Chooser - A Multi-Task Annotation Tool

Svetla Koeva, Borislav Rizov, Svetlozara Leseva
Department of Computational Linguistics, Institute of Bulgarian Language - BAS
1113 Sofia, 52 Shipchenski prohod blvd., bl. 17
E-mail: svetla@ibl.bas.bg, bobby@ibl.bas.bg, zara@ibl.bas.bg

Abstract

The paper presents a tool assisting manual annotation of linguistic data developed at the Department of Computational linguistics, IBL-BAS. Chooser is a general-purpose modular application for corpus annotation based on the principles of commonality and reusability of the created resources, language and theory independence, extendibility and user-friendliness. These features have been achieved through a powerful abstract architecture within the Model-View-Controller paradigm that is easily tailored to task-specific requirements and readily extendable to new applications. The tool is to a considerable extent independent of data format and representation and produces outputs that are largely consistent with existing standards. The annotated data are therefore reusable in tasks requiring different levels of annotation and are accessible to external applications. The tool incorporates edit functions, pass and arrangement strategies that facilitate annotators' work. The relevant module produces tree-structured and graph-based representations in respective annotation modes. Another valuable feature of the application is concurrent access by multiple users and centralised storage of lexical resources underlying annotation schemata, as well as of annotations, including frequency of selection, updates in the lexical database, etc. Chooser has been successfully applied to a number of tasks – POS tagging, WS annotation, syntactic annotation.

1. Introduction

Linguistic annotation has long since ceased to be a stand-alone effort. With the growing need of annotated linguistic resources of various kinds, reusability, applicability and compatibility have become crucial issues in NLP. One of the major applications of linguistic annotation nowadays has been to provide the foundation of supervised learning algorithms and models. Since training corpora were proved to have a positive impact on NLP systems' learnability resulting in increase in both precision and recall rates, they have found a huge application in probabilistic and hybrid approaches to handling language data regardless of the theoretical frameworks within which they have been employed. In parallel, the issue has been tackled of standardization of both the content and the format of annotated resources. In light of these considerations the development of general and extendable programming resources based on commonality and interoperability has become a central issue with respect to enabling the construction of diverse linguistic annotated resources based on uniformity, consistency and re-usability of the annotations (Ide & Romary, 2006, 2007; Ide & Suderman, 2006).

The paper presents Chooser - a multi-functional tool assisting development of language resources embedding one or more levels of annotation. It is intended as a modular annotation system based on an abstract architecture that is easily tailored to specific annotation tasks through the development of separate modules. With already several such modules made available for various purposes, the tool has proved to be extendable to diverse types of linguistic annotation. The underlying ideology subsumes level and theory independence as key features of the tool. Besides, the uniform pre-possessing of diverse input file and data representation formats allows resources created according to different methodologies to be imported, while the standardised outputs ensure that the annotated resources are compliant with other (higher) level modules, as well as with external programmes. The tool's user-friendly platform facilitates annotation and increases effectiveness of annotators' work in a number of ways concerning language data visualisation, editing and searching, and incorporates probabilistic techniques and reordering strategies. A key functionality of the tool is a multi-user client-server that allows annotators to work simultaneously. Chooser has so far been applied in the creation of the Bulgarian POS-tagged corpus (Koeva, Leseva & Todorova, 2006), the Bulgarian WSD-annotated corpus (Koeva et al., 2006), and the Bulgarian dependency treebank.

We start with an outline of the main features of the tool in Section two, followed by a brief presentation of its architecture in Section three and its current applications in Section four. Sections five and six, respectively, sketch the parallels with other language annotation tools and platforms and give an overview of the future work towards enhancing the system.

2. General Description

2.1 Input Data Pre-processing and Interoperability

Chooser supports various input data formats and ensures consistent representation of linguistic data by means of Unicode text encoding and a number of general normalisation procedures such as Unicode conversion, tokenisation, XML formatting, indexation, etc. More sophisticated task-specific pre-processing is employed, as well, such as tag assignment and pre-tagging, in which annotation schemata are mapped to tokens, and/or annotation at a more basic linguistic level is performed (e.g. POS pre-tagging in syntactic and word-sense annotation tasks).

