Integration of a Lexical Type Database with a Linguistically Interpreted Corpus

Chikara Hashimoto,[‡] **Francis Bond,**[*] **Takaaki Tanaka,**[*] **Melanie Siegel**[§]
[‡] Graduate School of Informatics, Kyoto University
[*] Machine Translation Research Group, NTT Communication Science Laboratories
[§] Language Technology Lab, DFKI
[‡] hasimoto@pine.kuee.kyoto-u.ac.jp
[*] {takaaki,bond}@cslab.kecl.ntt.co.jp
[§] siegel@dfki.de

Abstract

We have constructed a large scale and detailed database of lexical types in Japanese from a treebank that includes detailed linguistic information. The database helps treebank annotators and grammar developers to share precise knowledge about the grammatical status of words that constitute the treebank, allowing for consistent large scale treebanking and grammar development. In this paper, we report on the motivation and methodology of the database construction.

1 Introduction

Treebanks constructed with detailed linguistic information play an important role in various aspects of natural language processing; for example, grammatical knowledge acquisition; world knowledge acquisition (Bond et al., 2004b); and statistical language model induction. Such treebanks are typically semi-automatically constructed by a linguistically rich computational grammar.

A detailed grammar in turn is a fundamental component for **precise** natural language processing. It provides not only detailed syntactic and morphological information on linguistic expressions but also precise and usually language-independent semantic structures of them.

However, such a deep linguistic treebank and a grammar are often difficult to keep consistent through development cycles. This is both because multiple people, often in different locations, participate in a development activity, and because deep linguistic treebanks and grammars are complicated by nature. Thus, it is often the case that developers lose sight of the current state of the treebank and grammar, resulting in inconsistency.

We have constructed a linguistically enriched treebank named 'Hinoki' (Bond et al., 2004a), which is based on the same framework as the Redwoods treebank (Oepen et al., 2002) and uses the Japanese grammar JACY (Siegel and Bender, 2002) to construct the treebank.[1] In the construction process, we have also encountered the problem just mentioned. We are aiming to resolve this problem, which we expect many other project groups that are constructing detailed linguistic treebanks have encountered. Our strategy is to take a "snapshot" of one important aspect of the treebank and grammar for each development cycle. To be more precise, we extract information about lexical items that are being used in treebanking from the treebank and grammar and convert it into an electronically accesible structured database (the lexical-type database). Such a snapshot, the database, certainly helps treebank annotators and grammar developers to share precise and detailed knowledge of the treebank and grammar and thus to make them consistent throughout the development cycle.[2]

Lexical items whose information is included

[1] Currently, the Hinoki treebank contains about 121,000 sentences (about 10 words per sentence).

[2] We think we also need another snapshot, that of the grammar rules and principles being used. In this paper, however, we do not deal with it, and hopefully we will report on it some other time.

in the database are grouped together according to their grammatical behavior, and we will refer to each of the groups as a **lexical type** in the rest of the paper. A typical lexical item consists of an identifier, and then a triple consisting of the orthography, lexical-type and predicate: e.g., `inu_n_1` = ⟨ "犬", `common-noun-lex`, **`dog_n_animal`**⟩. The grammar treats all members of the same lexical type in the same way. the lexical type is the locus of syntactic and structural semantic information. Examples of lexical types will be described in §2.

The database could also benefit a wide range of language researchers, not just those who are treebanking. As the development of the treebank and grammar proceeds, together they describe the language (Japanese in this study) with increasing accuracy. As a result, the database that we obtain from the sophisticated treebank and grammar can be thought of as showing us the **real** view of the Japanese lexicon. Thus, though many of the details of the treebank and grammar are framework dependent, the database will provide NLP researchers who are aiming at deep linguistic processing of Japanese with a basic and reliable reference Japanese lexicon. The correctness can be verified by examining the treebanked examples. Such a resource is useful for Japanese language teachers, lexicographers, and linguists, in addition to NLP researchers.

The next section describes the framework of treebanking and motivates the need for the lexical type database. The third section discusses what information the lexical type database should contain to facilitate treebanking and grammar development; illustrates the contents of the database; and shows how the database is created. The fourth section discusses the usefulness of the lexical type database for many purposes other than treebanking. An overview of related works follows in the fifth section. Finally, we conclude the paper with a discussion of our plans for future work.

