Open Source Machine Translation with DELPH-IN

Francis Bond,[⊙♣] **Stephan Oepen,**[♣♠] **Melanie Siegel**[♡]
Ann Copestake,[◇] **Dan Flickinger,**[♣♠]

[⊙]NTT Communication Science Laboratories, 2-4 Hikaridai, Kyoto 619-0237 (Japan)
[♣]Universitetet i Oslo, Boks 1102 Blindern; 0317 Oslo (Norway)
[♠]Center for the Study of Language and Information, Stanford, CA 94305 (USA)
[♡]DFKI, Stuhlsatzenhausweg 3, D-66123 Saarbrücken, (Germany)
[◇]University of Cambridge, 15 JJ Thomson Avenue, Cambridge CB3 0FD (UK)

Abstract

The Deep Linguistic Processing with HPSG Initiative (DELPH-IN) provides the infrastructure needed to produce open-source semantic transfer-based machine translation systems. We have made available a prototype Japanese-English machine translation system built from existing resources include parsers, generators, bidirectional grammars and a transfer engine.

1 Introduction

Despite the recognized need for translation, especially in the international collaborative world of open source software, there is no widely used open source machine translation system. In contrast, there are well developed open source translation memory systems (e.g. OmegaT[1] or gtranslator[2]) and lexicons (e.g. Papillon[3] or Freedict[4]). Several projects have been started, but they were unable to produce a working machine translation system (e.g. Traduki or GPL trans).

One of the major reasons for this lack of success is the complexity of the task. A standard rule-based machine translation (RBMT) system using a transfer architecture needs at least a source language (SL) analysis module and its dictionaries, a target language (TL) generation module and its dictionaries, in addition to a transfer module and its dictionaries. Because natural language analysis and generation are still largely unsolved problems, each module involves a great deal of work. Further, compiling dictionaries requires walking through a minefield of copyright problems. Without all of these resources, one cannot even begin to build a RBMT system. Similarly, other approaches, such as example-based machine translation (EBMT) or statistical machine translation (SMT) rely on the existence of large amounts of well aligned bilingual text. Although much bilingual text is available on the web, little of it is well aligned.

In this paper we introduce a collection of open source natural language processing resources that simplifies the development of open source machine translation. The Deep Linguistic Processing with HPSG Initiative (DELPH-IN[5]) is coordinating the production of grammars, lexicons, parsers, generators, and a transfer engine that share a common framework. In principal, this means that the components can be combined together in various configurations, including transfer-based machine translation. The most important interface is the semantic representation, Minimal Recursion Semantics (MRS; Copestake, Flickinger, Pollard, & Sag, in press)—this is the output of the DELPH-IN parsers and the input to the generators. By using existing open source resources and tools we can start to work on the interesting task of machine translation right away.

In section 2 we introduce the overall system and the semantic transfer approach it uses. In section 3 we give an overview of the grammars and development tools we use. Finally, we compare our system to related work and outline some of the possible future extensions.

2 The DELPH-IN MT Architecture

The DELPH-IN machine translation architecture is a straightforward pipelined semantic transfer-based system. The transfer engine and the interfaces to the analysis and generation modules were developed by the Norwegian LOGON consortium (Oepen et al., 2004). We will give examples using a prototype that goes from Japanese to English. A similar system that uses a proprietary parser has been built for Norwegian–English, and we are currently also working on an English-to-Japanese prototype.

[1] http://www.omegat.org/omegat/omegat.html
[2] http://gtranslator.sourceforge.net/
[3] http://www.papillon-dictionary.org/
[4] http://www.dicts.info/
[5] http://www.delph-in.net/

We chose to develop a rule-based system, rather than example-based or statistical for three reasons. The first is that RBMT has a proven track record—it is used successfully in many commercial and research systems. The second is that the components of a RBMT system (the grammars, parsers and generators) are useful for other tasks, an important consideration for the overall DELPH-IN initiative. This means that work on the MT system benefits other projects, and work on other projects will also improve the MT system. EBMT and SMT systems provide results that are harder to re-use. The final reason is that SMT systems, although the current leaders in fixed domains with lots of well aligned parallel text, are not easy to use in domains without such text. In our experience, a lack of well aligned parallel text is the norm, rather than the exception.