Beside internal consistency, conformity to standardised formats such as XML or XML-convertible ones facilitates interaction with external resources and applications. The architecture supports import and export

of structured annotations (tree-structured, as well as graph-based ones), discussed briefly in Section 2.4.

2.2 Annotation Schemata (AS) and Data Representation

Chooser handles both raw corpora and corpora with pre-assigned tags at one or more linguistic level(s). The former case involves the association of items from an annotation schema (tagset) with corpus units having no additional linguistic specification assigned to them. In the latter, linguistic pre-processing prerequisite for a particular task or same-level annotation – e.g. POS or sense assignment in POS tagging and word sense disambiguation - is performed prior to annotation proper. Tagsets are derived from several (and as many as needed) lexical resources, such as an inflexional dictionary, a wordnet, a NE ontology, etc., or are defined by annotators. Annotation schemata are configured and stored in external files, and are therefore easily specifiable and redefinable. Besides, real-time specification and edit of tagsets allows users to make relevant changes in the annotation schemata during annotation.

Another important feature of the tool is the possibility of producing both hierarchical and non-hierarchical structured annotations, and displaying them as tree-structured and graph-based representations, respectively.

Uniformity is maintained by treating both as non-embedding (flat) structures. In Chooser's format all language units (henceforth LU) are considered same-level units. Instead of encoding phrase structure mark-up through embedded structures such as xml tags where the tags whose values are syntactic categories embed the tags whose values are their constituents, they are represented as a path attribute where parents' paths are defined as subpaths of their descendants' paths. Graph-based annotations are lists of label-node pairs. The two types of annotations are non-overlapping, which makes it possible for both to be maintained in a single annotated file. This allows Chooser to be theory-independent to a considerable extent.

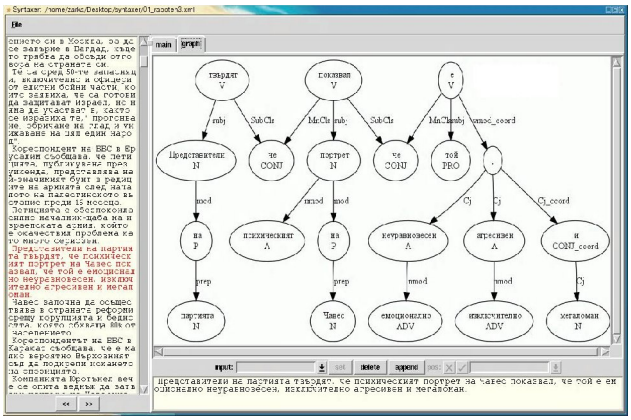

Figure 1. Dependency-grammar based annotation

As an illustration Figure 1 and Figure 2 show the treebank module's constituency-grammar and dependency-grammar based annotation, where the particular mode is selected by the user. The two types of data representations are associated with different annotation schemata - sets of phrasal categories and dependency relations, the first involving hierarchical units (phrases and lexical categories), the latter relation pointers and category labels.

