The Workshop Programme

8:00 – 8:15	Welcome
8:15 – 8:45	I. Alegria, M. Aranzabe, A. Ezeiza, N. Ezeiza, R. Urizar *Robustness and customization in an analyzer/lemmatizer for Basque*
8:45 – 9:15	M. O. Dzikovska, James F. Allen, Mary D. Swift *Finding the balance between generic and domain-specific knowledge: a parser customization strategy*
9 :15 – 9 :45	David M. de Matos, Nuno J. Marnede *Data-driven application configuration*
9 :45 – 10 :00	Break
10 :00 – 10:30	Svetlana Sheremetyva, Alexsei Pervuchin, Vladislav Trotsenko, Alexej Tkachev *Towards Saving on Software Customization*
10:30 – 11:00	Maria Nava *Resource integration and customization for automatic hypertext information retrieval in a corporate setting*
11:00 – 11:30	Patrice Lopez, Christine Fay-Vanier and Azim Roussanaly *Lexicalized Grammar Specialization for Restricted Applicative Languages*
11:30 – 11:45	Break
11:45 – 12:15	Hurskainen Arvi *A Versatile Knowledge Management Package*
12:15 – 12:45	Remi Zajac *Challenges in MT customization on closed and open text styles*
12:45 – 13:15	Anju Saxena and Lars Borin *Locating and reusing sundry NLP flotsam in an e-learning application*
13:15 – 13:30	Closing discussion

Workshop Organisers

Workshop Programme Committee

Table of Contents

Author Index

Robustness and customisation in an analyser/lemmatiser for Basque

Alegria I., Aranzabe M., Ezeiza A., Ezeiza N., Urizar R.

Informatika Fakultatea
649 P.K. E-20080 Donostia. Basque Country.
{i.alegria, jipecran}@si.ehu.es

Abstract

This paper describes the work carried out to improve the robustness of the morphological analyser/generator for Basque which can be adapted to several domains and variants of the language. This analyser is used as a lemmatiser in several IR applications such as an Intranet search engine.

We present an enhanced analyser that deals not only with standard words but also with linguistic variants (including dialectal variants and competence errors) and words, whose lemmas are not included in the lexicon, by relaxing the constraints of the standard analyser. In addition to this, a user's lexicon can be added to the system in order to customise the tool. This user's lexicon can be obtained by means of a semiautomatic process.

1. Introduction

The starting point of this research is a general morphological analyser/generator described in (Alegria et al., 1996), which reported 95% of coverage. This poor result was due (at least partially) to the recent standardisation and the widespread dialectal use of Basque.

Although in some systems lemmas corresponding to unknown words are included in the main lexicon in a previous step, this solution is not satisfactory if we want to build a flexible system. We decided that it was necessary to manage a user's lexicon, for linguistic variants and forms whose lemmas were not in the lexicon, if we wanted to develop a comprehensive or adapted analyser.

However, the enhancement of coverage leads, in some cases, to produce overgeneration, and, consequently, to increase ambiguity. Although this ambiguity is not real, it causes poor results (lower precision) in applications based on morphology or lemmatisation. Another important issue was the improvement of precision. We studied the results of the analyser and saw that most errors (50%-75%) were made when dealing with proper names. Therefore, we propose some solutions to avoid about 50% of the errors.

2. Architecture of the morphological analyser

Morfeus is a robust morphological analyser for Basque. It is a basic tool for current and future work on NLP of Basque. Some of the tools based on it are a tagger (Ezeiza et al., 1998), an Intranet search engine (Aizpurua et al., 2000) and an assistant for verse making (et al., 2001).

The analyser is based on the two-level formalism. The two-level model of computational morphology was proposed by Koskenniemi (Koskenniemi, 1983) and has had widespread acceptance due mostly to its general applicability, declarativeness of rules and clear separation of linguistic knowledge and program.

This tool is implemented using lexical transducers. A lexical transducer (Karttunen, 1994) is a finite-state automaton that maps inflected surface forms to lexical forms, and can be considered an evolution of the two-level morphology. The tool used for the implementation is the *fst* library of *Inxight*[1] (Karttunen and Bessley, 1992; Karttunen, 1993; Karttunen et al., 1996). A detailed description of the transducers can be found in (Alegria et al., 2001).