2.1 Architecture Overview

The architecture is shown in Figure 1. The source text is analyzed with a rule-based source language grammar. The most appropriate interpretation or interpretations are selected using a stochastic ranking model (Oepen et al., 2002). The result is a precise, but underspecified language specific semantic representation (MRS_S). This representation is transformed by the transfer engine using a resource-sensitive rewrite process, where rules rewrite MRS fragments (SL to TL) in a step-wise manner and every element of the SL utterance must be interpreted exactly once per translation. Rules are applied in a given order, and may apply more than once. If multiple rules could apply the system will apply them all and thus create multiple translations.

The target language semantic representation (MRS_T) is then given to the target language generator, where it is realized as a string. Despite the use of underspecification where applicable, each component will typically output multiple hypotheses—corresponding to distinct parses, for example, in analysis. Cascading ambiguity through the translation pipeline yields a fan-out tree, where specialized stochastic processes for each step allow ranking and pruning of intermediate results (see e.g. Velldal, Oepen, & Flickinger, 2004).

Because transfer involves only the semantic representation, it hides the grammar internal specifics. For example, the fact that the English degree specifier *enough* idiosyncratically follows its head (*That mountain is high enough* c.f. *That mountain is too high*) need not be a concern at the transfer level.

The general set-up is similar to transfer in Verb*mobil* (Wahlster, 2000), operating on semantic representations only, but adding two new elements: (i) the use of typing for hierarchical organization of transfer rules and (ii) a chart-like treatment of transfer-level ambiguity.

One difference is that the same grammars are used for both parsing and generation. This requires some care when writing the grammars but means that any development work produces gains in two modules.

Because the transfer is based on a well developed semantic representation it can also be used for grammars developed outside of DELPH-IN, as long as they can provide sufficiently informative semantic analyses. This is being done in the LOGON project, where the analysis grammar is written in Lexical Functional Grammar (LFG: Dyvik, 2003) and uses the proprietary XLE architecture.

2.2 Example of Translation

Consider the Japanese sentence (1) and its semantic representation (2). The building blocks are elementary predications (EPs), like biiru_n(x_1) "beer" and motsu_v(u_2, x_2), "hold" corresponding to atomic formulas in predicate logic. Quantifiers (e.g. udef_q) introduce special relations in an MRS, corresponding to generalized quantifiers. All EPs are labeled with *handles*, e.g. h_4 is the label on the predication biiru_n(x_1). The highest scoping handle h_1 is called the hook, and is listed first.

(1) ビールを 三つ もって きて
biiru-wo mittsu motte kite
beer-ACC three-CL hold come
ください
kudasai
please
Please bring three beers.

(2) $\langle h_1, \{h_1$: imp_m(h_3),
h_4: biiru_n(x_1),
h_6: udef_q(x_1, h_7, h_8),
h_9: card(u_1, x_1, "3"),
h_{11}: motsu_v(e_1, u_2, x_1),
h_{11}: kuru_v(e_2, u_3),
h_{15}: kudasaru_v(e_3, u_4, u_5, h_{17}),
h_{17}: proposition_m(h_{18}) $\}$,
$\{h_3 =_q h_{15}, h_7 =_q h_4, h_{18} =_q h_{11}\}\rangle$

MRS representations are language specific, rather than being an interlingua. However,

Figure 1: Schematic LOGON system architecture: the three core processing components are managed by a central controller that passes intermediate results (MRSs) through the translation pipeline.

there is some semantic decomposition and they abstract away from the syntactic structure in several ways. For example, in (2) the sense of the entire sentence is given in the topmost message type (imp[erative]_m[essage]) a (polite) imperative, this is shown in Japanese by using the final verb *kudasai* "give-honorific.imperative". The MRS also anchors the floating quantifier *mittsu* "three objects", so that it is represented as a restriction on the cardinality of the beer card$(u_1, x_1, \text{"3"})$.

The MRS of the English translation is given in (4):

(3) Please bring three beers.