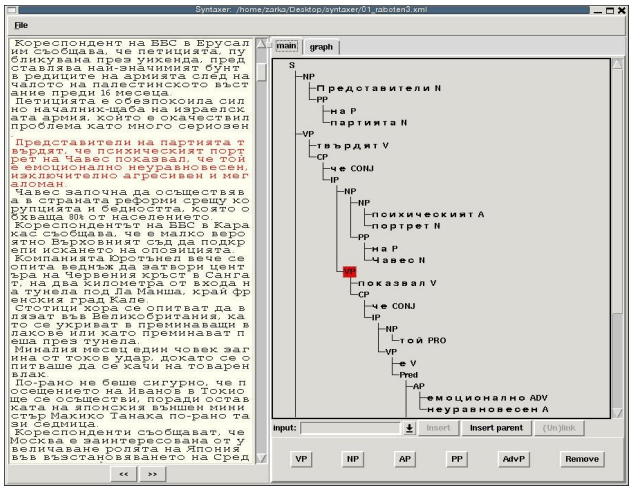

Figure 2. Constituency-grammar based annotation

2.3 Language Resources

As an annotation tool Chooser makes use of basically two types of resources - raw and pre-annotated corpora over which annotation is performed, and lexical resources such as diverse dictionaries and lexical databases which provide annotation schemata. The annotation corpora used so far are basically structured subsets of the Brown Corpus of Bulgarian sampled according to specified criteria. The Brown Corpus of Bulgarian (Koeva et al., 2006) is designed according to the Brown Corpus methodology (Francis & Kucera, 1979) with certain specific modifications of the model[1] resulting from the language and culture differences, as well as from the extensive distribution of electronic documents. It consists of 500 text units of approximately 2,000 words each, distributed proportionally across 15 "genres". Having the most clear-cut structure, it is the chief resource for deriving annotation corpora. The methodology of compilation of part-of-speech, word-sense and treebank annotation corpora involves stratified sampling based on frequency and weighted frequency of (particular classes of) content words. So far the following corpora have been designed: a POS-annotation corpus of app. 200,000 words (Koeva et al., 2006), a WS-annotation corpus of app. 100,000 words (Koeva, Leseva, Todorova, 2006), and a syntactic dependency bank of app. 200,000 words, the latter two currently under way.

The POS annotation schema used in the creation of the Bulgarian POS annotated corpus is derived from the Grammar Dictionary of Bulgarian containing app. 80,000 lemmas and more than 1M word forms with relevant grammatical features (Koeva, 1998). The same schema and the POS-tagging model were subsequently used in WS-annotation corpus and the treebank development.

Word senses defined in Bulgarian WordNet are used in annotation of the Bulgarian WS-annotated corpus.

[1] http://dcl.bas.bg/Corpus/home_en.html

WordNet (or BulNet) is a lexical-semantic network (Koeva, Tinchev & Mihov, 2004). Its overall structure follows the model of Princeton WordNet[2], but language-specific features and concepts have been introduced, as well. BulNet currently consists of app. 30,000 synonym sets (synsets) representing lexicalised concepts (word senses) with an overall of 64,750 synset members. Each synset is supplied with a gloss, and optionally usage examples, linguistic notes, etc. Bulgarian WordNet is stored as a VisDic-compliant xml data base (XDB) (Paveleck & Pala, 2002) and as a relational data base (RDB), of which Chooser currently employs the former.

User-defined annotation schemata are particularly applicable in syntactic annotation, since the number of syntactic labels is small, and to a considerable extent theory-specific. Thus, the syntactic module incorporates the possibility of defining and extending annotation schemata dynamically.

2.4 View and Annotation

Linguistic annotation involves either linear annotation of individual tokens and groups of tokens, or structured annotation, whereby a representation is assigned to a pair, or a larger group of tokens. The first type is represented by part-of-speech tagging, and word-sense annotation, whereas the second is employed in syntactic annotation, ontology mark-up, etc. The specific requirements have necessitated the design of independent modules that incorporate task-specific features within the general architecture. We will henceforth address in parallel the functionality and architecture of the two basic modules as realised in the POS and WSD modules, on the one hand, and the syntactic module, on the other.

The two principal modules have a bi- or tripartite display area that visualises different portions of information associated with LUs and annotation. Both have a main pane where the corpus is loaded and displayed. The linear module's window has also a listview pane where annotation options are viewed and selected, and a browser-based view that visualised additional information for LUs available in the lexical resource from which the annotation schema is created.

The second pane of the syntactic module contains a treeview and a graphview corresponding to the two modes of annotation, as well as control buttons and editable combo boxes where the annotation schemata are displayed, created, modified, or removed. The bottom of the graphview has an additional text area where the annotation unit (i.e. the sentence) is viewed and annotated.

Every LU is either mapped onto the annotation schema, whereby all relevant tags are associated with the LU (all POS-tags or all word-senses available for a lemma), or the user assigns tags from the schema to raw text. The former is adopted where annotation is performed over words (POS, WS assignment), the latter - where annotation involves labelling relations between two or more words. A note to be made at that point is that in principle the latter is appropriate for small finite tagsets such as syntactic categories and parts-of-speech, while not feasible for large and expanding annotation schemata such as word sense inventories.

[2] http://wordnet.princeton.edu/

An additional possibility adopted in the syntactic module is for annotators to create annotation schemata dynamically, usually where no linguistic resource is applicable or available.