2 Background to the Database

The treebanking process is illustrated in Figure 1. As the figure shows, our treebank is semi-automatically generated by a computational grammar (and a parser). Each sentence is parsed and the intended reading chosen from the possible interpretations. In doing so, we find the grammar's flaws such as insufficient coverage and spurious ambiguities. The feedback allows us to refine the grammar so that it can have wider coverage and be more appropriately restricted. Currently this process is carried out by several people, distributed over four continents.

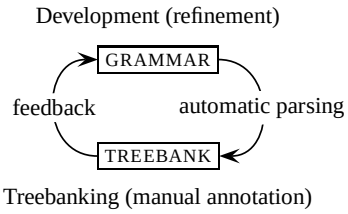

Figure 1: Treebanking Cycles

Although most treebanks are rarely updated, we consider the updating an integral part of the process. Thus our treebank is dynamic in the sense of Oepen et al. (2004).

As is often the case with detailed linguistic treebanking, our grammar and treebank consist of very fine-grained linguistic information. For example, our grammar, hence our treebank, distinguishes several usages of the Japanese dative marker *ni*. The Japanese sentence (1) can represent the two meanings described in (1a) and (1b). Lexical type names for each usage of *ni* are written in `typewriter` font.[3]

(1) hanasiai-wa sinya-**ni** itaru
discussion-TOP midnight-DAT reach

a. "The discussion comes (to a conclusion) at midnight."
ni as `adv-p-lex-1`

b. "The discussion continues until midnight."
ni as `ga-wo-ni-p-lex`

The dative phrase, *sinya-ni* (midnight-DAT), can act as either an adjunct (1a)[4] or an object of *itaru* "reach" (1b). Below is an example showing other usages of *ni*.

(2) Ken-wa yuka-o kirei-**ni** migaku
-TOP floor-ACC clean-DAT polish

[3] These are actual names of the lexical types implemented in our grammar and might not be understandable to people in general.

[4] The object, *a conclusion*, is expressed by a phonologically null pronoun. This is the so-called "pro-drop" phenomenon.

a. "Ken polishes a floor clean."
(The floor is clean.)
ni as `naadj2adv-end-lex`

b. "Ken cleanly polishes a floor."
(His way of polishing the floor is clean.)
ni as `adv-p-lex-6`

The dative phrase, *kirei-ni* (clean-DAT), is used as an adjunct in both (2a) and (2b), but their usages and meanings are different. The usage in (2b) is an ordinary adverb that describes the manner of Ken's polishing the floor as clean, while in (2a) the dative phrase describes the resulting situation of the floor after polishing as clean. In addition, the *ni*s in (1) and (2) are different in that the former takes nouns as its complement while the latter takes adjectives. Thus, the four usages in (1a), (1b), (2a) and (2b) must be distinguished so that we can obtain correct syntactic structures and semantic representations. In our terms, these *ni*s are said to belong to different lexical types.[5] Similarly, our grammar distinguishes usages of other words, notably functional ones.

However, as we augment the grammar with finer distinctions, the grammar becomes more and more opaque and difficult to maintain, and so is the treebank. This is problematic in three ways. Firstly, when we annotate parser outputs of one sentence, we have to see which parse is correct for the sentence. Consequently, we have to distinguish which word usage is correct for each word in the sentence. However, this task is not always trivial, since our grammar's word usage distinction is very fine grained as shown above. Secondly, when we add a word to the grammar to get wider coverage, we have to see which lexical type the word belongs to. That is, we are required to be familiar with lexical types of the grammar. Thirdly, in collaborative grammar development, it sometimes happens that a developer accidentally introduces a new lexical type that represents overlapping functionality with an existing type. This causes spurious ambiguity. As a result, the grammar will be unnecessarily bloated, and the treebank will also be easily inconsistent. Again, we see that comprehensive knowledge of the grammar's lexical types is indispensable.

[5]Usages of the Japanese dative marker, *ni*, are extensively discussed in, for example, Sadakane and Koizumi (1995).

In summary, it is important to make clear **(i)** what lexical types are assumed in a grammar and a treebank and **(ii)** how differently they are used from each other, so that we can make the treebank annotation and grammar development consistent.

Our solution to the problem is to construct a lexical type database of a treebank and a grammar. The database is expected to give us explicit information on **(i)** what lexical types are implemented in the grammar and are used in the treebank and **(ii)** how a word is used in Japanese and is distinguished from other words.