(4) $\langle h_0, \{h_0$: please_a(e_3, h_1),
h_1: imp_m(h_3),
h_2: pronoun_q(x_0, h_7, h_8),
h_4: pron$(x_0\{2nd\})$,
h_5: bring_v(e_2, x_0, x_1),
h_4: beer_n(x_1),
h_6: udef_q(x_1, h_{10}, h_8),
h_{11}: card$(u_1, x_1, \text{"3"})$ $\}$,
$\{h_3 =_q h_5, h_7 =_q h_4, h_{10} =_q h_{11}, \}\rangle$

Although the syntax is quite different, the MRS is similar. There are three main differences. The first, and most obvious, is that the predicate names are different. Therefore translation rules are required to transform biiru_n(x_i) into beer_n(x_i). A slightly more complicated rule takes two predicates with the same handle and compatible arguments (h_j: motsu_v(e_1, u_2, x_1) "hold" and h_j: kuru_v(e_2, u_3) "come") and translates them to h_j: bring_v(e_1, u_2, x_1). These simple rules can be semi-automatically compiled from a bilingual dictionary. The order of predicates shown here is also different, but this is unimportant, the EPs are an unordered bag—all the information necessary to order the output text is given in the handles. This flat structure makes it easy to apply rules to MRSs—rules can apply anywhere in the structure, so long as they preserve the relations between handles and arguments.

More interestingly, the English MRS instantiates the implicit second person pronoun *[you] please bring three beers.* This must be provided by a rule that triggers on the message type: the subject of a imperative must be a second person pronoun.

Finally, the verb *kudasaru*, which makes the imperative a request, is translated into the English adverb *please*, retaining the scope relations. This is in fact just the same type of rule as the first two. Even changing parts of speech is simple as long as the semantic structures are similar. The DELPH-IN semantic transfer architecture makes the source and target grammars do much of the work, allowing the transfer to be simpler.

Of course, this is far from solving the problems of machine translation. The problems of sense disambiguation remain (is 鳩 *hato* a *dove* or a *pigeon*), as well as the fundamental problems due to differences in languages. For example, Japanese does not distinguish between singular and plural, or countable and uncountable, so *biiru-wo kudasai* could be *a beer please* or just *beer please.* We currently approach these problems by producing multiple translations, one of which will be selected by the stochastic realization model (Velldal et al., 2004).

2.3 Summary

This introduction only gives a general idea of how the translation works. A more detailed description of the overall architecture can be found in Oepen et al. (2004). Documentation is also available on the DELPH-IN wiki pages (`http://wiki.delph-in.net/`).

The Japanese-to-English system is just a proof of concept, with fewer than a hundred transfer rules and a transfer lexicon in the thousands. We still have no measure of how many rules would be necessary for a grammar of a given size. Experience with previous systems suggests at least twice as many transfer rules

are required as source language lexical entries. Many words require only one rule, e.g. biiru_n → beer_n, but there are of course many words with multiple translations. In addition, rules are required to deal with language differences.

3 Grammars and Development Tools

The multi-lingual linguistic resources and development tools that we used to assemble our open-source MT system are all taken from the public repository of the Deep Linguistic Processing with HPSG Initiative (DELPH-IN)[6]. DELPH-IN is a loosely-organised, multi-national effort aiming to provide and maintain a pool of re-usable open-source tools for NLP rooted in 'deep' linguistic traditions. In-depth analysis, in this view, contrasts with 'shallow' (or statistical) approaches to language processing, where typical NLP applications to date incorporate limited (or no) linguistic expertise but focus on solving a specific task to some approximation. Mid- and long-term success in NLP will demand increasing degrees of precision.

The DELPH-IN partners have adopted Head-Driven Phrase Structure Grammar (HPSG; Pollard & Sag, 1994) and Minimal Recursion Semantics (MRS; Copestake et al., in press) as a common theoretical background, and aim to develop scalable linguistic resources that enable a variety of NLP tasks. Tools are implemented in several development and processing environments (that can serve differing purposes) and which enable the exchange of grammars and lexicons across platforms. Formalism continuity, on the other hand, has allowed DELPH-IN researchers to develop several comprehensive, wide-coverage grammars of diverse languages that can be processed by a variety of software tools.

DELPH-IN members share a commitment to re-usable, multi-purpose resources and active exchange. Based on contributions from several members and joint development over many years, an open-source repository of software and linguistic resources has been created that has wide usage in education, research, and application building.

3.1 Development Tools

Over time, the following configuration of core components has emerged as a typical grammar engineering configuration that is commonly used both by DELPH-IN members and other research initiatives. The resources are all available at the DELPH-IN web site.

