, originally constructed by Bannert and Czigler (1999) for a phonological study of variation in consonant clusters, a RADIO INTERVIEW database and a RADIO NEWS database, with recordings originating from *Sveriges radio* (Swedish public service radio).
The VAKOS database is a set of elicited monologues; ten speakers talk about some suggested topic or topics to a recording assistant (who is silent). About ten minutes from each speaker is included in the database. The VAKOS database also includes some manual annotation at different levels. The RADIO INTERVIEW database is a set of two 25 minute radio broadcast interviews, each including speech mainly from three speakers, the interviewee and two interviewers. The interviewees are experienced public speakers and are allowed to answer questions in length, rarely being interrupted. The RADIO NEWS database includes two radio news broadcasts, including speech from altogether three studio news announcers and eight reporters. Only studio environment recordings are included in the RADIO NEWS database.

3.4. A Multi-Layer Annotation System

The annotation used for pronunciation variation modelling is organised in six layers: 1) a discourse layer, 2) an utterance layer, 3) a phrase layer, 4) a word layer, 5) a syllable layer and 6) a phoneme layer. The layers are segmented into units, which are linguistically meaningful and can be synchronised to the speech signal. The segmentation of each layer is strictly sequential, i.e., every part of the signal belongs to some unit at all layers and there is no overlap between units within a layer.
Durational boundaries are inherited from higher order layers to lower order layers, so that a discourse boundary is always also an utterance boundary, a phrase boundary, a word boundary, a syllable boundary and a phoneme boundary. The layers are thus hierarchically ordered so that a higher

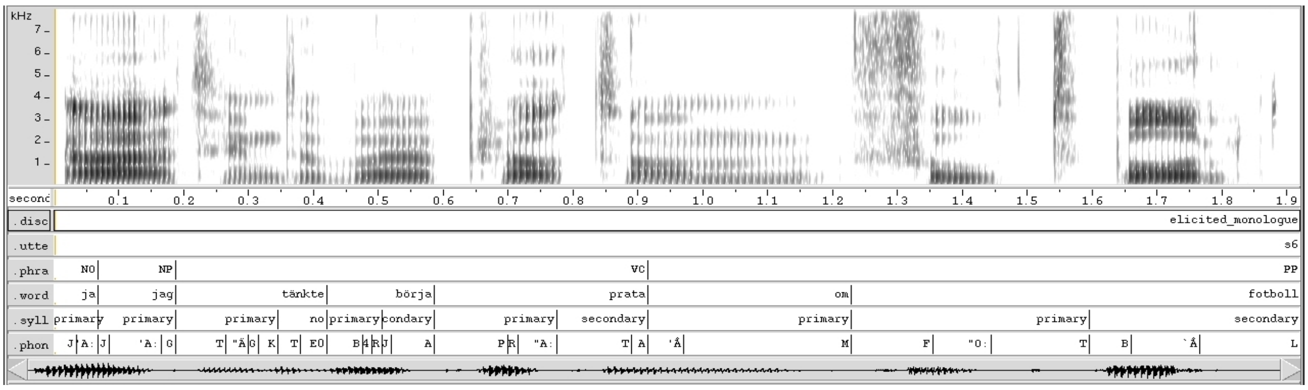


Figure 1: Annotation layers with example annotation aligned to the speech signal

order unit serves as the parent of all lower order units within its segmental bounds. An arbitrary amount of information can be supplied for each unit in each layer. Figure 1 shows an excerpt of a sound file with some aligned example annotation.

The most important feature of this system of annotation is that information can be unambiguously inherited from units on higher layers by units on the layers below. A unit can thus pass on its information to all the units within its bounds in the lower order layers. Consequently, information connected to syllable, word, phrase, utterance and discourse layer units, respectively, as well as to the phoneme layer units, is accessible from the phoneme layer. This is important since the pronunciation variation models will use phoneme-sized units as input. Sequential context information, i.e., properties of the units adjacent to the current unit at the respective layers is used at model induction together with information connected to the current unit. Having the information stored in different layers enables easy access to the sequential context information.

3.5. Segmentation

With some minor exceptions, automatic methods are used for segmentation, however with manual supervision to improve accuracy at some intermediate stages. The annotation process begins with segmenting each annotation layer into its respective type of unit. The next step is to retrieve, calculate or estimate a set of features for each unit. An utterance is in this context defined as a discourse turn uttered by a single speaker. This means that a monologue discourse is treated as a single utterance. For dialogues, the corpus is manually segmented into utterances.

Automatic segmentation begins at the word level. Given an orthographic string, the corpus is segmented into word units using an automatic aligner Sjölander (2003). Manual correction of the word layer segmentation is performed, since all succeeding annotation depends on this segmentation and increases in the segmentation accuracy on this level gives large improvements in the accuracy of successive annotation. Manual word layer segmentation was already included in the VAKOS database.

The phrase layer is segmented with the help of a shallow parser (Megyesi, 2002) using a string of tags produced by a part of speech and morphological tagger. The phrases are aligned to the signal using the word boundaries. The parser was created for parsing written text, but it is robust and produces parses also for tagged orthographic transcripts of spoken language.

The phoneme layer is segmented word-by-word using the word boundaries and phonemic representations from the CENTLEX database as input to an automatic aligner (Sjölander, 2003). The phonemes are clustered into syllables with forced syllable boundaries at word boundaries and the syllable layer is segmented using this clustering and the durational boundaries from the phoneme level segmentation.

Some units with special characteristics are introduced at the word layer to ensure that parts of the signal that are not speech (or non-analysable speech) can be annotated. The special unit types are <*overlap*> (overlapping speech), <*pause*> (including pauses, inhalation and exhalation sounds), <*non_speech*> (including laughter, smacks, clicks, coughs and hawking sounds etc.) and <*filled pause*>. The information supplied for normal word units is *not* included for these units. Within the boundaries of one of the special word layer units, a <*sil*> (for pauses) or a <*junk*> special phoneme unit is used as a place filler at the phoneme layer, but no additional annotation is supplied on lower order layers. Every special word layer unit is, however, included in a phrase unit, an utterance unit and in the discourse unit.

3.6. Adding Information to the Units

Values for a set of variables hypothesised to be important for predicting the realisation of a phoneme in its discourse context is attached to each unit at each layer of annotation. The following sections will briefly describe the information attached to the units at each layer.

3.6.1. The Discourse Layer

A set of 'inverted speech rate' measures based on the global *mean phoneme duration* is attached to discourse layer units. Phoneme durations are estimated from the automatic alignment of the phonemic word representations to the signal. The discourse layer information also includes four speaking style-related variables: *number of discourse participants*, *degree of formality*, *degree of spontaneity* and *type of interaction*.

3.6.2. The Utterance Layer

In the utterance layer, mostly speaker attributes are annotated. *Speaker pitch register* is a binary variable that differentiates speakers with a high pitch register from speakers with a low pitch register. This variable may interplay with measures based on pitch movement. A set of *mean phoneme duration* measures over the utterance and sets of *pitch range* and *pitch dynamics* ('speech liveliness') measures are also included in the utterance layer annotation.

3.6.3. The Phrase Layer

An attribute called *phrase type* corresponds to the type of the current phrase according to the shallow parser used for phrase chunking. Also included in the phrase layer annotation is a set of *phrase length* measures: the number of *words*, *syllables* and *phonemes*, respectively, contained by each phase unit. Further, two measures associated with the *prosodic weight* of a phrase are calculated: the number of *stressed syllables* and the number of *focally stressed words* contained by the phrase (focal stress annotation was manually provided for a subset of the speech data). Finally, *pitch dynamics*, *pitch range* and *mean phoneme duration* measures are calculated over each phrase unit.

3.6.4. The Word Layer

The word is generally conceived of as the most central linguistic unit, in that it is the principal conveyor of meaning in language and the principal syntactic unit. There is thus a large variety of features that can be attached to the word units. To begin with, *part of speech* and morphological information from the tagger is included in the annotation. *Morphology* is included as a set of tags corresponding to different morphological dimensions. Based on the part of speech tags, a division of words into *word types* (content words vs. function words) is made. A similar variable denoted *function word* has the entire closed set of function words and a generic 'content word' representation as its possible values. There are pronunciation variation strategies specific to certain function words and the *function word* variable should be a strong predictor of this behaviour.

The predictability of a word has been shown to be important for the realisation of the word, cf. e.g. Fosler-Lussier and Morgan (1999) and Jurafsky et al. (2001b). Many variables influence the predictability of a word in context. Measures related to word predictability included in the word layer annotation are *word repetitions* and *lexeme repetitions* (the number of times the full-form word and the lexeme, respectively, has been repeated thus far in the discourse), *the position of the word in a phrase*, *the position of the word in a frequent collocation* and *global word frequency*. A special measure termed *word predictability* is also included in the annotation. This measure is an estimation based on a weighted combination of unigram, bigram and trigram probabilities collected from the Göteborg Spoken Language Corpus (Allwood et al., 2002). The *part of speech* variable already mentioned also affects the predictability of a word in context, since there are syntactic constraints governing language production.

The distances to the preceding and the succeeding focally stressed word can be important factors in predicting the pronunciation of the current word and these distances (measured in number of words) are therefore included in the word layer annotation. Information about the presence of a *pause*, a *filled pause* or an *interrupted word* adjacent to the current word is also included. Prosodic boundaries are important for grouping coherent subunits in the speech signal. For listeners, this grouping facilitates parsing the sound stream. Manual *prosodic boundary* annotation has been supplied for the databases used.

Word length is measured as the number of syllables and as the number of phonemes, respectively, contained by the word. Finally, some measures of *pitch dynamics*, *pitch range* and *mean phoneme duration* over each word unit are included in the word layer annotation.

3.6.5. The Syllable Layer

Information about the stress and accent of the current syllable is derived from the phonemic representations. Swedish has two different types of word stress, *accent I* and *accent II*. In central standard Swedish, accent I has a single stressed syllable while accent II has a primary and a secondary stress. There is also a special compound accent similar to accent II, with primary stress on the first compound constituent and a secondary stress on the last compound constituent. The *stress* annotation is a simple division between stressed and unstressed syllables, while the *accent* annotation takes the word accent into account, thus making the *accent* classification a division into finer stress type classes.

Further, the distances to the nearest preceding stressed syllable and to the nearest preceding syllable with *primary stress* (measured in number of syllables) are included in the syllable layer annotation. The distances to succeeding stresses are also included. *Syllable length* is measured in number of phonemes. The initial and final syllables of a word are generally less prone to syllable reduction than medial syllables, which makes the *position of the syllable in the word* an important variable to include in the annotation. Lastly, a set of *mean phoneme duration* measures over the syllable are calculated.

3.6.6. The Phoneme Layer

The *phoneme identities* included in the phoneme layer annotation are represented by the phoneme symbols from CENTLEX. A set of *articulatory features* describing the phoneme is associated with each phoneme unit. The *position of the phoneme in the syllable* may be important for predicting the realisation of the phoneme. Hence, information about in which part of the syllable (*onset*, *nucleus* or *coda*) the phoneme is located is included in the annotation. A *consonant cluster length* variable takes as its value the length (phoneme count) of the consonant cluster of which the current phoneme is a part. This measure defaults to 0 for vowels.

The *phone* is the context-dependent realisation of the phoneme. Phonetic identity is the variable to be estimated by the pronunciation variation models and consequently, the phone is used as the key in model training. The phones are supplied by a hybrid automatic transcription system, using statistical decoding and a set of a posteriori correction rules.

A place filler ∅ symbol is used to signal that there is no realisation of a particular phoneme in the phonetic string.

The SNACK sound toolkit (Sjölander and Beskow, 2000) is used for building and decoding statistical models representing the possible realisations of a word. Models are built using an empirically compiled context-insensitive list of possible realisations (tentative phones) for each phoneme and a set of HMM monophone models. The speech signal is parameterised to form a sequence of observations. The path trough the statistical model most closely matching this observation sequence (using Viterbi decoding) can be represented as a string of phones, and this string is the output of the statistical decoder.

Evaluated against a small manually transcribed gold standard, statistical decoding alone was shown to give higher phone error rates (PER) than estimating the phonetic transcript with the phoneme string. However, due to the systematic nature of the errors made by the statistical decoder, a set of correction rules that significantly lowered the error rate could be compiled. The final hybrid transcription system produces an average of 15.5% errors on the phone level when compared to an enlarged gold standard transcription. This means that the PER is reduced by 40.4% compared to using the phoneme string for estimating the phone realisation.

Since manual transcription is restricted by a relatively small set of phone symbols, some decisions about phone identity are not obvious, most notably many cases of choosing between a full vowel symbol and a schwa. Defaulting to the system decision whenever a human transcriber is forced to make ad hoc decisions would increase the speed of manual transcript checking and correction considerably without lowering the transcription quality. It is worth noting that if this strategy had been used for compiling the gold standard transcript, the PER would have been somewhat lower. The 15.5% PER is thus a slight under-estimation of the system performance. Manual correction of the automatically obtained transcripts will most likely result in more accurate pronunciation variation models.

4. Creating Pronunciation Variation Models

Using the annotation from the speech databases, pronunciation variation models can be created with different types of machine learning methods. If the model is to be used for descriptive purposes, it must be transparent, i.e., it must contain information such that the model can be represented on a format interpretable by a human familiar with linguistic theory.

A machine learning paradigm that creates transparent models and is suitable for the type of data at hand is the *decision tree induction* paradigm. A decision tree inducer commonly needs no ad hoc knowledge and can induce models directly from training data. It is thus very easy to use once you have the data. For these reasons, the decision tree paradigm has been selected for creating the models reported in this paper. It is not claimed that the decision tree paradigm necessarily produces the best models. Other machine learning paradigms may be able to create more accurate models or models which meet certain application-specific demands.

4.1. Decision Tree Induction

Decision trees are induced from a set of training instances compiled from the structured annotation. The training instances are phoneme-sized and can be seen as a set of *context sensitive phonemes* with their respective phone realisations. Each training instance includes a set of 516 attribute values and the phone realisation, which is used as the classification key. The features of the current unit at each layer of annotation are included as attributes in the training examples. Where applicable, information from the neighbouring units at each annotation layer is also included in the attribute sets. The algorithm used for inducing the pronunciation variation models is that included in the DTREE program suite (Borgelt, 2004).

Decision tree induction is non-iterative and trees are built level by level, which makes the learning procedure fast. However, the optimal tree is not guaranteed. At each new level created during the tree induction procedure, the set of training instances is split into subsets according to the values of one of the attributes. The attribute selected is the attribute that best meets a given criterion, generally based on entropy minimisation. In the current case, a measure referred to as *symmetric information gain ratio* (Lopez de Mantaras, 1991) is used. The inducer is set to allow grouping of discrete values to obtain the optimal number of nodes at each level.

4.1.1. Pruning

Since training data generally contain some degree of noise, a decision tree may be biased toward the particular noise in the training data (over-trained). However, once a tree is constructed, it can be pruned to make it more generally applicable. The idea behind pruning is that the most common patterns are kept in the model, while less common patterns, with high probability of being due to noise in the training data, are deleted. At pruning, a sub-tree of a particular node is replaced by a leaf with the most common class of the leaves governed by the sub-tree, when some criterion is met.

4.2. Model Evaluation

A tenfold cross validation procedure was used for model evaluation. Under this procedure, the data is divided into ten equally sized partitions using random sampling. Ten different decision trees are induced, each with one of the partitions left out during training. The left out partition is then used for evaluation. A separate tenfold cross validation evaluation was performed for data from each of the three databases (VAKOS, RADIO INTERVIEW and RADIO NEWS) and for the collapsed data set.

The prosodic information cannot be fully exploited in its current form in e.g. a speech synthesis context. Thus, it was interesting topic investigate the influence of the prosodic information (variables based on f_0, duration, focal stress and prosodic boundary information) on model results. To investigate this, an experiment where the decision tree inducer did not have access to the prosodic information was performed for each of the four data sets. As a baseline, an evaluation of trees induced from phoneme layer information only was also performed for each data set. The same

Table 1: Mean and standard deviation of phone error rate (PER) for each data set

Database	All			VAKOS			RADIO INTERVIEW			RADIO NEWS		
# training instances	93,996			52,263			31,779			9,936		
# evaluation instances	10,444			5,807			3,531			1,104		
Trained on attributes	all	nopro*	pho†	all	nopro*	pho†	all	nopro*	pho†	all	nopro*	pho†
$\bar{\mathrm{x}}_{\mathrm{PER}}$ **(per cent)**	8.14	13.08	14.19	9.07	14.90	15.60	8.94	12.32	13.74	9.34	10.57	11.70
σ_{PER} **(per cent)**	0.15	0.25	0.23	0.39	0.49	0.53	0.42	0.30	0.54	1.23	1.23	1.34

**no prosodic attributes,* †*phoneme level attributes only*

Table 2: Error reduction as a result of making more information available for the decision tree inducer

Database	All		VAKOS		RADIO INT.		RADIO NEWS	
Tree types	pho†>all‡	nopro*>all‡	pho†>all‡	nopro*>all‡	pho†>all‡	nopro*>all‡	pho†>all‡	nopro*>all‡
Error reduction (per cent)	42.64	37.77	41.86	39.12	34.93	27.43	20.20	11.65

‡*trained with access to all attributes,* **trained access only to non-prosodic attributes,* †*trained with access only to phoneme level attributes*

randomisation was used under all conditions.

Each tree was pruned under a range of pruning criteria and the tree with the optimal performance on the evaluation data was selected to be used in the evaluation. The pruning criteria used all yielded the same pruned tree and the optimal tree could thus either be the *pruned* tree or the original, *unpruned* tree. The *symmetric information gain ratio* attribute selection measure created trees, which were near the optimal before pruning. Hence, the effect of pruning on model performance was small. In most cases, pruning affected model performance (on the test data) negatively and, on average, pruning gave rise to a *decrease* in model performance with 0.6%. The unpruned trees were actually subjected to *basic pruning*, at which the trees were pruned to the extent that no deterioration of accuracy on the training data occurred. Thus, following “Occam’s razor”, if there were several trees giving the same result, the simplest of these trees was selected.

5. Results and Discussion

Table 1 summarises the results from the cross validation experiments. On average, we get a phone error rate of 8.1% when training on 90% of the collapsed data set and allowing the decision tree inducer to use all available information.

5.1. Phone Error Rates

Using the phoneme string to estimate phone realisations gives a PER of 20.4%, which means that phone errors can be reduced by 60.2% by using an average pronunciation variation model in stead of using a phoneme string collected directly from a lexicon. Applying phonological sandhi rules to adapt the phonemic representations for isolated words to their context decreased the PER for the phoneme string only to 20.3%. The error reduction resulting from using the pronunciation variation model is thus significant. Further, as can be seen from Table 2, we get a reduction of PER by 42.6% when switching from a classifier trained on phoneme level information only to a classifier trained on all available information.

5.2. Data Size and Speaking Style

It is likely that the PERs presented in Table 1 reflect the fact that both the amount and the type of training data affects the performance of the models induced. If all attributes are used, neither models trained on the VAKOS database nor models trained on the RADIO NEWS database have the lowest PER, although the VAKOS database has the largest number of training instances and the RADIO NEWS database has the most formal type of speech. Instead, the models trained on the RADIO INTERVIEW database show the lowest PER. The RADIO INTERVIEW database has the advantages of having relatively formal speech compared to the VAKOS database, relatively few speakers and many more training instances than the RADIO NEWS database.

Further, we can see from Table 2 that models trained on the VAKOS database are more dependent on prosodic information and generally on information on layers above the phoneme, while the models trained on the RADIO NEWS database are less dependent on this type of information.

5.3. Attribute Ranking

Table 3 shows the 18 top ranking attributes over the ten optimal trees trained on all information from all databases. The layer from which the attribute is collected is used as a prefix in attribute names. Attributes can refer to the current unit or to units at ± 4 positions from the current unit at the specific annotation layer. Duration measures can be based on the duration of all *phonemes* or on the duration of *vowels* only, they can be based on *normalised* or *absolute* phoneme duration and they can be based on duration on a *log* scale.

The ranking in the first column of Table 3 is based on the position of the attribute in the ten trees. For this measure, the attribute governing the largest number of sub-trees (leaves excluded) will get the highest rank (1). The second column weights the sub-tree count with the number of classifications involving the attribute (over the training data). For this measure, an attribute involved in many classifications can climb in rank even if it does not appear in the absolute top of the tree. The *phoneme identity* attribute appears in the top node of all trees. This means that it governs all sub-trees and is involved in all classifications made by the trees.

5.4. Attributes Used by the Models

From Table 4, it can be seen that variables from all layers of annotation are used by the trees trained on all available information from all databases. In fact, from 516 available attributes, as many as 470 were used at least once in the

Table 3: The 18 top ranking attributes for trees trained on all information from all databases

Rank	Rank based on # sub-trees	Rank based on # sub-trees · # classifications
1	phoneme_identity	phoneme_identity
2	phoneme_identity+1	phoneme_identity+1
3	word_function_word-1	word_duration_phonemes_absolute
4	word_duration_phonemes_absolute	word_function_word+1
5	word_function_word+1	word_function_word
6	phoneme_identity+4	word_function_word-1
7	phoneme_identity-2	phoneme_identity-1
8	word_function_word	phoneme_identity+2
9	phoneme_identity-1	phoneme_identity-3
10	phoneme_identity-4	phoneme_identity+4
11	phoneme_identity+2	word_duration_vowels_absolute
12	phoneme_identity-3	phoneme_identity-2
13	phoneme_identity+3	phoneme_identity+3
14	word_duration_vowels_absolute	phoneme_identity-4
15	syllable_accent	syllable_accent
16	syllable_nucleus	phrase_duration_phonemes_absolute
17	word_duration_vowels_normalised	word_duration_vowels_normalised
18	word_duration_vowels_log_absolute	syllable_nucleus

Table 4: Probability of variables from each annotation layer at the top twelve tree levels

Level	P(Phoneme)	P(Syllable)	P(Word)	P(Phrase)	P(Utterance)	P(Discourse)	$\sum$
1	**1.0000**	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000
2	0.4101	0.0791	**0.4820**	0.0144	0.0144	0.0000	1.0000
3	**0.4113**	0.0493	0.3941	0.1404	0.0025	0.0025	1.0000
4	0.4052	0.0507	**0.4298**	0.0897	0.0145	0.0101	1.0000
5	0.3728	0.0310	**0.4281**	0.1294	0.0310	0.0077	1.0000
6	**0.3936**	0.0330	0.3729	0.1460	0.0348	0.0198	1.0000
7	**0.3952**	0.0316	0.3416	016.27	0.0383	0.0306	1.0000
8	**0.4338**	0.0408	0.3168	013.47	0.0397	0.0342	1.0000
9	**0.4140**	0.0440	0.3299	014.10	0.0543	0.0168	1.0000
10	**0.4180**	0.0384	0.3250	014.77	0.0561	0.0148	1.0000
11	**0.3958**	0.0545	0.3189	015.22	0.0529	0.0256	1.0000
12	**0.4096**	0.0422	0.3293	014.46	0.0562	0.0181	1.0000

ten trees. However, the phoneme and word layer attributes are the attributes most commonly used in the higher levels of the trees. The top ranking utterance layer attribute is a vowel-based duration measure showing up at place 50 using the first ranking strategy and on place 46 using the second ranking strategy. The top discourse layer attribute is also a vowel-based duration measure and shows up at place 31 and 35, respectively.

The *word frequency* and *word predictability* attributes both get ranks around 110. The relatively weak predictive strength of these variables may be due to the fact that they are obscured by the *function word* variable, which gets high ranks. Further, the *word frequency* and *word predictability* measures are estimated from a corpus of transcribed speech, relatively small in comparison to standard text corpora. These measures may be improved with text data.

A large variety of the duration and pitch based measures are represented among the variables used by the optimal trees (the first measure based on pitch shows up at place 42 using the fist ranking strategy and on place 55 using the second ranking strategy). Most of the duration measures seem to be nearly equivalent in terms of predictive power, with vowel-based measures working somewhat better over larger units. Units on higher order layers are both larger in terms of duration and conceptually more abstract than units on lower order layers. Because of this, it is not possible to make exact predictions from higher order layer units only and attributes from these levels end up in the lower levels of the decision trees, as a result of the 'greedy' induction algorithm used.

5.5. Effects of Noise

The erroneous classifications possible for a phoneme are limited to the set of realisations for the phoneme found in the training data. Both training and evaluation data contain up to 15.5% errors on the phone level, as previously discussed. Since the phone string is generated by an automatic transcription system with a priori restrictions on the possible realisations of each phoneme, the range of variation is probably less than it would have been if the transcripts had been produced by a human. It is not immediately obvious whether this translates into lower phone error rates for the pronunciation variation models than would have been the case if the phones in the training and evaluation data had been supplied by a human transcriber.

5.6. Gold Standard Evaluation

Although it is hard to speculate about how the model performance would be affected by more accurate training data, the transcriptions generated by the current models can be evaluated against actual target transcriptions. When evaluated against the small gold standard consisting of five minutes of manually transcribes speech from the VAKOS database, the models produce a PER of 16.9%, which means that the deterioration in performance when using the model instead of the automatic transcription system is only 8.5% and that the improvement using the model instead of the phoneme string is 34.9%

6. Conclusions

In this paper, two related spoken language-oriented projects have been described, each dealing with the issue of integrating linguistic information from multiple sources. First, the work with developing a multi-purpose central lexicon database including phonemic representations was described. The central ideas behind this project are central availability and incremental development. Tools for facilitating continuous and simultaneous lexicon development have been created.

Second, a project aimed at modelling phone-level pronunciation in discourse context was presented. A data-driven approach was taken for this task and the work involved annotating spoken language with linguistic and related information ranging from the discourse level down to articulatory feature level. Annotation was structured in six layers: 1) a discourse layer, 2) an utterance layer, 3) a phrase layer, 4) a word layer, 5) a syllable layer and 6) a phoneme layer. The layers were segmented into their specific unit types and linguistic information was attached to each unit at each level.

The resulting annotation was used for machine learning of models describing variation in phoneme realisation. Using the phoneme as the primary unit, a set of training instances, essentially being context-sensitive phonemes, were created. Each instance contained information about the current phoneme and about the current unit in all annotation layers above. Instances also contained information about the sequential context of the current unit in each layer.

In the evaluation of models created from the multi-layer linguistic annotation, emphasis was put on the effects of adding information of different types to the training data in addition to phoneme-level variables. It was shown that including information from multiple layers improves model performance, most notably for spontaneous speech, where the predictive power of phonological and grammatical information is relatively low.

Attributes from all layers of annotation were used in the models with the highest prediction accuracy and as many as 470 out of 516 available attributes were actually used by at least one of the models (optimally pruned decision trees) in a tenfold cross validation experiment. The optimal models produced an average phone error rate of 8.1%, which is an improvement with 60.2% compared to using the phoneme string for estimating phone-level realisation. A comparison between models trained only on phone layer attributes and models trained on attributes from all layers showed that the prediction accuracy could be improved by 42.6% by adding attributes for units above the phoneme layer.

The classification keys used at model training were generated by an automatic transcription system with access to the speech signal. Evaluated against gold standard transcriptions, the models produced a phone error rate of 16.9%. This means that the deterioration in performance when using the model instead of the automatc transcription system is only 8.5% and that the improvement using the model instead of a phoneme string from a lexicon is 34.9%.

Acknowledgements

The research reported in this paper is carried out at the Centre for Speech Technology, a competence centre at KTH, supported by VINNOVA (the Swedish Agency for Innovation Systems), KTH and participating Swedish companies and organisations.

7. References

J. Allwood, L. Grönqvist, E. Ahlsén, and M. Gunnarsson. 2002. Göteborgskorpusen för talspråk (The Göteborg spoken language corpus). In *Nydanske Sprogstudier 30*, pages 39–58. København: Akademisk Forlag.

R. Bannert and P. E. Czigler. 1999. *Variations in consonant clusters in standard Swedish*. Phonum 7, Reports in Phonetics. Umeå: Umeå University.

C. Borgelt. 2004. Dtree. http://fuzzy.cs.uni-magdeburg.de/∼borgelt/dtree.html.

G. Bruce. 1986. Elliptical phonology. In *Papers from the Scandinavian Conference on Linguistics*, pages 86–95.

M. Finke and A. Waibel. 1997. Speaking mode dependent pronunciation modeling in large vocabulary conversational speech recognition. In *Proceedings of Eurospeech*, pages 2379–2382.

E. Fosler-Lussier and N. Morgan. 1999. Effects of speaking rate and word frequency on pronunciations in conversational speech. *Speech Communication*, 29(2–4):137–158.

E. Gårding. 1974. Sandhiregler för svenska konsonanter (Sandhi rules for Swedish consonants). In *Svenskans beskrivning 8*, pages 97–106.

P. A. Jande. 2003. Phonological reduction in Swedish. In *Proceedings of ICPhS*, pages 2557–2560.

P. A. Jande. 2005. Inducing decision tree pronunciation variation models from annotated speech data. In *Proceedings of Interspeech*, pages 1945–1948.

D. Jurafsky, A. Bell, M. Gregory, and W. Raymond. 2001a. Probabilistic relations between words: Evidence from reduction in lexical production. In J. Bybee and P. Hopper, editors, *Frequency and the emergence of linguistic structure*, pages 229–254. Amsterdam: John Benjamins.

D. Jurafsky, A. Bell, M. L. Gregory, and W. D. Raymond. 2001b. The effect of language model probability on pronunciation reduction. In *Proceedings of ICASSP*, volume 2, pages 2118–2121.

R. Lopez de Mantaras. 1991. A distance-based attribute selection measure for decision tree induction. *Machine Learning*, 6(1):81–92.

B. Megyesi. 2002. Shallow parsing with PoS taggers and linguistic features. *Journal of Machine Learning Research*, 2:639–668.

K. Sjölander and J. Beskow. 2000. WaveSurfer – a public domain speech tool. In *Proceedings of ICSLP*, pages 464–467.

K. Sjölander. 2003. An HMM-based system for automatic segmentation and alignment of speech. In *Proceedings of Fonetik*, pages 93–96.

C. P. J. Van Bael, H. van den Heuvel, and H. Strik. 2004. Investigating speech style specific pronunciation variation in large spoken language corpora. In *Proceedings of ICSLP*, pages 586–589.

Argument Structure in TimeML

James Pustejovsky, Jessica Littman, Roser Saurí

Computer Science Department, Brandeis University
415 South Street
Waltham, MA 02454
{jamesp, jlittman, roser}@cs.brandeis.edu

Abstract

TimeML is a specification language for the annotation of events and temporal expressions in natural language text. In addition, the language introduces three relational tags linking temporal objects and events to one another. These links impose both aspectual and temporal ordering over time objects, as well as mark up subordination contexts introduced by modality, evidentiality, and factivity. Given the richness of this specification, the TimeML working group decided not to include the arguments of events within the language specification itself. Full reasoning and inference over natural language texts clearly requires knowledge of events along with their participants. In this paper, we define the appropriate role of argumenthood within event markup and propose that TimeML should make a basic distinction between arguments that are events and those that are entities. We first review how TimeML treats event arguments in subordinating and aspectual contexts, creating event-event relations between predicate and argument. As it turns out, these constructions cover a large number of the argument types selected for by event predicates. We suggest that TimeML be enriched to include certain temporally-related discourse relations, such as those that involve causal predicates, since these also involve event-event relations. We propose that all other verbal arguments be ignored by the specification, and any predicate-argument binding of participants to an event should be performed by independent means. In fact, except for the event-denoting arguments handled by the extension to TimeML proposed here, almost full temporal ordering of the events in a text can be computed without argument identification.

1. Introduction

TimeML, an annotation scheme for capturing temporal information, is used to create a temporal abstraction of what is presented in natural language text. The result both anchors and orders events with respect to time. This tells us what is happening when but not who is involved. To answer the question *Who does what when?* we must extend the annotation to include arguments of events such as entities or other events.

However, the goal of a TimeML annotation remains the capture of temporal information and entities are not temporal objects. In addition, there have already been great strides in the area of entity recognition. Rather than recreating that work and, thereby, extending the TimeML specification far beyond what it was desgined for, we propose a much smaller extension to TimeML that will allow for the merging of other annotation schemes that already specialize in this area.

In this paper, we begin with a brief overview of what TimeML currently is. Section 3. describes a way in which TimeML already deals with eventive arguments. In section 4., we describe another proposed extension to TimeML, the discourse link, which can also take advantage of other annotations such as the one provided in GraphBank. Finally, in section 5., we present the new `ARGLINK` tag and describe some ways in which linguistic information from other sources can be used.[1]

[1]This paper originally appeared in the Dagstuhl Seminar Proceedings (Pustejovsky et al., 2006). It has been updated for this workshop. The original text can be found at http://drops.dagstuhl.de/opus/volltexte/2006/449/.

2. Overview of the Current TimeML Specification

The TimeML specification language provides a standard for capturing all temporal information in a natural language text. This includes temporal expressions, events, and the relationships they share. To achieve such an annotation, TimeML uses four main tag types that fall into two categories, those that consume text and those that do not. `TIMEX3`, `SIGNAL`, and `EVENT` fall into the former group. The non-consuming tags are primarily of the `LINK` type with the exception of `MAKEINSTANCE`, a tag which completes the annotation of events. In the subsections that follow, we briefly describe each of these tags.

2.1. Temporal Expressions

TimeML expands on earlier attempts to annotate temporal expressions (Mani and Wilson, 2000; Schilder and Habel, 2001), with the introduction of the `TIMEX3` tag. Specifically, `TIMEX3` adds functionality to the TIMEX2 standard (Ferro et al., 2001).