3 Architecture of the Database

This section details the content of the database and the method of its construction. The database itself is on-line at `http://pc1.ku-ntt-unet.ocn.ne.jp/tomcat/lextypeDB/`.

3.1 Content of the Database

First of all, what information should be included in such a database to help treebank annotators and grammar developers to work consistently? Obviously, once we construct an electronic lexicon, whatever information it includes, we can easily see what lexical types are assumed in the grammar and treebank. But we have to carefully consider what to include in the database to make it clear how each of the lexical types are used and distinguished.

We include five kinds of information:

(3) Contents of the Database
- a. Linguistic discussion
  - i Name
  - ii Definition
  - iii Criteria to judge a word as belonging to a given lexical type
  - iv Reference to relevant literature
- b. Exemplification
  - i Words that appear in a treebank
  - ii Sentences in a treebank that contain the words
- c. Implementation
  - i The portion of grammar source file that corresponds to the usage

ii Comments related to the portion
iii TODOs

d. Links to "confusing" lexical types
e. Links to other dictionaries

That is, we describe each lexical type in depth (3a–3c) and present users (treebank annotators and grammar developers) explicit links to other lexical types that share homonymous words (3d) (e.g. `adv-p-lex-1` vs `ga-wo-ni-case-p-lex` in (1)) to make it clear what distinguishes between them. Further, we present correspondences to other computational dictionaries (3e).

Linguistic discussion To understand lexical types precisely, linguistic observations and analyses are a basic source of information.

Firstly, the requirements for naming lexical-types in a computational system (3ai) are that they be short (so that they can be displayed in large trees) and easily distinguishable. Type names are not necessarily understandable for anyone but the developers, so it is useful to link them to more conventional names. For example `ga-wo-ni-p-lex` is a *Case Particle* (格助詞).

Next, the definition field (3aii) contains a widely accepted definition statement of the lexical type. For example, `ga-wo-ni-p-lex` (1b) can be defined as "a particle that indicates that a noun it attaches to functions as an argument of a predicate." Users can grasp the main characteristics from this.

Thirdly, the criteria field (3aiii) provides users with means of investigating whether a given word belongs to the class. That is, it provides positive and negative usage examples. By such usage examples, developers can easily find differences among lexical types. For example, `adv-p-lex-1` (1a) subcategorizes for nouns, while `adv-p-lex-6` (2b) subcategorizes for adjectives. Sentences like (1a) and (2b) that fit such criteria should also be treebanked so that they can be used to test that the grammar covers what it claims. This is especially important for regression testing after new development.

Finally, the reference field (3aiv) points to representative papers or books dealing with the lexical type. This allows the grammar developers to quickly check against existing analyses, and allows users as well to find more information.

Exemplification Examples help users understand lexical types concretely. As we have constructed a treebank that is annotated with linguistic information, we can automatically extract relevant examples exhaustively. We give the database two kinds of examples: words, that are instances of the lexical types (3bi), and sentences, treebanked examples that contain the words (3bii). This link to the linguistically annotated corpus examples helps treebankers to check for consistency, and grammar developers to check that the lexical types are grounded in the corpus data.

Implementation Grammar developers need to know the actual implementation of lexical types (3ci). Comments about the implementation (3cii) are also helpful to ascertain the current status. Although this section is necessarily framework-dependent information, all project groups that are constructing detailed linguistic treebanks need to document this kind of information. We take our examples from JACY (Siegel and Bender, 2002), a large grammar of Japanese built in the HPSG framework. As actual implementations are generally incomplete, we use this resource to store notes about what remains to be done. TODOs (3ciii) should be explicitly stated to inform grammar developers of what they have to do next.

We currently show the actual *TDL* definition, its parent type or types, category of the head (SYNSEM.LOCAL.CAT.HEAD), valency (SYNSEM.LOCAL.CAT.VAL), and the semantic type (SYNSEM.LOCAL.CONT).

Links to "confusing" lexical types For users to distinguish phonologically identical but syntactically or semantically distinct words, it is important to link confusing lexical types to one another within the database. For example, the four lexical types in (1) and (2) are connected with each other in terms of *ni*. That way, users can compare those words in detail and make a reliable decision when trying to disambiguate usage examples.[6]

[6]Note that this information is not explicitly stored in the database. Rather, it is dynamically compiled from the database together with a lexicon database, one of the component databases explained below, when triggered by a user query. User queries are words like *ni*.