3.1.1 Linguistic Knowledge Builder

The Linguistic Knowledge Builder (LKB; Copestake, 2002) provides an interactive grammar development environment for typed feature structure grammars. The LKB includes a parser and generator, visualization tools for all relevant data structures (including trees, feature structures, MRSs, hierarchies, parse and generation charts), and a set of specialized debugging facilities (like interactive unification) and well-formedness tests for the grammar and the lexicon. The entire source is released under a very open license, essentially the MIT License.

3.1.2 The PET System

The PET System (Callmeier, 2002) for the high-efficiency processing of typed feature structure grammars complements the LKB as a run-time and application delivery component. PET interprets the same logical formalism (in fact reads the exact same grammar source files) and provides a parser that is (much) less resource-demanding than the LKB, more robust, portable, and available as a library that can be embedded into NLP applications. PET is released under the LGPL.[7]

3.1.3 The [incr tsdb()] Profiler and System Controller

The [incr tsdb()] Competence and Performance Profiler (Oepen & Callmeier, 2000) provides an evaluation and benchmarking tool to grammar writers and system developers alike. [incr tsdb()] (*'tee ess dee bee plus plus'*) acts like an umbrella application to a range of processing systems for typed feature structure grammars, including the LKB and PET, and defines a common format for the organization of test suites or corpora and the storage of precise and fine-grained measures of grammar and processor behavior. [incr tsdb()] also provides support for creating HPSG treebanks (Oepen et al., 2002). It is released under the LGPL.

The [incr tsdb()] profiler provides a powerful interface to component configuration and batch processing facilities, such that developers are able to perform high-frequency fully automated diagnostic and regression testing. For use in the MT scenario, the [incr tsdb()] profiler has been

[6] See 'http://www.delph-in.net/' for background information, including the list of current participants and pointers to available resources and documentation.

[7] http://www.gnu.org/copyleft/lesser.html

adapted for use with the LOGON transfer component and generator (Oepen et al., 2005). While test inputs for the parser are plain strings, typically each an individual root-level utterance, the latter two components take a complete profile as their input—essentially using the MRS meaning representations produced by an earlier processing stage in the pipeline (analysis or transfer, respectively) as their inputs. Our open-source MT pipeline, then, is realized as a cascade of general-purpose components that are invoked from the [incr tsdb()] controller sequentially. The uniform interface representation among components is MRS.

An immediate benefit of the use of [incr tsdb()] for MT is its built-in support for distributed and parallel computation across (standard) networked workstations. Using a mini-HPC cluster of four dual-Xeon Linux nodes, exhaustive batch translation of a development corpus (around 100 sentences with full fan-out) can be accomplished in a matter of minutes, while a strictly sequential batch on a single cpu would take time on the order of an extended lunch break. Given the internal complexity of each of the components and multiple dimensions of possible interactions, the ability to obtain an up-to-date system snapshot, empirically assessing the impact of most recent changes, is an important element in our highly data-driven approach to MT engineering. Furthermore, the controller itself—brokering intermediate results within the translation pipeline—utilizes the [incr tsdb()] API in communication with individual modules, such that component-level profiling and end-to-end evaluation are merely nested instances of the same protocol.

3.2 Grammars

Linguistic resources that are available as part of the DELPH-IN open-source repository include broad-coverage grammars for English, German, and Japanese and a set of 'emerging' grammars for French, Korean, Modern Greek, Norwegian, Spanish, Swedish, and Portuguese. Additionally, proprietary grammars for Danish and Italian use the same DELPH-IN formalism (and MRS interface representation) and are available for licensing (though not yet as open-source). The following sections discuss the grammars currently in use in our open-source MT prototype further. Finally we introduce a research program that seeks to find and codify commonalities between the grammars.

3.2.1 JACY

The JACY grammar is an HPSG-based grammar of Japanese which originates from work done in the Verb*mobil* project (Siegel, 2000) on machine translation of spoken dialogues in the domain of travel planning. It has since been extended to accommodate written Japanese and new domains (such as automatic email response and parsing machine readable dictionaries).