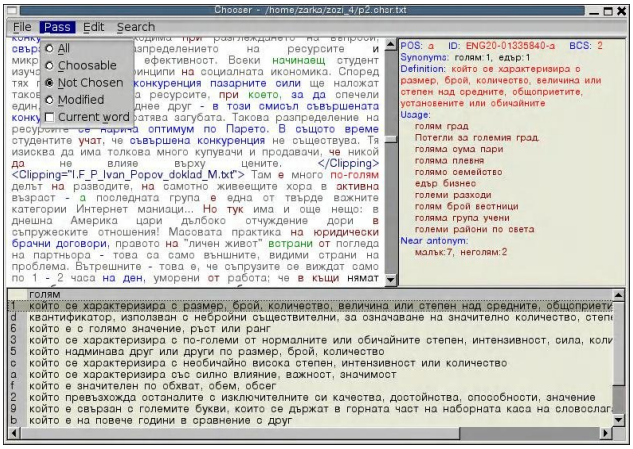

Figure 3. Chooser's WSD layout

The mappings between tokens and the annotation schema represent the annotation options suggested to annotators in tasks involving tag pre-assignment (POS tagging and WSD). Options are displayed in a separate listview pane where annotations are selected. Currently no pre-assignment of tags is involved in syntactic annotation.

The annotator associates a given LU with a tag from the tagset, e.g. a word form with a lexical category (in POS tagging), a word form with a word sense (in WSD), a phrase with a syntactic category or words with a dependency relation (in syntactic annotation), etc. In the POS/WSD module annotation options are browsed with the Arrow keys or click on a particular listview item. The selection of an item on the list results in the synchronisation with the html browser-based view which displays pre-configured linguistic information available for the particular listview item in the annotation schema. The displayed linguistic data may be redefined in terms of their content as more or less linguistic information may be made available to users.

Structured annotations are displayed in a different manner depending on the underlying data representation. Hierarchical relations (constituency trees, ontologies, etc.) are represented in a treeview, whereas dependency relations are represented as directed graphs. Phrase-structure markup takes place in the treeview pane by means of linking two or more words and selecting the type of phrase they form from the pre-defined inventory or on creating a relevant tag. Initially all words are sisters whose mother is S (sentence). On selecting the sisters that are constituents of a phrase and the syntactic category of their mother, a new node is created that corresponds to the phrasal category. Dependency relations are assigned by selecting the pair of words that form the relation (in the graphview text field) and specifying the type of relation. The graphview responds to the creation of a relation by drawing a labelled arc between the nodes in the graph.

2.5. Corpus Navigation, Data Management and Communication

The linear Chooser's module affords navigation of the corpus text according to different easily extendable and redefinable pass strategies selected from the main menu of the tool. These include consecutive linear pass of all tokens, as well as passes of LUs matching (i) all instances of the current LU, (ii) annotated LUs, (iii) markables, (iv) LUs modified after previous pass, etc. Independently, the tool provides a search function over: (i) wordforms, (ii) lemmata, (iii) parts of words with case sensitivity and search direction (forward/backward) selectable from the menu.

In the syntactic module passes of the corpus are performed over sentences by means of navigation buttons. Selection of a sentence is executed on mouse click on any word in the sentence.

An important feature of the tool is that it allows a number of operations over LUs on the word level. These include: (i) editing of corpus content; (ii) selection of compounds and other multi-word expressions, including ones with non-contiguous constituents whereby more than one word forms are associated with a single LU. The module maintains simple interactively updated edit operations over word forms and lemmas, which allows for correction of spelling errors that would otherwise interfere with annotation. However, Chooser currently does not handle issues concerning change in the number of tokens that might affect indexation.

Marking up multiple tokens as multi-word expressions is performed by selecting them by means of the Shift button and a mouse click. This results in the linking of the tokens through cycles without changing the tokens' individual indices. The selection of multiple tokens as one LU results in synchronization with corresponding mutli-word candidates available in the lexical resource employed in the disambiguation task, for instance word senses in BulNet. The advantage of this approach to multi-word expressions is that pre-annotation identification and re-indexation is avoided and contiguity of the constituents is made irrelevant. The latter proves particularly convenient with respect to free word order languages, such as Bulgarian, and with a view to the possibility of intervening non-constituents.