Links to other dictionaries This information helps us to compare our grammar's treatment with that of other dictionaries. This comparison would then facilitate understanding of lexical types and extension of the lexicon. We currently link lexical types of our grammar to those of ChaSen (Matsumoto et al., 2000), Juman (Kurohashi and Nagao, 2003), ALT-J/E (Ikehara et al., 1991) and EDICT (Breen, 2004). For example, `ga-wo-ni-case-p-lex` is linked to ChaSen's 助詞-格助詞-一般 (`particle-case_particle-general`), Juman's 格助詞 (`case_particle`), and ALT-J/E's 付属語-格助詞-名詞・助詞後接 (`adjunct-case_particle-noun/particle_suffix`).

Figure 2 shows the document generated from the lexical type database that describes the lexical type, `ga-wo-ni-p-lex`.

3.2 Method of Database Construction

The next question is how to construct such a database. Needless to say, fully manual construction of the database is not realistic, since there are about 300 lexical types and more than 30,000 words in our grammar. In addition, we assume that we will refer to the database each time we annotate parser outputs to build the treebank and that we develop the grammar based on the treebanking result. Thus the database construction process must be quick enough not to delay the treebanking and grammar development cycles.

To meet the requirement, our method of construction for the lexical type database is semi-automatic; most of the database content is constructed automatically, while the rest must be entered manually. This is depicted in Figure 3.

- Content that is constructed automatically
  - Lexical Type ID (Grammar DB)
  - Exemplification (3b) (Treebank DB)
  - Implementation (3ci,ii) (Grammar DB)
  - Link to "confusing" lexical types (3d) (Lexicon DB)
  - Link to Other Lexicons (3e) (OtherLex DB)
- Content that is constructed manually
  - Linguistic discussion (3a)
  - TODOs (3ciii)

3.2.1 Component Databases

To understand the construction process, description of the four databases that feed the lexical type database is in order. These are the grammar database, the treebank database, the lexicon database, and the OtherLex database.

- The grammar database contains the actual implementation of the grammar, written as typed feature structures using *TDL* (Krieger and Schafer, 1994). Although it contains the whole implementation (lexical types, phrasal types, types for principles and so on), only lexical types are relevant to our task.
- The lexicon database gives us mappings between words in the grammar, their orthography, and their lexical types. Thus we can see what words belong to a given lexical type. The data could be stored as *TDL*, but we use the Postgresql lexdb (Copestake et al., 2004), which simplifies access.
- The treebank database stores all treebank information, including syntactic derivations, words, and the lexical type for each word. The main treebank is stored as structured text using the [incr tsdb()] (Oepen et al., 2002). We have also exported the derivation trees for the treebanked sentences into an SQL database for easy access. The leaves of the parse data consist of words, and their lexicon IDs, stored with the ID of the sentence in which the word appears.
- We also use databases from other sources, such as ChaSen, Juman and Edict.

3.2.2 Automatic Construction

Next we move on to describe the automatic construction. Firstly, we collect all lexical types assumed in the grammar and treebank from the grammar database. Each type constitutes the ID of a record of the lexical type database.

Secondly, we extract words that are judged to belong to a given lexical type and sentences that contains the words (Example (3b)) from the treebank database compiled from the Hinoki treebank (Bond et al., 2004a). The parsed sentences

格助詞, ga-wo-ni-p-lex (が, に, を)

Linguistic Discussion

`ga-wo-ni-p-lex` particles attach to a noun and indicate what grammatical relation (e.g., subject or object) the noun takes on in relation to a predicate. It does not mean anything by itself.