The lexicon contains around 36,000 lexemes. The system also includes a mechanism to assume default lexical types for items that can be POS tagged by the ChaSen tokenizer and POS tagger (Asahara & Matsumoto, 2000), but are not included in the HPSG lexicon. As the grammar is developed for use in applications, it treats a wide range of basic constructions of Japanese. In the multilingual context in which this grammar has been developed, a high premium is placed on parallel and consistent semantic representations between grammars for different languages. Ensuring this parallelism enables the reuse of the same downstream technology, no matter which language is used as input.

3.2.2 The ERG

The LinGO English Resource Grammar (ERG; Flickinger, 2000) is a broad-coverage, linguistically precise HPSG-based grammar of English that has been under development at the Center for the Study of Language and Information (CSLI) at Stanford University since 1993. The ERG was originally developed within the Verb*mobil* machine translation effort, but over the past few years has been ported to additional domains (most notably in an e-commerce and financial services self-help product, and more recently translating text about hiking and tourism as part of the LOGON MT effort) and significantly extended. The grammar includes a hand-built lexicon of around 25,000 lexemes and allows interfacing to external lexical resources (like COMLEX: Grishman, Macleod, & Myers, 1994).

3.2.3 The Grammar Matrix

Essential to the idea of developing machine translation systems applicable to multiple language pairs are the dual concerns of facilitating and harmonizing the construction of the grammars and lexicons. Facilitation addresses the concern that the construction of a large grammar and lexicon from scratch is an extremely time-consuming task; harmonization addresses

the desirability of providing grammars and lexicons of different languages with a certain degree of uniformity, so as to enhance the cross-linguistic applicability of systems drawing on deep processing. To accommodate both of these concerns, a strategy of multilingual grammar engineering has been defined, crucially featuring a sub-grammar called "The HPSG Grammar Matrix" (Bender, Flickinger, & Oepen, 2002). The Matrix consists of a skeleton of grammatical and lexical types, combined with a system of semantic representation—Minimal Recursion Semantics. It therefore constitutes a possible formal backbone for a large scale grammar of—in principle—any language. New grammar resources (e.g. for Italian and Norwegian) were built using the Matrix as a 'starter-kit for grammar writing'. Three existing grammars (English, German and Japanese) were adapted to the Matrix restrictions.

Using the Matrix simplifies the translation process by normalizing not just the type hierarchies but also the names used for common types. For example, before the Matrix, the ERG and JACY used different names for the message types (imp_m and command_m) which would require a collection of unnecessary transfer rules to translate between them.

4 Discussion

In this section we discuss the current state of the DELPH-IN resources, and outline some future work. We then look at some issues related to the open source nature of our system.

4.1 Current Status

In theory, any of the grammars in the DELPH-IN repository could be used in machine translation systems using MRS transfer. In practice, most of the grammars are more robust in analysis than generation, therefore they could not be used without additional development effort. Further, none of the grammars cover more than a subset of the languages they handle. Precision (correctness of analyses) has been a higher priority than robustness.

The tools also vary in robustness and the amount of documentation. In particular, while the basic functionality of the LKB, PET and [incr tsdb()] is documented, recent developments (such as the machine translation architecture) are not. This is a common problem for open source software under rapid development.

On the other hand, DELPH-IN is following the open source development approach by setting up wikis and archived mailing lists for the major components[8]. These are starting to produce up-to-date documentation, and several research projects using the DELPH-IN tools have included documentation as deliverables. There are also several mailing lists which are read by most of the developers, and are being used to archive discussions.

Finally, there is a large and diverse community of users and developers who have managed to attract funding from various companies and agencies, thus ensuring that the resources are continually being developed. Current uses include question answering, ontology extraction, computer aided language learning, and teaching computational linguistics in addition to machine translation.

4.2 Future Work

From the potential MT user's point of view, the current prototypes would need to be developed in several ways before the systems are generally useful. The most basic is to increase the coverage of the system. This would require additions to the source, transfer and target grammars and lexicons. In addition, the interface to the MT system needs to be documented as an API.

From the research point of view, the prototype can already be used as a base for further research, but really needs some more documentation before anyone but the current developers can use it. Some of the open research questions currently being considered are:

1. How much of the semantic representation can be shared between languages (and thus require little or no transfer)?
2. How can we (semi-)automatically expand the grammars and lexicons? In particular, can we learn transfer rules from parallel text, or build them from bilingual dictionaries?
3. What role should lexical semantics (word senses) play in the analysis, transfer and generation?
4. What are the best features for the stochastic models, and what should they model (separate models for each stage or one large translation model)?