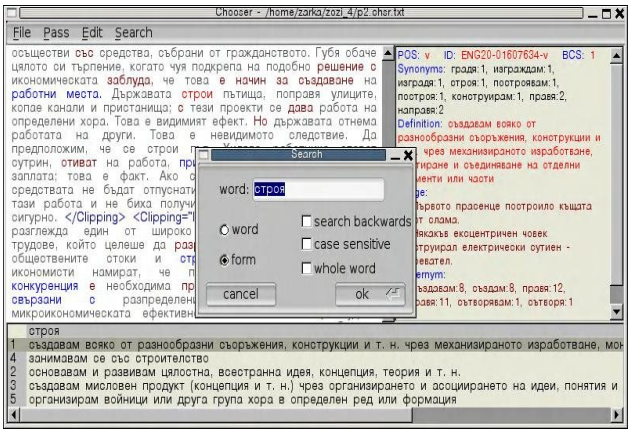

Figure 4. Chooser's WSD Search Option

Concurrent access and centralised storage of the annotation schema and the related lexical resource is required in tasks involving developing expandable schemata such as word sense databases, as a prerequisite to uniformity of both the schema and the annotated resources. To this end Chooser provides dynamic interaction between local users by means of a server that takes care of (i) the interaction between local users and the linguistic database; (ii) the interaction among local users. The first involves processes such as receiving and performing client requests, e.g. updates, while the second subsumes notification to the server about local clients' data status, processing clients' data and callbacks to local users. One of the applications of the latter is weight assignment to elements of the annotation schema according to frequency of selection, on the basis of which the tag option lists are periodically rearranged in frequency descending order.

3. Architecture and Implementation

3.1 Architecture

Chooser's architecture is intended to ensure fast and reliable corpus annotation environment, capable of accommodating general and task-specific requirements. The architecture is underlain by a Model-View-Controller paradigm enhanced with several design patterns such as Strategy, Chain of responsibility, Observer, Iterator which in conjunction afford sufficient and flexible design solutions (Buschmann et al. 1996; Gamma et al., 1994).

The linear module's **User Interface (UI)** provides text visualisation and navigation by means of the following objects: TextView, ChoiceView and InfoView. TextView takes care of text display by defining a vertically scrollable area and a Canvas object responsible for displaying the loaded text and storing information about LUs' colour (signalling mark-up status), and position (index). LUs themselves are stored in a Text object as a collection of Word instances. LU storage, text ordering strategy management and visualisation are delegated to several objects that communicate with Canvas. Compositor takes care of pass strategy management by receiving a collection of the sizes of the graphic objects to be arranged and the width of the display frame, and returns a collection of positions.

ChoiceView is a view control object that is responsible for displaying the set of annotation candidates for a LU, whereas InfoView visualises relevant portions of linguistic information associated with the selected or a default annotation candidate in the browser pane. The linguistic information is retrieved from the language resource used in the definition of the annotation schema, or may be created by client with user-defined schemata.

The application controller Framework centralizes retrieval and invocation of request-processing components. It performs action and view management between the User Interface and an abstract Document class by responding to users' actions and to updates in the document data, including ones made by external applications.

On its part, Document defines an interface for data loading from and saving to a stream, and provides access to LUs and the corresponding annotation candidates in the schema by means of two objects that are indexable collections of Word instances and annotation options, respectively. In this way a given LU in the corpus text is