Temporal expressions in TimeML fall into four categories: DATEs, TIMEs, DURATIONs, and SETs. A DATE is any calendar expression such as *July 3* or *February, 2005*. The annotation of such examples includes a `value` attribute that specifies the contents of the expression using the ISO 8601 standard. The example in (1) shows the annotation of a fully specified DATE `TIMEX3`.

(1) a. April 7, 1980

b.
```
<TIMEX3 tid="t1" type="DATE"
value="1980-04-07"
temporalFunction="false">
April 7, 1980
</TIMEX3>
```

April 7, 1980 is a fully specified temporal expression because it includes all of the information needed to give its value. Many temporal expressions are not fully specified and require additional information from other temporal expressions to provide their full `value`. We will say more about the annotation of these expressions shortly, but, for now, notice that the annotation in (1) includes an attribute called `temporalFunction` and that it is set to "false". When a temporal expression requires more information to complete its annotation, this attribute is set to "true" to indicate that a temporal function will be used. For more on this process, refer to the section below on temporal functions.
While the DATE type is used to annotate most calendar expressions, the TIME type is used to capture expressions whose granularity is smaller than one day. Examples of this include *4:20* and *this morning*. Example (2) shows the annotation of a fully specified TIME `TIMEX3`. Notice that for a TIME to be fully specified, it must include date information as well.

(2) a. 10:30am April 7, 1980

b.
```
<TIMEX3 tid="t1" type="TIME"
value="1980-04-07T10:30"
temporalFunction="false">
10:30am April 7, 1980
</TIMEX3>
```

Expressions such as *for three months* include a DURATION `TIMEX3`. The `value` attribute of a DURATION again follows the ISO 8601 standard. For example, *three months* receives a `value` of "P3M". Occasionally, a DURATION will appear anchored to another temporal expression. Since TimeML strives to annotate as much temporal information as possible, this information is also included in the annotation of a DURATION with the `beginPoint` and `endPoint` attributes as shown in (3).

(3) a. two weeks from December 17, 2005

b.
```
<TIMEX3 tid="t1" type="DURATION"
value="P2W"
beginPoint="t2" endPoint="t3">two
weeks</TIMEX3>
```

c.
```
<TIMEX3 tid="t2" type="DATE"
value="2005-12-17">
December 17, 2005</TIMEX3>
```

d.
```
<TIMEX3="t3" type="DATE"
value="2005-12-31"
temporalFunction="TRUE"
anchorTimeID="t1"/>
```

The example in (3a) contains to temporal expressions separated by a signal (see subsection 2.3.). The first, *two weeks*, is annotated as a DURATION. The second, *December 17, 2005*, is a fully specified DATE. Every `TIMEX3` annotation includes an identification number. This number is used to relate the temporal expression to other TimeML objects. In this case, the identification value in (3c), "t2", is included in the annotation of *two weeks* as the `beginPoint` of the duration. With this information, the `endPoint` of the duration can be calculated. An additional `TIMEX3` is created to hold its value. This is the `TIMEX3` given in (3d). Since the value of the new `TIMEX3` must be calculated, `temporalFunction` is set to "true" and a temporal anchor is suppled. This new attribute will be explained below. The final type of `TIMEX3` is used to capture regularly recurring temporal expressions such as *every three days*. This type, SET, uses the attributes `quant` and `freq` to annotated quantifiers in an expression and the frequency of the expression, respectively. An example is given in (4).

(4) a. two days every week

b.
```
<TIMEX3 tid="t1" type="SET"
value="P2D" quant="EVERY"
freq="1W">two days every
week</TIMEX3>
```

2.1.1. Temporal Functions

When a temporal expression is not fully specified, it requires the use of a temporal function to calculate its value. In a manual annotation, the user provides a particular anchor time ID that supplies the missing information. The user then gives the correctly calculated value for the `TIMEX3`. In automatic annotation, a library of temporal functions is used to perform the calculation.
The example in (3d) shows an annotation that uses a temporal function. In this case, the end point of a duration was calculated using the `beginPoint` and `value` of the duration given in (3b). For the new temporal expression in (3d), the `temporalFunction` attribute is set to "TRUE" and the `tid` for the duration is given as the `anchorTimeID`. Finally, the correct `value` is supplied. This same process is used for temporal expressions that are missing information such as *April 7*, which is missing the year, and for relative temporal expressions such as *today*.

2.2. Events

Events that can be anchored or ordered in time are captured with TimeML. Such events are predominantly verbs, but nouns, adjectives, and even some prepositions can also be eventive. The annotation of TimeML events is a two part process. First, they are tagged with the `EVENT` tag. This tag has two attributes: an ID number and an event class. The classification of an event can help determine what relationships that event may participate in. For example, an event classified as REPORTING will be the first element of an evidential `SLINK` (see the subsection on Subordinating Links in section 2.4.). There are seven event classes:

- REPORTING: *say, report, tell*
- PERCEPTION: *see, watch, hear*
- ASPECTUAL: *initiate, terminate, continue*
- I_ACTION: *try, investigate, promise*
- I_STATE: *believe, want, worry*
- STATE: *on board, live, seek*

- OCCURRENCE: *land, eruption, arrive*

Several of these classes introduce an event argument and are of particular interest to the work in this paper. The TimeML Annotation Guidelines (Saurí et al., 2005b) detail exactly which events fall into which classes.

2.2.1. Instances of Events

Besides the classification of an event, natural language documents supply much more information about events that we need to represent in an accurate annotation. In addition to the head of the event that is captured in the text, an event may include further tense and aspect indicators or modifiers that affect its modality or polarity. This information is captured with `MAKEINSTANCE`, a non-consuming timeML tag. Every event in TimeML has at least one instance annotated with this tag. A separate tag is used because one mention of an event in text can actually refer to multiple instances, as in example (5).

(5) John swims on Monday and Tuesday.

Here, there is one mention of *swim* that is tagged as an OCCURRENCE `EVENT`. TimeML will try to link this event to the temporal expressions also present in the sentence. However, it is clear that the *swim* event that takes place on *Monday* is not the same one that takes place on *Tuesday*. Instead, it is an instance of the event that is anchored to each temporal expression.
Instances of events can also have different tense, aspect, polarity, or modality properties. Again, this information is captured with the `MAKEINSTANCE` tag. Once an event has an instance annotated, that instance is elligible to take part in a `LINK` tag to show what relationship it has with other temporal objects (refer to section 2.4.).

2.3. Signals

When temporal objects are related to each other, there is often an additional word present whose function is to specify the nature of that relationship. These words are captured with the `SIGNAL` tag, which has one attribute that provides an identification number. Example (6) shows a typical use of preposition *at* as `SIGNAL`, and a complete annotation of all the temporal objects present.

(6) a. The bus departs at 3:10 pm.

b.
```
The bus
<EVENT eid="e1" class="OCCURRENCE">
departs
</EVENT>
<MAKEINSTANCE eiid="ei1"
eventID="e1" pos="VERB"
tense="PRESENT" aspect="NONE"
polarity="POS"/>
<SIGNAL sid="s1">
at
</SIGNAL>
<TIMEX3 tid="t1" type="TIME"
value="XXXX-XX-XXT15:10">
3:10pm
</TIMEX3>
```

2.4. Links

TimeML uses three varieties of `LINK` tag to represent relationships among temporal objects. In all cases, the `LINK` tag is non-consuming as there may not be any explicit text to capture or the relationship could be between objects whose locations vary greatly. Each link tag comes with a set of relation types to specify the nature of the relationship. In the following paragraphs, we briefly describe each of these tags: `TLINK`, `ALINK`, and `SLINK`.

2.4.1. Temporal Relationships

All temporal relationships are represented with the `TLINK` tag. `TLINK` can be used to annotate relationships between times, between events, or between times and events. In this way, TimeML can both anchor and order temporal objects. A `signalID` can also be used in a `TLINK` if it helps to define the relationship. The `TLINK` in example (7) completes the annotation of *The bus departs at 3:10pm.*

(7)
```
<TLINK lid="l1" eventInstanceID="ei1"
relatedToTime="t1"
signalID="s1" relType="IS_INCLUDED"/>
```

The possible `relType` values for a `TLINK` are based on Allen's thirteen relations (Allen, 1984). `TLINK` is also used to assert that two event instances refer to the same event using the IDENTITY `relType`.

2.4.2. Aspectual Links

Events classified as ASPECTUAL introduce an `ALINK`. The `ALINK` represents the relationship between an aspectual event and its argument event. This is an example of one way that TimeML already deals with event arguments.

2.4.3. Subordinating Links

As mentioned in section 2.2., certain event classes introduce a subordinated event argument. Some examples are verbs like *claim, suggest, promise, offer, avoid, try, delay, think*; nouns like *promise, hope, love, request*; and adjectives such as *ready, eager, able, afraid.* In the following sentences, the events selecting for an argument of situation or proposition type appear in bold face, whereas the corresponding argument is underlined:

(8) a. The Human Rights Committee **regretted** that discrimination against women <u>persisted</u> in practice.

b. Uri Lubrani also **suggested** Israel was **<u>willing</u>** to <u>withdraw</u> from southern Lebanon.

c. Kidnappers **kept** their **<u>promise</u>** to <u>kill</u> a store owner they took hostage.

In TimeML, subordination relations between two events are represented by means of a Subordinating Links (or `SLINK`s). The `SLINK` tag is perhaps the best example of the current treatment of arguments in TimeML. Reference to each event is expressed by a pointer to them (through the attributes `eventInstanceID` and `subordinatedEventInstance`), and the relation type is conveyed by means of the attribute `relType`, which captures the type of modality projected in each case onto the event denoted by the subordinated clause. `relType` can be any of the following types:

1. FACTIVE: When the argument event is entailed or presupposed. Here is an annotated example:[2]

(9) a. The Human Rights Commitee regretted that discrimination against women persisted in practice.

b.
```
The Human Rights Committee
<EVENT eID="e1" class="I_ACTION">
regretted
</EVENT>
that discrimination against
women
<EVENT eID="e2"
class="ASPECTUAL">
persisted
</EVENT>
in practice.
<SLINK eventInstanceID="e1"
subordinatedEventInstance="e2"
relType="FACTIVE"/>
```

2. COUNTERFACTIVE: When the main predicate presupposes the non-veracity of its argument:

(10) a. A Time magazine reporter avoided jail at the last minute...

b.
```
A Time magazine reporter
<EVENT eID="e1" class="I_ACTION">
avoided
</EVENT>
<EVENT eID="e2" class="STATE">
jail
</EVENT> at the last minute...
<SLINK eventInstanceID="e1"
subordinatedEventInstance="e2"
relType="COUNTERFACTIVE"/>
```

3. EVIDENTIAL: Typically introduced by REPORTING or PERCEPTION events, such as *tell, say, report* and *see, hear*, respectively.

4. NEGATIVE_EVIDENTIAL: Introduced by REPORTING and PERCEPTION events conveying negative polarity; e.g., *deny*.

5. MODAL: For annotating events introducing a reference to possible world.

(11) a. Uri Lubrani also suggested Israel was willing to withdraw from southern Lebanon.

b.
```
Uri Lubrani also
<EVENT eID="e1" class="I_ACTION">
suggested
</EVENT>
Israel was
<EVENT eID="e2" class="I_STATE">
willing
</EVENT>
to
<EVENT eID="e3"
class="OCCURRENCE">
withdraw
</EVENT>
from southern Lebanon.
<SLINK eventInstanceID="e1"
subordinatedEventInstance="e2"
relType="MODAL"/>
<SLINK eventInstanceID="e2"
subordinatedEventInstance="e3"
relType="MODAL"/>
```

[2]For the sake of simplicity, in this and the following examples we obviate the annotation of MAKEINSTANCE tags.

The following section goes into the detail of how SLINKs account for some arguments.

3. Events and their Participants

We will assume for our discussion that events can be represented as first order individuals, existentially quantified in a neo-Davidsonian manner where participants to the event are conjoined relations between individuals and the event (Davidson, 1967; Parsons, 1990). For each event, e, we will identify the participants to this event with a three-place relation, Arg:

(12) $\lambda k\colon \text{int}\lambda x\colon \text{ind}\lambda e\colon \text{event}[Arg(k,e,x)]$

Rather than labeling arguments with specific named semantic functions, such as agent, patient, and instrument, we identify the argument by an index, k. The idea is that a post-parsing procedure will identify the appropriate semantic role played by an argument.
Both named entity arguments and event arguments are expressible in this fashion. For example, for the sentence in (13a), the participants are directly identified by their indices 1 and 2, respectively, but not functionally, as *Agent* and *Patient*.

(13) a. John kissed Mary.

b. $\exists e[kiss(e) \wedge Arg(1,e,j) \wedge Arg(2,e,m)]$

Notice that the current TimeML representation of (13a) identifies the event predicate but not its arguments.

(14)
```
John
<EVENT eid="e1" class="OCCURRENCE">
kissed
</EVENT>
<MAKEINSTANCE eiid="ei1" eventID="e1"
pos="VERB"
tense="PAST" aspect="NONE"
polarity="POS"/>
Mary.
```

With the addition into TimeML of an Arg-relation, we would be able to identify the entity participants as represented in (13b) above. This should be done cautiously, however, without complicating the specification language or making the annotation task more difficult than it already is. We will take up this issue in Section 5 below.
By design, TimeML treats predicates that select for event arguments differently from those taking named entities. For

example, the event-embedding predicate *see*, in most cases, allows the same simple conjunctive representation over arguments that we saw in (13b), assuming the argument is extensional.[3]

(15) a. John saw Mary fall.

b. $\exists e_1 \exists e_2[see(e_1) \wedge Arg(1, e_1, j) \wedge Arg(2, e_1, e_2) \wedge fall(e_2) \wedge Arg(1, e_2, m)]$

In the next section, we turn to the question of how to generalize the encoding of an event argument as expressed in TimeML through SLINKs.

3.1. SLINK Encodes Partial Argument Structure

According to the TimeML specification, predicates in natural language that are encoded as introducing SLINKs in fact already identify the embedded complement as an argument to the verb.
For example, the TimeML markup of (16a) explicitly identifies the embedded complement (verb) as a subordinated argument to the event *regret*.

(16) a. John regretted that Sue marrried Bill.

b.
```
John
<EVENT eID="e1" class="I_ACTION">
regretted
</EVENT>
that Sue
<EVENT eID="e2" class="OCCURRENCE">
married
</EVENT>
Bill.
<SLINK eventID="e1" subEventID="e2"
relType="FACTIVE"/>
```

As it happens, with a factive predicate such as *regret* we can existentially quantify the event representing the embedded complement of the SLINK predicate. A first-order neo-Davidsonian representation of this sentence would, therefore, look like the following:

(17) $\exists e_1 \exists e_2[regret(e_1) \wedge Arg(1, e_1, j) \wedge Arg(2, e_1, e_2) \wedge marry(e_2) \wedge Arg(1, e_2, s) \wedge Arg(2, e_2, b)]$

The current TimeML representation of this sentence, however, expressed as a first-order expression, is closer to that shown in (18), since no entity arguments are represented in TimeML.

(18) $\exists e_1 \exists e_2[regret(e_1) \wedge Arg(2, e_1, e_2) \wedge marry(e_2)]$

For all other modality-introducing predicates, TimeML is generally descriptively adequate in differentiating the modal force of the complement expression. For example, the SLINK predicate *believe* is annotated as (19b) below.

(19) a. John believes that Bill went to Japan.

b.
```
Mary
<EVENT eID="e1" class="I_ACTION">
believes
</EVENT>
that Bill
<EVENT eID="e2" class="OCCURRENCE">
went
</EVENT>
to Japan.
<SLINK eventID="e1" subEventID="e2"
relType="MODAL"/>
```

The modal subordination introduced by the propositional attitude predicate *believe* is represented by an SLINK with a `relType` value of MODAL. To model this, we will introduce a special first order variable, $\hat{e}$, effectively encoding the modality of the event and the domain of its subordination. On this strategy, a first order expression representing the partial argument structure of (19b) would be that shown in (20).[4]

(20) $\exists e \exists \hat{e}[believe(e) \wedge Arg(2, e, \hat{e}) \wedge go(\hat{e})]$

4. Encoding Discourse Relationships in TimeML

One aspect of adding more arguments to TimeML is to capture a limited number of temporally-related discourse relations. This is done with a new kind of link, `DLINK`, with relationship types such as narrative and causal. Here, as an example, we focus on causal `DLINK`s, which involve the class of predicates that introduce causal relations between events explicitly in their lexical semantics.
The representation of causation between event denoting expressions within the same sentence is common in natural languages. For example, the following sentences express causal (and hence temporal) relations between events, which are largely ignored in TimeML.

(21) a. [The rain]$_{e1}$ caused [the flooding]$_{e2}$.
b. [The rioting]$_{e1}$ led to [curfews]$_{e2}$.
c. [Fifty years of peace]$_{e1}$ brought about [great prosperity]$_{e2}$.

To capture this relation, we make use of the new `DLINK` in order to express the causal relation between two events.

```
<DLINK>
attributes ::= [lid] [origin]
     [eventInstanceID] signalID
     relatedToEventInstance relType
lid ::= ID
{lid ::= LinkID
LinkID ::= l<integer>}
origin ::= CDATA
eventInstanceID ::= IDREF
{eventInstanceID ::= EventInstanceID}
relatedToEventInstance ::= IDREF
{relatedToEventInstance ::=
```

[3]We assume that the typing on the Arg relation can be generalized to allow events as arguments.

[4]This is similar to the first order representations in DAML for modal subordination (Jerry Hobbs (p.c); cf. http://www.daml.org/ontologies/.

```
        EventInstanceID}
    signalID ::= IDREF
    {signalID ::= SignalID | EventInstanceID}
    relType ::= 'CAUSES'|'NARRATIVE'
```

This solution can be adopted for the following verbs, in their causative senses: *cause, stem from, lead to, breed, engender, hatch, induce, occasion, produce, bring about, produce, secure*.
Now, a sentence such as (22a) can be explicitly annotated as involving a causal relation, as follows:

(22) a. The rioting led to curfews on November 22, 2004.

b.
```
The
<EVENT eid="e1" class="OCCURRENCE">
rioting
</EVENT>
<MAKEINSTANCE eiid="ei1"
eventID="e2" tense="NONE"
aspect="NONE"/>
<EVENT eid="e2" class="CAUSE">
led
</EVENT>
<MAKEINSTANCE eiid="ei2"
eventID="e2" tense="PAST"
aspect="NONE"/>
to
<EVENT eid="e3" class="OCCURRENCE">
curfews
</EVENT>
<MAKEINSTANCE eiid="ei3"
eventID="e2" tense="NONE"
aspect="NONE"/>
on
<TIMEX3 tid="t1" type="DATE
value="2004-11-22">
November 22, 2004
</TIMEX3>.

<DLINK eventInstanceID="ei1"
relatedToEventInstance="ei3"
relType="CAUSES" signalID="ei2"/>
<TLINK eventInstanceID="ei3"
relatedToTime="t1"
reltype="IS_INCLUDED"/>
```

Note that both the subject and object event expressions are syntactically arguments to the causal predicate. In this case, the Arg relation is not operative since the matrix predicate is itself a realization of a $Cause$ relation directly:

(23) a. The rioting led to curfews.

b. $\exists e_1 \exists e_2[rioting(e_1) \wedge Cause(e_1, e_2) \wedge curfews(e_2)]$

It is important to note that there are many cases where causation, expressed through explicit causative predicates such as those mentioned above, is not syntactically a relation between two events, but a relation between an individual and an event. Consider the sentences below:

(24) a. [John]$_x$ caused [a fire]$_{e2}$.
b. [The drug]$_x$ induced [a seizure]$_{e2}$.

In such cases of event metonymy (Pustejovsky, 1989; Pustejovsky, 1995), we will introduce a skolemized event instance, $ei1$, to act as the proxy in the causation relation. Hence, the TimeML for (24a) would be as follows:

(25)
```
John
<MAKEINSTANCE eiid="ei1" eventID="NONE"
tense="NONE"
aspect="NONE"/>
<EVENT eid="e2" class="CAUSE">
caused
</EVENT>
<MAKEINSTANCE eiid="ei2" eventID="e2"
tense="PAST"
aspect="NONE"/>
a
<EVENT eid="e3" class="OCCURRENCE">
fire
</EVENT>
<MAKEINSTANCE eiid="ei3" eventID="e3"
tense="NONE"
aspect="NONE"/>

<DLINK eventInstanceID="ei1"
relatedToEventInstance="ei3"
relType="CAUSES" signalID="ei2"/>
```

The interpretation of *John* as the agent of an event involved in the causation is out of the scope of TimeML; it would be the responsibility of subsequent semantic interpretation to bind the entity *John* to the causing event.

5. Binding Entity Arguments in TimeML

In this section, we propose an extension to the current specification of TimeML to accommodate the treatment of entity arguments. Our goal is to avoid any explicit mention of entities within the TimeML markup. There are two reasons for this move: first, entity arguments are not temporally sensitive text extents, unlike event-denoting predicates and temporal expressions; secondly, we wish to avoid complicating the TimeML specification. Therefore, our strategy will be to accomplish the argument binding independent of the event tag itself. Currently, the EVENT tag is defined as follows:

```
<EVENT>
attributes ::= eid class
eid ::= ID
{eid ::= EventID
EventID ::= e<integer>}
class ::= 'OCCURRENCE' | 'PERCEPTION'
      | 'REPORTING' | 'ASPECTUAL'
      | 'STATE' | 'I_STATE' | 'I_ACTION'
```

On our approach, this need not change. Rather than add an argument list to the event —similar to the SUBCAT list in HPSG (Pollard and Sag, 1994)— we will treat the binding of particpants to events in a parallel fashion to the treatment of event ordering; by introducing a new linking relation, called ARGLINK. This will encode, in TimeML, the binding accomplished by the Arg relation defined in (12) above.

```
<ARGLINK>
attributes ::= alid [origin]
       eventInstanceID ArgID
alid ::= ID
{alid ::= ArgLinkID
ArgLinkID ::= al<integer>}
origin ::= CDATA
eventInstanceID ::= IDREF
{eventInstanceID ::= EventInstanceID}
ArgID ::= IDREF
{ArgID ::= EntityID}
```

Now let us see how the two participants in sentence (26),

(26) John kissed Mary.

can be represented, using the ARGLINK tag. Recall that the desired logical form for this sentence is:

(27) $\exists e[kiss(e) \wedge Arg(1, e, j) \wedge Arg(2, e, m)]$

Assuming that the named entities in (26) have been identified and indexed, we can express the bindings shown in (27) as the two ARGLINKs below:

(28)
```
John (ai1)
<EVENT eid="e1" class="OCCURRENCE">
kissed
</EVENT>
<MAKEINSTANCE eiid="ei1" eventID="e1"
pos="VERB"
tense="PAST" aspect="NONE"
polarity="POS"/>
Mary (ai2).
<ARGLINK alid="al1"
eventInstanceID="ei1" ArgID ="ai1"/>
<ARGLINK alid="al2"
eventInstanceID="ei1" ArgID ="ai2"/>
```

The actual annotation and indexing of entity arguments can be carried out with the help of a robust parser such as RASP (Briscoe and Carroll, 2002). Within the TimeML annotation, `ARGLINK` will simply declare that a particular entity is a participant in a given event instance. The semantic role of arguments (e.g. who is the kisser and who is being kissed) is left underspecified, but it can be later identified along the lines of current research on semantic role labeling (Gildea and Jurafsky, 2002; Gildea and Palmer, 2002; Carreras and Màrquez, 2005); that is, based on knowledge induced from corpora annotated for argument structure, such as FrameNet (Fillmore et al., 2001) or PropBank (Palmer et al., 2005), along with some extensions to Evita (Saurí et al., 2005a), our TimeML compliant, open-domain event recognizer, so that it identifies sentence voice (active or passive). The end result will include both information on what entities participate in what events, and what are their semantic roles.

6. Conclusion

In this paper, we discussed the role of arguments in TimeML, an event annotation specification language, and explored possible ways to account for event argument structure without resulting in an overloaded annotation. We first described how TimeML handles event arguments in subordinating and aspectual contexts, where SLINKs and ALINKs create event-event relations between a predicate and an event-denoting argument. We proposed that TimeML be enriched slightly to include discourse relations such as causal predicates (DLINK), since these also involve event-event relations. Finally, we enriched the TimeML specification by introducing ARGLINK, a linking mechanism allowing entities to be identified with the event they participate in. This resource connects TimeML annotation, which handles event and temporal information, with argument structure-annotated resources such as PropBank and FrameNet, and consequently allows avoiding argument tagging as part of the TimeML spec, a move that would overload the annotation effort. Using components from other annotation schemes that account for event argument structure is therefore an optimal alternative in order to enrich the description of events in TimeML.

7. References

James Allen. 1984. Towards a general theory of action and time. *Artificial Intelligence*, 23:123–154.

Ted Briscoe and John Carroll. 2002. Robust accurate statistical annotation of general text. In *Proceedings of the Third International Conference on Language Resources and Evaluation*, pages 1499–1504.

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the conll-2005 shared task: Semantic role labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pages 152–164.

Donald Davidson. 1967. The logical form of action sentences. In *The Logic of Decision and Action*.

Lisa Ferro, Inderjeet Mani, Beth Sundheim, and George Wilson. 2001. Tides temporal annotation guidelines. Technical Report Version 1.0.2, MITRE Technical Report. MTR 01W0000041.

Charles J. Fillmore, Charles Wooters, and Collin F. Baker. 2001. Building a large lexical databank which provides deep semantics. In *Proceedings of the Pacific Asian Conference on Language, Informa tion and Computation*.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288.

Daniel Gildea and Martha Palmer. 2002. The necessity of parsing for predicate argument recognition. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 239–246, Philadelphia, PA.

Inderjeet Mani and George Wilson. 2000. Robust temporal processing of news. In *Proceedings of the 38th Annual Meeting of the ACL (AC L2000)*, pages 69–76, New Brunswick, New Jersey.

Martha Palmer, Dan Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1).

Terence Parsons. 1990. *Events in the Semantics of English*. MIT Press, Cambridge, MA.

Carl Pollard and Ivan Sag. 1994. *Head-Driven Phrase Structure Grammar*. CSLI, Stanford, CA.

James Pustejovsky, Jessica Littman, and Roser Saurí. 2006. Argument structure in timeml. In Graham Katz, James Pustejovsky, and Frank Schilder, editors, *Dagstuhl Seminar Proceedings*, Germany. Internationales Begegnungs- und Forschungszentrum (IBFI), Schloss Dagstuhl. http://drops.dagstuhl.de/opus/volltexte/2006/449/.

James Pustejovsky. 1989. Current issues in computational lexical semantics. In *ACL89*, pages xvii–xxv.

James Pustejovsky. 1995. *The Generative Lexicon*. MIT Press, Cambridge, MA.

Roser Saurí, Robert Knippen, Marc Verhagen, and James Pustejovsky. 2005a. EvITA: A robust event recognizer for QA systems. In *Proceedings of the HLT/EMNLP 2005*.

Roser Saurí, Jessica Littman, Robert Knippen, Rob Gaizauskas, Andrea Setzer, and James Pustejovsky, 2005b. *TimeML Annotation Guidelines*. http://www.timeml.org.

Frank Schilder and Christopher Habel. 2001. From temporal expressions to temporal information: Semantic tagging of news messages. In *Proceedings of the ACL-2001 Workshop on Temporal and Spatial Information Processing*, pages 65–72, Toulouse.

Dependency conversion and parsing of the BulTreeBank

Atanas Chanev*, Kiril Simov†, Petya Osenova†, Svetoslav Marinov‡

*Department of Cognitive Sciences, University of Trento
Italy, TN, 38068, Rovereto, via Matteo del Ben 5 &
Istituto Trentino di Cultura-irst
Italy, TN, 38050, Povo (Trento), via Sommarive 18
chanev@form.unitn.it
† Linguistic Modelling Laboratory, IPP, Bulgarian Academy of Sciences
Bulgaria, 1113, Sofia, Acad. G.Bonchev St. 25A
{kivs, petya}@bultreebank.org
‡School of Humanities and Informatics
University College Skövde &
Göteborg University
Graduate School of Language Technology
Sweden, Göteborg, 40530, Box 200, Faculty of Arts
svetoslav.marinov@his.se

Abstract

Recently dependency parsing is gaining popularity. It is broadly accepted that dependency representations are more suitable for free word order languages. Statistical dependency parsers are easy to port from one language to another, if there are dependency treebanks for learning a grammar for the particular language. However, many treebanks are based on constituency and have to be converted to dependency representations prior to learning statistical dependency parsers. In this paper we investigate the issues of the conversion of the BulTreeBank (Simov et al., 2002) from Head-driven Phrase Structure Grammar (HPSG) format to dependency-based format and its parsing. We have performed three different conversions to three different dependency formats. For two of the conversions we used head tables and dependency tables which were stated explicitly, as in (Xia, 2001). For the other conversion the tables were implicitly implemented by rules. Our choice of rules for the tables was guided by decisions rooted in different linguistic theories. We have parsed the converted treebank with the Malt parser (Nivre et al., 2004) for 'evaluating' our conversions. Then we made error analysis to find advantages and pitfalls of each conversion strategy.

1. Introduction

Generally porting existing algorithms for statistical Natural Language Processing from language to language can be done with limited effort. Statistical dependency parsers are not an exception from the rule. Nevertheless, for training statistical methods we need language resources which are often annotated according to different linguistic theories and annotation schemes.

Most of the current NLP technologies were firstly developed for English and some of them were then ported to other languages. In parsing, state-of-the-art statistical parsers like those reported in (Collins, 1997) and (Charniak, 2000) were ported to Czech[1] (the porting of the Collins' parser was documented in (Collins et al., 1999)). Another porting of state-of-the-art parsers, this time from English to Italian, was described in (Corazza et al., 2004). In most of the cases parsers have to be adapted to the annotation scheme of the treebank for the new language.

The most famous treebank for parser evaluation for English is undoubtedly the Penn Treebank (Marcus et al., 1993). It is constituency-based but information about heads of the phrases can be found in the most recent version of the treebank. While information about heads seems to be very useful for learning and parsing English, this kind of information is crucial for free word order languages.

Constituency notion does not seem to be very convenient for free word order languages. It can be successfully substituted by fully dependency-based approach as in (Hajič, 1998), or richly extended with head information as in HPSG-based treebanks like the BulTreeBank (Simov et al., 2002). Pure constituency treebanks can be converted to dependency, if we want to benefit from dependency representations for free word order languages.

There are several studies for parsing Bulgarian. A shallow parsing module has been used in the annotation of the BulTreeBank. Chunks have been identified with a manually constructed grammar. We should also mention the work on constituency parsing within a larger system for text analysis for Bulgarian (Tanev, 2001) which was not evaluated on a treebank because there was not a broadly available treebank for Bulgarian at that time.

Another study (Krushkov and Chanev, 2005) reports full constituency parsing of simple sentences in Bulgarian with a grammar extracted from a small syntactically annotated collection of sentences. Evaluation was done manually and this makes the results biased and not reliable.

The first experiments on dependency parsing of Bulgarian were performed by (Marinov and Nivre, 2005). They report 84.2% unlabelled and 78.0% labelled precision on a limited subset of the BulTreeBank converted to dependency graphs. The conversion tables from that experiment were used in one of the experiments that this paper describes.

[1]The interested reader can find information about the performance of various parsers for Czech on: http://ufal.mff.cuni.cz/czech-parsing/

However, they were extended with more rules in order to cover the larger amount of data that we had.

Besides the head and dependency tables for Bulgarian that were first introduced in (Marinov and Nivre, 2005), another table and another conversion method are reported in this paper. All the methods for conversion that we used are evaluated on a small set of gold standard annotation. Finally an inductive dependency parser (Nivre et al., 2004), (Nivre, 2005) is used on the converted versions of our treebank to find out which conversion has been learned and parsed best.

The paper is structured in the following way: In Section 2. the treebank that we used in our experiments is presented. Section 3. discusses the 'head tables' that we have implemented. In section 4. we present our 'dependency tables' and argue about the shortcomings for each of the approaches that we followed. Section 5. reports error analysis of our conversions. In the next 2 sections (6. and 7.) we briefly present the parser that we use and show our parsing results. In section 8. we conclude our work.

2. The BulTreeBank Annotation Scheme

Treebank annotation is of great importance to a successful cross-theoretic portability. Since for different tasks and applications various types of information are needed, it is practical for a treebank to have not entirely constituency or dependency encodings, but rather some combination of both. They might be presented with different degrees of explicitness. It is important the appropriate information to be easily derivable. Having all this in mind, we pursued hybrid annotation in BulTreeBank. HPSG language model was explored. It views the linguistic data as a set of constituent structures with head-dependant markings.

Currently the treebank comprises 214000 tokens, a little more than 15000 sentences. Each token is annotated with elaborate morphosyntactic information. Additionally the Named Entities are annotated with ontological classes as `person`, `organization`, `location`, and `other`. The HPSG-based annotation scheme defines a number of phrase types which reflect both – the *constituent structure* and *the head-dependent* relation. Thus we have phrase labels with the explication of the dependent types like `VPC` (verbal head complement phrase), `VPS` (verbal head subject phrase), `VPA` (verbal head adjunct phrase), `NPA` (nominal head adjunct phrase) etc. We consider coordinations as non-headed phrases, where the grammatical function overrides the syntactic labels (Simov and Osenova, 2003). This fact causes problems if some head is always needed within the dependency relation. However, modelling coordination still remains one of the 'tough nuts' in all frameworks.