At the moment, the main advantages of making the DELPH-IN machine translation system

[8] http://wiki.delph-in.net, http://lists.delph-in.net/

open source are to the research community. Researchers working on grammatical phenomena can implement them in one of the DELPH-IN grammars and compare their results to other languages. Similarly, it is possible to test the effects of different computational techniques within the same framework, for example to objectively evaluate transfer strategies. However, we cannot hope for many people to become involved at this level.

In the future, as the components become more robust and the coverage wider, we hope to make the system available as an open source MT system within other applications. Kamei (1999) showed that, with some care given to the interface, people will share user dictionaries on the web with a proprietary system. Further, users will usefully critique each others entries. Translators and people interested in translation have a sophisticated knowledge of the problems of translation. We hope to one day harness this knowledge to improve open lexicons in the same way.

4.3 Open Source Machine Translation

There is a good summary of open source machine translation projects at the Wikipedia Meta-Wiki.[9] As it shows, there are several usable translation memories and lexicons, but no usable MT systems. The only open source translation tools currently usable send text to on-line web translation services. For example, the AbiWord plugin to Babelfish[10] which offers on-line translation support for Abiword by connecting to Altavista's Babelfish[11] web translation tool, which is powered by Systran.

At present, our system has nowhere near the coverage or quality of Babelfish, so could not be considered a replacement. However, now that the framework exists, it is possible to build on it, in a way that was not possible for earlier attempts which started from scratch. The existence of earlier projects shows that there are members of the open source community interested in machine translation. Plugins such as those for AbiWord mean that the infrastructure for using MT already exists. It is not yet clear to us how good a system has to before it will attract developers other than researchers. We hope to find out.

We find some encouragement in the success of Jim Breen's EDICT.[12] It is a Japanese-to-multilingual lexicon, largely built by one person, but with many contributors and dictionaries in compatible formats. It currently contains over 460,000 Japanese-to-English entries (including 350,000 proper names). Producing lexical entries for HPSG lexicons and the transfer lexicon is more work than producing plain bilingual entries. However, if adding this extra information made automatic translation possible we expect this would motivate people.

We can already semi-automatically construct translation rules from the SL and TL lexicons and a plain bilingual lexicon (e.g. for our Japanese-English prototype, JACY, the ERG and EDICT). While this provides only the most simple of rules (biiru_n → beer_n), we consider that this is a reasonable baseline to start work on the interesting problems.

Finally, we wish to offer a hypothesis on one of the reasons for the current success of SMT in machine translation research. One of the major bottlenecks for research into knowledge-based machine translation is the high start-up cost. Before research can start on the interesting problems specific to machine translation, one has to build a source language analysis module and its dictionaries, a target language generation module and its dictionaries, and a transfer module and its dictionaries. Because most RBMT systems were built before the idea of open source software became widespread, most of the systems and lexicons developed remain proprietary. This means that components are usually not shared between systems, making it hard to compare one particular component in two different systems.

In comparison, statistical machine translation (SMT) can be attempted as soon as a fairly literally translated bilingual aligned corpus becomes available. Code and algorithms were shared during SMT research, and a basic and widely used SMT tool kit is available on-line (The EGYPT Statistical Machine Translation Toolkit: `http://www.clsp.jhu.edu/ws99/projects/mt/toolkit/`). Standard corpora are widely available with fewer restrictions on their use than lexicons, at least for the research community. This has made it relatively easy for SMT to advance rapidly, as each generation of software could build directly on the one

[9] `http://meta.wikimedia.org/wiki/Wikipedia_Machine_Translation_Project`

[10] `http://abiword.pchasm.org/builds/`

[11] `http://babelfish.altavista.com/`

[12] `http://www.csse.monash.edu.au/~jwb/j_edict.html`

before it, using all of the best tested methods.

Of course, the difference in methodologies is only one factor, the fact that SMT gains linguistic knowledge from bitexts with little manual intervention where RBMT has to manually encode knowledge in the form of grammars and lexicons is the main difference. These differences may give RBMT the potential to derive an even greater benefit from the open source development model than SMT.[13] Because changes to the rules in RBMT systems directly affect the capabilities of the system, every time a linguist contributes something the performance of our MT system should go up.