We have defined the architecture of the analyser using three main modules (Schiller (Schiller, 1996) and others propose only two levels):

1. The standard analyser that uses a general lexicon and a user's lexicons. This module is able to analyse/ generate standard language word-forms. In our applications for Basque we defined about 75,000 entries in the general lexicon, more than 130 patterns of morphotactics and two rule systems in cascade, the first one for long-distance dependencies among morphemes and the second for morphophonological changes. The three elements are compiled together in the standard transducer. To deal with the user's lexicon the general transducer described below is used.
2. The analysis and normalization of linguistic variants (dialectal uses and competence errors). Due to non-standard or dialectal uses of the language and competence errors, the standard morphology is not enough to offer good results when analysing real text corpora. This problem becomes critical in languages like Basque in which standardisation is in process and dialectal forms are still of widespread use. For this process the standard transducer is extended with new lexical entries and phonological rules producing the *enhanced transducer.*
3. The guesser or analyser of words without lemmas in the lexicons. In this case the standard transducer is simplified removing the lexical entries in open categories (nouns, adjectives, verbs, …), which constitute the vast majority of the entries, and is substituted by a general automata to describe any combination of characters. So, the *general transducer* is produced combining this general set of lemmas with affixes related to open categories and general rules.

[1] Inxight Software, Inc., a Xerox Enterprise Company (www.inxight.com)

The analyser of non-standard words (steps 2 and 3) may sometimes produce overgeneration, and it is important to reduce this ambiguity as soon as possible.

3. Customizing the analyser

In order to deal with unknown words, a general transducer has been designed to relax the need of lemmas in the lexicon. This transducer was initially (Alegria et al., 1997) based on an idea used in a speech synthesis system (Black et al., 1991) but it has been now simplified. Daciuk (Daciuk, 2000) proposes a similar way when he describes the *guessing automaton*, but the construction of our automaton is simpler.

The new transducer is the standard one modified in this way: the lexicon is reduced to affixes corresponding to open categories and generic lemmas for each open class, while standard rules remain. There are seven open classes and the most important ones are: common nouns, personal names, place nouns, adjectives and lexical verbs. Grammatical categories and semantic ones (personal names or place names) are separated because they have different declension.

So, the standard rule-system is composed of a mini-lexicon where the generic lemmas are obtained as a result of combining alphabetical characters and can be expressed in the lexicon as a cyclic sublexicon with the set of letters (some constraints are used with capital/non-capital letters according to the part of speech). In fig. 1 the graph corresponding to the mini-lexicon is shown.

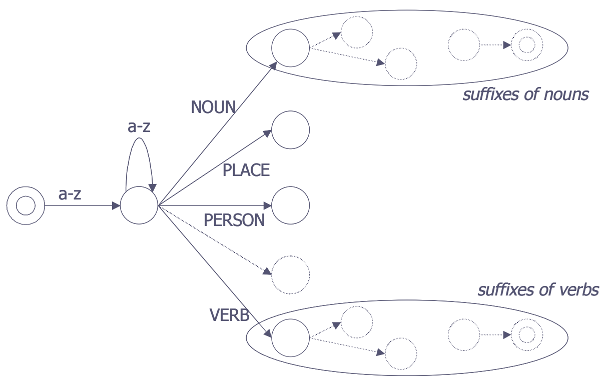


Figure 1. Simplified graph of the mini-lexicon

This transducer is used in two steps of the analysis:

1. in the standard analysis, in order to analyse declension and derivation of lemmas in the user's lexicon.
2. in the analysis without lexicon (called *guesser* in taggers).

The user's lexicon is composed of a list of lemmas along with their parts of speech defined by the users. The general transducer suggests possible interpretations of the word, and these lemmas are searched in the user's lexicon. When any lemma and class given by the general transducer matches the information on the user's lexicon, the analyser selects the corresponding interpretation and gives it as a result.

So, the user's lexicon is an editable resource which can be inferred from corpora or be managed on-line by the user. The use of this lexicon combined with the general transducer allows to customise the applications and it has been included successfully in three tools:

1. A spelling corrector for Basque (Aldezabal et al., 1999) in which for each lemma included in the user's lexicon any inflected form or derivative is accepted.
2. An Intranet search engine (Aizpurua et al., 2000) in which lemmatisation plays an important role and which can be customised when adapted to a special domain. In this case a semiautomatic process is carried out. First, the whole analyser (in the three steps above mentioned) is used to analyse a big corpus and the possible lemmas obtained by the guesser. After being sorted by frequency, they are presented to the user in order to include them in the user's lexicon[2]. The site www.zientzia.net, devoted to scientific documents, was built in this way.
3. A general part-of-speech tagger including customisation similar to the search engine.