Right	Wrong
健が机に本を置く。	勉強したがわからない。
子供に愛情が必要だ。	10時に出発する。

Literature

[1] Koichi Takezawa. *A Configurational Approach to Case Marking in Japanese*. Ph.D. dissertation, University of Washington, 1987.
[bib]

[2] Shigeru Miyagawa. *Structure and Case Marking in Japanese (Syntax and Semantics 22)*. Academic Press, 1989.
[bib]

Examples

Lexical Entries (6)

が (ga), に (ni-case), を (o)

Example Sentences (54280)

Examples for が (ga)

- 人や物に対して、報酬 が無くても尽くしたいと思ったり、自分の手元に置きたいと思ったりする、 暖かい感情
- 記帳するとき、 不足額を赤色で示すことから、収入より支出 が多いこと
- 人が移動するために使う交通機関

Examples for に (ni-case)

- 中身を出して、空 に成った缶
- 飾り に成る付属品
- 衣服の飾り に成る付属品

Examples for を (o)

- 人を慕う心
- 犬をかわいがること
- 異性を恋して慕う心

More Examples

TDL Summary

TDL Definition

```
ga-wo-ni-p-lex := case-p-lex &
                  [SYNSEM.LOCAL.CAT.VAL.COMPS.FIRST.LOCAL.CAT.HEAD noun_head].
```

Supertype	Head Category	Valency	Content
case-p-lex	overt-case-p_head	p_sat	mrs

TODO

Dative subjects of stative predicates are not recognized.
"おまえにこれがわかるか。"
See also mental-stem-lex.

Links

CHASEN's Lexical type	JUMAN's Lexical type	ALT-J/E's Lexical type
助詞-格助詞-一般	格助詞	付属語-格助詞-名詞・助詞後接

Lexical Type List

Figure 2: Screenshot of the lexical type `ga-wo-ni-p-lex`

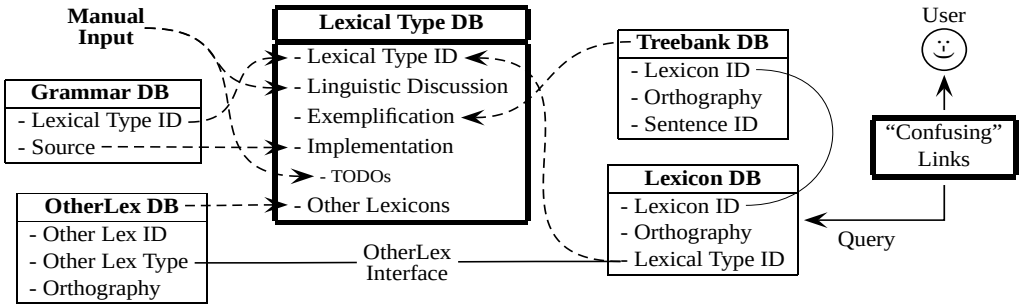

Figure 3: The Lexical Type Database Construction

can be seen in various forms: plain text, phrase structure trees, derivation trees, and minimal recursion semantics representations. We use components from the Heart-of-Gold middleware to present these as HTML (Callmeier et al., 2004).

Thirdly, implementation information except for TODOs is extracted from the grammar database (3ci,ii).

Fourthly, in order to establish "confusing" lexical type links (3d), we collect from the lexicon database homonyms of a word that users enter as a query. To be more precise, the lexicon database presents all the words with the same orthography as the query but belonging to different lexical types. These lexical types are then linked to each other as "confusing" in terms of the query word.

Fifthly, we construct links between our lexical types and POS's of other lexicons such as ChaSen from OtherLex DB (3e). To do this, we prepare an interface (a mapping table) between our lexical type system and the other lexicon's POS system. As this is a finite mapping it could be made manually, but we semi-automate its construction. The similarity between types in the two databases (JACY and some other lexicon) is calculated as the Dice coefficient, where $W(L_A)$ is the number of words W in lexical type L:

$$\text{sim}(L_A, L_B) = \frac{2 \times |(W(L_A \cap L_B)|}{|W(L_A)| + |W(L_B)|} \quad (1)$$

The Dice coefficient was chosen because of its generality and ease of calculation. Any pair where $\text{sim}(L_A, L_B)$ is above a threshold should potentially be mapped. The threshold must be set low, as the granularity of different systems can vary widely.

3.2.3 Manual Construction

Linguistic discussion (3a) and implementation TODOs (3ciii) have to be entered manually. Linguistic discussion is especially difficult to collect exhaustively since the task requires an extensive background in linguistics. We have several linguists in our group, and our achievements in this task owe much to them. We plan to make the interface open, and encourage the participation of anyone interested in the task.

The on-line documentation is designed to complement the full grammar documentation (Siegel, 2004). The grammar documentation gives a top down view of the grammar, giving the overall motivation for the analyses. The lexical-type documentation gives bottom up documentation. It can easily be updated along with the grammar.