Behind the constituent structures and the head-dependent relations the treebank also represents phenomena like ellipsis, pro-dropness, word order, secondary predication, control. As an important mechanism for dealing with these phenomena we are using co-reference relations.

The treebank is in XML format, hence the restrictions over the language relations of dominance are encoded in a DTD. In most cases the head within phrases can be uniquely derived. For example, under the phrase VPC the head is the verb, while the complement is a nominal or a clause. Only in some combinations more specific rules are needed. For example, in NP phrases of the type NN. The head might be the former or the latter NP depending on the semantics of the phrase. In such cases manual annotation of the head is necessary.

3. Head Tables

We have performed three different conversions of the BulTreeBank from HPSG-based to dependency-based format. From now on we will refer to them in the following way: conversion 1 – the conversion of Svetoslav Marinov for the first ever experiments on dependency parsing of the BulTreeBank, with an extended head table by Atanas Chanev; conversion 2 – the conversion of Atanas Chanev and conversion 3 – the conversion of Kiril Simov and Petya Osenova for the CoNLL-X shared task[2].

In two of our conversions from constituency to dependency representation, head tables (Xia, 2001) were used to determine the head of each constituent. For conversion 3, rules for identification of the head were applied, then all non-head daughters were made to point to the head daughter of the constituent. Once the head of each phrase of the sentence is known, the conversion approach can vary from recursively top-down as in (Daum et al., 2004) to iteratively bottom-up as in all the conversions described in this study.

Conversion procedures from constituency-based to dependency-based representation can be traced back to (Gaifman, 1965). He showed that if one knew the head daughter of each constituent in a sentence, the unlabelled dependency graph of that sentence could be easily retrieved. Information about heads is kept in a table that is known in the literature as 'head table'. In addition to head tables, (Xia, 2001) introduced dependency tables which are needed for adding labels to the unlabelled dependency arcs of the sentence graph.

Besides in (Xia, 2001), conversions from constituency to dependency performed for English on the WSJ part of the Penn treebank have been reported in (Collins, 1997), (Yamada and Matsumoto, 2003) and (Nivre and Scholz, 2004). All these conversions benefit from a head table that consists of records containing the constituent that can have daughters, the direction of searching for the head constituent and a list of possible head constituents ordered by priority.

There are several studies in which German treebanks have been converted to dependency. Conversions have been reported in (Kübler and Telljohann, 2002) and (Ule and Kübler, 2004) for the TüBa-D treebank (Hinrichs et al., 2000). However, information about some dependencies is explicitly annotated in TüBa-D and only a few treebank specific issues have to be addressed for a successful conversion to dependency format.

The German NEGRA treebank (Skut et al., 1997) has been converted using the script DEPSY in (Daum et al., 2004). DEPSY is based on (Magerman, 1994) and implements a top-down recursive algorithm. However, the script can convert treebanks in only two formats: Penn Treebank and NEGRA treebank.

There are studies on conversion from constituency to the Prague Dependency Treebank (PDT) format. One of them

[2]http://nextens.uvt.nl/˜conll/

Table 1: An extract from the collection of rules used in conversion 3.

Rule	Head
AdvPA -> Adv Adv	Adv[2]
AdvPC -> Adv Adv	Adv[1]
NPA -> NPA NPA	NPA[1]
NPA -> (N NPA) (N NPA)	*[1]
NPA -> (N H) (H N)	*[1]
NPA -> N N	N[1]
NPA -> CoordP PP	CoordP
Nomin -> *	*

Table 2: An extract from the head table for conversion 1.

Constituent	Head daughter
AdvPC	<Adv> <Gerund>
Ako	<Ako> <C>
AkoP	<Ako> <C>
APA	<A> <Participle> <Pron> <Adv> <M> <Prep>
APC	<A> <Participle>
C	<C> <Prep>
CLCHE	<C>
CLDA	<T>

Table 3: An extract from the head table for conversion 2.

Constituent	Head daughter
AdvPC	<Adv> <AdvPA> <AdvPC> <Gerund> <CoordP>
Verbalised	<T> <I>
Subst	<Pron> <M> <A> <Participle>
APA	<A> <APA> <APC> <Participle> <CoordP> <Pron>
APC	<A> <APC> <APA> <Participle> <CoordP>
C	<C>
CLCHE	<V> <VPA> <VPS> <VPC> <VPF> <Participle> <CLDA> <CLCHE> <CoordP>
CLDA	<V> <VPA> <VPS> <VPC> <VPF> <CoordP>

is about the conversion of an English treebank (Žabokrtský and Kučerová, 2002). And another is about the conversion of an Arabic treebank (Žabokrtský and Smrž, 2003). The conversion algorithm used in these studies has been supplemented with a procedure for removing the traces from the treebanks.

In all the transformations mentioned above, except in the transformation of the TüBa-D treebank, the conversion has been performed similarly, usually in a recursive top-down fashion together with processing of treebank annotation specific constructions. Having a constituency treebank and a head table, if the mentioned algorithms are used, the resulting dependency treebank should always be the same, except in the case of the TüBa-D conversion where the whole process is strongly dependent on the treebank.

Two of our conversion methods (namely conversion 1 and 2) are very similar to the conversion method for the WSJ part of the Penn treebank. The difference in the head tables is that there is not an option for right to left search for the head among the daughters of the constituent. However, this is not a big disadvantage, because in most of the cases there is very little ambiguity which daughter to be the head. In conversion 3 the head table was substituted by rules which allow even more precise specifications for the choice of the head than the method described in (Xia, 2001). The 'head table' was encoded in 250 rules. Several constructions were converted by hand.

Our head tables have 44 records for conversion 1 and 38 records for conversion 2. The differences in the head tables for the two conversions are due to different treatment of several linguistic structures, e.g. clauses. Table 4 shows the percentage of the heads from each of the conversions (the rows) that were found in every conversion and the gold standard data (the columns).

The gold standard data (last column) that we annotated ourselves is used for evaluation. It consists of 60 sentences (976 tokens). For all the other columns the training part of the BulTreeBank was used. It consists of 10911 sentences (159394 tokens). All the punctuation marks were skipped in the evaluation.

In Table 1 we give a few rules that were used in conversion 3. The first column contains the rule from the grammar used in the annotation of the BulTreeBank and its left-hand side corresponds to the constituent whose head daughter should be selected. The constituents on the right hand side of the rule are its daughters among which the head should be chosen. Relying on the second column of the table the choice can be made.

For example, the head of the mother constituent given in the first record of Table 1 is the second Adv daughter constituent. Wildcards are used with the meaning 'no matter which constituent' and in some cases the meaning 'or no constituent' can be added. This approach is different from the approach of conversion 1 and 2 in its richer possibilities of specification which daughter to be the head of the constituent. The rule from the first row of Table 1 cannot be encoded using the head tables from conversions 1 and 2.

In addition to the rules from conversion 3 we give extracts from the head tables of conversion 1 (Table 2) and 2 (Table 3). Each rule from these tables starts with a mother constituent and then the possible head daughters are given ordered by priority.

All the records from the head tables from conversion 1 and 2 can be encoded with rules like those used in conversion 3. We can do that using rules of the type `<Const1> -> * <Const2> *` which are very common in conversion 3. A record from conversions 1 and 2 has the form `Const1 <Const2> <Const3> ...<ConstN>` meaning that `Const2` is the head

Table 4: Comparison of the different conversions to one another as well as on the gold standard data.

Conv.	1	2	3	Gold
1	100%	82.18%	69.06%	62.94%
2	82.18%	100%	79.42%	74.49%
3	69.06%	79.42%	100%	70.76%

daughter of `Conjst1` and if it is not present, then `Const3` is, etc. This record can be translated to the rules: `<Const1> -> * <Const2> *` – the head is `Const2`, `<Const1> -> * <Const3> *` – the head is `Const3`, ..., `<Const1> -> * <ConstN> *` – the head is `ConstN`.

If the rules of conversion 3 encode the same information as the head tables of conversion 1 or 2, the dependency arcs in the resulting dependency treebank will not differ from the arcs of the treebanks obtained by performing conversion 1 or 2 directly. With this we put an accent on the conversion table but not on the conversion algorithm. However, we will not prove here our assumption that the processing approach is irrelevant for a broader set of conversion algorithms, since it is beyond the scope of this paper.

The differences in our conversions are not due to the different conversion methods but dependent on the different sets of head rules. Undoubtedly there are head rules that treat the same linguistic constructions differently in conversions 1, 2 and 3. This indroduces different types of errors in the three converted dependency treebanks. We will discuss some of the interesting cases of erroneously converted graphs in Section 5.

4. Dependency labels

4.1. Three sets of dependency labels

There had been three sets of dependency labels that we used in the dependency tables in conversions 1 and 2 and in the rules in conversion 3. The labels are taken from a Swedish treebank (Nilsson et al., 2005) in conversion 1 and where possible from an Italian treebank (the Turin University Treebank – TUT) (Bosco, 2004) in conversion 2. The labels in conversion 3 had been chosen according to linguistic principles more than taken from another treebank.

The labels used in conversion 1 are 14: `ADV` (adverbial modifier), `APP` (apposition), `ATT` (attribute), `CC` (coordination), `DET` (determiner), `ID` (non-first element of multi-word expression), `IP` (punctuation), `OBJ` (object), `PR` (complement of preposition), `PRD` (predicative complement), `SUBJ` (subject), `UK` (head-verb of subordinate clause dependent on complementizer), `VC` (verb chain), `ROOT` (dependent of a special root node). The labels used in conversion 1 were adapted from Swedish to Bulgarian without significant effort in (Marinov and Nivre, 2005).

The labels used in conversion 2 are generally following (Bosco, 2004) and more specifically a reduced version used in (Chanev, 2005). Although reflecting most of the basic principles of the TUT annotation scheme, the number of tags is greatly reduced to 14. The current tag set includes the tags: `SUBJ` (subject), `OBJ` (object), `RMOD` (adjectival or adverbial modifier, PP or relative clause), `ARG` (argument), `INDCOMPL` (locative or theme complement), `EMPTYCOMPL` (reflexive personal pronoun modifying verb), `PREDCOMPL` (predicative complement), `INTERJECTION`, `APPOSITION`, `COORDINATOR` (coordinating conjunctions and arguments of coordination), `CONTIN` (part of an expression), `TOP` (root label), `SEPARATOR` (punctuation) and `DEPENDENT` (default label).

Although a reduced number of tags was used in conversion 2, it gave best results in some of the experiments. However, a more precise set of dependency tags should increase parsing accuracy[3]. Besides being somehow incomplete, the labels from conversion 2 were taken from an annotation scheme which is more semantically oriented and which was originally developed for Italian. Several changes were made in order the labels to represent syntactic more than semantic relations and fit the language (Bulgarian) better.

An example of such a change is using subjects and objects only in their shallow sense and not in prepositional phrases, for example[4].

The dependency set from conversion 3 is more fine-grained than the dependency labels set of conversions 1 and 2. The the number of labels is 16: `subj` (subject), `obj` (object), `mod` (modifier), `indobj` (indirect object), `comp` (complement), `prepcomp` (complement of preposition), `adjunct`, `xcomp` (clausal complement), `xmod` (clausal modifier), `clitic` (clitic form), `xadjunct` (clausal adjunct), `marked` (clauses introduced by a subordinator), `pragadjunct` (pragmatic adjunct), `xsubj` (clausal subject), `xprepcomp` (clausal complement of preposition), `conj` (coordinated conjunction), `conjarg` (argument of a coordinated construction), `ROOT` (root label), `punc` (punctuation).

Whereas there are labels with the same role in the three dependency label sets there are labels from each set with no strict analogues from the other two. Basic categories as root nodes, subjects, objects as well as punctuation are treated in the same way in all the dependency sets. Coordinations are treated differently. Conversion 3 has two different labels for coordinated constructions, namely 'conj' and 'conjarg'. For the other sets there is only one ('CC' in conversion 1 and 'COORDINATOR' in conversion 2).

The variety of the other labels concerns mainly the detailness and different priorities of the relation encodings. For example, in conversions 1 and 3 'complement of preposition' is set 'PR' and 'prepcomp', while in conversion 2 there is no such distinction. Then, in conversions 1 and 2 'predicative complement' is set 'PRD' and 'PREDCOMPL', while in conversion 3 this kind of complement is a part of a broader label – 'comp'.

4.2. Problematic issues

The dependency table guides the choice of the appropriate dependency label for the arc that has already been found using the head table. Relying on two constituents above the word in the original treebank a dependency label should be

[3]See Section 7.

[4]Besides being inconvenient to process, these constructions cannot be converted using a common head table.

chosen. This was the approach in conversion 2. In conversion 1, there were rules in which only one constituent above one of the words from the relation and two – above the other were enough to determine the dependency labels of some arcs in the graphs of some sentences.

Using one or two constituents above the words for determining the labels of each dependency relation might not be very appropriate for languages with free word order. In particular, if only two constituents are taken in mind when determining the label of the relation we may end up with errors like mistaken subjects and objects, especially if the structure of the trees in the treebank is too flat or it is too deep and both subject and object candidates have the same two constituents above them.

In languages like Bulgarian, where long-distance dependencies are common, it is difficult to keep the annotation scheme of the treebank uniform. A typical example for 'breaking' the boundaries of a constituent is in cases where a noun or pronoun subject is included in the verb phrase. Having such structures in our treebank it is harder to convert in a plausible way. The trees from the BulTreeBank are generally more deep than the trees from the Penn treebank. We consider one of the reasons for that to be the free-word-orderness of Bulgarian.

One of the problems with increasing the number of constituents above each word in the sentence tree to guide the decision which label to be used for each particular dependency arc is that the number of rules in the dependency table will increase. In some cases two constituents above the words from the tree are enough but in other cases more constituents should be considered. Using a dependency table that relies on two or three constituents above each word in the tree with the rules with three constituents having higher priority might be worth trying.

A common problem in conversion 2, for example, was mistaking the subject and the object in the sentence. In these cases two constituents above the words in the tree were not enough for selecting the appropriate label.

The following sentences will clarify this issue: 'V dushata i se pojavi omraza sreshtu men' ('Hatred against me arose in her soul.'). Here 'omraza' ('hatred') is the subject of the verb 'pojavi' ('arose'). The constituents above 'omraza' are `N` and `NPA`. The constituents above 'poiavi' are `V` and `V`. But in the sentence: 'I shte napishat kritika za mene / pod formata na policejski akt' ('And they will write criticism about me / in the form of a policeman's statement') 'kritika' ('criticism') is the object of the verb 'napishat' ('write'). The constituents above 'kritika' are `N` and `NPA` and the constituents above 'napishat' are `V` and `V` – the same as the constituents from the first example sentence.

5. Error analysis

Whereas in the previous section we described problems with incorrectly assigned dependency labels, if having a correctly retrieved dependency graph, this part of the paper adresses cases of incorrectly attached dependents. We show where our conversions are wrong with respect to the gold standard. There might be at least two different reasons for incorrect attachment corresponding to two different types of errors: 1) relations which are errors according to any theory of dependency grammar and 2) relations where the errors are not errors according to some theory of dependency grammar, especially a theory that is consistent with the head tables of some of our conversions.

Inconsistencies between the conversions can be observed in the verb chain treatment and clauses. If an auxiliary verb is considered the head in a VP, sometimes arguments that normally should be attached to the main verb, like subject and object, will be attached by the conversion algorithm to the auxiliary verb instead. Having auxiliaries as heads can be syntactically more plausible, but we unwillingly neglect the valency of the verb, if not attaching its arguments immediately[5].

According to (Tesnière, 1959) the main verb and the auxiliary form a nucleus and all the dependents of the verb should be attached to the nucleus. But we would not like to pass the boundary by introducing relations among words other than 'dependent'. The auxiliary, if present, is chosen to be the head of the VP in conversions 1 and 3, contrary to the solution implemented in conversion 2 where the main verb is chosen to be the head. We have agreed to use the auxiliary as the head in the gold standard.

The other major difference in the conversions are clauses. In conversions 1 and 3 subordinating conjunctions were chosen to be the heads of the clauses. However, the main verb of the clause was chosen to be the head of the construction in conversion 2. We have the subordinating conjunction being the head of the different types of clauses in the gold standard set. Similarly to the wrong attachment of arguments of the main verb in the auxiliary case, the conversion algorithm can attach the arguments of the main verb to the subordinating conjunction rather than the main verb and this is an error.

In table 5 the three conversions of the sentence 'I shte napishat kritika za mene / pod formata na policejski akt.' ('And they will write criticism about me / in the form of a policeman's statement.') are given together with the gold standard heads. Each row from the table contains a numerated word from the sentence together with information about it part-of-speech, the number of the head word, extracted from conversions 1, 2, and 3 as well as the gold standard head and the dependency labels from each of the conversions.

Conversions 1 and 2 of the sentence from table 5 are identical. The root of conversion 1 should be the same as the root of conversion 3 and the gold standard. This error is probably due to the rules from the head table of conversion 2 that were added to the head table of conversion 1.

Another serious problem reflecting on the conversion procedures is the presence of errors in the treebank. If a head rule is introduced just for the reason to deal with an error annotation, the quality of the conversion will decrease. Sometimes it is hard to distinguish erroneous annotations from proper ones. It is especially tricky to convert linguistic constructions which were not specified in the annotation guidelines of the treebank.

[5]These clarifications were discussed in personal communication with Joakim Nivre.

W No	Word	PoS	Head 1	Head 2	Head 3	Head gold	Dep 1	Dep 2	Dep 3
1	I	Cp	3	3	2	2	-	COORDINATOR	conj
2	shte	Tx	3	3	0	0	VC	ARG	ROOT
3	napishat	Vpptf-r3p	0	0	2	2	ROOT	TOP	comp
4	kritika	Ncfsi	3	3	2	2	OBJ	SUBJ	obj
5	za	R	4	4	4	4	ATT	RMOD	mod
6	mene	Ppelas1	5	5	5	5	PR	ARG	prepcomp
7	/	pt	3	3	2	2	IP	SEPARATOR	punct
8	pod	R	3	3	2	2	ADV	INDCOMPL	adjunct
9	formata	Ncfsd	8	8	8	8	PR	ARG	prepcomp
10	na	R	9	9	9	9	ATT	RMOD	mod
11	policejski	Amsi	12	12	12	12	ATT	RMOD	mod
12	akt	Ncmsi	10	10	10	10	PR	ARG	prepcomp
13	.	pt	3	3	2	2	IP	SEPARATOR	punct

Table 5: Three different analyses and the gold standard for a sentence from the BulTreeBank.

6. The parser

We used the Malt parser (Nivre et al., 2004) in our experiments. Malt parser is a data-driven dependency parser that uses a dependency treebank to learn the actions of a shift-reduce parsing algorithm. It had been tested on several languages, including Swedish, Italian and Bulgarian among others. It has proven to be easily portable from one language to another and is suitable for parsing the data that we have converted.

Malt parser is more generally a framework for construction of different parsers. Different features as part-of-speech tags, lexical units and dependency labels can be used for preparing feature models for learning. Some feature models had proven to be language independent to a large extent. For example, the model m4 is always better than the model m2. Using the m7 model (Marinov and Nivre, 2005) report very good results for Bulgarian. The model consists of six part-of-speech features, four lexical features and four dependency features.

Several learning methods are available as well as a few parsing algorithms within the framework of Malt parser. In this paper, however, we report results that were obtained using only the arc-eager algorithm from (Nivre et al., 2004) together with memory based learning (Daelemans and den Bosch, 2005).

7. Parsing results

We annotated a small set of gold standard data on which to evaluate our conversions, because we did not have a large scale manually annotated dependency treebank. The reason for our decision to use the conversions for training and parsing with a statistical dependency parser was that this could give us evidence for dependency parsing performance of the BulTreeBank.

If the parsing results for the three conversions are different from each other, we can trace the inconsistencies and conclude good and bad conversion practices. However, parsing is not the best measure for the accuracy of our conversions, because a wrongly converted construction could be learned and parsed properly as well as a properly converted construction could be learned and parsed wrongly and the other two combinations are also possible. Although the parsing task is not entirely appropriate for evaluation of the accuracy of our conversions we can conjecture about their applicability.

Table 6: Parsing results for the BulTreeBank converted to dependency

Accuracy	Conv. 1	Conv. 2	Conv. 3
Unlabelled:	76.04%	86.00%	85.27%
Labelled:	20.69%	79.24%	79.48%

All the results are obtained on the same training and test sets with the original gold standard part-of-speech tags of the BulTreeBank. The results are given in Table 6. The metric that we use for evaluation is labelled and unlabelled accuracy measured per word as defined by (Lin, 1998).

If using memory based learning, conversion 2 gives the best unlabelled accuracy and labelled accuracy which is very close to the best. A defect of conversion 2 is the significant number of arcs in the training and test sets (around 4%) that were given the default dependency label ‘DEPENDENT’. We believe that if we reduce that percent, parsing results will improve. The same statement is valid for conversion 1 where there are too many default labels. This is due to the rules from the head table of conversion 2 that we artificially added to the head table of conversion 1.

Finally we should mention some preliminary experiments with the Malt parser using another learner within the parsing framework – Support Vector Machines (Chang and Lin, 2005). We obtained better results for conversions 2 and 3. We haven’t performed tests on conversion 1.

8. Conclusions and future work

We can conclude that in general terms the head table is more important than the conversion algorithm. It should be easy to use different conversion algorithms with the same head table and obtain the same results in most of the cases (with minor corrections in the algorithms which have something to do with treebank specific phenomena).

We conclude that the choice of dependency labels for automatically converted constituency compatible treebanks should be linguistically motivated and specific for the language.

We showed that obtaining applicable dependency parsing results for Bulgarian is achievable if we use the BulTreeBank, even though it is not a dependency treebank. Our results can be used in areas like Question Answering and other tasks from NLP where the syntactic structure of the sentence can provide clues for better analysis and disambiguation.

Our work demonstrates that a treebank than combines the constituent and the dependency information is a valuable source for extraction of dependency treebanks with different inventory of dependency labels and with different granularity of specificity. The experiments with the Malt parser show that the quality of the parsing output depends on the information in treebank. This gives us area for future research how to extract the most appropriate treebank for a given task.

Having achieved state-of-the-art parsing for Bulgarian we can further improve our conversions in several directions. The first one is for the dependency tables of conversion 1 and 2. The table of conversion 2 is not large enough and there are still around 4% of the words in both the training and test data having the default 'DEPENDENT' label. The same problem can be observed to a greater extent in conversion 1.

The second direction for further research is to choose a unified representation for all the dependency structures, combining approaches from the three conversions. A first step in that direction could be to perform several further transformations in order to fix errors, e.g. in the clauses and VPs' treatment. Combined with a unified approach to dependency representation this step could gain some parsing accuracy.

Optimizing the different options and feature models of the Malt parser for Bulgarian and using the SVM learner can improve further parsing results.

Besides the dependency parser we could try a good constituency parser on the BulTreeBank as well. Our intuition is that such a parser would not give better parsing accuracy, because of the free word order nature of the Bulgarian language. Nevertheless, evaluating and comparing a dependency and a constituency parser on the BulTreeBank can point some interesting directions for future research.

Acknowledgements

We would like to thank Joakim Nivre for providing the Malt parser as well as hints about best performing feature models and issues concerning treebank conversion. We thank Alberto Lavelli and two anonymous reviewers for their valuable comments and suggestions on a previous version of this paper. This work was partly funded by a grant provided within the project BIS-21++ at the Institute for Parallel Processing, Bulgarian Academy of Sciences.

9. References

C. Bosco. 2004. *A grammatical relation system for treebank annotation*. Ph.D. thesis, University of Torino.

A. Chanev. 2005. Portability of dependency parsing algorithms – an application for Italian. In *Proc. of the fourth workshop on Treebanks and Linguistic Theories (TLT)*, Barcelona.

C.-C. Chang and C.-J. Lin. 2005. LIBSVM: A library for Support Vector Machines. *URL: http://www.csie.ntu.edu.tw/ cjlin/papers/libsvm.pdf.*

E. Charniak. 2000. A maximum-entropy-inspired parser. In *Proc. of the First Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL)*, Seattle.

M. Collins, J. Hajič, E. Brill, L. Ramshaw, and C. Tillmann. 1999. A statistical parser for Czech. In *Proc. of the 37th Meeting of the Association for Computational Linguistics (ACL)*, College Park.

M. Collins. 1997. Three generative, lexicalized models for statistical parsing. In *Proc. of the 35th Annual Meeting of the Association for Computational Linguistics (ACL)*, Madrid.

A. Corazza, A. Lavelli, G. Satta, and R. Zanoli. 2004. Analyzing an Italian treebank with state-of-the-art statistical parsers. In *Proc. of the 3rd workshop on Treebanks and Linguistic Theories (TLT 2004)*, Tübingen.

W. Daelemans and A. Van den Bosch. 2005. *Memory-Based Language Processing*. Cambridge University Press.

M. Daum, K. A. Fith, and W. Menzel. 2004. Automatic transformation of phrase treebanks to dependency trees. In *Proc. 4th Int. Conf. on Language Resources and Evaluation, LREC-2004*, Lisbon.

H. Gaifman. 1965. Dependency systems and phrase-structure systems. *Information and control*, 8:304–337.

J. Hajič. 1998. Building a syntactically annotated corpus: The Prague Dependency Treebank. In *Issues of Valency and Meaning*, Prague. Karolinum.

E. Hinrichs, J. Bartels, Y. Kawata, V. Kordoni, and H. Telljohann. 2000. The Verbmobil treebanks. In *Proc. of 5. Konferenz zur Verarbeitung natürlicher Schprache*, Ilmenau.

H. Krushkov and A. Chanev. 2005. Automatic parsing: a probabilistic approach for Bulgarian. In *Proc. of Annual Spring Conference of the Union of Bulgarian Mathematicians (UBM)*, Borovetz.

S. Kübler and H. Telljohann. 2002. Towards a dependency-based evaluation for partial parsing. In *Proc. of the LREC-Workshop Beyond PARSEVAL – Towards Improved Evaluation Measures for Parsing Systems*, Las Palmas.

D. Lin. 1998. A dependency-based method for evaluating broad-coverage parsers. *Natural Language Engineering*, 4 (2):97–114.

D. M. Magerman. 1994. *Natural language parsing as statistical pattern recognition*. Ph.D. thesis, Stanford University.

M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19 (2):273–290.

S. Marinov and J. Nivre. 2005. A data-driven parser for

Bulgarian. In *Proc. of the fourth workshop on Treebanks and Linguistic Theories (TLT)*, Barcelona.

J. Nilsson, J. Hall, and J. Nivre. 2005. MAMBA meets TIGER: Reconstructing a Swedish treebank from antiquity. In *Proc. from the special session on treebanks at NODALIDA 2005*, Joensuu.

J. Nivre and M. Scholz. 2004. Deterministic dependency parsing of English text. In *Proc. of 20th International Conference on Computational Linguistics (COLING 2004)*, Geneva.

J. Nivre, J. Hall, and J. Nilsson. 2004. Memory-based dependency parsing. In *Proc. of the Eighth Conference on Computational Natural Language Learning (CoNLL)*, Boston.

J. Nivre. 2005. *Inductive Dependency Parsing of Natural Language Text*. Ph.D. thesis, University of Växjö.

K. Simov and P. Osenova. 2003. A treatment of coordination in the Bulgarian HPSG-based treebank. In *Proc. from FDSL-5*, Leipzig. in press.

K. Simov, P. Osenova, S. Kolkovska, E. Balabanova, D. Doikoff, K. Ivanova, A. Simov, and M. Kouylekov. 2002. Building a linguistically interpreted corpus of Bulgarian: the BulTreeBank. In *Proc. of LREC 2002*, Canary Islands.

W. Skut, T. Brants, B. Krenn, and H. Uszkoreit. 1997. Annotating unrestricted German text. In *Proc. of 6. Fachtagung der Section Computerlinguistic der Deutschen Gesellschaft für Sprachwissenschaft*, Heidelberg.

H. Tanev. 2001. *Automatic Text Analysis and Ambiguities Resolution in Bulgarian*. Ph.D. thesis, University of Plovdiv.

L. Tesnière. 1959. *Éléments de syntaxe structurale*. Editions Klincksieck.

T. Ule and S. Kübler. 2004. From phrase-structure to dependencies, and back. In *Proc. of the International Conference on Linguistic Evidence*, Tübingen.

Z. Žabokrtský and I. Kučerová. 2002. Transforming Penn Treebank phrase trees into (Praguian) tectogrammatical dependency trees. *Prague Bulletin of Mathematical Linguistics*, 78:77–94.

Z. Žabokrtský and O. Smrž. 2003. Arabic syntactic trees: from constituency to dependency. In *Proc. 10th Conference of the European Chapter of the Association of Computational Linguistics, EACL 2003*, Budapest.

F. Xia. 2001. *Automatic Grammar Generation from Two Different Perspectives*. Ph.D. thesis, University of Pennsylvania.

H. Yamada and Y. Matsumoto. 2003. Statistical dependency analysis with Support Vector Machines. In *Proc. of IWPT*, Nancy.

NLP Tools Integration Using a Multi-Layered Repository

João Graça[*], Nuno J. Mamede[*], João D.Pereira[†]

[*]L^{2}F – INESC-ID Lisboa/IST
Rua Alves Redol 9, 1000-029 Lisboa, Portugal
{joao.graca, nuno.mamede}@l2f.inesc-id.pt
[†] Eng. Group –INESC-ID Lisboa/IST
Rua Alves Redol 9, 1000-029 Lisboa, Portugal
joao@inesc-id.pt

Abstract

Natural Language processing (NLP) systems are typically characterized by a pipeline architecture in which several independently developed NLP tools, connected as a chain of filters, apply successive transformations to the data that flows through the system. Hence when integrating such tools, one may face problems that lead to information losses, such as: (i) tools discard information from their input which will be required by other tools further along the pipeline; (ii) each tool has its own input/output format. Moreover, the tools formats do not establish relations between their Input/Output. These relations are useful for keeping the information of different levels aligned. Another problem is that each tool developer must normally concern himself with the definition of a data model to represent linguistic information and Input/Output facilities for the tool he's developing. Usually, these models and facilities are very similar, so their redefinition for each tool represents a waste of time.

This work proposes a solution that solves these problems. We offer a framework for NLP systems. Our framework uses a client-server architecture, in which the server acts as a repository where all tools add/consult data. Data is kept in the server under a data model that is independent of the client tools. This model is able to represent a broad range of linguistic information. The tools interact with the server through a generic API which allows the creation of new data and the navigation through all the existing data. Moreover, we provide libraries implemented in several programming languages that abstract the connection and communication protocol details between the tools and the server, and provide several levels of functionality that simplify server use.

1. Introduction

Natural Language processing (NLP) systems are typically characterised by a pipeline architecture, in which several NLP tools connected as a chain of filters apply successive transformations to the data that flows through the system. Usually, each tool is independently developed by a different person whose focus is on his/her own problem rather than on the future integration of the tool in a broader system. Hence when integrating such tools, several problems arise, which are mainly related to the following: (i) how the tools communicate with each other, (ii) what kind of information flows between the several tools (may cause information lost).

At the Spoken Language Systems Lab (L^{2}F), where this work was developed, several NLP systems have already been created. Most of the detected problems when bulding those systems where related with the information flow between the tools composing the system. These problems are:

- Architectural problems - the information discarded along the system may be required further ahead by other tools;
- Conversion between data formats - conversions are necessary between the data formats produced by different tools that wish to interchange information. Moreover, if the expressiveness of each format is different, then one format may not be completely mappable into another.

Besides these problems, which lead to information losses, there is another problem concerning the data: how to maintain the data lineage between information produced by different tools composing a NLP system? When viewing each tool output as a layer of linguistic information over a primary data source and considering that layers are normally related to each other, it is desirable to maintain relations between those layers. First, these relations enable the navigation through related linguistic information produced by different tools. Secondly, tools can reference data from other layers in order to avoid the repetition of common data. These types of relations are called cross-relations because they span across linguistic information layers.

Finally, a last problem concerns the fact that each NLP tool programmer usually develops its own data model to represent the linguistic information, and Input/Output facilities to that data model. Since, these different data models normally represent similar information they tend to be very similar. The redefinition of such similar models represents a waste of time.

The main objective of this work is to build an NLP framework for the creation of NLP systems with the following properties:

- Avoid information losses between tools composing a system;
- Simplify new NLP tools implementation by providing general Input/Output facilities and a data model to represent linguistic information;
- Simplify the reutilisation of existing NLP tools by minimising the changes required in each individual tool;

- Maintain the data from tools aligned allowing the navigation through related data from different tools.

The proposed solution is a framework using a client-server architecture instead of a pipelined architecture. In our solution, the server acts as a repository where all NLP tools (clients) add/consult data. The server maintains cross-relations between the existing layers of data. The data is kept in the repository under a data model independent of the client tools. This data model allows the representation of a broad range of linguistic information. The tools interact with the repository through a generic remote API that permits the creation of new data and the navigation through all the existing data. Moreover, this work provides libraries implemented in several programming languages that abstract the connection and communication protocol details between NLP tools and the server, and provide several levels of functionality that simplify the creation and integration of NLP tools.