4. Increasing coverage

The analyser was designed with the main objective of being robust, that is, capable of treating both standard and non-standard forms in real texts. For this reason, the morphological analyser has been extended in two ways:

1. The treatment of linguistic variants (dialectal variants and competence errors) (Aduriz *et al.*, 1994)
2. A two-level mechanism for lemmatisation without lexicon to deal with unknown words, which has been explained above

Important features of this design are homogeneity, modularity and reusability because the different steps are based on lexical transducers, far from *ad hoc* solutions, and these elements can be used in different tools. This could be considered a variant of constraint relaxation techniques used in syntax (Stede, 1992), where the first constraint demands standard language, the second one combines standard and linguistic variants, and the third step allows free lemmas in open categories. Only if the previous steps fail, the results of the next step are included in the output. Oflazer also uses relaxation techniques in morphology (Oflazer, 1996).

With this design the obtained coverage is 100% and precision over 99.5%. The ambiguity measures of the morphological analyser, taken from a balanced corpus of about 27,000 tokens and from a news collection of about 9,000, are shown in table 1. These measures have been obtained using all the morphological features.

Ambiguity Rate	Interpretations per ambiguous token	Interpretations per token
66.95%	4.38	3.26

Table 1: Ambiguity measures[3]

However, sometimes overgeneration is produced in order to improve robustness. Overgeneration increases ambiguity but often this ambiguity is not real and causes poor results (low precision) in applications based on morphology such as spelling correction, morphological generation or tagging.

[2] At this moment it is a not friendly off-line process

[3] Ambiguity Rate: #ambiguous_token / #token; Interpretations per token: #analyses / #token; Interpretations per ambiguous token: #analyses_ambiguous_token / # ambiguous_token

	Distribution	Ambiguity Rate	Interpretations per ambiguous token	Interpretations per token	Precision
standard	77.90%	80.73%	3.81	3.27	99.73%
variant	1.75%	80.53%	4.23	3.60	92.31%
unknown	2.65%	99.79%	18.05	18.01	98.12%
average	100.00%	66.95%	4.38	3.26	99.61%

Table 2: Ambiguity measures in the output of the analyser

	tokens	standard	variant	unknown	other[4]
corpus1	116,720	76.66%	1.02%	3.28%	19.04%
corpus2	1,288,257	78.44%	0.94%	3.80%	16.82%
corpus3	587,515	74.98%	2.03%	2.92%	20.07%
corpus4	33,232	77.32%	1.42%	4.92%	16.34%
corpus5	148,333	77.91%	1.01%	6.23%	14.85%
corpus6	29,939	60.54%	11.50%	7.90%	20.06%

Table 3: Distribution of tokens in different types of corpora

5. Decreasing ambiguity

The ambiguity for linguistic variants and unknown words is higher and the precision measures are poorer, but they form a small group of the input words (5%-10%) and the influence on average results is not significant.

The morphological analyser may sometimes overgenerate analyses of linguistic variants and unknown lemmas (table 2). Even if most words in texts are analysed in the first phase (see table 3), the small proportion of non-standard words constitutes a great amount of the superfluous interpretations. Yet, the rate of non-standard words varies depending on the type of corpus.

For instance, corpus3 is a balanced corpus with a high rate of standard Basque texts. On the contrary, corpus6 is a subset of texts from corpus3 written mainly in two dialects. Obviously, this corpus has a higher rate of non-standard uses. Corpus1 is a compilation of texts from the Web, and, generally, there is a trend to write these documents following standard rules of the language. Finally, corpus2, corpus4 and corpus5 are texts from the Basque newspaper *Euskaldunon Egunkaria*, and, even if the language variant used on them is standard, there is a relatively high amount of unknown words.

The treatment of non-standard words has been added to the previously developed analyser for two main reasons:

1. The average number of interpretations in non-standard words is significantly higher than in standard words (see table 2).
2. There could be multiple lemmas for the same or similar morphological analysis. This is a problem when we want to build a lemmatiser. For example, if *bitaminiko* (vitaminic) is not in the lexicon the results of the analysis of *bitaminikoaren* (from the vitaminic) as adjective can be multiple: *bitamini+ko+aren*, *bitaminiko+aren* and *bitaminikoaren*, but the only right analysis is the second one.