associated with a corresponding (co-indexed) Choice object.
From a current instance of Choice a Colorer object called by Framework retrieves information about the mark-up status of the current LU, and basically serves to distinguish between annotated, ambiguous, modified words, etc. Another function of the controller is to provide an interface for search and edit LU (misspellings, mistypings, wrong lemmatisation, etc.).
Framework also takes care of corpus navigation management by means of an Iterator object that encapsulates different pass strategies including (i) identification of markable LUs and making them available for annotation; (ii) determination of the direction of movement. The former responsibility is delegated to a Pass object that employs the LU's corresponding Choice object, the latter - to an Increment object. The Iterator object itself defines an interface for initiating iteration, moving on to a next passable LU (as requested by the user), moving on to any LU and making sure when the iteration is completed.
Choice class objects are generic complex components central to the module's design that provide the abstract representation of annotation data (the linguistic level, candidate annotations and their data type, selected tags, etc.) and define a protocol for annotations access, i.e. calling annotation candidates and additional linguistic information, as well as retrieving and saving selected tags.
Compound LU selection is realised through cyclic lists of Choice objects, so that every Choice object stores reference to the next. In such a way users are allowed to annotate MWEs regardless of contiguity and word order.
Chooser maintains a centralised linguistic data base currently implemented for the linear module. A Choice object successor called Server_Choice encapsulates the process of passing information for a LU from a server program to a local instance of the application. The server extracts linguistic information corresponding to LUs and annotation options from the database, stores and maintains it and sends it to every server client. Reported changes in the database are passed back to clients through a single Choice_Info instance whose responsibility is to store all the information for LUs and the corresponding options, maintain connection with the server and "listen" for updates.
The syntactic module's **User Interface (UI)** has a window that is divided in two panes; a TextView where navigation between sentences is performed by means of buttons, generally corresponding to the same object in the linear module. The right pane contains a TreeView object and a GraphView object. TreeView is a treeview control that serves to represent sentence structure. Structured representations are created by means of Insert, Delete and Insert Parent controls, where the first inserts empty nodes, the second deletes nodes, and the third creates parent nodes. GraphView contains a drawable area where the current state of graphs is displayed, and a HyperTextView object - a text field that stores the current sentence where each word is a selectable hyperlink.
The users may select one or two words at most. One word selection allows assignment of a tag (e.g. POS). When two words are selected relations between them may be inserted and deleted. Since only one relation may be defined for a pair of words, the insertion of a second relation overrides the previously selected one. Relations are picked up from an editable combo-box.
The module's controller has similar functions to those of Framework of centralising retrieval, and conducting action and view management. Communication between objects is performed by means of an Observer pattern. The main controller centralises synchronisation between the data and their representation in the views. View objects subscribe as observers to events (changes in the data) and receive notification of events' occurrence from the controller.

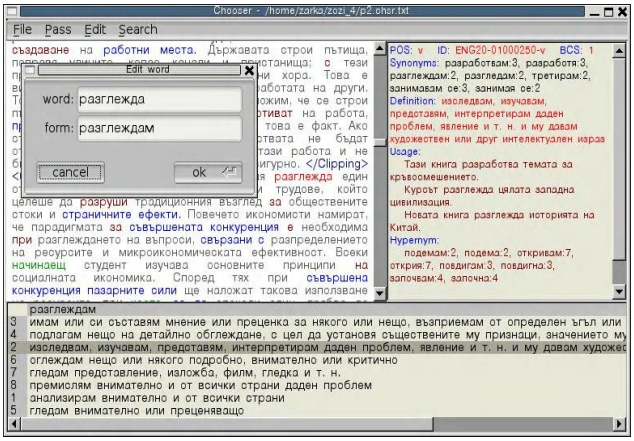

Figure 5. Edit interface

As in the linear module a Document object takes care for stream loading/saving functions and access to LUs and the annotation schema. Data representation and access is ensured by means of the xml format and an implementation of the Document-Object Model (DOM) in the following way. Along with a Node object and a Sentence object Document maintains data by means of a list of Sentence objects and the index of the current sentence. Sentence is a list of Node objects that provides an interface for the string representation of the sentence. Every Node object encapsulates a DOM node whose "path" attribute represents the path from the tree's root to itself in tree-structured representations. The DOM node also provides an interface for editing relations between nodes in graph-based annotations where relations are stored as the DOM node's children.

3.2 Implementation

The framework is a platform-independent C++ implementation with certain components written in Python and Perl. The User Interface uses FLTK[3], a cross-platform open source library for GUI. Other system-dependent features like the use of threads are implemented with another open-source portable library - Boost[4]. Graphs are displayed with Graphviz[5]. The utilisation of portable libraries along with the server's implementation in Perl makes it possible for Chooser to

[3] Fast Light Toolkit, cf. www.fltk.org
[4] A free open source and cross-platform library, cf. www.boost.org
[5] An open source graph visualization software, cf. http://www.graphviz.org/

be recompiled for and run on a number of operating systems (UNIX/LINUX, MacOS, Windows, OS/2, etc.).