2. Solution Requirements

We defined a set of requirements for a solution that supports and simplifies the integration of independently developed NLP tools into NLP systems. These requirements were used to validate the proposed solution, and are the following:

- All information produced during the execution of a system should be available;
- Tools should only produce directly related information;
- The solution should simplify the creation of new tools, by providing: (i) an Input/Output interface, which handles the loading and saving of data used by the tool; (ii) a data model that can be used by each tool to represent linguistic information;
- The solution should minimise the number of conversion components required to build an NLP system, when integrating existing NLP tools that do not comply with the system's model;
- The provided interface should allow the navigation between information produced by different NLP tools.

To achieve the previous generic requirements we defined two groups of requirements that the solution must fulfil, namely:

- Data Model Requirements - Represents the requirements for a data model capable of representing and relating a broad range of linguistic information, which are described in Subsection 2.1.;
- System requirements - Represents requirements of the underlying system related with the interaction between the system and the NLP tools, which are described in Subsection 2.2..

2.1. Data Model Requirements

The data model main requirement is that it must be able to represent a broad range of linguistic information produced by different NLP tools. Furthermore, the data model must be extensible because it is impossible to foresee all kinds of linguistic information that may appear in the future. We begin by distinguishing two conceptually different kinds of information that the data model must represent:

- Primary data sources such as a text, or a speech signal;
- Linguistic information produced by NLP tools, over primary data sources, or previously defined linguistic information.

We also identify four types of actions that NLP tools may perform:

- Creation and edition of primary data sources, for example, the incremental creation of a new primary data source containing the phonetic transcription of the text belonging to another primary data source. This newly created data source can be the target of the linguistic information of other tools;
- Identification of linguistic elements from a primary data source, for instance, the segmentation of a sentence into words;
- Creation of relational information between linguistic elements, such as the relation between a verb and the corresponding subject.
- Assignment of characteristics to a linguistic element, or a relation, for example, the morphological features of a word;

Each NLP tool may produce several types of information at the same time. The linguistic information generated by an NLP tool is normally derived from linguistic information created by other tools. For example, a part of speech tagger will use the segments produced by a tokenizer and add morphologic information to those segments. A morphological disambiguator may use the classifications produced by several part of speech taggers to select the most appropriate classification.
The data model must be able to represent several layers of both primary data sources and linguistic information. We have defined the following requirements regarding these two types of information:

- The data model must be able to represent any kind of primary data source such as text, speech, video, or any combination of these;
- The data model must support the creation and edition of primary data sources;
- All linguistic information except primary data sources produced by an NLP tool must be associated with the same layer;
- The data model must allow the selection of linguistic information through the identification of the layer that contains it;

- Each layer is associated with the identification of the tool that produced it.

The last three requirements are necessary to simplify the identification of information inside the data model. This way all linguistic information is organised into layers.
The data model must represent the three types of linguistic information that each NLP tool can produce: (i) the identification of linguistic elements; (ii) the creation of relations between linguistic elements; (iii) and the assignment of characteristics to linguistic elements and relations.
We have defined the following requirements for the representation of those linguistic elements:

- The model must be able to represent ambiguity in the identification of linguistic elements, for example, a compound term can be segmented as only one segment containing the compound term or several segments for each word.
- The model must be able to represent trees of linguistic elements, for example, syntactic trees.
- The model must allow the creation of relations between linguistic elements from the same layer.
- It must be possible to represent classification ambiguity, which correspond to associating disjunct sets of characteristic to the same linguistic element. For example, distinct morphological features for the same word.
- The model must allow the association of characteristic to linguistic elements, or relations from other layers.

Besides the representation of linguistic information produced by each NLP tool, the data model has the following requirements concerning the relations between linguistic information from different layers:

- The model must be able to represent relations between linguistic elements from different layers. These relations represent dependencies between layers of information, and allow the navigation between layers;
- The data model must allow linguistic elements to reference data belonging to a primary data source, without having to copy its value, thus avoiding the repetition of the same data in several layers;
- The model must be able to represent data which may not exist in any primary data source, for example, the separation of contractions.

2.2. System Requirements

This subsection presents the general requirements of the system which are not related to its data model, but with the interaction between the system and the NLP tools, which are the following:

- The system must simplify the iteration of data in the *repository*, e.g. all segments from a segmentation. This is required because iteration is the most common way of interaction between an NLP tool and its data;
- Any NLP tool can select data based on a layer's identification. This way an NLP tool only needs to handle the data it requires;
- It is possible to access the data of an unfinished *Analysis*. This way an NLP tool may consume information that is being produced at the same time by another NLP tool, allowing the parallel processing of data;
- The system must make the data persistent;
- The system must be able to interact with NLP tools written in any programming language.

3. Related Work

During the development of this work we analysed several architectures whose goal was to simplify the creation or integration of NLP tools, towards their usage in NLP systems, namely: the Emdros text database system (Petersen, 2004), a text database engine for analysis, and retrieval of analysed or annotated text; the Natural Language Toolkit (Loper and Bird, 2002), a suite of libraries, and programs for symbolic, and statistical natural language processing; the Gate architecture (Bontcheva et al., 2004), a general architecture for text engineering that promotes the integration of NLP tools by composing them into a pipes and filters architecture; and the Festival speech synthesis system (Taylor et al., 1998), a general framework for building speech synthesis systems.
We also compared some work from the linguistic annotation field, whose focus is on the definition of a logical level for annotation independent of the annotations' physical format. This logical level should be able to represent the most common types of linguistic annotations to promote reuse of annotated corpora. Our data model can be seen as this logical level. In this field we compared two works: the Annotation Graphs Toolkit (AGTK) (K. Maeda and Bird, 2002) that is an implementation of the Annotation Graphs formalism (Bird and Liberman, 1999), the most cited work in this area; and the ATLAS architecture (Bird et al., 2000), a generalisation of the Annotation Graphs formalism to allow the use of multidimensional signals.
The AGTK and the ATLAS architectures do not allow the separation of information into layers. In these architectures, to avoid the loss of information, each tool has to load all previous annotations, and then save them together with its results. This strategy has several drawbacks: first, each tool must know how to manage data which may be unrelated with the tool itself. Second, each tool may have to load and parse extra data upon its initialisation and consequently save extra data when terminating. Finally, it is difficult for a tool to handle data from several tools at the same time, because it must merge the common data from the input tools. Concerning the expressiveness power of the data model used to represent linguistic information, the adoption of the Annotation Graphs model is not possible, mainly because it does not allow the representation of relational information, nor the representation of cross-relations between several data layers. The extensions performed by the ATLAS architecture provide a better representation for conceptually different linguistic phenomena, such as hierarchic trees, and ambiguous segmentations. Furthermore,

ATLAS allows the use of every type of data sources. But, even so, this model still presents the same problems as the Annotation Graphs formalism.
The Emdros framework has a representation model that is too restrictive for our objectives. For example, it restricts the media type to text. In Emdros it is not possible to properly represent some types of linguistic phenomena, such as classification ambiguity, or relations between elements.
The NLTK restricts development to the Python programming language, and relies on the Python interpreter to work. Moreover, its underlying data model is not able to fulfil all our requirements, for instance, the representation of ambiguities, such as, classification ambiguity.
The GATE framework presents the same problems as the AGTK formalism, concerning its data model. Moreover, it limits its use to the Java programming language. GATE promotes the integration of NLP tools into a pipes and filters architecture, which as we mentioned in the introduction, has some problems that led to the development of this work.
Finally, the Festival framework has a model that does not allow the fulfilment of all our requirements, namely: i) its source data must be text, ii) it cannot represent all types of ambiguities defined in the requirements, iii) it does not allow the concurrent execution of different tools.
The requirements defined in the previous section were compared against the requirements that are being defined by ISOTC37/SC4 (Terminology and other language resources) to define a standard for linguistic annotation (Ide et al., 2003; Ide and Romary, 2001). We found them to be very similar, which strengthened our conviction that any model used to represent linguistic information should follow these requirements.

4. Proposal

Our proposal consists of a client-server architecture. The clients are NLP tools while the server consists of a centralised repository of linguistic information and data sources represented under a data model. Each NLP tool can interact with the server in two ways:

- By using a remote interface independent of the tool's programming language;
- By using a module in its programming language that abstracts the communication and protocol details between the client and the server, and offers an implementation of the data model. The use of the server is simpler using the client library than using the remote API, but the use of the client library requires an implementation of the client library for each programming language used.

4.1. Data Model

The data model is able to represent and relate various types of linguistic information produced by several NLP tools. Besides representing the different linguist phenomena, the model simplifies the use of linguistic information.
The entities composing the data model, described in Fig.1 are:

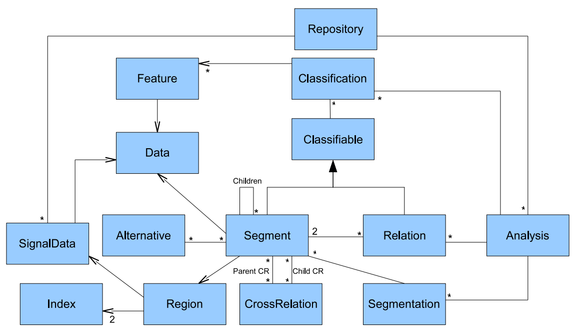


Figure 1: Data Model class diagram.

- *Repository*: a centralised linguistic repository that stores the output of several NLP tools, and organises that information into layers. Each layer is univocally identified inside the repository. There are two types of layers: *SignalData* and *Analysis*;
- *Data*: an abstraction of a data type that can be used by the *Repository*, e.g. *String*;
- *SignalData*: an abstraction of a raw data source, e.g. a text, abstracting details such as its physical location or its data type. All *SignalData* have a minimal granularity unit. In a text that unit might be a character, while in an audio file it might be the sample;
- *Index*: a point in a *SignalData* using its minimal unit;
- *Region*: defines a region in a *SignalData*, using a pair of *indexes*. The *Region* encapsulates the details about the specifics of *SignalData* and *Index* elements;
- *Analysis*: linguistic information other than *SignalData* elements produced by an NLP tool. An *Analysis* may be open or closed indicating whether the tool has already finished the addition of data into the *Repository*. An *Analysis* can only be changed if it is open. The *Analysis* is responsible for the creation of *Segmentations*, *Relations* and *Classifications*;
- *Segment*: a linguistic element, e.g., a word. A *Segment* may contain two Data elements: the original data and the derived data. The original data corresponds to a linguistic element identified in a *SignalData*, while the derived data corresponds to a possible transformation performed over the original data. A *Segment* may be ambiguous, meaning that it has a set of *Alternative* elements for the linguistic element that it identifies. It may be hierarchic, meaning that it has a *Parent Segment* and *Child Segments*. A *Segment* has a set of disjunct *Classification* elements, where each *Classification* assigns a set of characteristics to the *Segment*. It also has a set of Relations, that establish links between two Segments from the same Analysis. A *Segment* may belong to a set of *CrossRelations*, which are used to establish structural relations between *Segments* from different *Analyses*;

- *Segmentation*: a set of sequentially ordered *Segments*, e.g., the words in a sentence. The *Segmentation* is responsible for the creation of *Segments*;
- *Relation*: a link between two segments. For example, the relation between the subject and the verb in a phrase;
- *Classification*: a set of characteristics of a *Segment* or *Relation*. For instance, the morphological features of a word;
- *Alternative*: a set of *Segments* representing different alternatives for an ambiguous linguistic element;
- *Cross-Relation*: a structural relation between *Segments* of distinct *Analyses*. It is used to navigate between different layers of information.

Figure 2 shows how the derived data attribute can be used to represent the separation of a contraction. It shows a *Repository* with two layers. The first layer is a *SignalData* that contains the text *"They're waiting outside"*. The second layer is an *Analysis* with two *segmentations*. The first *Segmentation* contains a *Segment* for each token of the *SignalData* which references a *Region* in the *SignalData*. The second *Segmentation* shows the use of *derived data* to represent the separation of the contraction *"They're"*. It contains two *Segments* with *derived data* (represented in italic) one for each word composing the contraction. These segments still contain the *Region* referencing the *original data*, which allows the access to the original state of the text. To clarify the example, we have repeated the representation of the *SignalData* layer to not overlap the arrows. Note that, there is only one layer with that *SignalData*.

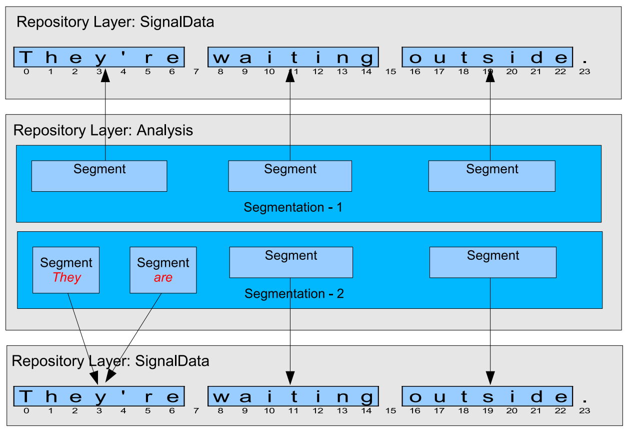


Figure 2: Use of derived data example.

Figure 3 shows the representation of an example of classification ambiguity. The figure shows a *Repository* with 3 layers. The first layer contains the original *SignalData* while the second layer contains an *Analysis* with the segmentation of the original text into words. The third layer represents the output of a part-of-speech tagger. Each *Segment* from the second layer has several *Classifications*. In this example, each *Classification* only contains one attribute-value

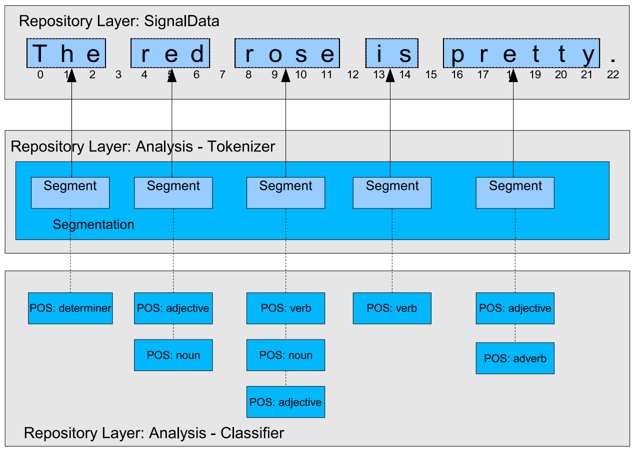


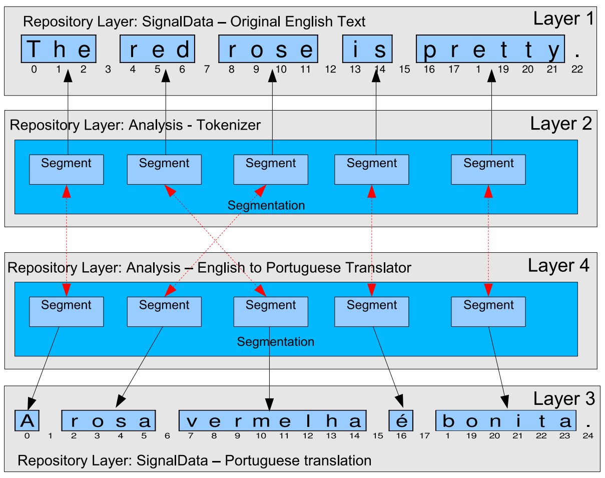
Figure 3: Classification ambiguity example.

pair: the word's part of speech. This example shows the association of *Classifications* to *Segments* from other layers.

Figure 4 shows how *CrossRelations* can be used to align different *SignalData* elements. The figure shows the translation of the sentence *"The red rose is pretty"* from English to Portuguese, *"A rosa vermelha é bonita"*. The *Repository* contains four layers. The first layer corresponds to the original *SignalData* containing the English text, where the possible *Indexes* are represented as integers under each character, and the possible *Regions* identifying the words are represented inside a box. The second layer is an *Analysis* which contains the segmentation of that text into words: it contains one *Segment* for each word, that uses a *Region* (indicated by an arrow from the *Segment* to the *Region* in the *SignalData*) to refer to the word's text. The *Segmentation* contains those *Segments* in accordance with the word's order in the text. The third layer is a *SignalData* containing the translation of the English text from the first layer, and the fourth layer is an *Analysis* produced by an English to Portuguese translator tool, which creates a *Segmentation* where each *Segment* represents a Portuguese word from the third layer. The representation of the third and fourth layers is the same as the explained for the first two layers. The alignment between the two texts is achieved by adding *CrossRelations* between *Segments* from the corresponding *Analyses*. The *CrossRelations* are represented as dotted arrows between *Segments*.

4.2. Server Architecture

The server architecture consists of a shared data style. This architecture has the advantages of allowing clients to be added without the server knowledge, and of allowing the integration of the data produced by all the tools. The linguistic information, described using the data model, is managed by the server. The server is organised in three layers: the data layer, the service layer, and the remote interface layer. Each layer can only use the adjacent layers through their interfaces. The layered approach promotes portability, and maintainability since the role of each layer is well identified, and the implementation of each layer can be changed without affecting the other layers.



Repository Layer: SignalData – Original English Text
Layer 1
T h e r e d r o s e i s p r e t t y .
Repository Layer: Analysis - Tokenizer
Layer 2
Segment
Segmentation
Repository Layer: Analysis – English to Portuguese Translator
Layer 4
Segment
Segmentation
Layer 3
A r o s a v e r m e l h a é b o n i t a .
Repository Layer: SignalData – Portuguese translation

Figure 4: Data alignment example.

4.2.1. The Data Layer

The data layer contains the logic of the application, and uses the data model to represent the linguistic information. Besides representing all the linguistic information, the Data Layer is also responsible for guaranteeing the persistence of all linguistic information stored in the server.

4.2.2. The Service Layer

The service layer (Fowler, 2002) defines the server's boundary by providing a set of methods and coordinates the server's response to each method. It is used by the Remote Interface Layer, which handles the specific protocol details, and encapsulates the Data Layer. The Service Layer is responsible for hiding the details regarding the representation of the domain elements of the Data Layer. It transforms the domain elements into Data Transfer Objects (DTO) which will be passed to the client. A DTO (Fowler, 2002) is an object with no semantics, and is used to pass information between the client and the server. Each DTO can hold two kinds of information from domain objects: i) identification information used to access domain objects; ii) read-only information that may be required by the client.

The Service Layer is also responsible for providing methods that allow the creation of iteration facilities on the client side. Moreover, and since the Service Layer is a single entry point into the server,it is an ideal place to perform logging and authentication actions.

The Service Layer together with the DTOs works as a Remote Facade (Fowler, 2002) thus diminishing the number of remote calls required for certain operations.

4.2.3. The Remote Interface Layer

The remote interface layer provides the methods that are available to client tools according to a selected protocol. It communicates with the Service Layer, and is responsible for serialising the DTOs provided by the Service Layer into their external representation, which will be sent across the connection. It is also responsible for assembling the DTOs back, and pass them to the Services Layer.

4.3. Client Library Architecture

The use of the client library allows an NLP tool to abstract from details concerning the communication with the server, and the data exchange protocol. It also provides some high level interfaces that may simplify the integration of NLP tools. The client library uses a layered architecture, each layer is described in the rest of this subsection.

4.3.1. The Client Stub Layer

The client stub layer is responsible for communicating with the server under the chosen protocol, through the server's Remote Interface layer. All the other layers of the client library depend and use this layer. This way the other layers are independent of the specific communication protocol that is being used.

4.3.2. The Data Model Layer

The data model layer implements the data model. It allows NLP tools to use the data model as their object model, thus simplifying their creation. Since the concepts used by NLP tools are usually similar, by using the data model we desire to avoid the definition of an equivalent one every time a new tool is created. In addition, by using only the interfaces provided by the Data Model layer its concrete implementation can be changed without changing the tool. This way, an NLP tool can be used as a stand-alone tool or as a client tool connected to the shared repository just by changing the implementation of the Data Model Layer.

The Data Model layer elements are proxies(Gamma et al., 1995) for the elements of the Data Model in the server. The methods performed on those elements are delegated into the corresponding elements in the server.

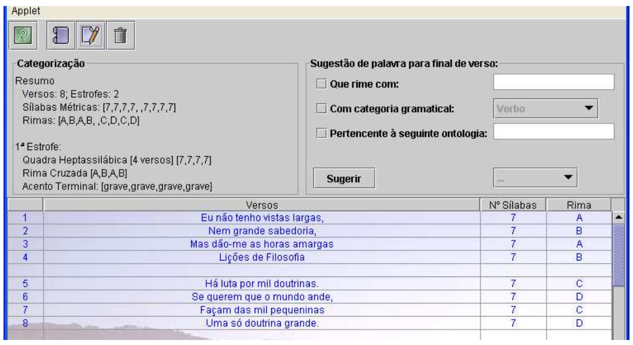
The repository can be used concurrently by several NLP tools, so it is possible that a tool consumes information that is being produced by another tool at the same time. If the consumer tool is faster than the producer tool and depletes the data that is being produced, the consumer tool may finish its processing due to a lack of data before what was expected. To avoid this situation the iterators on the client side have a blocking behaviour. The method `hasNext()` only returns false when the Analysis that contains the data that is being iterated is closed and there are no more elements to iterate. However, this policy can result in a deadlock to the consumer tool if the producer tool ends abruptly without closing its Analysis. So, we introduced a time limit for which a client method can be blocked in the method `hasNext()`.

4.3.3. The Extra Layers Layer

The extra layers layer provides extensibility to the client library. It represents new layers that can be added on top of previous ones, enabling the creation of domain specific layers, which may simplify the creation of new NLP tools. For example, a part-of-speech tagger could use a layer that provides the concepts of word, phrase and text, with methods such as `nextWord()`, and `addGender(Word w)`. The usage of Extra Layers can also provide semantic meaning to the linguistic information kept in the Repository for a given NLP system.

5. Poetry Learning Aiding System

The Poetry Learning Aiding System (Araújo, 2004)is a didactic NLP system for learning the concepts of Portuguese poetry as well as aiding the creation of poems. The system major functionalities are the characterization of poems and the suggestion of words to complete a verse. Figure 5 shows the system's user interface. The user can add a poem contained in a text file to be characterized, or ask for a suggestion to end the last verse of the current poem.



Categorização
Resumo
Versos: 8; Estrofes: 2
Sílabas Métricas: [7,7,7,7, ,7,7,7,7]
Rimas: [A,B,A,B, ,C,D,C,D]

1ª Estrofe:
Quadra Heptassilábica [4 versos] [7,7,7,7]
Rima Cruzada [A,B,A,B]
Acento Terminal: [grave,grave,grave,grave]

Sugestão de palavra para final de verso:
☐ Que rime com:
☐ Com categoria gramatical: Verbo
☐ Pertencente à seguinte ontologia:

	Versos	Nº Sílabas	Rima
1	Eu não tenho vistas largas,	7	A
2	Nem grande sabedoria,	7	B
3	Mas dão-me as horas amargas	7	A
4	Lições de Filosofia	7	B
5	Há luta por mil doutrinas.	7	C
6	Se querem que o mundo ande,	7	D
7	Façam das mil pequeninas	7	C
8	Uma só doutrina grande.	7	D

Figure 5: Poet User Interface example.

The system is composed by several tools, some of them were already developed in our group, while others where developed specifically for this system. The initial input of the system is a text file containing the source poem where verses are separated by blank lines. The system contains the following tools:

- The poem structure identifier - This tool places the original poem in the repository and creates two analyses. One containing the segmentation of the poem into verses and the other containing the separation of the poem into lines. These two analyses are aligned using cross-relations (each line is cross-related with the corresponding verse);
- A word tokenizer and part of speech tagger - This tool segments the poem into words (these new segments are aligned with the sentence segments) and assigns each word's possible part of speech tags as several classifications;
- Syllable segmentation and phonetic transcriber - This tool segments each word into its grammatical syllables (kept aligned with the corresponding words) and adds a classification for each syllable containing its phonetic transcription;
- Poem characterization tool - This tool classifies the poem based on the information produced by the previous tools. It uses for instance, the number of syllables of each sentence, the number of sentences of each verse and the the type of rhyme of each verse;
- Word suggester - This tool suggests a word to end the last line of the last verse. It uses a n-gram model that based on the pos tags of the last words of the last line suggests the part of speech required. Then based on the phonetic termination of the last words from the other lines in the verse suggests a word to end the verse. This suggestion may be restrained to a certain pos tag or to a certain phonetic termination if required.

Figure 6 shows an example of the repository during the execution of the system before the execution of the poem characterization. At this stage all information required for the next two tools is cross-related in the repository.

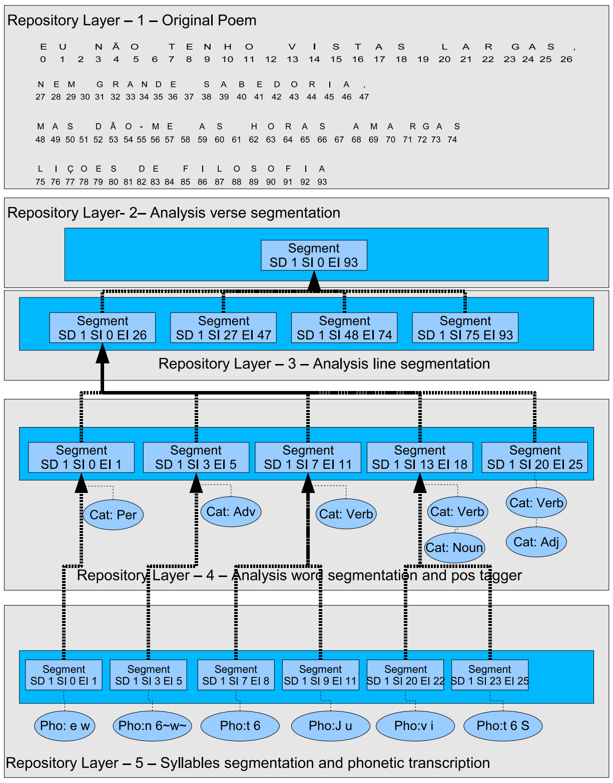


Figure 6: Poet Repository state example.

This system is the first being created using the repository and it allowing us to test the feasibility of the framework. The existing tools (pos-tagger and the phonetic transcriber) where integrated using converters from their existing formats into the repository data model format. The new tools were implemented using the data model as their object model, avoiding some issues which are usually the developer's responsibility, as the definition of an object model, and the definition of an IO interface. Moreover, having the different levels of information cross-related the creation of the poem characterization tool and the word suggester was simplified, since instead of having several input files from each tool and having to merge all information, each tool accesses one particular level and then navigates through the information using the existing cross-relations.

6. Results and Future work

We implemented our framework in *Java* (around 57 classes) using the XML-RPC protocol provided by the APACHE XML-RPC package. This protocol was chosen because of its simplicity and because it does not impose any restrictions on the programming language used by client

Merging FrameNet and PropBank in a corpus of written Dutch

Paola Monachesi*, Jantine Trapman *

* Utrecht University, Uil-OTS
Trans 10, 3512 JK Utrecht, The Netherlands
{Paola.Monachesi, Jantine.Trapman}@let.uu.nl

Abstract

We discuss the development of a schema for the semantic annotation of a corpus of written Dutch. Our focus is on the annotation of semantic roles. We rely on the proposals made within initiatives such as PropBank and FrameNet. Our aim is to reconcile the PropBank approach to role assignment which is essentially corpus based and syntax driven with the more semantic driven FrameNet approach which is based on a network of relations between frames. A comparison with similar initiatives such as Omega is carried out.

1. Introduction

The creation of semantically annotated corpora has lagged dramatically behind. As a result, the need for such resources has now become urgent. Several initiatives have been launched at the international level in the last years, however, they have focussed almost entirely on English and not much attention has been dedicated to the creation of semantically annotated Dutch corpora. The Flemish-Dutch STEVIN-programme has identified semantic annotation as one of its priorities.[1] Within the project Dutch Language Corpus Initiative (D-Coi) we are developing guidelines for the semantic annotation of Dutch and our focus is on two types: semantic role assignment and temporal and spatial semantics.[2] In this paper, however, we will only be concerned with semantic role annotation.

During the last few years, corpora with semantic role annotation have received much attention, since they offer rich data both for empirical investigations in lexical semantics and large-scale lexical acquisition for NLP and Semantic Web applications. Several initiatives are emerging at the international level and within the D-coi project we have taken the following ones into account:

- PropBank (Kingsbury et al. 2002) which aims at adding a layer of semantic annotation to the Penn English TreeBank;
- FrameNet (Johnson et al. 2002) which is a corpus-based lexicon-building project that documents the links between lexical items and the semantic frame(s) they evoke;

Our aim is to reconcile the PropBank approach to role assignment which is essentially corpus based and syntax driven with the more semantics driven FrameNet approach which is based on a network of relations between frames.

In this paper, we will discuss two pilot studies we have carried out to evaluate the feasibility of this approach: one aiming at the annotation of sentences containing predicates belonging to the *Communication* frame while the other one takes the (adjunct) middle construction into consideration.

Furthermore, the approach sketched here which aims at *merging* certain features of FrameNet and PropBank will be compared with the efforts made within the Omega project (Philpot et al. 2005) which deals with the *alignment* of several resources among which FrameNet and PropBank.

2. The Dutch language Corpus Initiative: description of aims and goals

The realization of an appropriate digital language infrastructure for Dutch is one of the objectives of the STEVIN programme which has been recently launched in The Netherlands. In particular, the need for a large corpus of written Dutch, comprising 500-million-words has been identified as one of the top priorities. This corpus should be tailored to the needs of scientific research and commercial applications and should improve the development of other resources and tools. Applications such as information extraction, question-answering, document classification, and automatic abstracting that are based on underlying probabilistic techniques should benefit from it.

All texts in the corpus will conform to standards for character encoding and markup. Furthermore, the corpus will be linguistically annotated. Given its size, linguistic annotation is only feasible if automatic or semi-automatic procedures can be employed: post-editing can be undertaken only for a subset of the data. For the various annotation layers, annotation schemes must be decided upon and the aim is to revise and adapt the protocols which have been developed for the Spoken Dutch Corpus (Oostdijk et al. 2002)
.

A pilot study is being carried out to this end: the Dutch language Corpus Initiative is a project launched within the STEVIN programme whose aim is a blueprint for the construction of the 500-million-word corpus. The project is concerned with issues related to the design of the corpus and the development (or adaptation) of protocols, procedures and tools that are needed for sampling data, text regularization, converting file formats, marking up, annotating, post-editing, and validating the data. Within the D-coi project, a 50 million word pilot corpus will be compiled, parts of which will be enriched with (verified) linguistic annotations. The pilot corpus is intended to demonstrate the feasibility of the approach. It will provide the necessary testing ground on the basis of which feedback can be obtained about the adequacy and practicability of various annotation schemes and procedures, and the level of success with which tools can be applied.

[1]http://taalunieversum.org/taal/technologie/stevin/
[2]http://lands.let.ru.nl/projects/d-coi/

3. Semantic annotation in D-coi

One of the innovative aspects of the D-coi project is that it will focus not only on the revisions of those protocols which have been already developed within the Spoken Dutch Corpus for PoS tagging, lemmatization and syntactic annotation but it will also explore the possibility of integrating an additional annotation layer based on semantic information. This annotation layer was not present in the Spoken Dutch Corpus.

The need for semantically annotated corpora has now become urgent. Several initiatives have been launched at the international level in the last years, showing that the time is ripe for activities in this direction. However, they have focussed almost entirely on English and not much attention has been dedicated to the creation of semantically annotated Dutch corpora. One of the goals of the D-Coi project is the development of a protocol for such an annotation layer. Only a small part of the corpus will be annotated with semantic information (i.e. 3000 words), in order to yield information with respect to its feasibility. A more substantial annotation effort could be carried out in the framework of the 500 million word corpus. In this follow-up project other types of semantic annotation might also be taken into consideration, as well as their interaction with other levels like PoS tagging and syntactic analysis. We are therefore taking this interaction into consideration when developing the protocol.

For the moment, we are only dealing with two types of semantic annotation and their interaction, that is semantic role assignment and temporal and spatial semantics. The reason for this choice lies in the fact that semantic role assignment (i.e. the semantic relationships identified between items in the text such as the agents or patients of particular actions), is one of the most attested and feasible types of semantic annotation within corpora. On the other hand, temporal and spatial annotation was chosen because there is a clear need for such a layer of annotation in applications like information retrieval or question answering. In the rest of this paper, however, we will focus only on semantic role assignment.

4. Semantic role assignment in D-coi

During the last few years, corpora enriched with semantic role information have received much attention, since they offer rich data both for empirical investigations in lexical semantics and large-scale lexical acquisition for NLP and Semantic Web applications. Several initiatives are emerging at the international level to develop annotation systems of argument structure, within the D-coi project we intend to exploit existing results as much as possible and to set the basis for a common standard. We want to profit from earlier experiences and contribute to existing work by making it more complete with our own (language specific) contribution given that most resources have been developed for English.

The following projects have been evaluated in order to assess whether the approach and the methodology they have developed for the annotation of semantic roles could be adopted for our purposes:

- PropBank (Kingsbury et al. 2002);
- FrameNet (Johnson et al. 2002);

Given the results they have achieved, we have taken their insights and experiences as our starting point. In the rest of this section, we will consider them more in detail in order to evaluate their strengths and weaknesses and to assess which features of the existing systems we want to include in the scheme for the semantic annotation of the D-coi corpus.