We think that it is important to reduce the ambiguity at this stage, so that the input of subsequent processes is more precise. But, we do not use information about surrounding words because a tagger will be used later. The process is limited to the word we want to treat, and we only need to know, in some cases, if the previous token was a full stop.

This module consists of different methods for linguistic variants and unknown words, because overgeneration is produced by different facts in each case, as will be described below.

5.1. Disambiguation of linguistic variants

In the case of linguistic variants a heuristic tries to select the lemma that is "nearest" to the standard one according to the number of non-standard morphemes and rules applied. It chooses the interpretation that has less non-standard uses for each POS tag.

For example, analysing the word-form *kaletikan* (dialectal form) two possible analyses are obtained: *kale+tik* (from the street) and *kala+tik* (from the cove). Both analyses have a non-standard morpheme (*-tikan*) but the first analysis is more probable because it applies no other transformation rule and to obtain the second one it has been necessary to apply another rule at the end of the lemma to transform *kale* into *kala*.

Thus, we must decide which of the analyses need to be selected or discarded based on the amount of transformation rules applied to obtain each analysis, but the enhanced transducer does not detail this information. The output of the enhanced transducer displays the normalised lemma/morphemes along with their corresponding morphological features. In the case of non-standard morphemes linked in the lexical database to their normalised form, the analysis details both normalised and variant morphemes.

Thus, the procedure uses these results to select the most probable lemmas for each POS tag. The results of applying this procedure are shown in table 4. The error rate of the procedure is 1.7%, so the error rate added to the whole process is 0.03%. It does not mean a significant drop in overall ambiguity, but it discards 40% of superfluous analyses.

[4] This group represents punctuation marks and other symbols.

	Ambiguity Rate	Interpretations per ambiguous token	Interpretations per token	Precision
before	80.53%	4.23	3.60	92.31%
after	75.35%	2.98	2.49	90.42%

Table 4: Ambiguity measures on linguistic variants before and after the procedure

	Ambiguity Rate	Interpretations per ambiguous token	Interpretations per token	Precision
initial	99.79%	18.06	18.01	98.12%
typographical	99.58%	8.18	8.15	96.46%
derivational	99.58%	7.94	7.91	96.46%
proper names	85.21%	6.93	6.05	95.94%
statistical 3+2+1	83.33%	3.99	3.49	91.98%

Table 5. Ambiguity measures on unknown words using all the procedures

	Distribution	Ambiguity Rate	Interpretations per ambiguous token	Interpretations per token	Precision
standard	77.90%	80.73%	3.81	3.27	99.73%
variant	1.75%	75.35%	2.98	2.49	90.42%
unknown	2.65%	85.21%	4.06	3.61	93.02%
average	100.00%	66.46%	3.80	2.86	99.43%

Table 6. Ambiguity measures in the output of the improved analyser

However, this heuristic treats every rule equally, but not all of them have the same probability of being applied. We think that it could be interesting to use a probabilistic transducer (Mohri, 1997) to improve the precision measures of both the analyser and the disambiguation procedure of variants.

5.2. Disambiguation of unknown words

We have tested several procedures to detect and treat unknown words using different criteria:

1. Typographical disambiguation. Some analyses are discarded based on capital letters.
2. Disambiguation of derivational words to counterbalance overgeneration of the analyser. The goal of this procedure is to discard one of several interpretations when the morphological analyser assigns analyses as derivational and non-derivational word.
3. Identification and disambiguation of proper names not included in the lexicon. Some analyses can be disambiguated when identical lemmas are found in the same document.
4. Disambiguation based on both statistical and linguistic information. These statistics relates final trigrams of characters and POS tags. is used. The main features of the heuristic are: a) for each POS tag, leave at least one interpretation; b) assign a weight to each lemma according to the final trigram and the POS tag; c) select the lemma according to its length and weight –best combination of high weight and short lemma.

These procedures were designed to be applied consecutively. To decide the order in which they must be applied, we tried different combinations.

Finally, table 5 shows the best result of applying all the procedures in cascade.