Writing implementation TODOs also requires expertise in grammar development and linguistic background. But grammar developers usually take notes on what remains to be done for each lexical type anyway, so this is a relatively simple task.

After the database is first constructed, how is it put to use and updated in the treebanking cycles described in Figure 1? Figure 4 illustrates this. Each time the grammar is revised based on treebank annotation feedback, grammar developers consult the database to see the current status of the grammar. After finishing the revision, the grammar and lexicon DBs are updated, as are the corresponding fields of the lexical type database. Each time the treebank is annotated, annotators can consult the database to make sure the chosen parse is correct. Following annotation, the treebank DB is updated, and so is the lexical type database. In parallel to this, collaborators who are

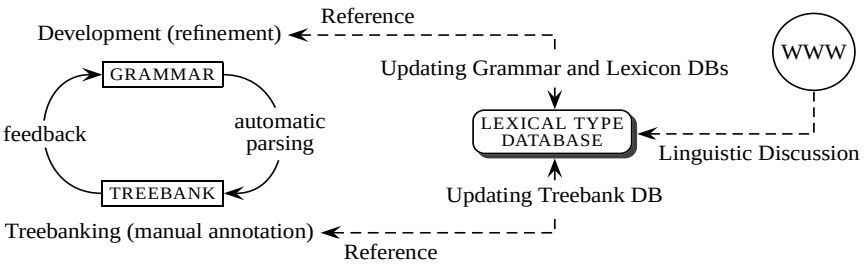

Figure 4: Database Construction Intergrated with Treebanking Cycles

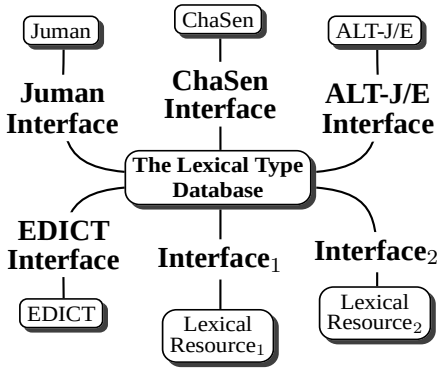

Figure 5: Synthesis of Lexical Resources

familiar with linguistics continue to enter relevant linguistic discussions via the WWW.

4 Lexical Type Database as a General Linguistic Resource

In this section, we discuss some of the ways the database can benefit people other than treebank annotators and grammar developers.

One way is by serving as a link to other lexical resources. As mentioned in the previous section, our database includes links to ChaSen, Juman, ALT-J/E, and EDICT. Currently, in Japanese NLP (and more generally), various lexical resources have been developed, but their intercorrespondences are not always clear. These lexical resources often play complementary roles, so synthesizing them seamlessly will make a Japanese lexicon with the widest and deepest knowledge ever. Among our plans is to realize this by means of the lexical type database. Consider Figure 5. Assuming that most lexical resources contain lexical type information, no matter how fine or coarse grained it is, it is natural to think that the lexical type database can act as a "hub" that links those lexical resources together. This will be achieved by preparing interfaces between the lexical type database and each of the lexical resources. Clearly, this is an intelligent way to synthesize lexical resources. Otherwise, we have to prepare ${}_nC_2$ interfaces to synthesize n resources. The problem is that construction of such an interface is time consuming. We need to further test generic ways to do this, such as with similarity scores, though we will not go on further with this issue in this paper.

Apart from NLP, how can the database be used? In the short term our database is intended to provide annotators and grammar developers with a clear picture of the current status of the treebank and the grammar. In the long term, we expect to create successively better approximations of the Japanese language, as long as our deep linguistic broad coverage grammar describes Japanese syntax and semantics precisely. Consequently, the database would be of use to anyone who needs an accurate description of Japanese. Japanese language teachers can use its detailed descriptions of word usages, the links to other words, and the real examples from the treebank to show for students subtle differences among words that look the same but are grammatically different. Lexicographers can take advantage of its comprehensiveness and the real examples to compile a dictionary that contains full linguistic explanations.

The confidence in the linguistic descriptions is based on the combination of the precise grammar linked to the detailed treebank. Each improves the other through the treebank annotation and grammar development cycle as depicted in Figure 1.

5 Related Work

Tsuchiya et al. (2005) have been constructing a database that summarizes multiword functional

expressions in Japanese. That describes each expression's linguistic behavior, usage and examples in depth. Notable differences between their database and ours are that their database is mostly constructed manually while ours is constructed semi-automatically, and that they target only functional expressions while we deal with all kinds of lexical types.