4.1. PropBank

PropBank aims at adding a layer of semantic annotation to the Penn English TreeBank (Marcus et al. 1993). It provides a semantic representation of argument structures that are labeled consistently in such a way that the data are usable for automatic extraction. The set of argument labels is a very restricted one, as Table 1 shows. The PropBank lexicon, which was added first to facilitate annotation and later evolved into a resource on its own, is constructed following a 'bottom-up' strategy: starting from the various senses of a word, a framefile is created for every verb. Such a framefile contains thus all possible senses of the verb plus a set of example sentences that illustrate the context in which the verb can occur. For each sense of the verb, a roleset and example sentences are available. Therefore, when a verb has two senses its framefile contains two different rolesets as is the case with *leave* from (Babko-Malaya 2005):

(1) a. Frameset leave.01 *move away from*
Arg0: entity leaving
Arg1: place left
Mary left the room

b. Frameset leave.02 *give*
Arg0: giver
Arg1: thing given
Arg2: beneficiary
Mary left her daughter-in-law her pearls in her will

To create a framefile, relevant sentences are extracted from the corpus. Based on those sentences, the most frequent and/or necessary roles are selected and one or more rolesets are formed. In this way, the most common senses of the verb are stored in the framefile. An interesting feature of the PropBank project is that the corpus has been annotated automatically with 83% accuracy and then corrected by hand on the basis of the developed lexicon. Furthermore, the goal of the ptoject is to provide training data for supervised automatic role labelers. This is a desirable objective since it will be possible to annotate corpora of the size of D-coi with semantic role information only if the the process is semi-automatic.

4.2. FrameNet

Contrary to PropBank, FrameNet does not annotate a complete corpus, but one that contains example sentences that illustrate all possible syntactic and semantic contexts of the lexical items taken into consideration. Besides the corpus, two other components can be distinguished in FrameNet, that is a set of lexical entries and a frame ontology.The development of the ontology is based on the frames. A

Arg0	extern argument (proto-Agent)
Arg1	intern argument (proto-Patient)
Arg2	indirect object / beneficiary / instrument / attribute / end state
Arg3	start point / beneficiary / instrument / attribute
Arg4	end point
ArgA	external causer

Table 1: PropBank argument labels

frame represents a certain prototypical situation which is described by the frame definition. Every frame contains also a list of frame elements and a set of lexical units that can evoke the frame. The term lexical unit is used for a word in combination with one of its senses (Johnson et al. 2002). The frame elements fulfill a certain semantic role within the situation that is evoked by one of the lexical units. For every lexical unit a set of sentences is selected that illustrate all possible occurences of the lexical unit; all possible semantic roles are annotated in these sentences.
For example, the verb *leave* would evoke the frame *Departing* which is (partly) shown below:

- Departing
  An object (the Theme) moves away from a Source. The Source may be expressed or it may be understood from context, but its existence is always implied by the departing word itself.
- Frame Elements: Source, Theme, Area, Depictive, Distance, Manner, Goal etc.

A sentence annotated with semantic roles on the basis of the FrameNet information, would receive the following representation:

(2) $[_{Theme}$ We all] left $[_{Source}$ the school] $[_{Time}$ at four oclock].

Although FrameNet is still under development, its approach has been adopted for the annotation of semantic roles for languages other than English. An example is provided by the German project *Saarbrücken Lexical Semantics Annotation and analysis* (SALSA) (Erk et al. 2003), but there are also projects based on FrameNet for languages such as Spanish, French and Japanese. However, SALSA distinguishes itself from the others by the fact that it is not restricted to building a lexicon but it annotates the complete German Tiger corpus (Brants et al. 2002) using the FrameNet dictionary and adapting it to German. Unlike FrameNet, SALSA is not committed to always assigning a single sense (frame) to a target expression, or a single semantic role to a constituent but either more than one or an *Underspecified* sense tag can be assigned in case of vagueness or ambiguity.

4.3. Comparing approaches

The main differences we have noticed between FrameNet and Propbank are related to the methodology employed in the construction of the lexicon and the way the lexicon is structured. More generally, the classification attested in PropBank is based on *word senses* which are grouped in the 'shallow' framefiles while the FrameNet classification is driven by the *concepts* which are structured in the ontology of frames and thus based on hierarchically structured semantic classes.
Furthermore, the two projects differ with respect to the granularity of the role labels employed. FrameNet uses labels which immediately reflect the semantic role of the constituent and its annotation is rich in information. PropBank labels require more careful investigation about the meaning of the constituent in question. The difference is clearly illustrated in the following examples by (Palmer et al., 2005):

(3) FrameNet

a. $[_{Buyer}$ Chuck] bought $[_{Goods}$ a car] $[_{Seller}$ from Jerry]$[_{Payment}$ for 1000 dollars].

b. $[_{Seller}$ Jerry] sold $[_{Goods}$ a car] $[_{Buyer}$ to Chuck]$[_{Payment}$ for 1000 dollars].

(4) PropBank

a. $[_{Arg0}$ Chuck] bought $[_{Arg1}$ a car]$[_{Arg2}$ from Jerry] $[_{Arg3}$ for 1000 dollars].

b. $[_{Arg0}$ Jerry] sold $[_{Arg1}$ a car]$[_{Arg2}$ to Chuck]$[_{Arg3}$ for \$1000].

In (3a) and (3b) the labels immediately and clearly reflect who is buying and who is selling, but that is not the case in (4a) and (4b); there the seller and the buyer carry different labels depending on the verb. On the other hand, FrameNet does not immediately indicate that the subject in both cases is the Agent, as in the PropBank notation which uses the Arg0 tag to label it. The subject is not necessarily the Agent, as shown by the passive sentences below:

(5) PropBank

a. $[_{Arg1}$ A car] was bought $[_{Arg0}$ by Chuck].

b. $[_{Arg1}$ A car] was sold $[_{Arg2}$ to Chuck]$[_{Arg0}$ by Jerry].

c. $[_{Arg2}$ Chuck] was sold $[_{Arg1}$ a car]$[_{Arg0}$ by Jerry].

(6) FrameNet

a. $[_{Goods}$ A car]was bought $[_{Buyer}$ by Chuck].

b. $[_{Goods}$ A car] was sold $[_{Buyer}$ to Chuck]$[_{Seller}$ by Jerry].

c. $[_{Buyer}$ Chuck] was sold $[_{Goods}$ a car]$[_{Seller}$ by Jerry].

The examples above also indicate that the PropBank scheme maintains a more direct relation with the syntactic structure of the sentence than the FrameNet one.
The FrameNet labels are rather rich in information, however, they might not always be transparent for users and annotators. On the other hand, the advantage of the PropBank appoach is that by employing neutral labels, less effort is required from annotators to assign them. Furthermore, it creates the basis for the development of semi-automatic annotation of role labels, which is a necessary requirement if we want to annotate large corpora.

5. Merging approaches

In developing a scheme for the semantic annotation of the D-coi corpus, we are faced with several options.
We could assume the FrameNet approach and develop a Dutch lexicon based on the English (and German) one and employ it for the annotation of the Dutch corpus. We would thus follow the strategy employed within the SALSA project and we could even exploit their results given the similarity between Dutch and German. A disadvantage of this choice is related to the fact that in order to annotate the corpus further we are bound to construct new frames (with their definitions and their frame entities) manually and this is a rather expensive process. Furthermore, we believe that the labels used to identify the frame entities are not very transparent and difficult for annotators to use.
The other possibility would be to employ the PropBank approach which has the advantage of providing clear role labels and thus a transparent annotation for both annotators and users. Furthermore, the annotation process could be at least semi-automatic. However, a disadvantage of this approach is that we would have to give up the classification of frames in an ontology which could be very useful for certain applications, especially those related to the Semantic Web.
Within the D-coi project, we have chosen for a third option which wants to reconcile the rather pragmatic PropBank approach to role assignment which is essentially corpus based and syntax driven with the more semantic driven FrameNet approach which is based on a network of relations between frames. More generally, we would like to adopt the conceptual structure of FrameNet, but not necessarily the granularity of its role assignment approach. With respect to role assignment, we would lile to adopt the annotation approach of PropBank.[3]
In order to assess the feasibility of our approach we have carried out two pilot studies. The former one is based on the integration of PropBank and FrameNet with respect to what we consider a language independent phenomenon such as the classification of verbs of communication while the latter one considers a more language specific phenomenon, such as the classification of (adjunct) middle verbs. The goals behind these pilots are:

[3]A risk we take is that we will end up with a semantic annotation layer which is too similar to the syntactic representation which is assumed in D-coi. This will be an extension of that developed for the Spoken Dutch Corpus, that is a dependency structure which carries information about heads, complements and modifiers.

- to assess whether it is possible to merge FrameNet frames with PropBank role labels and whether this merging has to be manual or whether it is possible to make it at least semi-automatic;
- to investigate to which extent we can use resources already developed for other languages;
- to assess whether we can extend existing resources on the basis of our language specific annotation and whether we should include the language specific features in the original resource;
- to investigate whether it is possible to extend the merged resources by exploiting the best features of both and in this way facilitate the process.

5.1. Merging FrameNet and PropBank: the Communication frame

As previously discussed, FrameNet provides a rich semantic representation of language because its lexicon not only encodes word senses but also relations among words. Words can be related to each other on the basis of the frame they share, but also because a relationship among frames is established (Fillmore et al. 2004)). The ontological relations add extra information to word senses.
In FrameNet, every frame has its own definition which distinguishes it from other frames while the frame elements can be the same across frames due to (partial) inheritance. However, there is a great variety of elements that are frame specific creating thus a quite complex structure which is not always very transparent for annotators and users. Furthermore, by taking into account the *Communication Frame*, which comprises a mother frame with six daughters, we have noticed that the inheritance relation is not as strict as we had assumed.
Our aim is thus to reduce the FrameNet frames to a simpler form in which the set of frame elements is restricted to a number of elements that is comparable with the PropBank arguments. Since the interpretation of a PropBank argument label depends on the word senses of the individual word, we wanted to make their interpretation more uniform as well. This can be achieved by assuming Levin's classes and diathesis alternation (Levin 1993) and the revisions implemented within VerbNet (Kipper et al. 2002). Verbs within the same Levin class, sharing the same diathesis alternations, should have the same roleset. Thus, the second step is to group together those verbs that share the same FrameNet frame, the same Levin class and diathesis alternation and assign this group one roleset; this roleset is derived from PropBank by selecting the most common arguments from one group of verbs. Regrouping the verbs this way decreases the number of rolesets in comparison with the number of PropBank rolesets. The advantage is that we can determine rolesets using a simple algorithm that results in an intersection of the FrameNet and the Levin classification (cf. also (Giuglea and Moschitti 2004)). Assigning rolesets to these newly created classes takes less effort than manual role assigment for every individual verb. We then compared this roleset with the frame elements that are normally used.

As a test case we took the frame *Communication* which has six daughters: *Communication-manner, Communication-noise, Communication-response, Gesture, Reassuring and Statement*. For example, the verbs comprised in *Communication-noise* belong to four different Levin classes with the majority of the verbs belonging to the class *Verbs of Manner of Speaking*. These are verbs like *babble, bellow, croon, hiss, wail, whine* etc. A general roleset for this group could be:

PropBank	FrameNet
Arg0: speaker, communicator	Speaker
Arg1: utterance	Message
Arg2: hearer	Addressee

Table 2: Roleset for *Communication-Noise*

The alignment in the case of the communication verbs was rather straightforward and it seems to suggest that indeed this methodology might be appropriate, the question is whether this would be in general the case and whether we would encounter more difficulties with other predicates. It is for this reason that we worked out another case based on a language specific construction, that is adjunct middles.

5.2. (Adjunct) middles in Dutch

The second test case we have taken under consideration involves the argument structure of the middle construction in Dutch and thus the annotation of sentences in which this construction is attested (Trapman 2005). The middle construction can be characterized by an active sentence which receives a passive interpretation, as exemplified by (7b). Example (7a) represents the active equivalent of (7b). Even though the verb in (7b) exhibits active morphology, the object represented by *zijn laatste roman* in (7a) fills the subject position in (7b) and the sentence receives a passive interpretation:

(7) a. De winkel verkocht zijn laatste roman helemaal niet.
’ The store didnt sell his last novel at all.’

b. Zijn laatste roman verkocht helemaal niet.
’His last novel didn’t sell at all.’

It should be noticed that the Agent represented by *de winkel* doesn’t surface in (7b) but it is still present since the sentence receives a generic interpretation. Another property of the middle construction is that the verb is always followed by a modifier. The modifier can be an adjective, but also negation or simply stress. It has a dyadic character: it pertains to the subject and at the same time it implies the presence of an Agent.
Dutch differs from English in that the middle construction is attested not only with objects surfacing in the subject position but also with adjuncts appearing in this position. Example (8a) shows an active sentence containing a middle verb and an adjunct which is a prepositional phrase. In the next sentence, which receives a middle interpretation, the Agent is not present and the adjunct is in the subject position. Notice that the preposition has disappeared as well:

(8) a. Hij zit lekker op deze stoel.
’He sits well on this chair.’

b. Deze stoel zit lekker.
’This chair sits well.’

This kind of middle formation is subject to certain restrictions. First, adjunct middle formation occurs almost only with intransitive verbs. (Peeters 1999) divides them in three classes:

- verbs of position (sit, rest, lean etc);
- verbs of physical activity, implying no locomotion (knit, shake, write etc.);
- (agentive) verbs of manner of motion (expressing no directional endpoint) (travel, skate, walk etc.).

Second, there are only three types of adjuncts that are suitable for middle formation: adjuncts that represent an instrument, a location or an external circumstance (Peeters 1999). Sentences (9a), (9b) and (9c) give an example of each type:

(9) a. Deze stoel zit lekker.
’This chair sits comfortably.’

b. Deze zee vaart rustig.
’This see sails peacefully.’

c. Regenweer wandelt niet gezellig.
’Rainy weather does not walk pleasantly.’

The semantic annotation of the adjunct middle construction represents an interesting challenge both for FrameNet and PropBank. The examples given below are an attempt to annotate sentence (9a), (9b) and (9c) with FrameNet and PropBank role labels:[4]

(10) FrameNet

a. $[_{Goods}$ Zijn laatste roman$]$ verkocht helemaal niet (CNI: Seller) .

b. $[_{Location}$ Deze stoel $]$ zit lekker (CNI: Agent).

c. $[_{Area}$ Deze zee $]$ vaart rustig (CNI: Driver).

d. $[_{Depictive}$? Regenweer $]$ wandelt niet prettig (CNI: Self-mover).

(11) PropBank

a. $[_{Arg1}$ Zijn laatste roman $]$ verkocht $[_{ArgM-MNR}$ helemaal niet $]$.

b. [? Deze stoel $]$ zit $[_{ArgM-MNR}$ lekker $]$.

c. [? Deze zee $]$ vaart $[_{ArgM-MNR}$ rustig $]$.

d. [? Regenweer $]$ wandelt $[_{ArgM-MNR}$ niet prettig $]$.

These few examples immediately reveal an important difference between FrameNet and PropBank: the former adopts a more fine grained distinction in role labelling which is driven by semantics while the latter adopts a less fine grained distinction which is syntactically driven. In

[4]CNI: Constructionally licensed Null Instantiation. This means that the external argument frame element is unexpressed (Johnson et al. 2002).

the case of the adjunct middles, we notice that PropBank doesn't offer us the means to label the constituent located in the subject position. Even if the role set contains an argument label that is suitable, this label represents only a location, sometimes an instrument, but it is never possible to represent all three types of adjunct middle correctly using the available role set. In this case, FrameNet gives a more satisfactory result but even the more exhaustive representation of FrameNet is not sufficient to represent every type of adjunct middle. Example (10d) represents an adjunct middle with an adjunct of external circumstance on subject position. We have labeled it as a Depictive, but it is not quite correct since the definition of Depictive is 'Depictive phrase describing the actor of an action'. This points at an internal circumstance, but for now this label is the best alternative that is available.[5]

To conclude, this construction poses problems both for FrameNet and for PropBank, thus the merging approach doesn't provide a solution since we are dealing with a language specific phenomenon which needs a language specific solution. Actually, even linguistic theory doesn't provide us with a clear answer on what the role of the element in subject position is supposed to be. This case can be considered a typical example of how D-coi could contribute to the further development of language resources by extending the available notation on the basis of language specific input. However, we still need to investigate how the modifications due to the language specific phenomena relate to the original annotation scheme.

6. Omega: a comparison

The approach sketched in this paper aims at an integration of PropBank and FrameNet, that is we would like to adopt the conceptual structure of FrameNet and to merge it with the role assignment procedure of PropBank. There seem to be similarities with the integration of language resources which has taken place within the Omega project (Philpot et al. 2005).

Omega, is a 120,000-node terminological ontology concived as the reorganization and synthesis of different language resources which include WordNet and Mikrokosmos (O'hara et al. 1998), a conceptual resource originally developed to support translation. A more recent addition to the ontology has assigned frame information to each word sense of a predicate based on FrameNet and PropBank lexicons. However, due to lack of detailed description of the methodology adopted it is hard to evaluate whether the approach sketched here and the one developed within Omega are equivalent.

It seems to us that Omega *aligns* different resources, including thus FrameNet and PropBank while we are *merging* certain specific features of FrameNet (i.e. ontology of frames) with others of PropBank (role labelling approach). It would be relevant to assess whether the two methodologies provide ultimately the same results. A relevant test case in this respect would be how to extend an ontology such as Omega, which is the result of the alignment of several resources. That is, whether all the resources will be extended separately and then subsiquently aligned or whether specific features of a given resource are exploited in order to extend further the resources it is aligned with. The latter would be the case in our approach since we would like to exploit the semi-automatic role assignment of PropBank to annotate new predicates and combine it with the conceptual structure of FrameNet: through inheritance roles are shared among predicates belonging to the same frame.

In oder to carry out a more detailed comparison between our approach and that of Omega, we plan to carry out a pilot study which will employ Omega to annotate the D-coi corpus. A possibility would be to exploit the alignment already established in Omega among Wordnet, FrameNet and PropBank so that the semantic annotation of the Dutch corpus could be achieved automatically through an additional alignment of the Dutch equivalent of Wordnet, that is either the Dutch Wordnet developed within Euroword-net (Vossen 1999) or the new lexical resource which will be developed within CORNETTO, which is one of the new projects launched within the STEVIN programme.

7. Integration of annotation schemas in D-coi

As already mentioned, within the D-coi project a choice has been made, with respect to which types of semantic annotation should be developed: annotation of semantic roles as well as of temporal and spatial semantics. This choice has been motivated both by pragmatic reason (i.e. semantic role assignment is one of the most attested and feasible types of semantic annotation within corpora) and by the need for such a layer of annotation within several applications such as Semantic Web, informational retrieval or question answering.

At the moment, we keep the two annotation levels separately, to make it easy to produce alternative annotations of a specific type of semantic information without need to modify the annotation at the other level. By keeping the different types of annotation separately, it will be possible to enable progress on techniques for one type of semantic processing without need to wait for the development of high-performing systems for other aspects of semantic interpretation. However, we are aiming at a comprehensive annotation scheme which should ensure compatibility among the various types of semantic information. The need would be even more urgent if in a follow-up project, it will be possible to integrate the results obtained with respect to semantic annotation in current STEVIN projects. It would desirable to incorporate the results of COREA,[6] which deals with annotation of coreference and CORNETTO which will build a lexical semantic database for Dutch with rich vertical and horizontal semantic relations and with concepts aligned with the English Wordnet.

Since all linguistic levels interact closely in order to determine the meaning of a whole sentence, the meaning of an expression will be characterized not only by its word

[5]It should be noted that the original classification of (Peeters 1999) cannot be traced back in FrameNet. Instead, the 56 middle verbs that were investigated have a direct inheritance or use relationship with one of the frames of *Motion, Posture and Intentionally-act.*

[6]http://www.cnts.ua.ac.be/ hoste/corea.html

meanings, but also by the manner in which they are put together: syntactic structure plays thus a relevant role. In the D-coi project, the two different types of semantic annotations will be carefully integrated with the other layers of annotation, that is syntactic and morphological. Allowing semantic annotation to proceed in parallel with the other levels of annotation is a great advantage. There are several examples of treebanks which were extended with semantic information at a later stage such as PropBank or the Prague Dependency Treebank. While these additions are possible, as discussed in (Simov and Osenova 2004), they are not trivial since they often require modifications in the previous annotations, such as changing the labels or some design principles. With D-coi, we are in the privileged position of developing these annotations in parallel, taking thus into account possible interactions and being able to exploit the available information. The guidelines developed in this respect can constitute the basis for further research as well as a reference for similar initiatives.

In particular, our input sentences are syntactically analysed in another layer using the Alpino parser,[7] in this way the meaning of an entity (expression, sentence) will not only be characterized by the meaning of the constituting words, but also by the manner in which these are put together. Note that for example in temporal semantics the interaction of a verb and other constituents (especially their prepositional and/or nominal heads) is crucial in order to decide whether the verb is refering to a bounded or an unbounded event. PoS information is also still available at the syntactic level, for example with respect to temporal information associated with the verbal forms. Similarly, in the case of semantic role labelling, the syntactic structure encoded will guide the assignment of the role labels, this is the case in the distinction between arguments and modifiers.

8. Conclusion

In order to develop a scheme for the annotation of semantic roles within the D-coi project, we want to built on the results of similar initiatives. Particularly relevant for our purposes are projects like PropBank, FrameNet, SALSA and, more recently, Omega. In this paper, we have discussed their properties, their methodologies and assessed their approaches, in order to decide which one would be the most suited for our needs. However, in order to make a choice, it is crucial to establish which criteria play a role in the development of a (semantic) annotation scheme.

One criterion is the kind of applications which will make use of our corpus. This means that further research should be carried out to investigate what do applications in the area of Information Extraction, Semantic Web, QA etc. require from a linguistic resource of this kind. Apart from being useful, we believe that such an annotation scheme should provide an appropriate representation of language. Therefore, a relevant question is whether a shallow semantic annotation such as that provided by PropBank will suffice or whether we need a richer scheme, as the one in FrameNet. However, adding more fine grained semantic information will take more effort while the necessary means might not be available. Finally, linguistic theory might play a relevant role in the choices we make and one wonders whether it is possible to be as theory neutral as possible. We believe that our proposal to merge the PropBank approach to role assignment which is essentially corpus based and syntax driven with the conceptual structure of FrameNet, takes the criteria mentioned above into consideration.

Furthermore, we have attempted a comparison between our approach and the alignment carried out within the Omega project and we have focussed on the distinction between alignment and merging. In our opinion these are two different operations. Alignment means to link two different schemes with each other, but keeping them as separate modules. This may cause problems if one of the modules is modified which makes the alignment either incomplete or maybe even not useful anymore. Merging implies alignment but goes one step further by integrating one scheme into the other, or perhaps collapsing two schemes into a third, new scheme. This requires careful consideration about which properties the new scheme will inherit from its sources and which additional actions have to be taken to make it satisfactory.

[7]http://odur.let.rug.nl/ vannoord/alp/

9. References

O. Babko-Malaya. 2005. *Guidelines for Propbank framers.*

S. Brants, S. Dipper, S. Hansen, W. Lezius, and G. Smith. 2002. The TIGER treebank. In *Proceedings of the Workshop on Treebanks and Linguistic Theories.* Sozopol.

K. Erk, A. Kowalski, S. Pado and M. Pinkal. 2003.Towards a Resource for Lexical Semantics: A Large German Corpus with Extensive Semantic Annotation. In *Proceedings of ACL 2003*, Sapporo.

C. Fillmore, J. C.F. Baker and H. Sato. 2004. FrameNet as a net. In *Proceedings of LREC*, Volume 4, 1091-1094. Lisbon, Elra.

A. Giuglea and A. Moschitti. 2004. Knowledge Discovery using FrameNet, VerbNet and PropBank. In *Proceedings of the Workshop on Ontology and Knowledge Discovering at ECML 2004*, Pisa.

C. R. Johnson, C. J. Fillmore, M. R. L. Petruck, C. F. Baker, M. J. Ellsworth, J. Ruppenhofer, and E. J. Wood. 2002. FrameNet: Theory and Practice.

P. Kingsbury, M. Palmer and M. Marcus. 2002. Adding Semantic Annotation to the Penn TreeBank. In *Proceedings of the Human Language Technology Conference. HLT-2002.* San Diego, California.

K. Kipper, M. Palmer, and O. Rambow. 2002. Extending PropBank with VerbNet semantic predicates. Unpublished manuscript, presented at Workshop on Applied Interlinguas, AMTA-2002, October.

B. Levin.1993. English Verb Classes and Alternations: A Preliminary Investigation. University of Chicago Press, Chicago.

M. Marcus, B. Santorini, and M. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics*, 19(2):313-330, June.

T. O'Hara, K. Mahesh and S. Nirenburg. 1998. Lexical acquisition with WordNet and the Mikrokosmos Ontology. *Proceedings of the COLING/ACL Workshop on Usage* *of WordNet in Natural Language Processing Systems.* Montreal, Canada.

N. Oostdijk, W. Goedertier, F. van Eynde, L. Bovens, J.P. Martens, M. Moortgat, and H. Baayen. 2002. Experiences from the Spoken Dutch Corpus Project. In M. Gonzalez Rodriguez and C. Paz Saurez Araujo, editors, *Proceedings of LREC-2002*, pages 340-347.

M. Palmer, D. Gildea, P. Kingsbury. 2005. The Proposition Bank: A Corpus Annotated with Semantic Roles, *Computational Linguistics*, 31:1.

A. Philpot, E. Hovy and P. Pantel. 2005.The Omega Ontology. University of Southern California.

R. J. Peeters. 1999. The adjunct middle construction in Dutch. In: *Leuvense Bijdragen*, 88, pp. 355-401

K. Simov and P. Osenova. 2004. A Treebank-Driven Approach to Semantic Lexicons Creation. In *Proceedings of the Third Workshop on Treebanks and Linguistic Theories.* Tübingen, Germany.

J. Trapman. 2005. Where FrameNet meets the Dutch Spoken Corpus: in the middle. Bachelor thesis. Utrecht University. 2005.

P. Vossen. 1999. EuroWordNet Final Report. EuroWordNet (LE-4003 and LE-8328) Deliverable D041, University of Amsterdam.

XML-Based Representation of Multi-Layered Annotation in the PDT 2.0

Petr Pajas, Jan Štěpánek

Institute of Formal and Applied Linguistics
Charles University in Prague, Faculty of Mathematics and Physics
Ke Karlovu 3, 121 16 Praha 2, Czech Republic
E-mail: {pajas,stepanek}@ufal.ms.mff.cuni.cz

Abstract

In this paper we introduce a generic XML-based data format called PML (Prague Markup Language), which was specifically designed for the needs of representing rich multi-layered linguistic annotation and other data sources such as annotation dictionaries. PML is the main data format for the upcoming release of version 2.0 of the PDT (Prague Dependency Treebank). We first present the fundamental concepts and ideas behind the format, then describe how the annotation of PDT is represented in PML, and finally outline our plans for further improvement.

1. Introduction

During the last phases of the Prague Dependency Treebank 2.0 (PDT 2.0)[1] production, a final decision about the distribution format had to be made. In PDT 1.0 (Hajič et al., 2001), the prior version of the treebank, several legacy formats were used (namely a SGML based format named CSTS and an application specific text format named FS). Their common major disadvantage was that they followed the all-in-one approach where the data as well as all layers of annotation were intermixed in a single document. This becomes unacceptable for PDT 2.0 where two separate layers of annotation featuring dependency trees (namely the analytical and the tectogrammatical layer) are present. After investigating other possibilities and various existing data formats, including TEI (Sperberg-McQueen and Bournard, 2005), Annotation Graphs (Bird and Liberman, 2001), works of Ide and Romary (Ide and Romary, 2003) and the preliminary output from the ISO/TC 37/SC 4 (Ide and Romary, 2001) initiative, we have concluded that none of them (at least at that time) meets the specific demands of representing such a rich annotation as that of PDT 2.0. We thus decided to implement an entirely new data format.

Rather than following the unfortunate habit of creating a project-specific format, we came to a decision for seeking a more generic solution, to be potentially reused by us in our future projects as well as by others for their specific purposes. Based on our present experience with PDT 2.0, we have recognized some demands on data representation which seemed universally applicable within treebanking and other NLP-oriented projects. Some of the most fundamental of these demands that we carried in mind when designing the first version of PML are summarized in the following assorted list:

Uniformity of representation: Same or analogous constructions should be represented in uniform or analogous ways.

Following the stand-off annotation principle: Layers of annotation should be cleanly separated, both one from the other as well as from the original data set to which the annotations relate. This allows for making changes only to a particular layer without affecting the other parts of the annotation and data.

Unified cross-referencing and linking system: Both links to external document and data resources and links within a document should be represented coherently. It should be noted that a variety of types of external links is required by the stand-off approach. By following the same principles in all related data resources (data, tag-sets, and dictionaries), we may achieve their tighter interconnection.

Linearity and structure: The data format ought to be able to capture both linear and structured types of annotation. Linear type includes e.g. word and sentence order (in case of written text) or temporality (in case of speech data). As for the structural annotation, our primary concern is to allow capturing tree-like structures in a way that naturally mirrors their logical nesting.

Structured attributes: The representation should allow for associating the annotated units with complex and descriptive data structures, with expressive power similar to that of feature-structures.

Handling ambiguity: The vague nature of the language often leads to more than one linguistic interpretation and hence to alternative annotations. This phenomenon occurs on many levels, from atomic values to compound parts of the annotation, and should be treated in a unified manner.

Human-readability: It appears necessary to retain certain level of human-readability of the data representation. It is useful not only in the first phases of the annotation process, when the tools are not yet mature enough to reflect all evolving aspects of the annotation, but also later, especially for emergency situations such as unexpected data corruption, etc. It also helps the programmers while creating and debugging new tools.

Extensibility: On a general level, the format should be extensible to allow new data types, link types, and sim-

[1]See (Hajič et al., in preparation).

ilar properties to be added. The same should apply to all specific annotation formats derived from the general one, so that one could incrementally extend the vocabulary with markup for additional information.

XML based: XML (the eXtensible Markup Language) is a widely deployed format very easy to build on. Using XML as the underlying data representation allows us to make use of existing validation tools and schema languages for XML.

2. The format

PML (The Prague Markup Language), as of the specification released on the PDT 2.0 CD-ROM (in preparation), is our first step towards the new generic XML based data format. It proved capable of representing rich linguistic annotation of texts, such as morphological tagging, dependency trees, etc. in adequate and straightforward way.

The full specification of PML as well as various technical aspects are given in (Pajas and Štěpánek, 2005). In this paper, we only briefly introduce its internal conception.

The fundamental feature of PML is that it uses a metalanguage called *PML schema language* to provide a uniform way of defining individual XML-based vocabularies, or data formats, for specific applications. Each of these individual vocabularies, which we call *PML applications*, is thus formally described in an XML document called *PML schema*. PML schema provides not only syntactic information about data types and declarations of data structures, nesting, etc. like other schema languages such as DTD, W3C XML Schemas, Relax NG, but also elementary semantical information about the data which may help the processing tools "understand" the data. This semantical information is inserted into the declaration via so called *PML roles* which we discuss later in this paper.

When launching a new annotation project, one typically makes a decision about the annotation schema and underlying mathematical models used for representing the linguistics phenomena studied by the project. As both annotation schema and the underlying mathematical model are most often specified rather vaguely, the actual representation of the annotation very much depends on the data format used. From this perspective, the concept of PML schema also serves (to a reasonable extent) as a formal description of an annotation schema for a single layer of annotation with definition of all data types and structures involved and declarations of roles they play in the abstract annotation schema.

XML documents conforming to a PML schema are called *instances* of the PML schema or rather of the particular PML application the schema represents.

To illustrate these concepts, imagine the relations between an annotation schema, PML schema, PML application, and a PML instance, to be analogical to the relations between the abstract idea of a book, the DocBook XML DTD, the XML vocabulary that DocBook XML represents, and a particular XML document conforming to the DocBook DTD, respectively.

Standard validation schemata, such as Relax NG, can be derived from a PML schema automatically (using a XSLT stylesheet). Hence, formal consistency of instances of the PML schema can be verified using conventional XML-oriented tools.

All PML schemata are defined in the XML namespace `http://ufal.mff.cuni.cz/pdt/pml/schema/`, whereas all instances share a common dedicated XML namespace `http://ufal.mff.cuni.cz/pdt/pml/`.

2.1. Data types

PML identifies several basic types of abstract data structures (or annotation constructions) and offers a unified way of representing them in XML. These include:

attribute-value structures (AVS) i.e. structures consisting of attribute-value pairs. To avoid confusion with XML attributes, we usually refer to attributes of an attribute-value structure as *members*.

lists allowing several values of the same type to be aggregated in either an ordered or unordered list.

alternatives used for aggregating alternative annotations, ambiguity, etc. From the technical point of view, alternatives are very similar to unordered lists, however, the difference lies in the semantics. If A is a value of a list type and $B_1, \ldots, B_n$ are the lists members, then the whole tuple is considered to be the value of A, i.e. $A = \{B_1, \ldots, B_n\}$. With alternatives, the situation is different. If the value of A is an alternative of $B_1, \ldots, B_n$, then each of B_i should be considered as a possible value of A individually, that is $A = B_i$ for all $i = 1, \ldots, n$ (in fact, $A = B_i$ can be true for only one $i = 1, \ldots, n$, yet we do not know or cannot decide which one of them it is, hence the ambiguity).

sequences representing sequences of values of different types and also providing rudimentary support for XML-like mixed content. Whereas lists are homogeneous (all list members are of the same type), sequences are not. They consist of an ordered sequence of XML elements from a given list or text content (as specified in the PML schema). The type of each element in a sequence is implied by its tag name. Also, unlike list members, elements in a sequence may be further annotated using XML attributes.

links providing a unified method for cross-referencing within a PML instance and linking both among various PML instances (which includes links between layers of annotation) and to other external resources (in the present revision, these resources have to be XML-based).

enumerated types which are atomic data types whose values are literal strings from a fixed finite set. A special case of enumerated type is a constant, where the set of possible values is a singleton.

cdata types representing all character-based data without internal structure or whose internal structure is not expressed by means of XML elements, etc. These types are further specified by a format field, in order to allow validation and optimal in-memory representation, distinguishing between numbers, identifiers, arbitrary character data, etc.