3.3 Functionality

The already discussed features of Chooser's architecture can be summarised as:
Modular - different annotation tasks are handled in different interacting modules;
Task-independent – Chooser has been applied in various annotation projects involving different linguistic levels;
Customisable – Chooser's features may be easily extended and redefined to accommodate other both linguistic and non-linguistic tasks involving data annotation;
Multi-user – the architecture supports concurrent access by multiple users;
User-friendly – the design affords easy and compatible incorporation of new features, visualization and editing strategies, etc.;
Language-independent – under UTF-8 text encoding different languages can be managed, and language-dependent parameters can be easily re-configured;
Theory-independent – the tool supports annotation solutions couched in different linguistic frameworks that may be concurrently performed and maintained.

4. Putting into Practice

A POS disambiguated training corpus developed with the tool has been successfully used in the training of a Brill-designed POS taggers (Doychinova & Mihov, 2004), (Koeva 2007) and has come to be applied on a regular basis in a variety of other applications, the latest being a corpus search engine[6] (Tinchev et al. 2008). Major ongoing work employing Chooser involves the construction of a training word annotated corpus needed for the implementation of an HMM learning algorithm for WSD and machine translation (Koeva et al. 2006). Recently, a dependency bank has been initiated with the help of the tool.

5. Related Work

Principally, tools assisting manual annotation are either stand-alone applications, or multi-modular environments designed to handle different levels of annotation and personalised configuration including definition of annotation schemata (Callisto[7], GATE[8], MMAX2[9], WorkFreak[10]).
In the approach adopted in the design of Chooser different annotation tasks are handled in independent modules with a straightforward user interface that ensures fast and secure data storage and server-client data exchange, easy-to-do annotation through simple keyboard actions, and provides access to and data exchange with lexical resources while dispensing with complex level-specific configuration. At the same time the reusability of the annotated data enables the interaction with other Chooser modules and external applications.
Another Chooser's major characteristic as compared with other annotation tools is that it makes use of a single, uniform and re-importable linguistic database that is open to modifications. This has proved vital in the WSD implementation since instead of resorting to unspecified senses (Raileanu et al., 2002), linguistically motivated missing senses have been encoded in Bulgarian WordNet as new entries, and linked to the network according to established standards and criteria (Koeva et al. 2006)
Unlike corpus-oriented systems such as Word Sketch Engine[11] (Kilgarriff et al., 2004) designed for exploring corpora and retrieving particular chunks of information for LUs Chooser provides corpus search to the extent to which annotation is concerned such as finding particular word form(s) or lemmas in the annotation corpus, and passing tokens according to different strategies. As mentioned before edit features allow modifications of word forms and lemmas.
Chooser is largely compliant with widely used lexicographic tools such as VisDic and DebVisDic[12]. The current WSD application uses an XDB which is fully compatible with VisDic. Currently BulNet is constructed with an independently developed lexicographic tool – Hydra - using a relational database, described in (Rizov, 2008). The tool is comparable to DebVisDic in most of its main functionalities (except that the main implementation is a local client-server application), adding to them a powerful modal logic data representation and query language originally introduced by (Koeva, Tinchev & Mihov, 2004). The migration to Hydra was enabled through the implementation of an XML-to-RDB mapping algorithm. Nevertheless, a VisDic compliant XDB is maintained by means of an "export to xml" function which generates a VisDic compatible xml-formatted version of the database. As far as migration from VisDic to DebVisdic is ensured, compatibility with the latter should be unproblematic.

6. Conclusion and Future Work

The features outlined in the previous section, especially the modifiability and extendibility of the linguistic resources imported in the application and the multiple-user design make the tool valuable in other annotation activities related to manual disambiguation, such as post-editing of automatically annotated corpora.
Future work will be directed to enhancements aimed at streamlining annotators' work and the integration of new functionalities such as a web interface and coupling with the wordnet development tool Hydra.

References

Buschmann et al. (1996). F. Buschmann, R. Meunier, H. Rohnert, P.Sommerlad, M. Stal. Pattern-Oriented Software Architecture. John Wiley and Sons Ltd, Chichester.

Gamma et al. (1994). Erich Gamma, Richard Helm, Ralph Johnson, John M. Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley: Reading, MA, 1994.