Hypertextual Grammar development (Dini and Mazzini, 1997) attempted a similar task, but focused on documenting the grammar, not on linking it to a dynamic treebank. They suggested creating the documentation in the same file along with the grammar, in the style of *literate programming*. This is an attractive approach, especially for grammars that change constantly. However, we prefer the flexibility of combining different knowledge sources (the grammar, treebank and linguistic description, in addition to external resources).

The Montage project (Bender et al., 2004) aims to develop a suite of software whose primary audience is field linguists working on underdocumented languages. Among their tasks is to facilitate traditional grammatical description from annotated texts by means of one of their products, the Grammar export tool. Although in the paper there is little explicit detail about what the "traditional grammatical description" is, they seem to share a similar goal with us: in the case of Montage, making grammatical knowledge assumed in underdocumented languages explicit, while in our case making lexical types assumed in the treebank and the computational grammar understandable to humans. Also, some tools they use are used in our project as well. Consequently, their process of grammatical description and documentation looks quite similar to ours. The difference is that their target is underdocumented languages whose grammatical knowledge has so far not been made clear enough, while we target a familiar language, Japanese, that is well understood but whose computational implementation is so large and complex as to be difficult to fully comprehend.

Another notable related work is the COMLEX syntax project (Macleod et al., 1994). Their goal is to create a moderately-broad-coverage lexicon recording the syntactic features of English words for purposes of computational language analysis. They employed elves ("elf" = enterer of lexical features) to create such a lexicon by hand. Naturally, the manual input task is error-prone. Thus they needed to prepare a document that describes word usages by which they intended to reduce elves' errors. It is evident that the document plays a role similar to our lexical type database, but there are important divergences between the two. First, while their document seems to be constructed manually (words chosen as examples of lexical types in the documentation are not always in the lexicon!), the construction process of our database is semi-automated. Second, somewhat relatedly, our database is electronically accessible and well-structured. Thus it allows more flexible queries than a simple document. Third, unlike COMLEX, all the lexical types in the database are actually derived from the working Japanese grammar with which we are building the treebank. That is, all the lexical types are defined formally. Fourth, examples in our database are all real ones in that they actually appear in the treebank, while most of the COMLEX examples were created specifically for the project. Finally, we are dealing with all kinds of lexical types that appear in the treebank, but the COMLEX project targets only nouns, adjectives, and verbs.

6 Future Work

We are currently experimenting with moving some of the information (in particular the type name and criteria) into the actual grammar files, in the same way as Dini and Mazzini (1997). This would make it easier to keep the information in sync with the actual grammar.

We have discussed the motivation, contents and construction of the lexical type database. We plan to evaluate the database **(i)** by measuring treebank inter-annotator agreement and **(ii)** by evaluating the coverage, the amount of spurious ambiguity, and efficiency of the grammar before and after introducing the database in the treebanking and grammar development cycles. We expect that treebank annotators will be more consistent when they can refer to the database and that grammar developers can more easily find the grammar's flaws (like lack of lexical items and overlapping implementations of the same lexical

type) by looking into the database.

Although this paper deals with a lexical type database of Japanese, the importance of such a database certainly holds for any large scale deep grammar. We use the tools from the DELPH-IN collaboration[7] and plan to make our additions available for groups working with other languages. In particular, we plan to construct a lexical type database for the Redwoods treebank, which is semi-automatically constructed from the English Resource Grammar (ERG) (Flickinger, 2000).

Acknowledgements

We would like to thank the other members of Machine Translation Research Group, Dan Flickinger, Stephen Oepen, and Jason Katz-Brown for their stimulating discussion.

References

Emily M. Bender, Dan Flickinger, Jeff Good, and Ivan A. Sag. 2004. Montage: Leveraging Advances in Grammar Engineering, Linguistic Ontologies, and Mark-up for the Documentation of Underdescribed Languages. In *Proceedings of the Workshop on First Steps for the Documentation of Minority Languages: Computational Linguistic Tools for Morphology, Lexicon and Corpus Compilation, LREC2004*, Lisbon, Portugal.