2.2. PML Roles

As already mentioned, PML introduces a concept of so called *PML roles*, which is orthogonal to the concept of data typing. PML roles are assigned to a particular annotation construction (be it a specific member of an AVS, an element of a sequence, its attribute, etc.) defined in the PML schema.

The information provided by PML roles identifies a construct as a bearer of additional higher-level property of the annotation, such as being a node of a tree, being a unique identifier, etc. This information can be used by tools such as annotation editors, like TrEd (Hajič et al., 2001), to determine an adequate way of presenting the PML instances to users.

The set of roles available in the current version of PML specification was only chosen to cover the needs of the annotations of PDT 2.0 and is therefore very incomplete from the general point of view. So far, the following roles are defined:

TREES is a role indicating that members of a list or sequence with this role represent dependency or constituency trees.

NODE identifies a node of a dependency or constituency tree (this role may be assigned to structures or sequence-members).

CHILDNODES identifies node member representing a list of child-nodes of the node in a dependency or constituency tree (this role is only applicable to a member of a list type in a structure with role `NODE` or to a sequence in an element of role `NODE`).

ID is a role assigned to members or attributes uniquely identifying a particular parent construct (an XML element, structure, sequence, etc.) in the PML instance. Applications may use this information to provide look-up indexes for particular constructs. All values with role `ID` within a PML instance should be distinct.

KNIT is a role which can be used to mark certain links interconnecting two annotation layers as suitable for merging the two layers into a single annotation by embedding the referred construct of the lower annotation layer into the referring construct of the higher annotation layer.

ORDER is a role assigned to node members whose numerical value can be used to define a total ordering of a tree.

Of course other roles come to mind when one considers representing in PML other types of annotation, lexicons, and other linguistic data sources. We currently evaluate both possible extensions to the current set of PML roles as well as the possibility to open the set of roles entirely to the user.

2.3. Layering and Links

In PML, individual layers of annotation can be stacked one over another in a stand-off fashion. Typically, each layer of annotation has its own PML schema. The interconnection between annotation layers is represented by (preferably ID-based) links.

Presently, PML only provides guidelines for ID-based linking and cross-referencing. Other types of links may be represented in PML on a per-application basis.

Since PML is specifically designed for the purposes of representing linguistic data and resources, where each file typically contains many links to only a few external resources, we have decided to adopt indirect linking model for linking to external documents. It works as follows: Every PML instance starts with a standard header which, among other, contains a section where all referenced external documents are listed, in the following form:

```
<references>
  <reffile id="a" name="adata"
           href="doc73.a"/>
  <reffile id="v" name="vallex"
           href="vallex.xml"/>
</references>
```

Each `<reffile>` element contains information about one referenced document, namely an ID assigned to the document within the referencing instance, an optional symbolic name of the particular type of resource (e.g. of the annotation layer or lexicon it represents), preferably one that is stable over all instances of this particular layer of annotation, and the URL of the document. Links to these documents from an instance containing the above declaration could look like in the snippet below:

```
...
  <a>
    <lex.rf>a#doc73-w5</lex.rf>
    <aux.rf>
      <LM>a#doc73-w3</LM>
      <LM>a#doc73-w4</LM>
    </aux.rf>
  </a>
  <t_lemma>chodit</t_lemma>
  <functor>PRED</functor>
  <val_frame.rf>v#f2234</val_frame.rf>
...
```

The above snippet contains four PML references, the first three of which lead to elements with IDs `doc73-w5`, `doc73-w3`, and `doc73-w4` in the document `doc_73.a`, while the last one leads to an element with ID `f2234` in the document `vallex.xml`. We can see on this example that instead of repeating URLs of target documents, each target document is specified using the ID (`a` and `v` in this case) associated with it in the `references` section of the header of the instance.

By a convention, not a requirement of the current PML guidelines, all names of AVS members containing cross-references have the suffix `.rf`. Note also, that in the above example, the second and third links are aggregated as list members of a single AVS member named `aux.rf`.

We conclude this section with an example on how pointers into non-XML data, such as audio file, can be represented in PML.

```
...
<references>
  <reffile id="au1" href="spk1_129.ogg"/>
  ...
</references>
...
  <w id="w-12941_01-p1s1w1">
   <token>_SIL_</token>
   <audio>
     <time_start>600000</time_start>
     <time_end>4700000</time_end>
     <file.rf>au1</file.rf>
   </audio>
  </w>
...
```

2.4. Examples

To give the reader a better idea about the actual XML representation of the concepts discussed so far, we now present as an example a simplified version of the PML schema for PDT 2.0 analytical layer and a corresponding instance.

```
<?xml version="1.0" encoding="utf-8"?>
<pml_schema xmlns=
 "http://ufal.mff.cuni.cz/pdt/pml/schema/">
...
<root name="adata">
 ...
 <element name="trees"
          role="#TREES"
          required="1">
  <list type="a-root.type" ordered="1"/>
 </element>
</root>

<type name="a-root.type">
 <structure role="#NODE" name="a-root">
  ...
  <member name="children"
          role="#CHILDNODES">
   <list type="a-node.type" ordered="1"/>
  </member>
 </structure>
</type>

<type name="a-node.type">
 <structure role="#NODE" name="a-node">
  <member name="id" role="#ID"
          as_attribute="1" required="1">
   <cdata format="ID"/>
  </member>
  <member name="m.rf" role="#KNIT"
          type="m-node.type"/>
  <member name="afun" required="1">
   <choice>
    <value>Pred</value>
    <value>Sb</value>
    <value>Obj</value>
    ...
   </choice>
  </member>
  <member name="ord"
          role="#ORDER" required="1">
   <cdata format="nonNegativeInteger"/>
  </member>
  ...
  <member name="children"
          role="#CHILDNODES">
   <list type="a-node.type" ordered="1"/>
  </member>
 </structure>
</type>
  ...
```

We see that this PML schema defines the root element as a container for a list of trees represented by attribute-value structures of the `a-root` type. In PDT 2.0 the roots of the dependency trees are technical nodes which do not map to actual tokens of the annotated sentence. We have therefore skipped all declarations of its subsidiary members, except for the member `children` which comprises a list of nodes immediately descending from the technical root. These nodes are attribute-value structures of the type `a-node` whose members include the following: `id` for capturing a unique ID of the node (notice that the declaration requires that the `id` member is serialized into XML as attribute), an ID-based link `m.rf` associating an a-node with the corresponding unit (word) on the morphological annotation layer suitable for “knitting”, e.g. embedding into the a-node AVS, member `afun` for analytical function, `ord` whose integer value defines total order on all nodes of the tree, and finally, as above, a list of child-nodes of the type `a-node`. A corresponding instance might look as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<adata
 xmlns="http://ufal.mff.cuni.cz/pdt/pml/">
 <head>
  <schema href="adata_schema.xml" />
  <references>
   <reffile id="m"
            name="mdata"
            href="sample4.m.gz"/>
  </references>
 </head>
 ...
 <trees>
  <LM id="a-d79p1s1">
   ...
   <children>
    <LM id="a-d79p1s1w2">
     <m.rf>m#m-d79p1s1w2</m.rf>
     <afun>Pred</afun>
     <ord>2</ord>
     <children>
      <LM id="a-d79p1s1w1">
       <m.rf>m#m-d79p1s1w1</m.rf>
       <afun>Sb</afun>
       <ord>1</ord>
      </LM>
      <LM id="a-d79p1s1w3">
       <m.rf>m#m-d79p1s1w3</m.rf>
       <afun>AuxP</afun>
       <ord>3</ord>
       <children>
        <LM id="a-d79p1s1w4">
         <m.rf>m#m-d79p1s1w4</m.rf>
         <afun>Obj</afun>
         <ord>4</ord>
        </LM>
        </children>
       </LM>
      </children>
     </LM>
    </children>
   </LM>
   ...
  </trees>
</adata>
```

In the head section, the instance is associated with a corresponding PML schema and documents on the lower layers of annotation. Notice that all list members are uniformly represented by elements LM.

The following figure presents an analytical tree represented by the preceding PML instance as rendered by the annotation tool TrEd (Hajič et al., 2001). Note that the actual word forms appearing as labels under the nodes are obtained by following the links to the morphological layer.

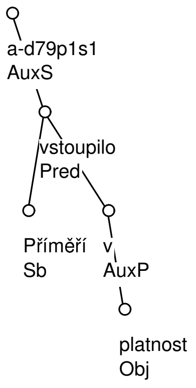


Figure 1: Visual representation of the example tree.

3. Representation of PDT 2.0

PDT 2.0 comprises of four layers of annotation, namely word layer (w-layer), morphological layer (m-layer), analytical layer (a-layer) and tectogrammatical layer (t-layer). Thus, in order to represent them in PML, we have defined four PML schemata, one for each layer.

The lowest layer is the word layer. It segments the text into documents (there is one document per file) and paragraphs, each of which consists of a flat sequence of tokens. Each document, paragraph, and token on the w-layer has a unique identifier and also a pointer to the original source of the data, which in the particular case of PDT 2.0 are documents in a SGML based format used by the Institute of the Czech National Corpus.

The morphological layer attaches to each w-layer token a structure comprising of morphological form, lemma, and tag (Zeman et al., 2005). The m-layer also introduces segmentation on sentence boundaries. Each morphological form may relate to zero or more tokens of the w-layer. The case where a form relates to no w-layer token only occurs in cases where the sentence was originally misspelled and a certain token (usually a punctuation mark) otherwise required by grammar rules was completely skipped in the original text. The relation between the m-layer and w-layer is represented as links from a m-layer PML instance into the corresponding instance of the w-layer.

Annotation on the analytical layer (Hajič, 1998) in PDT 2.0, as already mentioned in Section 2.4., consists of a sequence of (analytical) trees pertaining to sentences of the m-layer. There is a 1:1 correspondence between nodes of the analytical tree and forms on the m-layer, represented by links from an a-layer instance into the corresponding m-layer instance.

Annotation on the tectogrammatical layer (Sgall et al., 1986) consists of a sequence of (tectogrammatical) trees each of which pertains to a certain analytical tree. The mapping between analytical and tectogrammatical trees is 1:1. However, the mapping between nodes of a tectogrammatical tree and nodes of the corresponding analytical tree is in general $N{:}M$ (with 0 possible both for N and M), and in some cases the later mapping even crosses tree boundaries (it never crosses a document/file boundary, though). Again, these relations are represented by links from t-layer instances into the corresponding a-layer instances.

Tectogrammatical and analytical trees are dependency trees, represented in PML commonly as nested attribute-value structures. In this representation, a node of a tree is an AVS with the PML role NODE. Each node has a dedicated member with the PML role CHILDNODES, which contains a list of child-nodes of the node. Because of the auxiliary character of root nodes of the dependency trees of PDT 2.0, the structure representing the technical root of the tree is of a different type then the rest of the nodes (i.e. has a different set of members).

The dependency trees both on the a-layer and t-layer are ordered trees, although the semantics for these two orderings is very different. The ordering of nodes of analytical trees is rather technical and simply mirrors the ordering of the underlying m-layer, while the ordering on the tectogrammatical trees is an integral part of the tectogrammatical annotation and has a strong linguistic interpretation based on communicative dynamism. The implementation of the ordering in PML is, however, common for both cases. Each node has a dedicated member with PML-role ORDER and integer value type. The value of this member represents the position of a particular node in the total order of the tree it belongs to.

The t-layer further contains annotation of coreference, represented in PML as ID-based links between tectogrammatical nodes (from an indexing word or pronoun to its antecedent).

Another part of the t-layer annotation which is worth mentioning here because of the specific way which was chosen for representing it in PML is the annotation of quoted parts of the sentences, which aggregates tectogrammatical nodes into so called quotation sets, consisting of nodes pertaining to a part of the text in quotes. The annotation further distinguishes between several types of quoting, such as citation, direct speech, use of meta-language, etc. Each tectogrammatical node may belong to zero or more quotation sets.

We represent quotation sets in PML by means of "coloring" nodes that belong to a quotation set, which technically means the following: Each node has a member quot which is a list of structures, each of which consists of two members, the first being an ID ("color") identifying a quotation set and the second a value determining the type of the quotation. In this representation, a node belongs to a quotation set if and only if the ID of the quotation set is listed in the `quot` attribute of the node.

The solution used for annotation of quotation sets has the obvious advantage that it does not introduce any new annotation structure parallel to the tectogrammatical tree. On the other hand, it has some undeniable disadvantages, too, namely the following:

1. although each quotation set has its own unique ID, it is not represented by any explicit object. Instead, the ID serves as an artificial name of a common property of nodes belonging to the same quotation set.
2. the information about the type of the quotation is repeated with each member of the quotation set, although it is constant over the quotation set. Hence, it falsely appears as a property of an individual node rather than property of the set it belongs to.

It should be noted that no other part of the tectogrammatical annotation actually relates to the annotation of the quotation-sets. Hence introducing an extra annotation layer stacked over the t-layer would seem to be a cleaner solution in this case. In Section 5.6. below, we discuss several criteria for choosing an adequate representation in similar situations.

4. Current limitations and problems

Although PML proved mature enough to capture all aspects of annotation as complex as the four annotation layers of PDT 2.0, we have observed several problems that we should deal with to improve applicability of PML to other resources.

On the technical level, the fact that mutli-layered annotation of a particular piece of data is divided into several PML instances (one per layer) with internal links from one to another causes that PML instances are not copy safe. Using relative filenames works around the problem as long as all related PML instances are always copied together. Similar problem of this kind arises when PML-encoded files are compressed with `gzip` (e.g. in order to fit on a CD-ROM), even in case that all tools used in further processing are able to handle `gzip`-compressed files transparently. The reason is that `gzip` compression adds a `.gz` suffix to each file, which also breaks internal links. All this complicates implementation of functions such as *Save As* in GUI applications working with PML documents.

On the data-representation level, there are several limitations that we are aware of and will try to eliminate in a future revision of the PML specification. These include: no direct support for versioning, no support for inclusion of raw XML data from a foreign namespace, no support for modularization of large PML schemata, no type inheritance or ambiguity (e.g. it is not possible to specify that, say, a section either contains a list of other sections or a list of paragraphs without allowing both at the same type). We outline possible solutions to these limitation in the following section.

5. Future improvements

5.1. Modularity

The PML schemata for PDT 2.0 are strict and intended as an output format. Since PML is being adopted as the base format for several further projects building on top of PDT 2.0, there is a natural demand for new schemata that would be more open and therefore suitable for storing intermediate annotation data. Since we want to prevent over-using the "copy-and-modify" strategy where each project maintains its own modified copy of the original PML schemata, we plan adding extensive modularization support to PML. It includes:

Versioning of both the PML specification (i.e. the schema language) as such as well as of individual PML schemata.

Inclusion of one PML schema into another. This will also eliminate present duplication of schema code, which is currently a necessity in cases where one layer of annotation allows for embedding some units from another layer (PML role `KNIT`).

Overriding allows the top-level PML schema to selectively override declarations of certain types within the included PML schema, greatly reducing the amount of duplication of declarations when deriving a new PML schema from an existing one. Since both inclusion and overriding can be handled by a PML schema pre-processor, it is completely transparent to existing applications.

Type inheritance for AVS data types. Currently, the set of structure members and their value types, as specified in a PML schema, is strictly unambiguous, which makes it very difficult to define complex type hierarchies. If PML provided basic support for type inheritance for attribute-value structures, then one could use abstract types to specify common properties of data structures of a certain kind and then use them as a basis for deriving specific types, with the important aspect that values of all the derived types could be used equally as instances of the abstract type.

5.2. Per-instance typing

We are also evaluating the possibility of adding certain level of support for instance-driven typing. In this approach, similar to inheritance, but not limited to AVS-based data types, the exact type of a certain data field is not fully specified by the PML schema but may be on some level freely controlled by the instance.

5.3. Foreign XML namespaces

In certain situations it seems useful to allow preserving non-PML XML markup as direct part of the annotation. One of such cases is when there is an existing widely recognized XML vocabulary representing a given type of information. For example, should some formal representation of mathematical expressions and formulae be part of the annotation, then directly embedding MathML (http://www.w3.org/Math/) vocabulary into the PML annotation would seem the most natural solution. In order to make such a thing possible, we only need to identify foreign XML fragments as a new PML data type.

5.4. Metadata

Since PDT 2.0 data are not accompanied by much meta-data, recommendations for a uniform representation of such information have not been included in the first version of the PML specification. In order to fill this gap, we consider adding support for associating PML documents with meta-data encoded using some of the existing meta-data and resource description vocabularies, specifically Dublin Core (http://dublincore.org) and RDF (http://www.w3.org/RDF/).

5.5. Lexicon support

PDT 2.0 annotation is based on several annotation dictionaries such as morphology and PDT Valency Lexicon ValLex, (Hajič et al., 2003). We currently experiment with representing these types of resources in PML. Although lexicon data can often be represented as a tree-like hierarchy, these trees, unlike those occurring in syntactic annotation, do not involve recursion. In fact, their nodes are better viewed as items of (usually ordered) lists, often indexable for random access according to various index keys. We currently evaluate extending the set of PML roles by dedicated roles suited exactly for lexicon purposes, such as the following:

ITEM intended for structures representing lexicon items, such as words or phrases.

INDEXABLE specifically marks large lists for which applications might want to provide lookup support.

INDEXKEY marks AVS members as intended lookup keys in indexable lists (similarly, INDEXSUBKEY for lookup subkeys).

5.6. Representing class membership

At the end of Section 3., we have already mentioned some problems related to choosing adequate annotation model and representation for set membership (in the particular case of quote sets) and suggested introducing an extra annotation layer as a possible solution. In general, there can be found several criteria for choosing an appropriate approach to annotation of membership relation:

- Does each class represent a specific property of its members? Can these properties be associated with meaningful labels? If so, and the set of labels is finite and known in advance, then the labels should be listed in the PML schema and, in the annotation, attached to the class members. Otherwise it is advisable to add an extra layer representing the classes.

- Are the classes pairwise disjoint? If yes, then in order to assure more easily that each object belongs to at most one class, it is better to link from objects to classes.

- Should all the annotated objects belong to some class? If so, then, to assure that each object belongs to at least one class, the links should again lead from objects to classes. Also, the most common class (if there is one) could be taken as default, in case the classes are pairwise disjoint at the same time.

- If the classes are assumed to merge or divide often during the annotation process, then it is probably advisable to link from the class layer to the object layer.

- The expected number of classes and the number of objects belonging to at least one class should be considered. If both the numbers are low (compared to the total number of objects) then it is easier to link from classes to objects.

5.7. More atomic types

As already mentioned, in order to provide an extensive set of atomic data formats, we consider adopting external libraries of simple type borrowed from validation languages such as W3C XML Schema.

5.8. Intermediate layers

It has been noted that the PML-based formats of PDT 2.0 are designed as output formats and are not very suitable for the annotation process. One of the places where this becomes apparent, at least from the perspective of machine processing, is the fact that the annotation of sentence boundaries is part of the morphological layer. Although it is, at least in theory, possible that a sophisticated morphological tagger was responsible for breaking the token stream into sentences (note that for some languages such as Arabic certain knowledge of the morphological structure is necessary for both tokenization and segmentation), in practice it is most common that this step is done separately, prior to morphological tagging (or sometimes even prior to analysis). This results in demand for isolation of segmentation in an intermediate layer, produced by the segmentation tool and loaded up together with the word-layer by the morphological tagger. PML schema of such a layer could be as simple as:

```
<?xml version="1.0" encoding="utf-8"?>
<pml_schema xmlns=
 "http://ufal.mff.cuni.cz/pdt/pml/schema/">
  <root name="sb">
    <element name="start.rf">
      <list ordered="1">
        <cdata format="PMLREF"/>
      </list>
     </element>
  </root>
</pml_schema>
```

The corresponding instance would then look like (each link simply pointing to the first token in a sentence):

```
<?xml version="1.0"?>
<sb xmlns=
 "http://ufal.mff.cuni.cz/pdt/pml/">
  <head>
    <schema href="sb_schema.xml"/>
    <references>
      <reffile id="w" href="sample0.w"/>
    </references>
  </head>
  <start.rf>
    <LM>w#w-d56p1s1w1</LM>
    <LM>w#w-d56p2s1w1</LM>
    <LM>w#w-d56p2s2</LM>
    ...
  </start.rf>
</sb>
```

A similar approach can be used to define a generic PML schema for annotation of segment alignment in multilingual corpora, only in that case the list would consist of pairs (or vectors) of links with targets being segments (sentences, paragraphs, etc.) of data in the individual languages.

5.9. Generating API

The strict nature of data-typing in PML and its formal declaration in form of a PML schema allows us to create sophisticated code-generating tools which would transform the PML schema into API libraries in various programming languages providing optimal in-memory representation, parser, serialization, indexing, and other convenient routines, ready-to-use and tailor-made just for the data conforming to the particular PML schema.

6. Conclusion

We have introduced a new generic data format suitable for multi-layered annotation as well as various other NLP applications. We also reviewed how the format was applied to capture the rich structural annotation of the Prague Dependency Treebank 2.0. In the last part we have pointed out some current limitations and discussed in detail several areas in which we plan to further extend and refine the specification of the format.

7. Acknowledgment

This paper was written with the support of the grant GA AV ČR 1ET101120503 (Integration of language resources in order to extract information from natural language texts).

8. References

Steven Bird and Mark Liberman. 2001. A formal framework for linguistic annotation. *Speech Communication*, 33(1,2):23–60.

Jan Hajič, Eva Hajičová, Petr Pajas, Jarmila Panevová, Petr Sgall, and Barbora Vidová Hladká. 2001. Prague Dependency Treebank 1.0 (Final Production Label). CD-ROM. CAT: LDC2001T10.

Jan Hajič, Jarmila Panevová, Zdeňka Urešová, Alevtina Bémová, Veronika Kolářová-Řezníčková, and Petr Pajas. 2003. PDT-VALLEX: Creating a Large-coverage Valency Lexicon for Treebank Annotation. In E. Hinrichs J. Nivre, editor, *Proceedings of The Second Workshop on Treebanks and Linguistic Theories*, pages 57–68. Vaxjo University Press.

Jan Hajič, Barbora Vidová-Hladká, and Petr Pajas. 2001. The Prague Dependency Treebank: Annotation Structure and Support. In *Proceedings of the IRCS Workshop on Linguistic Databases*, pages 105–114, Philadelphia, USA. University of Pennsylvania.

Jan Hajič, Marie Mikulová, Alla Bémová, Eva Hajičová, Jiří Havelka, Veronika Kolářová-Řezníčková, Lucie Kučová, Markéta Lopatková, Petr Pajas, Jarmila Panevová, Magda Razímová, Petr Sgall, Jan Štěpánek, Zdeňka Urešová, Kateřina Veselá, and Zdeněk Žabokrtský. in preparation. The Prague Dependency Treebank 2.0. CD-ROM. http://ufal.mff.cuni.cz/pdt2.0/.

Jan Hajič. 1998. Building a Syntactially Annotated Corpus: The Prague Dependency Treebank. In Eva Hajičová, editor, *Issues of Valency and Meaning: Studies in Honour of Jarmila Panevová*, pages 106–132. Karolinum - Charles University Press, Prague.

Nancy Ide and Laurent Romary. 2001. Standards for language resources. In *Proceedings of the IRCS Workshop on Linguistic Databases*, pages 141–149, University of Pennsylvania, Philadelphia, 11-13.

Nancy Ide and Laurent Romary. 2003. Encoding syntactic annotation. In A. Abeillé, editor, *Building and Using Parsed Corpora*. Kluwer, Dordrecht.

Petr Pajas and Jan Štěpánek. 2005. A Generic XML-Based Format for Structured Linguistic Annotation and Its Application to Prague DependencyTreebank 2.0. Technical Report TR-2005-29, ÚFAL MFF UK, Prague, Czech Rep., December.

Petr Sgall, Eva Hajičová, and Jarmila Panevová. 1986. *The Meaning of the Sentence and Its Semantic and Pragmatic Aspects*. Academia/Reidel Publishing Company, Prague, Czech Republic/Dordrecht, Netherlands.

C. M. Sperberg-McQueen and Lou Bournard, editors. 2005. *TEI P5 - Guidelines for Electronic Text Encoding and Interchange*. The TEI Consorcium, January.

Dan Zeman, Jiří Hana, Hana Hanová, Jan Hajič, Barbora Hladká, and Emil Jeřábek. 2005. A Manual for Morphological Annotation, 2nd edition. Technical Report 27, ÚFAL MFF UK, Praha.

Sustainability of Linguistic Resources

Stefanie Dipper[†], Erhard Hinrichs[‡], Thomas Schmidt[*], Andreas Wagner[**], Andreas Witt[‡]

[†]Universität Potsdam
dipper@ling.uni-potsdam.de
[‡]Eberhard-Karls-Universität Tübingen
eh@sfs.uni-tuebingen.de
andreas.witt@uni-tuebingen.de
[*]Universität Hamburg
thomas.schmidt@uni-hamburg.de
[**]Universität Duisburg-Essen
andreas.wagner@uni-due.de

Abstract

This paper describes a new research initiative addressing the issue of sustainability of linguistic resources. This initiative is a cooperation between three linguistic collaborative research centres in Germany, which comprise more than 40 individual research projects altogether. These projects are involved in creating manifold language resources, especially corpora, tailored to their particular needs. The aim of the project described here is to ensure an effective and sustainable access of these data by third-party researchers beyond the termination of these projects. This goal involves a number of measures, such as the definition of a common data format to completely capture the heterogeneous information encoded in the individual corpora, the development of user-friendly and sustainably usable tools for processing (e.g. querying) the data, and the specification of common inventories of metadata and terminology. Moreover, the project aims at formulating general rules of best practice for creating, accessing, and archiving linguistic resources.

1. Introduction

This paper describes a new DFG[1] funded project (10/2005 – 12/2008) on preparation of language resources for assuring an accessible dissemination and a sustainable storing of these corpora. A main aim of the project is a practical one: resources acquired in long-term projects from three 'Collaborative Research Centres' have to be converted in one or several formats to be sustainably usable by researchers and applications. Furthermore it is envisaged to provide a unified access for the heterogeneous data acquired in the different involved projects. In addition to the preparation of already existing language corpora, general methodologies and 'Rules of Best Practice' should be developed.

The paper is structured as follows: Section 2. describes the resources of the three Collaborative Research Centres. These Centres are the SFB 538 'Multilingualism' at the University of Hamburg, the SFB 632 'Information Structure' at the University of Potsdam and the Humboldt University Berlin, and the SFB 441 'Linguistic Data Structures' at the Eberhard Karls University Tübingen.

Section 3. describes the technical aspects of the project. Especially the aspect of a data format usable as a general or meta-format for the formats in the three SFBs is adressed. Because of the heterogeneity of the formats we expect that such a format could serve as a meta-format for a wide range of XML-based annotation schemes.

Section 4. addresses the use of an appropriate set of meta data and the integration of formally defined termologies for enhancing interoperability of the annotated data.

[1]Deutsche Forschungsgemeinschaft, i.e. the German Research Foundation.

The paper ends with some remarks on the 'Rules of Best Practice', copyright issues and rights of personality.

2. Annotated resources and annotation schemes

2.1. Hamburg

2.1.1. Annotated resources

The research centre on multilingualism at the University of Hamburg comprises 14 projects doing research on diverse aspects of multilingualism. All projects work empirically, basing their analyses on digital corpora of written or transcribed spoken language. Apart from the spoken/written distinction, these data differ with respect to many more dimensions, but very roughly fall in one of the following categories:

- Longitudinal first language acquisition data of bilingual children - these are mostly transcriptions of video recordings of child/caretaker interactions;
- Other language acquisition data - this comprises sampled (as opposed to longitudinal) L1 acquisition data of mono- or bilingual children as well as data from L2 learners and from children with specific language impairments;
- Multilingual spoken communication data - this includes, for instance, transcribed radio broadcasts of Inter-Scandinavian communication, transcriptions of interpreter-mediated doctor/patient communication, Japanese/German expert discourse (e.g. business or academic communication) and semi-structured interviews with bilingual speakers from the Faroe Islands;

- Historical texts - examples of these are Old Swedish and Old Danish bible translations, 19th century letters by Irish emigrants and Old French legal documents;
- Modern texts - this comprises a parallel corpus of English and German business texts as well as a parallel corpus of popular science writing.

Apart from this conceptual diversity, the data in their original form also exhibited a great diversity on the technical level, in particular with respect to their storage formats (ranging from RDB-like over text-based to binary formats) and the tools with which they could be created, edited and analysed. Since this diversity made data exchange and data reuse extremely difficult, the EXMARaLDA system was developed to give these data a common structural backbone and thus to facilitate data exchange and data reuse as well as the construction of multi-purpose transcription and query tools.

2.1.2. Annotation schemes

Building on the idea of the annotation graph framework (Bird and Liberman, 2001), EXMARaLDA uses a time-based data model. This means that the primary relation between any two entities in a data set is established via their reference to a timeline, and not via their position in some other structure like, for instance, an ordered hierarchy. All non-temporal relations, like hierarchical inclusion or entity/feature relations, are regarded as secondary features that can be derived from this temporal structure. EXMARaLDA defines a *basic* and an *extended* data model for working with linguistic data.

The basic data model (a "Basic-Transcription") is a variant of the "Single Timeline, Multiple Tiers" model which is also used by a number of other systems or tools like Praat, ELAN, the TASX annotator or ANVIL. In general, these kinds of data model organise individual descriptions (*events*) into a number of *tiers* (or layers) and relate them to one another by assigning each description a start and an end point from a single, fully ordered *timeline*. In addition to that, the basic data model in EXMARaLDA requires that, firstly, no two events within a tier must overlap. Secondly, each tier can be assigned a *speaker* and must be assigned a *category*. Categories, in turn, fall into three *types*: T(ranscription) for tiers in which verbal behaviour is described, D(escription) for tiers in which non-verbal behaviour is described, and A(nnotation) for tiers in which stretches of transcribed speech are further categorised. This data model has proven adequate for the process of data creation as well as for many data visualisation tasks. In particular, its theory-neutrality makes it applicable for a wide range of researchers, its comparative simplicity facilitates the construction of intuitive user interfaces, and its similarity to the models of other systems (mentioned above) makes data exchange between EXMARaLDA and these systems a fairly straightforward matter.

The extended data model (a "Segmented-Transcription") caters for more complex tasks like querying and extensively annotating data, as well as for additional types of visualisation and for long term archiving. On top of the temporal structure encoded in the basic data model, it allows for the representation of additional linguistic structure. Most importantly, this means a segmentation of transcribed speech events into words and entities like turns, utterances or intonation units. Since these linguistically motivated units and the temporally motivated units in a Basic-Transcription do not have a uniform relation to one another (i.e. neither do their boundaries coincide in a regular way nor is one systematically included in the other), encoding them both in one document requires additional structural complexity. This is attained by allowing for a *bifurcating*, partially ordered timeline instead of the fully ordered one in the basic data model. In practice, the additional linguistic structure in a Segmented-Transcription is calculated automatically from the transcription convention regularities used in describing the temporal structure. Since different transcription conventions exhibit different such regularities, (and because they also define different linguistic units to begin with), data expressed in the extended data model is more dependent on specific linguistic theories than data expressed in the basic data model. For a more extensive discussion of EXMARaLDA's data model, see (Schmidt, 2005a) and (Schmidt, 2005b).

2.2. Potsdam / Berlin

The research centre SFB 632 at Potsdam University and Humboldt University Berlin investigates various facets of Information Structure (IS). IS concerns the means exploited by the speaker to structure discourse and utterances in order to convey information in a way that is optimised for the hearer in the given context. Languages differ a lot with regard to the means to express IS: by intonation, particles, word order, etc. The exact nature and interplay of many of these factors, however, is yet to be determined.

2.2.1. Annotated resources

The SFB consists of 13 individual research projects from disciplines such as theoretical linguistics, psycholinguistics, first and second language acquisition, typology, and historical linguistics. Following the overarching objective of providing a clearer picture of information structure, several of these projects are involved in collecting and analysing empirical data: Two projects examine the phenomenon of focus in different Western African languages; both carry out field studies for collecting data, which later is being annotated. One project investigates the role of IS in diachronic change, based on manuscripts of Old High German and Old English. Another project is developing a typology of the means for expressing IS. To this end, they have developed a language-independent questionnaire that is used to collect language data relevant for IS from speakers of typologically diverse languages, such as Hungarian, Greek, Georgian, Prinmi, Niue, Teribe, and Yucatec Maya; see, e.g., (Götze et al., to appear). Data sets elicited by the questionnaire consist of question-answer pairs, map-task dialogues, and short scenario descriptions. Finally, two projects focus on rhetorical and co-reference relations to address the relationship between discourse structure and IS.

According to the specific research interests of the individual projects, this data is annotated at different levels, according to SFB-wide common guidelines. Diachronic data is annotated by morpho-syntactic features and givenness information; the Old High German translation of Tatian is furthermore word-aligned to the Latin source text. Typological data is annotated by phonetic/phonological information (breaks, pitch-range, tones, etc.), morpheme-to-morpheme translations, part of speech, syntactic constituents and their thematic roles, animacy, etc. Discourse-related data is enriched by annotations according to Rhetorical Structure Theory (RST) (Mann and Thompson, 1988), co-reference and syntax annotations.

Currently, the corpora of these projects consist of several hundreds of data sets (for each of the languages of the typological data) and 20,000 German sentences (for discourse-related data).

2.2.2. Annotation scheme

To promote the active exchange of research hypotheses, the data is being collected in a single, uniform database, ANNIS. The database has to deal with highly heterogeneous data: First, primary data itself is heterogeneous, differing with respect to size (e.g., single sentences vs. entire articles), modality (monologue vs. dialogue), and language. Second, the annotations require data structures of various types (attribute-value pairs, trees, pointers, etc.). And finally, data is annotated by means of different, task-specific annotation tools: phonological, morphological and IS-related information, such as givenness, is annotated by EXMARaLDA, syntax by annotate, discourse structure by the RST Tool, and co-reference by MMAX2.[2]

Prior to import into the database, the data is mapped to a generic interchange format, PAULA[3]. This allows us to represent data annotations from different sources in a homogeneous way.

In our context, segments that annotations are attached to quite often overlap. The following example features an overlap between the phonemic and syntactic levels: at the phonemic level (= third tier), tokens 1 and 2, *de la* 'of the', are treated as one unit, whereas at the syntactic level, tokens 2–3, *la crème glacée* 'the ice-cream', form an NP constituent, cf. tier 4.

Token	*de*	*la*	*crème*	*glacée*
Gloss	some	the	cream	iced
Phonemic	**dla**		krEm	glase
Syntax	P	**NP**		

To account for such overlapping segments and for the heterogeneity of the data in general, PAULA uses an XML-based standoff architecture such that each annotation type is stored in a separate file. Annotations refer to the source text or to other annotations, by means of XLinks and XPointers. Building on proposals like in LAF (Linguistic Annotation Framework (Ide et al., 2003)), PAULA defines generic XML elements like `<mark>` (markable), `<feat>` (feature), `<struct>` (structure), and `<rel>` (relation), which allows us to represent, e.g., annotations attached to simple tokens as well as discontinuous segments, directed relations encoding anaphoric relations, and graphs to encode TIGER-like syntax trees or RST trees; for more details on the format, see (Dipper, 2005).

Currently, PAULA is used to represent data annotated by phonetic/phonological information, part of speech, morphology and lemma, syntax, rhetorical relations, anaphoric relations, and information structure.

For manual inspection of the data at multiple levels, we have developed the database ANNIS, (Dipper et al., 2004). ANNIS supports the concurrent visualisation of different types of annotations. The discourse view gives an overview on the discourse, while the table view enables easy and interactive access to multilayer annotations. The tree view displays syntactic structures.

The query facility of ANNIS offers a rich set of search operators that apply to primary data and annotations. It supports the use of wildcards and operators like precedence and dominance. Complex queries can be formulated by means of negation, logical "&" and "|" ('or'). Query results are displayed with the matching data (text and/or annotations) highlighted.

2.3. Tübingen

The principal concern of the collaborative research centre SFB 441 at University of Tübingen are the empiric data structures which feed into linguistic theory building. In order to approach this general issue from a considerable variety of research perspectives, SFB 441 comprises a number of projects (currently 15) each of which investigates a particular linguistic phenomenon, either concerning general methodological issues, or with regard to a particular language or language family. The respective research interests range from syntactic structures (such as coordination) in German and English, local and temporal deictic expressions in Bosnian/Croatian/Serbian or Portuguese and Spanish, to semantic roles, case relations, and cross-clausal references in Tibetan, to mention just a few.

2.3.1. Annotated resources

As empirical basis for their research, many projects create electronically accessible collections of linguistic data and prepare them to fit their particular needs. In most cases, these collections are corpora. However, a couple of projects deal with data (e.g. lexical information) which are more adequately represented by an Entity-Relationship based data model and thus are implemented in relational databases.

All data collections built within SFB 441 projects are assembled in one repository called TUSNELDA[4]. Especially, the different corpora are integrated into a common XML-based environment of encoding, storage, and retrieval. This integration is particularly challenging due

[2]`http://www.rrz.uni-hamburg.de/exmaralda/; http://www.coli.uni-saarland.de/projects/sfb378/negra-corpus/annotate.html; http://www.wagsoft.com/RSTTool/; http://mmax.eml-research.de/.`

[3]*Potsdamer AUstauschformat für Linguistische Annotation*, Potsdam Interchange Format for Linguistic Annotation.

[4]*TUebinger Sammlung Nutzbarer Empirischer Linguistischer DAtenstrukturen*, Tübingen collection of reusable, empirical, linguistic data structures.

to the heterogeneity of the individual corpora, which differ with regard to the following aspects:

- languages (e.g. German, Russian, Portuguese, Tibetan,...)
- text types / data types (e.g. newspaper texts, diachronic texts, dialogues, treebanks, ...)
- categories of information covered by the annotation / annotation levels (e.g. layout, textual structure, morpho-syntax, syntax, ...)
- underlying linguistic theories

The size of the individual corpora ranges from about 10,000 (Spanish/Portuguese spoken dialogues) to approx. 200 million words (German newspaper texts, automatically chunk-parsed). (Wagner, 2005) provides an overview of the corpora built by the individual SFB 441 projects.

2.3.2. Annotation scheme

Despite the diversity of the corpora in TUSNELDA, they all share the same generic data model: hierarchical structures. It is most appropriate to encode the phenomena captured in the TUSNELDA corpora by means of nested hierarchies, augmented by occasional "secondary relations" between arbitrary nodes in these hierarchies. This distinguishes TUSNELDA fundamentally from corpora whose annotation is based on other data models such as timeline-based markup of speech corpora or multimodal corpora (see especially subsection 2.1.). Such corpora encode the exact temporal correspondence between events on parallel layers (e.g. the coincidence of events in speech and accompanying gesture or the overlap of utterances) whereas hierarchical aspects are secondary. In TUSNELDA, however, hierarchical information (e.g. textual or syntactic structures) is prevalent, while capturing the exact temporal coincidence of different events in general is not of primary relevance in the research conducted within SFB 441.

Consequently, the annotation scheme developed for TUSNELDA encodes information as embedded (rather than standoff) annotation, immediately modelling hierarchical structures by XML hierarchies. Essentially, this decision rests on two major considerations. Firstly, this procedure makes it possible to utilise standard XML-aware tools (such as XML editors, format conversion tools, XML databases, or query engines), which are optimised for processing hierarchical XML structures so that they are well suited for embedded annotation, while providing at best rudimentary support for standoff annotation. Secondly, embedded annotation indeed is sufficient for encoding the data captured by the TUSNELDA corpora. Standoff annotation would be necessary if the structures to be encoded formed overlapping hierarchies, which cannot be modelled within a single XML document. However, the structures primarily encoded in the TUSNELDA do not overlap but can be integrated into a single hierarchy. For example, whereas syntactic structures constitute sub-sentential hierarchies, text structures define super-sentential hierarchies. Hence, these structures can be captured straightforwardly within a single XML document structure. Concurrent hierarchical units occur only marginally and are not of primary importance. These units concern the (physical) layout structure of the annotated texts, e.g. page boundaries. Such boundaries are marked by milestone elements (e.g. <pb/> for a page break), which do not violate the well-formedness of the document.

The following example, taken from the Tibetan Corpus in TUSNELDA (Wagner and Zeisler, 2004), illustrates the annotation of syntactic constituent structure, argument structure, and cross-clausal reference within an embedded environment. Syntactic constituents are encoded by the elements <clause>, <ntNode> (non-terminal node), and <tok> (token); their catogories are specified by <clauseCat>, <ntNodeCat>, and <pos> elements, respectively. Additional descriptions concerning individual constituents may be encoded within <desc> elements. A special case of such a description is the specification of the argument structure of a verb token. Especially, the subcategorisation frame realised in the current clause is encoded as <realFrame>, where each complement is represented by a <realComplement>. In the example, the first complement is not overtly realised within the clause (status="*empty*"). However, it is implicitly given by the context, i.e. it corresponds to the first complement of the previous clause. This correspondence is modelled by a <ref> (reference) element including a pointer (target) to the corresponding complement.