Doychinova & Mihov. (2004). Diychinova V., Mihov S.

[6] http://dcl.bas.bg/dictionaries_en.html
[7] http://callisto.mitre.org
[8] http://gate.ac.uk/
[9] http://mmax.eml-research.de/
[10] http://wordfreak.sourceforge.net
[11] http://www.sketchengine.co.uk/
[12] http://nlp.fi.muni.cz/projekty/visdic/

High Performance Part-of-Speech Tagging of Bulgarian. In *Proceedings of the Eleventh International Conference on Artificial Intelligence: Methodology, Systems, Applications* LNAI #3192, pp. 246-255.

Ide & Romary (2006). Ide, N., Romary, L. Representing Linguistic Corpora and Their Annotations. Proceedings of the Fifth Language Resources and Evaluation Conference (LREC), Genoa, Italy.

Ide & Romary (2007). Ide, N., Romary, L. Towards International Standards for Language Resources. In Dybkjaer, L., Hemsen, H., Minker, W. (Eds.), Evaluation of Text and Speech Systems, Springer, 263-84.

Ide & Suderman (2006). Ide, N., Suderman, K. Integrating Linguistic Resources: The American National Corpus Model. Proceedings of the Fifth Language Resources and Evaluation Conference (LREC), Genoa, Italy.

Francis and Kucera. (1979). Brown Corpus Manual. Available:http://khnt.hit.uib.no/icame/manuals/brown/INDEX.HTM

Kilgarriff et al. (2004). Kilgarriff, A., Rychly, P., Smrz, P., Tugwell, D. The Sketch Engine. In *Proceedings of the Eleventh EURALEX International Congress*, pp. 105-116.

Koeva (1998). Grammar Dictionary of Bulgarian. Representation of Linguistic Data. *Bulgarian Language*, 7 (6).

Koeva (2007). Multi-word Term Extraction for Bulgarian, ACL 2007, Proceedings of the Workshop on Balto-Slavic NLP, p. 59-66.

Koeva, Tinchev & Mihov. (2004). Koeva S., Tinchev T., Mihiv S. Bulgarian Wordnet Structure and Validation. *Romanian Journal of Information Science and Technology*, 7 (1-2), pp. 61-78.

Koeva, Leseva & Todorova. (2006). Koeva S., Leseva S., Todorova M. Bulgarian Sense Tagged Corpus. In *Proceedings of the 5th SALTMIL Workshop on Minority Languages: Strategies for Developing Machine Translation for Minority Languages*, pp.79-87.

Koeva et al. (2006). Koeva, S., S. Leseva, I. Stoyanova, E. Tarpomanova, M. Todorova. Bulgarian Tagged Corpora. In *Proceedings of the Fifth International Conference Formal Approaches to South Slavic and Balkan Languages*. pp. 78-86.

Miller & Fellbaum. (1992). Semantic Networks of English. In B. Levin & S. Pinker (Eds.), *Lexical and Conceptual Semantics*. Blackwell, Cambridge and Oxford, England, pp. 197-229.

Pavelek & Pala (2002). Pavelek, T., and Pala K. VisDic - A New Tool for WordNet Editing. Proceedings of the International Wordnet Conference, January 21-25, Mysore, India, 192-195.

Raileanu et al. (2002). Raileanu D., Buitelaar P., Vintar S., Bay J. Evaluation Corpora for Sense Disambiguation in the Medical Domain. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation*, (LREC'02), May 29-31.

Rizov. (2008). Hydra – A Modal Logic Tool for Wordnet Development, Validation and Exploration. In *Proceedings of the 6th International Conference on Language Resources and Evaluation* (to appear).

Stamou et al. (2002). Stamou S., Oflazer K., Pala K., Christodoulakis D., Cristea D., Tufis D., Koeva S., Totkov G., Dutoit D., Grigoriadou M. BALKANET: A Multilingual Semantic Network for the Balkan Languages. In *Proceedings of the International Wordnet Conference.*

Tinchev et al. (2008). Tinchev, T., Koeva, S., Rizov, B., Obreshkov, N. A System of Advanced Corpus Search. In *Literature*. Sofia University, 2008 (to appear).