Francis Bond, Sanae Fujita, Chikara Hashimoto, Shigeko Nariyama, Eric Nichols, Akira Ohtani, Takaaki Tanaka, and Shigeaki Amano. 2004a. The Hinoki Treebank — Toward Text Understanding. In *Proceedings of the 5th International Workshop on Linguistically Interpreted Corpora (LINC-04)*, pages 7–10, Geneva.

Francis Bond, Eric Nichols, and Sanae Fujita Takaaki Tanaka. 2004b. Acquiring an Ontology for a Fundamental Vocabulary. In *20th International Conference on Computational Linguistics (COLING-2004)*, pages 1319–1325, Geneva.

J. W. Breen. 2004. JMDict: a Japanese-mulitlingual dictionary. In *Coling 2004 Workshop on Multilingual Linguistic Resources*, pages 71–78, Geneva.

Ulrich Callmeier, Andreas Eisele, Ulrich Schäfer, and Melanie Siegel. 2004. The DeepThought core architecture framework. In *Proceedings of LREC-2004*, volume IV, Lisbon.

Ann Copestake, Fabre Lambeau, Benjamin Waldron, Francis Bond, Dan Flickinger, and Stephan Oepen. 2004. A lexicon module for a grammar development environment. In *4th International Conference on Language Resources and Evaluation (LREC 2004)*, volume IV, pages 1111–1114, Lisbon.

[7] `http://www.delph-in.net/`

Luca Dini and Giampolo Mazzini. 1997. Hypertextual Grammar Development. In *Computational Environments for Grammar Development and Linguistic Engineering*, pages 24–29, Madrid. ACL.

Dan Flickinger. 2000. On building a more efficient grammar by exploiting types. *Natural Language Engineering, 6 (1) (Special Issue on Efficient Proceeding with HPSG*, pages 15–28.

Satoru Ikehara, Satoshi Shirai, Akio Yokoo, and Hiromi Nakaiwa. 1991. Toward an MT system without pre-editing – effects of new methods in **ALT-J/E**–. In *Third Machine Translation Summit: MT Summit III*, pages 101–106, Washington DC. (`http://xxx.lanl.gov/abs/cmp-lg/9510008`).

Hans-Ulrich Krieger and Ulrich Schafer. 1994. $\mathcal{TDL}$ — a type description language for constraint-based grammars. In *Proceedings of the 15th International Conference on Computational Linguistics*.

Sadao Kurohashi and Makoto Nagao. 2003. Building a Japanese parsed corpus — while improving the parsing system. chapter 14, pages 249–260.

Catherine Macleod, Ralph Grishman, and Adam Meyers. 1994. The Comlex Syntax Project: The First Year. In *Proceedings of the 1994 ARPA Human Language Technology Workshop*.

Yuji Matsumoto, Akira Kitauchi, Tatsuo Yamashita, Yoshitaka Hirano, Hiroshi Matsuda, Kazuma Takaoka, and Masayuki Asahara, 2000. *Morphological Analysis System ChaSen version 2.2.1 Manual*. Nara Institute of Science and Technology, Dec.

Stephan Oepen, Dan Flickinger, Kristina Toutanova, and Christoper D. Manning. 2002. LinGO Redwoods: A Rich and Dynamic Treebank for HPSG. In *Proceedings of The First Workshop on Treebanks and Linguistic Theories*, pages 139–149, Sozopol, Bulgaria.

Stephan Oepen, Dan Flickinger, and Francis Bond. 2004. Towards Holistic Grammar Engineering and Testing. Grafting Treebank Maintenance into the Grammar Revision Cycle. In *Proceedings of the IJCNLP Workshop Beyond Shallow Analysis*, Hainan,China.

Kumi Sadakane and Masatoshi Koizumi. 1995. On the nature of the "dative" particle *ni* in Japanese. *Linguistics*, 33:5–33.

Melanie Siegel and Emily M. Bender. 2002. Efficient Deep Processing of Japanese. In *Proceedings of the 3rd Workshop on Asian Language Resources and International Standardization*, Taipei, Taiwan.

Melanie Siegel. 2004. JACY a practical Japanese HPSG. ms.

Masatoshi Tsuchiya, Takehito Utsuro, Suguru Matsuyoshi, Satoshi Sato, and Seiichi Nakagawa. 2005. A corpus for classifying usages of japanese compound functional expressions. In *Proceedings of Pacific Association for Computational Linguistics 2005*, Tokyo, Japan.