```
<clause>
  <ntNode>
    <tok>
      <orth>khra·phru·gu</orth>
      <pos>NOM:anim~pers</pos>
    </tok>
    <ntNodeCat>NP</ntNodeCat>
    <desc>
      <case>Abs</case>
    </desc>
  </ntNode>
  <tok id="v6">
    <orth n="2">med-tshug</orth>
    <pos>VFIN</pos>
    <desc>
      ...
      <realFrame>
        <realComplement id="v6c1"
                        status="empty">
          <role>POSS</role>
          <ref target="v5c1"> </ref>
        </realComplement>
        <realComplement id="v6c2">
          <role>EXST2</role>
        </realComplement>
      </realFrame>
    </desc>
  </tok>
  <clauseCat>simple</clauseCat>
</clause>
```

3. Technical aspects

The description of the three 'Collaborative Research Units' in Hamburg, Potsdam, and Tübingen demonstrates the large variety of language data and research interests. Consequently, different annotation schemes are used in these projects. In a way, this is a common situation in nearly every annotation related project and several standard solutions, e.g. the use of XSLT-based conversions, exist for dealing with this problem. But since this variety is also due to the variety of the given original data, i.e. audio, video, already annotated text, and raw text, we have to deal with a more fundamental problem. The different annotation schemes are based on different basic annotation methodologies. While some of the projects, especially projects in the SFB "Multilingualism", are using a graph-based methodology, others, especially the projects in the SFB "Linguistic Data Structures", use embedded markup where several annotation levels are mapped on a single annotation layer.[5]

3.1. Development of data formats

The data formats of the diverse collections of linguistic data should be converted to a uniform data format. This format must conform to widely accepted public standards. Furthermore, the data format must be supported by a wide variety of – ideally non-proprietary – software. Consequently the standards XML and Unicode have been chosen as a starting point. But using these standards does not suffice for a sustainable representation and storing of the data. Indeed, most of the existing data already use these standards.

XML and Unicode can be regarded as a base level of annotation. Two other important aspects of data formats for linguistic annotation are the use of the appropriate tagsets or annotation vocabularies and the use of a suitable data model for corpus annotation.

In the recent years, several general corpus annotation standards have been developed, e.g. TEI (Sperberg-McQueen and Burnard, 1994) or XCES (Ide et al., 2000). But, since in concrete projects specialised annotation schemes are important, further developments became necessary. The ISO TC37/SC4 developed an infrastructure, the already mentioned "Linguistic Annotation Framework (LAF)", to allow for combining general-purpose annotation formats (a dump-format) with specific annotation schemes (Ide et al., 2003).

Moreover, LAF defines a user extensible set of Data Categories and a user extensible Data Category Registry allowing for linking a corpus-specific annotation to a generic format. We intent to follow the LAF approach by combining the existing annotations, a generic annotation format (see below) and a linguistic terminology or ontology. (see also subsection 4.2.)

One of the main tasks of the project will be the development and implementation of a generic annotation format, i.e. a data model for the existing language data. The model must be applicable for all the language data already annotated in the projects involved.

[5]In this distinction a layer (or tier) is a technical realisation of an annotation, e.g. a single XML-file or a named directed path in an annotation graph, whereas level refers to an abstract level of description, e.g. in linguistics the levels of morphology, syntax, or semantics. (Bayerl et al., 2003)

In linguistics, hierarchical annotations are essential for embedding syntactic information in a corpus. Consequently a large percentage of the corpus data, especially the TUSNELDA data (see subsection 2.3.), require a hierarchical data model.

Graph based annotations, on the other hand, are the predominant data model for transcriptions of audio and video data and are the base of the EXMARALDA format. (see subsection 2.1.) Consequently, also these annotations must be represented in the uniform data format.

The data represented in PAULA (see subsection 2.2.) combine characteristics of hierarchical annotations and graph-based models. This is a typical situation for linguistic data annotated according to the standoff methodology (see (Thompson and McKelvie, 1997), (McKelvie et al., 2001)).

The variety of the data formats is a common situation for projects dealing with linguistic resources. Finding a meta-format suitable for covering all the data formats of the involved projects is a major task for the sustainable representation of corpus data. What is needed is a data format suitable for hierarchically annotated corpora as well as for graph based annotations.

As a starting point for such a format, we are currently evaluating the Nite Object Model (NOM, see (Carletta et al., 2003)).

3.2. Development of methods and tools for data distribution and data access

It is intended to produce and generate several distributions of language data. These distributions are optimised for distributing a whole collection of data, for a sustainable storing and for querying the data. The following methods of distribution are planned:

1. A human readable hardcopy of all corpus data;
2. An electronic version distributed as an offline medium (e.g. DVD);
3. A query interface accessible via the Internet.

For generating a printed version of the corpora XSLT stylesheets will be developed. The generated printable versions of the corpora can be archived and offered by libraries. For the electronic distributions (points 2 and 3) tools are to be implemented for the linguistic search in the data.

As described in section 2., in the SFBs involved query mechanisms for the respective data collections are already realised. However, the SFB's query mechanisms do have another focus, namely: power (the possibility of specifying complex search criteria), efficiency (short response time), and ease of use (input interfaces and output formats, should be comprehensible for linguists without advanced technical knowledge). Unfortunately, the criterion of sustainability is quite often in opposition to these criteria. For this reason new query mechanisms will be developed. For achieving a sustainable query interface, new tools will be based on XSLT and XQuery, since we expect these standards to be supported by software for a relatively long time.

4. Data Integretion

For an accessible storing of language corpora, the corpora must contain additional information. This additional information can be subdivided into two classes: (1) Information on the corpus itself, e.g. information on the participants of a conversation, the languages, the names of the transcribers, and (2) information on the meaning of the annotations, e.g. the tag w is used for annotating a word. The first class is traditionally termed "metadata". The second class of additional information is traditionally provided with the help of tag set documentations. At the moment, however, there is a tendency to use more or less formal apparata for this, namely terminologies or ontologies.

4.1. Metadata

It is intended to compile a comprehensive set of metadata. This set must adequately describe all the corpora of the SFBs. This implies that all the metadata already in use will be integrated and if necessary extended. Of course, in a second step the individual corpora are to be classified according to the extended set of metadata.

The metadata should be compatible with existing linguistic metadata standards, especially with IMDI[6] and the the metadata set of OLAC[7]. However in different aspects the new set of metadata will be more specific.

4.2. Integration of terminologies

As already recognized by several researchers the problem of combining existing, real annotation vocabularies with a repository of linguistic categories is a crucial one (Ide et al., 2003). Since we expect standard based solutions to meet the need of sustainability most appropriately, we intend to use and/or to produce a data repository on the base of OWL (McGuinness and van Harmelen, 2004), such as the resource GOLD[8].

We would like to start with an ontology such as GOLD and to successively extend the existing ontology with sub-ontologies[9] for all the annotated phenomena in the projects of the SFBs. Since it has been shown that GOLD is extensible and therefore applicable for diverse kinds of linguistically motivated annotation vocabularies (Goecke et al., 2005), we are quite confident, that GOLD is a good candidate for an appropriate base ontology for linguistic categories. A first study on the integration of the GOLD-Ontology will be presented in (Chiarcos et al., to appear).

Following the LAF proposal, in a second step a mapping from the annotation vocabularies to the ontology will be defined and implemented.

5. Outlook

This paper has focussed on data models, data formats and software tools for sustainable linguistic resources. There are, however, less technical aspects that have an equally relevant impact on sustainability. On the one hand, this concerns the way individual researchers or research projects approach their data handling in the first place - a lot of problems that arise with respect to sustainability of linguistic data could be avoided or at least mitigated if some basic agreement on a set of best practices (e.g. use of open standards and non-proprietary software, or a minimum set of metadata) could be achieved on a broad basis in the research community. Suggestions for such rules of best practice have been made, e.g. (Bird and Simons, 2003), and the project described here intends to elaborate on this work and contribute to its spreading in the research community. On the other hand, insecurities about questions of copyright and of individual rights of persons recorded for linguistic studies often constitute a major obstacle to making linguistic corpora available to a broader public. Here too, the project aims to investigate possible ways of overcoming these obstacles and to formulate rules of best practice. A more comprehensive description of these tasks will be provided in (Chiarcos et al., to appear).

[6]ISLE (International Standard for Language Engineering) Meta Data Initiative, see (Wittenburg et al., 2002)

[7]Open Language Archives Community, see (Bird and Simons, 2004)

[8]General Ontology for Linguistic Description, see (Farrar and Langendoen, 2003)

[9]These specific sub-ontologies are named Community-specific extensions (COPEs) in the GOLD-Terminnology)

6. References

Petra Saskia Bayerl, Harald Lüngen, Daniela Goecke, Andreas Witt, and Daniel Naber. 2003. Methods for the semantic analysis of document markup. In C. Roisin, E. Munson, and C. Vanoirbeek, editors, *Proceedings of the ACM Symposium on Document Engineering (DocEng 2003). pp. 161 - 170)*, pages 161 – 170.

Steven Bird and Mark Liberman. 2001. A formal framework for linguistic annotation. *Speech Communication.*

Steven Bird and Gary Simons. 2003. Seven dimensions of portability for language documentation and description. *Language*, 79:557 – 582.

Steven Bird and Gary Simons. 2004. Building an open language archives community on the dc foundation. In Diane I. Hillmann and Elaine L. Westbrooks, editors, *Metadata in practice*, pages 203 – 222. American Library Association., Chicago.

Jean Carletta, Jonathan Kilgour, Timothy J. O'Donnell, Stefan Evert, and Holger Voormann. 2003. The NITE Object Model Library for Handling Structured Linguistic Annotation on Multimodal Data Sets. In *Proceedings of the EACL Workshop on Language Technology and the Semantic Web (3rd Workshop on NLP and XML).*

Christian Chiarcos, Erhard Hinrichs, Timm Lehmberg, Georg Rehm, Thomas Schmidt, and Andreas Witt. to appear. From project data to sustainable archiving of linguistic corpora. In *Paper accepted at the E-MELD workshop 2006*, Ypsilanti.

Stefanie Dipper, Michael Götze, Manfred Stede, and Tillmann Wegst. 2004. ANNIS: A linguistic database for exploring information structure. In Shinichiro Ishihara, Michaela Schmitz, and Anne Schwarz, editors, *Interdisciplinary Studies on Information Structure (ISIS)*, volume 1, pages 245–279. Universitätsverlag Potsdam, Potsdam, Germany.

Stefanie Dipper. 2005. XML-based Stand-off Representation and Exploitation of Multi -Level Linguistic Annotation. In *Proceedings of Berliner XML Tage 2005 (BXML 2005)*, pages 39–50, Berlin, Germany.

Scott Farrar and Terry Langendoen. 2003. A linguistic ontology for the semantic web. *GLOT International*, 7(3).

Daniela Goecke, Harald Lüngen, Felix Sasaki, Andreas Witt, and Scott Farrar. 2005. Gold and discourse: Domain- and community-specific extensions. In *Proceedings of the E-MELD workshop on Morphosyntactic Annotation and Terminology: Linguistic Ontologies and Data Categories for Language Resources.*, Cambridge, MA.

Michael Götze, Stavros Skopeteas, Thorsten Roloff, and Ruben Stoel. to appear. Towards an exploration infrastructure for a cross-linguistic production data corpus. In *Proceedings of the Sixth International Tbilisi Symposium on Language, Logic and Computation*, Batumi, Georgia.

Nancy Ide, Patrice Bonhomme, and Laurent Romary. 2000. XCES: An XML-based Standard for Linguistic Corpora. In *Proceedings of the Second Language Resources and Evaluation Conference (LREC)*, pages 825 – 830, Athens.

Nancy Ide, Laurent Romary, and Eric de la Clergerie. 2003. International standard for a linguistic annotation framework. In *Proceedings of HLT-NAACL'03 Workshop on The Software Engineering and Architecture of Language Technology*.

William C. Mann and Sandra A. Thompson. 1988. Rhetorical Structure Theory: Toward a functional theory of text organization. *Text*, 8(3):243–281.

Deborah L. McGuinness and Frank van Harmelen. 2004. OWL Web Ontology Language. Technical report, World Wide Web Consortium.

David McKelvie, Amy Isard, Andreas Mengel, Morten Baun Møller, Michael Grosse, and Marion Klein. 2001. The mate workbench — an annotation tool for xml coded speech corpora. *Speech Communication*.

Thomas Schmidt. 2005a. *Computergestuetzte Transkription - Modellierung und Visualisierung gesprochener Sprache mit texttechnologischen Mitteln*. Peter Lang.

Thomas Schmidt. 2005b. Time based data models and the text encoding initiative's guidelines for transcription of speech. *Working Papers in Multilingualism, Series B*.

C. M. Sperberg-McQueen and Lou Burnard, editors. 1994. *Guidelines for Electronic Text Encoding and Interchange (TEI P3)*. Chicago, Oxford: Text Encoding Initiative.

Henry S. Thompson and David McKelvie. 1997. Hyperlink semantics for standoff markup of read-only documents. In *Proceedings of SGML Europe '97: The Next Decade - Pushing the Envelope*, Barcelona, Spain.

Andraes Wagner and Bettina Zeisler. 2004. A syntactically annotated corpus of Tibetan. In *Proceedings of LREC 2004*, pages 1141–1144, Lisboa, May.

Andreas Wagner. 2005. Unity in diversity: Integrating differing linguistic data in TUSNELDA. In Stefanie Dipper, Michael Götze, and Manfred Stede, editors, *Heterogeneity in Focus: Creating and Using Linguistic Databases*, volume 2 of *ISIS (Interdisciplinary Studies on Information Structure), Working Papers of the SFB 632*, pages 1–20. Potsdam.

Peter Wittenburg, Wim Peters, and Daan Broeder. 2002. Metadata proposals for corpora and lexica. In *Proceedings of the Third Language Resources and Evaluation Conference (LREC)*, pages 1321 – 1326, Las Palmas.

DEB Tools for Merging Linguistic Resources

Aleš Horák, Karel Pala

Faculty of Informatics, Masaryk University Brno
Botanická 68a, 602 00 Brno, Czech Republic
E-mail: {hales,pala}@fi.muni.cz

Abstract

In this paper we present new tools based on client/server XML database system called DEB II. Thanks to the versatility of the XML format used, this platform enables us to cover various applications, namely the management of the electronic readable dictionaries, wordnet-like lexical databases as well as ontologies for Semantic Web applications. In the paper the attention will be paid mainly to the tool DEBVisDic which allows us to merge different lexical resources and appears as an appropriate tool for future standardization of wordnet-like databases. First, the main features of the DEB II dictionary writing platform are outlined, and the implementation strategies of both server and the client part of the DEB II platform are briefly characterized. Second, the tool DEBVisDic, a complete re-implementation of the successful wordnet editor and browser VisDic used in the Balkanet EU project, is presented and its functionality is described. We also pay attention to merging lexical data, particularly the list of Czech valency frames VerbaLex, developed separately in XML format, with Czech Wordnet. We show an example of the merge which also indicates how DEBVisDic can serve as a means for such kind of integration. We also point out that this type of merge can be extended to other languages as it was done with Bulgarian and Romanian in Balkanet project.

1. Introduction

There is a need to handle various lexical resources that take the form of wordnets, ontologies, valency lexicons, framenets and others. For this purpose researchers seek for software systems that are able to store dictionary-like data using XML as the core element. Many dictionary publishing houses operate large systems with the complex functionality of so called lexicographic stations that manipulate XML in the last years (DPS Longman (McNamara, 2003), TshwaneLex (Joffe and de Schryver, 2004), iLEX (Erlandsen, 2004) or ShoeBox (sho,)). However, these and similar tools are not able to efficiently merge and manipulate resources obtained from data-driven NLP applications. Therefore, they cannot provide a universal environment for lexical database management as well as semantic networks and ontologies.

One of the most popular lexical resources in the NLP field is the Princeton WordNet (Fellbaum, 1998). It was followed by multilingual EuroWordNet 1, 2 projects (1998-99) (Eur,) and Balkanet project (2001-4) (Bal,) in which the wordnets for 13 languages have been developed (English, Dutch, Italian, Spanish, French, German, Czech, Estonian, Bulgarian, Greek, Romanian, Serbian and Turkish). Both projects are taking advantage of the Top Ontology which classifies the word stock of the mentioned languages and offers a link to ontologies studied nowadays in the field of Semantic Web (e.g. SUMO/MILO ontologies).

In the course of the Balkanet project's work the specialized software tools for browsing and editing wordnets have been designed and implemented, without whose the job could hardly have been performed. The first browser developed at Princeton is still used there and can be found at `http://www.cogsci.princeton.edu/~wn/` or `charity.princeton.edu`, within the EuroWordNet project the Polaris (and Periscope) tools have been implemented and used (M., 1998).

For Balkanet project the browser and editor VisDic has been prepared at the NLP Laboratory at the Faculty of Informatics Masaryk University (Horák and Smrž, 2003) since the development of the Polaris tool has been closed by 1999.

From what has been said so far it may seem that VisDic has been oriented mainly to the browsing and editing WordNet-like or other lexical databases. It would be more correct to say that it can be used for any kind of semantic network, particularly, we have in mind the networks that are related to the Semantic Web.

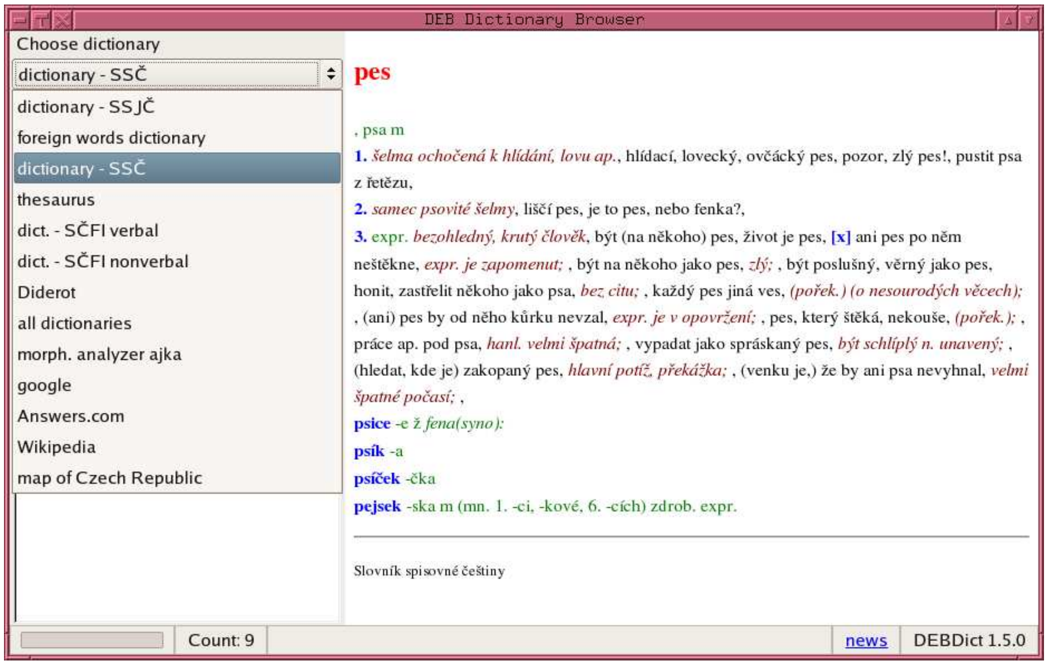
2. The Features of a Common XML Platform for Lexicographers' Tools

The DEB platform (DEB II, i.e. its second version) follows a strict client-server architecture. The actual development of applications within the DEB platform can be divided into the server part (the server side functionality) and the client part (graphical interfaces with only simple functionality). The server part is built from small parts, called *servlets*, which allow a modular composition of all services.

The clients communicate with servlets using HTTP requests in a manner similar to recently popular concept in web development called AJAX (Asynchronous JavaScript and XML (Rosenfeld and Morville, 1998)) with the usage of SOAP protocol (soa,). The data are transported (using plain HTTP) in RDF, generic XML or plain-text formats or they are marshalled using SOAP.

The actual data storage backend on the server side is provided by Berkeley DB XML, which is a native XML database providing XPath and XQuery access into a set of document containers. The metadata are stored in widely-used Berkeley DB embedded database which runs on many systems and devices ranging from Linux and Windows operating systems to mobile phones. Berkeley DB XML comes in form of a C++ library with interfaces to many scripting languages.

Since the client applications are mostly oriented to the graphical user interfaces (GUI), we have decided to



pes

, psa m

1. *šelma ochočená k hlídání, lovu ap.*, hlídací, lovecký, ovčácký pes, pozor, zlý pes!, pustit psa z řetězu,

2. *samec psovité šelmy*, liščí pes, je to pes, nebo fenka?,

3. expr. *bezohledný, krutý člověk*, být (na někoho) pes, život je pes, [x] ani pes po něm neštěkne, *expr. je zapomenut;* , být na někoho jako pes, *zlý;* , být poslušný, věrný jako pes, honit, zastřelit někoho jako psa, *bez citu;* , každý pes jiná ves, *(pořek.) (o nesourodých věcech);* , (ani) pes by od něho kůrku nevzal, *expr. je v opovržení;* , pes, který štěká, nekouše, *(pořek.);* , práce ap. pod psa, *hanl. velmi špatná;* , vypadat jako spráskaný pes, *být schlíplý n. unavený;* , (hledat, kde je) zakopaný pes, *hlavní potíž, překážka;* , (venku je,) že by ani psa nevyhnal, *velmi špatné počasí;* ,

psice -e ž *fena(syno):*

psík -a

psíček -čka

pejsek -ska m (mn. 1. -ci, -kové, 6. -cích) zdrob. expr.

Slovník spisovné češtiny

Figure 1: The DEBDict common interface to several dictionaries with different structures.

adopt the concepts of the Mozilla Development Platform (Oeschger, 2002). Firefox Web browser is one of the many applications created using this platform. Other applications include Mozilla Thunderbird mail client, Netscape Web browser, Komodo integrated development environment or Nvu web page editor.
The Mozilla Cross Platform Engine provides a clear separation between application logic and definition, presentation and language-specific texts. The application design is simple and allows the possibility of concurrent work of different team members which leads to significant time savings.
The main "programming language" used for the GUI design of the DEB clients is called XUL (XML User-interface Language, pronounced "zool"). XUL is a user interface description language based on XML. It allows relatively simple creation of cross platform applications with possibility of easy customization of design, texts and localization. XUL itself is aimed only on creation of user interface, e.g. windows, buttons or toolbars, but it incorporates wide range of standardized technologies:

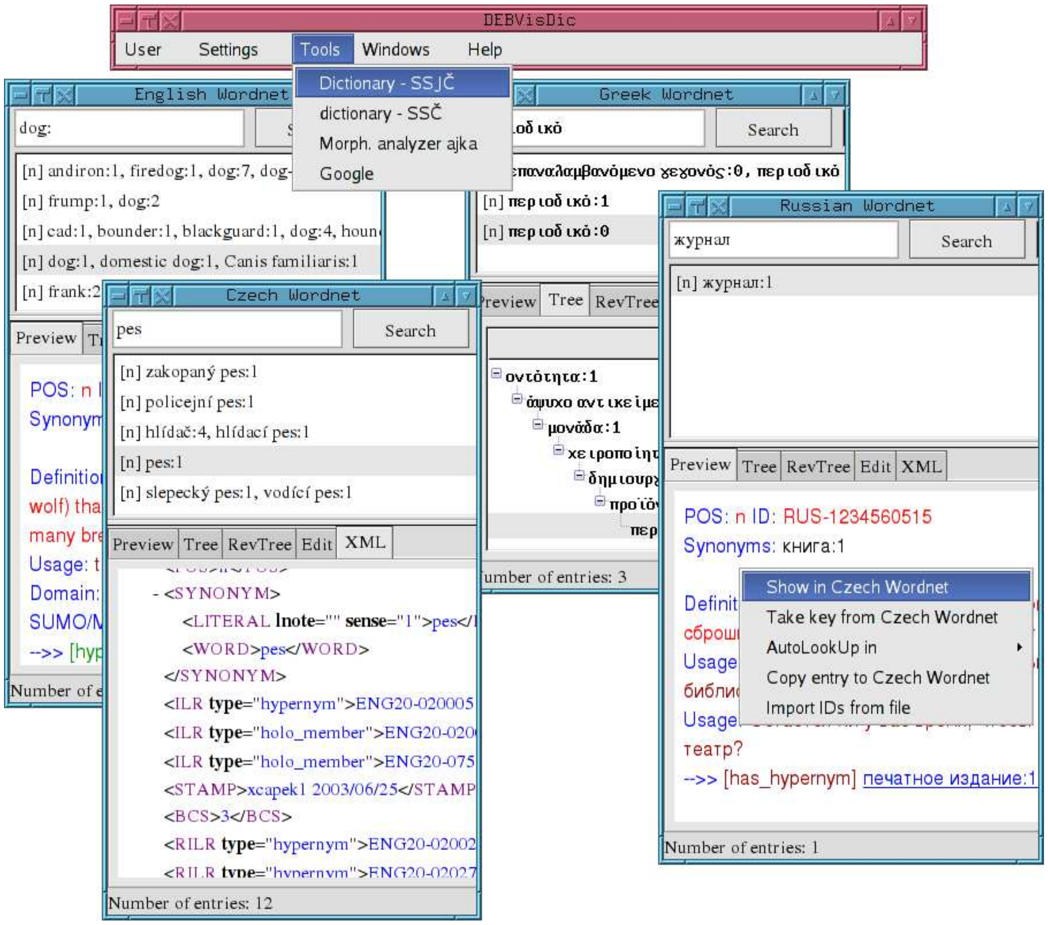
- Cascading Style Sheets (CSS) for describing the graphic appearance of the application,
- JavaScript as a programming language for simple application logic,
- Document Object Model (DOM), XSLT and XPath to work with HTML and XML documents,
- DTD for easy localization,
- RDF as data source.

2.1. The Users' Interfaces

The DEB clients are written in XUL and JavaScript and integrate with Mozilla Firefox Web browser. This allows us to use both Mozilla's user interface engine and its HTML/XHTML rendering engine as well as built-in components for interaction with filesystem on client computers, XPath interpreter, RDF processor etc.
Due to the feature-rich client architecture the developers may decide whether certain operations should be done on the server or on client parts – e.g. XSLT transformation can be done on both sides.
The particular DEB clients that are currently being implemented within the DEB platform include:

- DEBVisDic – complete new version of the successful wordnet semantic network editor and browser VisDic, see the Section 3.
- DEBDict – general dictionary browser. This simple DEB client demonstrates several basic functions of the system:
  - multilingual user interface (English, Czech, others can be easily added)
  - queries to several XML dictionaries (of different underlying structure) with the result passed through an XSLT transformation
  - connection to Czech morphological analyzer
  - connection to an external website (Google, Answers.com)
  - connection to a geographical information system (display of geographical links directly on their positions within a cartographic map) or any similar application

  The version of DEBDict that is currently running on our server provides a common interface to 7 dictionaries (see the Figure 1):