5 Conclusion

By making the building blocks for RBMT available as open source, we aim to make substantive research possible for anyone. In particular, we are making it possible to compare different techniques within the same framework, thus allowing for objective evaluation of transfer strategies, and the adoption of the best strategies as a new baseline.

Acknowledgments

This work is in part supported by the Research Collaboration between NTT Communication Science Laboratories, Nippon Telegraph and Telephone Corporation and CSLI, Stanford University. We would also like to thank the anonymous reviewers for their extremely informative and interesting comments.

References

Asahara, M., & Matsumoto, Y. (2000). Extended models and tools for high-performance part-of-speech tagger. In *Proceedings of the 18th International Conference on Computational Linguistics* (pp. 21–27). Saarbrücken, Germany.

Bender, E. M., Flickinger, D., & Oepen, S. (2002). The grammar Matrix. An open-source starter-kit for the rapid development of cross-linguistically consistent broad-coverage precision grammar. In *Proceedings of the Workshop on Grammar Engineering and Evaluation at the 19th International Conference on Computational Linguistics.* Taipei, Taiwan.

Callmeier, U. (2002). Preprocessing and encoding techniques in PET. In S. Oepen, D. Flickinger, J. Tsujii, & H. Uszkoreit (Eds.), *Collaborative language engineering. A case study in efficient grammar-based processing.* Stanford, CA: CSLI Publications. (forthcoming)

Copestake, A. (2002). *Implementing typed feature structure grammars.* Stanford, CA: CSLI Publications.

Copestake, A., Flickinger, D., Pollard, C., & Sag, I. A. (2005). Minimal Recursion Semantics. An introduction. *Research on Language and Computation.*

Dyvik, H. (2003). ParGram. Developing parallel grammars. *ELSNews, 12*(2), 12–14.

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

Grishman, R., Macleod, C., & Myers, A. (1994). COMLEX syntax: Building a computational lexicon. In *Proceedings of the 15th International Conference on Computational Linguistics* (Vol. I, p. 268-272). Kyoto, Japan.

Kamei, S. (1999). Sharing dictionaries among MT users by common formats and social filtering framework. In *Machine translation summit VII* (pp. 180–181). Singapore.

Oepen, S., & Callmeier, U. (2000). Measure for measure: Parser cross-fertilization. Towards increased component comparability and exchange. In *Proceedings of the 6th International Workshop on Parsing Technologies* (pp. 183–194). Trento, Italy.

Oepen, S., Dyvik, H., Flickinger, D., Lønning, J. T., Meurer, P., & Rosén, V. (2005). Holistic regression testing for high-quality MT. Some methodological and technological reflections. In *Proceedings of the 10th Annual Conference of the European Association for Machine Translation.* Budapest, Hungary.

Oepen, S., Dyvik, H., Lønning, J. T., Velldal, E., Beermann, D., Carroll, J., Flickinger, D., Hellan, L., Johannessen, J. B., Meurer, P., Nordgård, T., & Rosén, V. (2004). Som å kapp-ete med trollet? Towards MRS-based Norwegian–English Machine Translation. In *Proceedings of the 10th International Conference on Theoretical and Methodological Issues in Machine Translation* (pp. 11–20). Baltimore, MD.

Oepen, S., Toutanova, K., Shieber, S., Manning, C., Flickinger, D., & Brants, T. (2002). The LinGO Redwoods treebank. Motivation and preliminary applications. In *Proceedings of the 19th International Conference on Computational Linguistics.* Taipei, Taiwan.

Pollard, C., & Sag, I. A. (1994). *Head-Driven Phrase Structure Grammar.* Chicago, IL and Stanford, CA: The University of Chicago Press and CSLI Publications.

Siegel, M. (2000). HPSG analysis of Japanese. In W. Wahlster (Ed.), (2000) 265–280.

Velldal, E., Oepen, S., & Flickinger, D. (2004). Paraphrasing treebanks for stochastic realization ranking. In *Proceedings of the 3rd Workshop on Treebanks and Linguistic Theories.* Tübingen, Germany.

Wahlster, W. (Ed.). (2000). *Verbmobil. Foundations of speech-to-speech translation.* Berlin, Germany: Springer.

[13] Thanks to an anonymous reviewer for this suggestion.