```
- <SYNONYM>
    <LITERAL lnote="" sense="1">pes</
    <WORD>pes</WORD>
  </SYNONYM>
  <ILR type="hypernym">ENG20-020005
  <ILR type="holo_member">ENG20-020
  <ILR type="holo_member">ENG20-075
  <STAMP>xcapek1 2003/06/25</STAMP
  <BCS>3</BCS>
  <RILR type="hypernym">ENG20-02002
```

Figure 2: The DEBVisDic interface.

- the Dictionary of Literary Czech Language (SSJC, 180.000 entries)
- the Dictionary of foreign words (46.000 entries)
- the Dictionary of Literary Czech (SSC, 49.000 entries)
- the Dictionary of Czech Synonyms (thesaurus, 23.000 entries)
- two dictionaries of Czech Phrasal Words and Idioms (4.000 and 10.000 entries)
- the Diderot encyclopedia (90.000 entries)

As an addition, DEBDict features an interconnection to several web systems and the geographical system with the list of the Czech towns and cities.

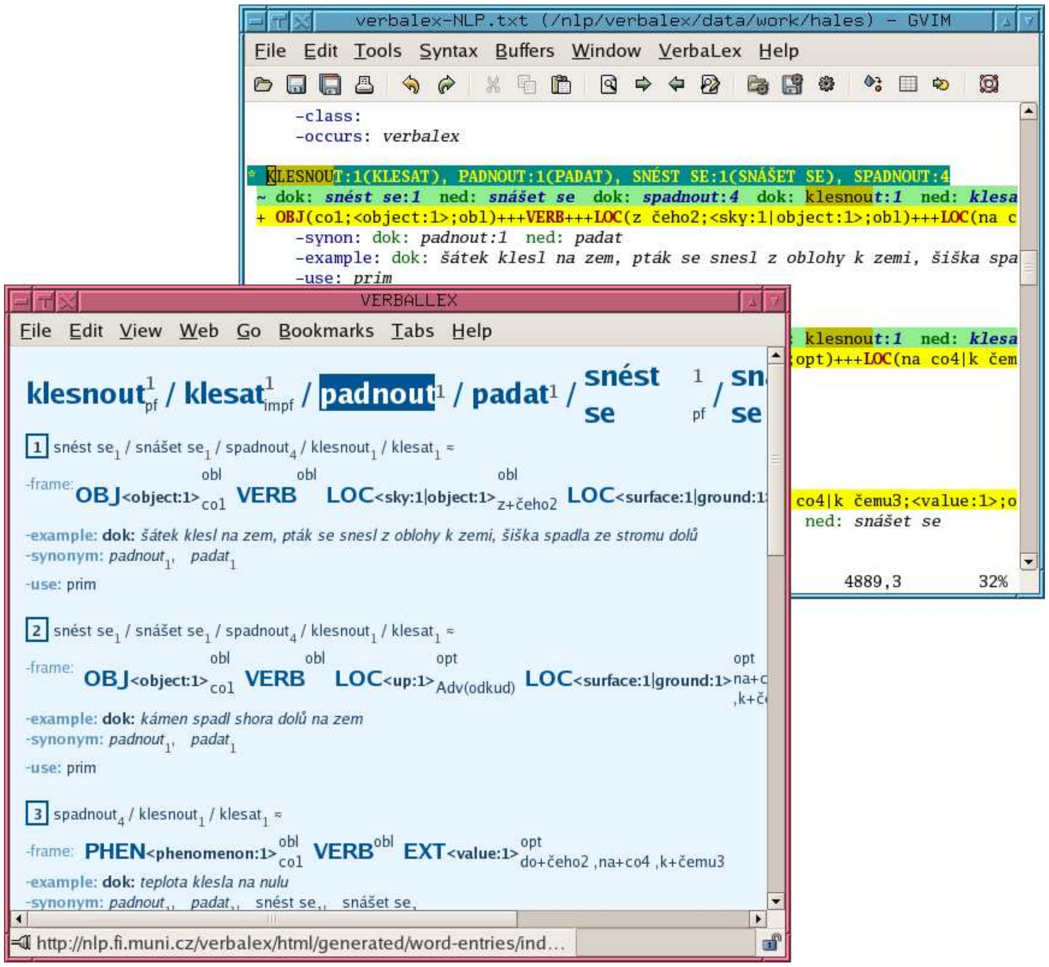
- PRALED – designed for the development of the Czech Lexical Database, CLD. This application serves as a main tool in preparation of the new comprehensive and exhaustive database of lexicographic information for Czech language. The user's part of PRALED is presently under the development in the Institute of Czech Language, Czech Academy of Sciences, Prague. Here DEB II is used as a dictionary writing system.
- VisualBrowser – the client-server architecture allows an easy connection of other existing applications to the DEB server. An example of such application is a direct interface to the VisualBrowser (Nevěřilová, 2005) tool that now displays the graphical representation of relations between elements stored in various DEB server databases.

3. DEBVisDic – a Tool for Handling Wordnets

One of our current goals has been to create a tool for working with two independent resources, the Czech Wordnet and the Czech verb valency lexicon VerbaLex, enabling us to merge and standardize the information contained in them.

DEBVisDic has been conceived as a reimplementation of the previous tool for wordnet semantic networks editor – VisDic. VisDic already exploits the XML data format thus making the wordnet-like databases more standard and exchangeable. Moreover, thanks to its general configuration, VisDic can serve for developing various types of dictionaries, i.e. monolingual, translational, thesauri and multilingually linked wordnet-like databases. The experience with the VisDic tool during Balkanet project has been extremely positive (Horák and Smrž, 2004) and it was used as the main tool with which all 6 Balkanet national wordnets were developed.

The experience with VisDic has led us to more systematic research into the usage of XML data formats within


verbalex-NLP.txt (/nlp/verbalex/data/work/hales) - GVIM
File Edit Tools Syntax Buffers Window VerbaLex Help
VERBALLEX
File Edit View Web Go Bookmarks Tabs Help
4889,3
32%
http://nlp.fi.muni.cz/verbalex/html/generated/word-entries/ind…

```
    -class:
    -occurs: verbalex

* KLESNOUT:1(KLESAT), PADNOUT:1(PADAT), SNÉST SE:1(SNÁŠET SE), SPADNOUT:4
 ~ dok: snést se:1  ned: snášet se  dok: spadnout:4  dok: klesnout:1  ned: klesa
 + OBJ(co1;<object:1>;obl)+++VERB+++LOC(z čeho2;<sky:1|object:1>;obl)+++LOC(na c
    -synon: dok: padnout:1  ned: padat
    -example: dok: šátek klesl na zem, pták se snesl z oblohy k zemi, šiška spa
    -use: prim
```

```
klesnout¹pf / klesat¹impf / padnout¹ / padat¹ / snést se¹pf / sn

[1] snést se₁ / snášet se₁ / spadnout₄ / klesnout₁ / klesat₁ ≈
-frame: OBJ<object:1>obl co1  VERB obl  LOC<sky:1|object:1>obl z+čeho2  LOC<surface:1|ground:1
-example: dok: šátek klesl na zem, pták se snesl z oblohy k zemi, šiška spadla ze stromu dolů
-synonym: padnout₁,  padat₁
-use: prim

[2] snést se₁ / snášet se₁ / spadnout₄ / klesnout₁ / klesat₁ ≈
-frame: OBJ<object:1>obl co1  VERB obl  LOC<up:1>opt Adv(odkud)  LOC<surface:1|ground:1>opt na+c ,k+č
-example: dok: kámen spadl shora dolů na zem
-synonym: padnout₁,  padat₁
-use: prim

[3] spadnout₄ / klesnout₁ / klesat₁ ≈
-frame: PHEN<phenomenon:1>obl co1  VERB obl  EXT<value:1>opt do+čeho2 ,na+co4 ,k+čemu3
-example: dok: teplota klesla na nulu
-synonym: padnout₁,  padat₁,  snést se₁,  snášet se₁
```

Figure 3: The VerbaLex editing and presentation interface.

the field of the computational lexicography. In parallel, we also pay attention to the relations between wordnets and Semantic Web. This interest gives us a strong motivation for studying the properties of the XML data formats and tools for working with them.

DEBVisDic uses a new windowed interface (see the Figure 2) that allows a user to arrange the client layout without any limitations. Of course, DEBVisDic contains all the main features that were present in VisDic, like multiple views of multiple wordnets, hypero-hyponymic tree browsing, inter-dictionary linking or synset editing. With the help of the DEB platform reusability, DEBVisDic is supplemented with a number of new features that were so far accessible only as separate tools or resources such as a connection to a morphological analyzer (for languages, where it is available), language corpora including Word Sketches statistics, access to any electronic dictionaries stored within the DEB server or searching for literals within encyclopedic web sites.

The client-server architecture allows an easy connection of other existing applications to the DEB wordnet server. An example of such application is a direct interface to the VisualBrowser (Nevěřilová, 2005) tool that now displays the graphical representation of the semantic network from the same database which is displayed in the DEBVisDic tool.

3.1. Using DEBVisDic for Merging Lexical Resources

A good example of merging various lexical data is the work going on in NLP Lab at FI MU where the data from Czech Wordnet and Czech Valency Lexicon VerbaLex are combined together. The VerbaLex lexicon is currently being developed separately and independently of Czech Wordnet using a particular XML format (see the Figure 4) describing so called *complex valency frames* (Hlaváčková and Horák, 2005). However, the entries in VerbaLex are written in form of Wordnet synsets, which enables combining the data from both these resources, see the Figure 3.

The Czech Wordnet currently contains a smaller set of valency frames in a plain PCDATA format (see the Figure 5). The current work is directed to merging the VerbaLex valency frames with the Czech Wordnet synset structures.

The Czech verbs for which valency frames already exist (approx. 5000) are or will be linked to their English (Princeton-2.0) equivalents by means of ILI (Inter-Lingual Index). If Czech and English verbs (synsets) are linked correctly, the deep valency frames developed for Czech can be also valid for English (surface valencies are obviously different since Czech is a synthetic language whereas English is an analytic one).

```
<word_entry>
  <headword_lemmata>
    <lemma ord='1' sense='1' aspect='pf'
        aspectual_counterpart_lemma='dodávat'>dodat</lemma>
    ...
  </headword_lemmata>
  <frame_entry frame_index='1'>
    <frame_lemmata>
      <lemma  sense='8' aspect='pf'>dát</lemma>
      ...
    </frame_lemmata>
    <synonym_lemmata>
      <lemma aspect='pf' sense='1'>vložit</lemma>
      ...
    </synonym_lemmata>
    <example>dok: připojili ke smlouvě své podpisy</example>
    <use>prim</use>
    <frame_slots>
      <slot number='1' functor='AG' type='obl' class='person:1'>
        <form type='direct_case' case='kdo1' />
      </slot>
      <slot number='2' type='obl' functor='VERB'/>
      <slot number='3' functor='INFO' type='obl' class='info:1'>
        <form type='direct_case' case='co4' />
      </slot>
      <slot number='4' functor='COM' type='obl'
            class='written communication:1'>
        <form type='prepos_case' prepos_lemma='k' case='čemu3' />
      </slot>
    </frame_slots>
  </frame_entry>
  ...
</word_entry>
```

Figure 4: An example of (a part of) an entry in the VerbaLex XML format for the synset `dát:8, vložit:1, vsunout:1, přidat:2, připojit:1, dodat:1` (i.e. insert:1, infix:1, enter:7, introduce:6 in the Princeton WN)

Synset: dát:8, vložit:1, vsunout:1, přidat:2, připojit:1, dodat:1

```
<VALENCY>
<FRAME>{dát, vložit, vsunout}
    kdo1*AG(person:1)=co4*OBJ(object:1) &do čeho2*OBJ(container:1)
</FRAME>
<FRAME>{dát, vsunout, přidat, vložit, dodat}
    kdo1*AG(person:1)=co4*INFO(info:1)
        & do čeho2*COM(written communication:1)
    %dodal do textu nové poznámky, přidal k článku obrázek
</FRAME>
<FRAME>{dát,přidat, připojit, dodat}
    kdo1*AG(person:1)=co4*INFO(info:1)
        & k čemu3*COM(written communication:1)
    %připojili k smlouvě své podpisy
</FRAME>
<FRAME>{přidat, připojit, dodat}
    kdo1*AG(person:1)=co4*OBJ(object:1) ? k čemu3*OBJ(object:1)
    %připojil hadici ke kohoutku
</FRAME>
```

Figure 5: An example of valency frames in the Czech Wordnet for the same synset `insert:1, infix:1, enter:7, introduce:6`

4. Why Client-Server Architecture?

In the client-server environment, the server provides different interfaces using the same data structure and these interfaces can be reused by many client applications. For example, several client applications are using the same interface to query XML dictionaries (with different underlying structure).

One of the main benefits of developing DEBVisDic on the DEB platform is the homogeneity of the data structure and presentation. If the DEBVisDic administrator commits a change in the data presentation, this change will automatically appear in each client software. And of course, any data flaws discovered can be instantly corrected, there is no need to change the client software or provide new data files to each client.

The data sources can even be implemented with different structures, that the server transforms seamlessly to a homogeneous form, which is then provided to client applications.

5. Conclusions and Future Directions

We have presented the DEB implementation platform and its main features. The DEB platform exploits the client/server architecture and offers several different clients allowing to perform various lexicographic tasks. The relevant features of the DEB platform are high modularity, configurability and flexibility which results in an easy adaptability for the various tasks. Thanks to them, the DEB platform represents a versatile base, on which the individual and powerful dictionary writing tools (clients) such as DEBVisDic are implemented.

In our view the XML formats within the DEB platform present a reasonable base not only for merging various lexical resources but also for their future standardization. This has been convincingly shown in the Balkanet Project where the VisDic XML format has been employed for building all 6 Balkanet languages (Bulgarian, Czech, Greek, Romanian, Serbian and Turkish) plus English. In fact, VisDic tool played the role of the instrument through which the first wordnet standardization steps have started. It is our belief that DEBVisDic tool can play the same role in the near future as well.

The experiments with exploiting Czech valency frames for building Bulgarian and Romanian frames in Balkanet project have been evaluated as more than promising, see (Tufiş et al., 2006) and S. Koeva in (bal, 2004). This kind of merge is even more inspiring than the former one and the DEBVisDic tool is enabling us to go beyond the mere experiments.

Acknowledgements

This work has been partly supported by the Academy of Sciences of Czech Republic under the projects T100300414 and T100300419, and by the Ministry of Education of CR within the center of basic research LC05113202.

6. References

Balkanet project website, `http://www.ceid.upatras.gr/Balkanet/`.

2004. *Balkanet Final Report*. University of Patras, DBLAB. No. IST-2000-29388, led by D. Christodoulakis.

Jens Erlandsen. 2004. iLEX – an ergonomic and powerful tool combining, effective and flexible editing with easy and fast search and retrieval. In *EURALEX 2004*, Lorient, France. demonstration.

Eurowordnet project website, `http://www.illc.uva.nl/EuroWordNet/`.

C. Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. The MIT Press, Cambridge, Massachusetts, London, England.

Dana Hlaváčková and Aleš Horák. 2005. Verbalex – new comprehensive lexicon of verb valencies for czech. In *Proceedings of the Slovko Conference*, Bratislava, Slovakia.

Aleš Horák and Pavel Smrž. 2003. VisDic – wordnet browsing and editing tool. In *Proceedings of the Second International WordNet Conference – GWC 2004*, pages 136–141, Brno, Czech Republic.

Aleš Horák and Pavel Smrž. 2004. New features of wordnet editor VisDic. In *Romanian Journal of Information Science and Technology*, volume 7, pages 1–13.

D. Joffe and G-M. de Schryver. 2004. Tshwanelex – professional off-the-shelf lexicography software. In *Third International Workshop on Dictionary Writing Systems: Program and List of Accepted Abstracts*, Brno, Czech Republic. Masaryk University, Faculty of Informatics.

Louw M. 1998. Polaris user's guide. Technical report, Belgium.

Michael McNamara. 2003. Dictionaries for all: XML to final product. In *XML Conference 2003*, Philadelphia, USA.

Z. Nevěřilová. 2005. The Visual Browser Project. `http://nlp.fi.muni.cz/projects/visualbrowser`.

Ian et al. Oeschger. 2002. *Creating Applications with Mozilla*. O'Reilly and Associates, Inc., Sebastopol, California.

Louis Rosenfeld and Peter Morville. 1998. *Information Architecture for the World Wide Web*. O'Reilly and Associates, Inc., Sebastopol, California.

The linguist's shoebox. `http://www.sil.org/computing/shoebox/`.

SOAP 1.1 – W3C Simple Object Access Protocol (SOAP) 1.1 Specification, `http://www.w3.org/TR/SOAP/`.

Dan Tufiş, Verginica Barbu Mititelu, Luigi Bozianu, and Cătălin Mihăilă. 2006. Romanian wordnet: New developments and applications. In *GWC 2006*, Jeju Island, Korea. Masaryk University, Brno.

Merging Layered Annotations

Nancy Ide, Keith Suderman

Department of Computer Science
Vassar College
Poughkeepsie, New York 12604-0520 USA
{ide,suderman}@cs.vassar.edu

Abstract

The American National Corpus and its annotations are represented in a stand-off XML format compliant with the specifications of ISO TC37 SC4 WG1's Linguistic Annotation Framework. Because few systems that enable search and access of the corpus currently support stand-off markup, the project has developed a SAX *like* parser that generates ANC data with annotations in-line, in a variety of output formats.

1. Introduction

The American National Corpus (ANC) project[1] (Ide and Macleod, 2001; Ide and Suderman, 2004 recently released its 2nd release consisting of approximately 22 million words of data, representing a variety of genres of both written and spoken data. The corpus is annotated with several layers of automatically produced linguistic information, including sentence and token boundaries, part of speech using two different POS tagsets (a version of the Penn tagset[2] and the Biber tagset[3]), and noun chunks and verb chunks. For a complete description of the ANC 2nd release and its contents, see http://AmericanNationalCorpus.org.

ANC primary documents are plain text (UTF-16) documents and are treated as "read only" resources. All annotations are represented in stand-off XML documents referencing spans in the primary data or other annotation documents, using the XCES[4] implementation of the specifications of ISO TC37 SC4's Linguistic Annotation Framework (LAF) (Ide and Romary, 2004). Because few systems that enable search and access of the corpus currently support stand-off markup, the project has developed a parser that generates ANC data with annotations in-line, in a variety of output formats.

This demonstration will show the "life-cycle" of an ANC document, from acquisition of a document in any of a variety of formats (MS Word, PDF, HTML, etc.) through annotation and final representation in the stand-off format. The ANC tool for merging annotations of the user's choice with the primary data to produce a single document with in-line annotations will also be demonstrated.

2. Document Life-Cycle

Documents to be included in the ANC are acquired in many different formats, including MS Word, PDF, HTML, Quark Express, etc. Processing involves a series of steps, which are outlined below.

2.1. Conversion from original format to "rudimentary" XML

The ANC receives documents in a variety of different formats. The first step in processing is to convert the input documents into XCES XML with basic structural annotations included. The most common types of file formats encountered are:

- **Microsoft Word.** The release of OpenOffice 2 has greatly simplified the processing of MS Word documents. OpenOffice uses XSL and XSLT stylesheets to export files to XML and ships with stylesheets to generate DocBook and TEI-compliant formats. We modified the TEI stylesheet to create XCES XML. OpenOffice's Java API enables us to automate and integrate OpenOffice with later processing steps.
- **XML/SGML/HTML**. Processing of XML files typically involves using XSLT to map element names to XCES. SGML and HTML files typically require pre-processing to render them into valid XML, followed by the application of an XSLT stylesheet to convert them to XCES.
- **Quark Express.** Several publishers provided documents prepared for publication using Quark Express. Quark documents can be exported in XML, but doing so is worthwhile only if the creator of the document takes advantage of Quark's style-definition facilities (which was not the case for any of the contributed Quark documents). We therefore exported the documents in RTF; however, many fonts and special characters are improperly rendered, and fairly extensive manual editing was therefore required to render the files into a format that could be used. Once edited, the same procedures for MS Word documents are used to generate XCES.
- **PDF.** Bitmap PDF files are unusable for our purposes. Adobe Acrobat can generate plain text from PDF, although this process loses much of the formatting information that would be desirable to retain to facilitate later processing. In some cases, ligatures and other special characters are improperly represented in the text version, and it is not always possible to automatically detect and convert them to conform to the original. PDF documents with two or

[1] http://AmericanNationalCorpus.org

[2] http://americannationalcorpus.org/FirstRelease/gatetags.txt

[3] http://americannationalcorpus.org/FirstRelease/Biber-tags.txt

[4] http://www.xces.org

more columns cannot, to our knowledge, be extracted without some mis-ordering of the text in the results.

- **Other formats.** Other formats in which the ANC has acquired documents include plain text and plain text that employed a variety of proprietary markup languages. These documents are processed on a case by case basis, using specialized scripts.

2.2. Creation of standoff annotation documents

We have developed several custom processing resources that plug into GATE to generate standoff annotations in the XCES implementation of the LAF format. The last step in our GATE pipeline is to create the primary text document and generate all the required standoff annotation files. It is also possible to import annotations generated by software outside the GATE environment and render them into the standoff format.

3. Standoff Format

The ANC standoff format for annotations is a simple graph representation, consisting of one node set and one, or more, edge sets. The node set is represented by the text itself, with an implied node between each character. Each edge set is represented by an XML document and may contain one or more annotation types: logical structure, sentence boundaries, tokens, etc.

An ANC header file for each document is used to associate the source text with the standoff annotation documents; for example:

```
<cesHeader>
  ...
  <annotations>
    <annotation type="content"
       ann.loc="en_4065.txt">
       Text content</annotation>
    <annotation type="logical"
       ann.loc="en_4065-logical.xml">
       Logical structure</annotation>
     <annotation type="s"
       ann.loc="en_4065-s.xml">
       Sentence boundaries</annotation>
     <annotation type="hepple"
       ann.loc="en_4065-hepple.xml">
       Hepple POS tags</annotation>
     <annotation type="biber"
       ann.loc="en_4065-biber.xml">
       Biber POS tags</annotation>
     <annotation type="vp"
       ann.loc="en_4065-vp.xml">
       Verb chunks</annotation>
     <annotation type="np"
       ann.loc="en_4065-np.xml">
       Noun chunks</annotation>
  </annotations>
      ...
</cesHeader>
```

ANC annotation documents are marked up with the XCES representation of the nodes and edge sets of the annotation graph. The following shows a segment of the document containing part of speech annotation:

```
<cesAna xmlns="http://www.xces.org/schema/2003"
   version="1.0.4">
   <struct type="tok" from="4" to="6">
      <feat name="base" value="in"/>
      <feat name="msd" value="IN"/>
   </struct>
   <struct type="tok" from="7" to="11">
      <feat name="msd" value="DT"/>
      <feat name="base" value="this"/>
   </struct>
   <struct type="tok" from="12" to="19">
      <feat name="base" value="chapter"/>
      <feat name="msd" value="NN"/>
   </struct>
   ...
</cesAna>
```

Each `<struct>` element represents an edge in the graph; values of the *from* and *to* attributes denote the nodes (between characters in the primary text document) over which the edge spans.

3.1. Annotating discontiguous spans

Presently, the ANC includes standoff annotations that reference contiguous spans of data in the original (primary) document. However, we plan to add a wide variety of automatically-produced annotations for various linguistic phenomena to the ANC data, some of which will reference discontiguous regions of the primary data, or may reference annotations contained in other standoff documents. This is handled as follows: given an annotation graph, *G*, we create an edge graph *G'* whose nodes can themselves be annotated, thereby allowing for edges between the edges of the original annotation graph *G*.

For example, consider the sentence "My dog has fleas." The standoff annotations for the tokens would be:

```
                   1
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6
|M|y| |d|o|g| |h|a|s| |f|l|e|a|s|

<struct … id="t1" from="0" to="2"/>
<struct … id="t2" from="3" to="6"/>
<struct … id="t3" from="7" to="10"/>
<struct … id="t4" from="11" to="16"/>
```

Now consider the dependency tree generated by Minipar[5] given in Figure 2. The tree can be represented by annotating the token elements in the standoff annotation document as follows:

```
<!-- Define some pseudo nodes -->
<node type="root" id"E0" ref="t3"/>
<node type="clone" id="E2" ref="t2"/>

<!-- Define edges in dependency tree -->
<struct type="subj" id="s1"
       from="t3" to="E2"/>
<struct type="s" id="s2"
       from="t3" to="t2"/>
<struct type="gen" id="gen"
       from="t2" to="t1"/>
<struct type="obj" id="obj"
       from="t3" to="t4"/>
```

[5]http://www.cs.ualberta.ca/~lindek/minipar.htm

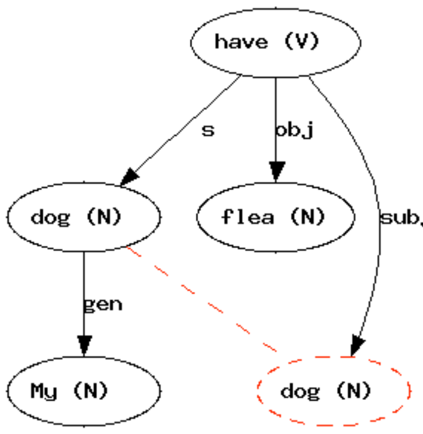


Figure 2. Dependency tree generated by Minipar.[6]

4. Creating In-line Annotation Documents

We have developed an "XCES Parser"[7] that implements the org.xml.sax.XMLReader interface to create ANC documents containing in-line annotations in XML (or any other format).

The XCES parser works as follows: annotations to be loaded are selected with the `org.xml.sax.XMLReader.setProperty()` method. The selected annotation sets are then loaded into a single list in memory and sorted, first by offset and, if the offsets are the same, secondly by annotation type. At present, the ordering of annotation types are hard coded into the parser; work is underway to make the XCES parser "schema aware" so that embedding specifications can be provided by the user. Once the text is loaded and sorted, the appropriate SAX2 events are generated and dispatched to the org.xml.sax.ContentHandler (if one has been registered with the parser) in sequence to simulate the parsing of an XML document. While the parser will allow the programmer to specify an ErrorHandler, DTDHandler, or EntityResolver, at this time no methods from those interfaces will be invoked during parsing.

In the current version of the XCES parser, when overlapping annotations are encountered, they are "truncated". For example:

```
<s>Sentence <em>one.</s><s>Sentence</em>
two.</s>
```

becomes

```
<s>Sentence <em>one.</em></s><s>Sentence
two.</s>
```

Work is underway to provide the option to generate milestones in CLIX/HORSE (DeRose, 2004) format to represent overlapping hierarchies.

4.1. Using the XCES parser

The XCES parser can be used in three ways:

- from the command line. The xces-parser.jar file can be run as a command line program to print XML with inline annotation to standard output.
- as the XML parser used by other applications. For example, Saxon[8] can take the name of the parser to use to parse the source document as a command line parameter. This allows us to apply XSLT stylesheets to ANC documents without having to translate them into XML first.
- as a library for use in other Java applications. For example, The ANC Tool[9] is a graphical front end to the XCES parser.

4.2. The ANC tool

The ANC Tool provides a graphical user interface for the XCES parser and is used to convert ANC documents to other formats. Users specify their choice of annotations to be included. Currently, the ANC Tool can be used to generate the following output formats:

- XML XCES format, suitable for use with the BNC's XAIRA[10] search and access interface;
- Text with part of speech tags appended to each word and separated by an underscore;
- WordSmith/MonoConc Pro format.

The ANC Tool uses multiple implementations of the `org.xml.sax.DocumentHandler` interface, one for each output format, which the XCES parser uses to generate the desired output. Additional output formats can be easily generated by implementing additional interfaces.

Of course, if the target application understands annotation graphs, there is no need to bother with the XCES parser or conversion to XML. For example, we provide several resources for GATE that permit GATE to open and read ANC documents with standoff annotations, or to load standoff annotations into an already loaded document.

5. Next Steps

Currently the XCES parser is a *proof of concept* rather than a production grade tool. The parser is being augmented to invoke all the appropriate methods from the `org.xml.sax.*Handler` interfaces and throw the proper `SAXExceptions` at the appropriate times. We are also providing for some level of SAX conformance, rather than simply "doing what Xerces does".

A top priority in the development of the XCES parser is to adapt it to be schema-aware, in order to enable specification of nesting as well as to allow for including only some parts of a given annotation in the merged version. This capability will become increasingly important as we include annotations for a wider variety of annotation types. For example, merging annotations such as those provided by PropBank (Palmer *et al.*, 2005), NomBank (Meyers, *et al.*, 2004), and TimeML (Pustejovsky *et al.*, 2003) in some cases demands that interactions among the various annotations are taken into account in order to produce a single, coherent merged

[6] Image generated by http://ai.stanford.edu/~rion/parsing/minipar_viz.html
[7] http://americannationalcorpus.org/tools/index.html#xces-parser

[8] http://saxon.sourceforge.net/
[9] http:// americannationalcorpus.org/tools/anctool.html
[10] http://sourceforge.net/projects/xaira

annotation that may include only some of the information from one or more of the separate annotations, include both annotations from different sources, or create a new annotation based on information from all three. A schema dictating the inter-relations coupled with a style sheet to render the merged document can be used to specify the information in and format of the result.

6. Conclusion

The ANC has implemented an efficient pipeline for the processing of text into a corpus of machine usable documents. For some document types this process is almost completely automated and can be regarded as a *Corpus-Builder-in-a Box*: raw data goes in one end, and a fully annotated corpus with standoff annotations comes out the other.

The use of standoff annotations allows for an accurate representation of the ANC data as provided by the contributors and allows us to easily provide several modular annotation sets that can be included or excluded by the end user as desired. By providing a SAX like parser for ANC documents, we are able to leverage a number of available XML tools without the restrictions imposed by an XML representation of the documents. For users who are not interested in XML or standoff annotations, the plain text version is preserved.

7. References

DeRose, S. J. (2004) Markup Overlap: A Review and a Horse. In *Extreme Markup Languages 2004: Proceedings*. http://www.mulberrytech.com/Extreme/Proceedings/html/2004/DeRose01/EML2004DeRose01.html

Ide, N., Romary, L. (2004) International Standard for a Linguistic Annotation Framework. *Journal of Natural Language Engineering*, 10:3-4, pp. 211-225.

Meyers, A., Reeves, R., Macleod, C., Szekely, R., Zielinska, V., Young, B., & Grishman. R. (2004) The NomBank Project: An Interim Report. In *HLT-NAACL 2004 Workshop: Frontiers in Corpus Annotation*, Boston, Massachusetts.

Palmer, M., Gildea, D., & Kingsbury, P. (2005) The Proposition Bank: An Annotated Corpus of Semantic Roles. *Computational Linguistics*, 31(1).

Pustejovsky, J., Castaño, J., Ingria, R., Saurí, R., Gaizauskas, R., Setzer, A., & Katz. G. (2003) TimeML: A Specification Language for Temporal and Event Expressions. In *IWCS, International Workshop of Computational Semantics.* Kluwer Academic Publishers.

Author Index

Chanev, Atanas 8
Dipper, Stefanie 48
Graça, João 24
Hinrichs, Erhard 48
Horák, Ales 55
Ide, Nancy 61
Jande, Per Anders 1
Littman, Jessica 8
Mamede, Nuno J. 24
Marinov, Svetoslav 16
Monachesi, Paola 31
Osenova, Petya 16
Pajas, Petr 40
Pala, Karel 55
Pereira, João D. 24
Pustejovsky, James 8
Saurí, Roser 8
Schmidt, Thomas 48
Simov, Kiril 16
Stepánek, Jan 40
Suderman, Keith 61
Trapman, Jantine 31
Wagner, Andreas 48
Witt, Andreas 48