This treatment has been designed to discard some of the interpretations of unknown words. Even if unknown words are only 2%-3% of the words, they constitute 15%-20% of the analyses. After applying the procedures, they only represent 3%-4.5% of the analyses, depending on the combination of procedures we use, and the average number of interpretations decreases from 18-19 down to 3,5-4,5. The overall results of treating the reference text are shown in table 8. This has been measured using the second level tagset both for disambiguation of linguistic variants and for statistical disambiguation of unknown words, thus leaving (at least) one lemma per class and subclass.

Precision decreases in average around 0.2%, even if the results for unknown words fall from 98% to 93%. Finally, we want to point out that each combination of the procedures may be used for different applications.

6. Improving precision

The main reason for these errors is the incremental architecture of the analyser. The first step in the process, the standard analyser, causes wrong interpretations, primarily when very short or very rare lemmas are involved in the analysis. However, the process stops when the analyser finds (at least) one interpretation of the word.

A clear example of these misinterpretations is *Barak*. This name, when it appears in its base form, is interpreted as *bara*, a common noun of very low frequency. When it appears inflected, i.e. *Barak-ek* (*Barak* in ergative case), the standard analyser assigns no interpretation and the analyser without lexicon interprets it correctly as a proper noun.

	Distribution	Ambiguity Rate	Interpretations per ambiguous token	Interpretations per token	Precision
standard	77.88%	81.02%	3.86	3.32	99.88%
variant	1.66%	81.36%	4.40	3.76	96.51%
unknown	2.76%	99.90%	18.20	18.18	98.34%
average	100.00%	67.21%	4.46	3.32	99.80%

Table 7: Ambiguity measures in the output of the analyser

Most of the errors are avoidable enriching the user's lexicon, but it is necessary to improve the results when this is not done.

So we must avoid rare and improbable analyses when a word has an initial capital letter. In order to avoid odd analyses we have marked short or conflicting lemmas with low probability as rare in the lexical database. Using this information, when all the possible interpretations for a word are marked as rare, the process follows using the next module. If at the next step the analyser does not find a non-rare analysis for the word, the word will be tagged just as the standard analyser did.

In the case of low frequency lemmas, words written with initial capital letter are also analysed by the guesser and only proper name interpretations are added to the ones suggested by the standard analyser.

In order to increase the precision in the analyser of linguistic variants, we limit the number of rules applied to obtain the interpretations. If all the interpretations have been obtained applying a higher value of rules than the threshold, the word will be treated using the guesser, thus, discarding the other interpretations.

We have implemented these proposals and the results are encouraging (see table 7). As a result, we have avoided 50% of the errors relaxing the constraints of the morphological analyser.

7. Conclusions

We have presented the work carried out to improve the robustness of a morphological analyser and to adapt it to new domains. We have made a proposal for the architecture of a morphological analyser combining different transducers to increase flexibility, coverage and precision. The design we propose is quite new as far as we know and we think that our design could be interesting for the robust treatment of other languages.

On the other hand, we have also defined some local disambiguation procedures, which don't take into account the context of the word, so as to discard many of the overgenerated analysis for non-standard words. The results of the research are very encouraging.

8. Acknowledgements

This work has been partial supported by the Education Department of the Government of the Basque Country (UE1999-2) and the Spanish Science and Technology Ministry (*Hermes* research project; 8/DG00141.226-14247/200).

We would like to thank to Xerox for letting us using their tools, and also to Ken Beesley and Lauri Karttunen for their help.

9. References

Aduriz I., I.Alegria, J. M. Arriola, X. Artola, A. Díaz de Ilarraza, N. Ezeiza, K. Gojenola and M. Maritxalar 1995. Different issues in the design of a lemmatizer/tagger for Basque. *From text to tag SIGDAT, EACL Workshop*.

Aizpurua I., I. Alegria, N. Ezeiza, 2000. GaIn: un buscador Internet/Intranet avanzado para textos en euskera. *Actas del XVI Congreso de la SEPLN.*

Aldezabal I., I. Alegria, O. Ansa, J. Arriola, N. Ezeiza, 1999. Designing spelling correctors for inflected languages using lexical transducers. *Proceedings of EACL'99*, 265-266. Bergen, Norway. 8-12.

Alegria I., M. Aranzabe, A. Ezeiza, N. Ezeiza, R. Urizar, 2001. Using Finite State Technology in Natural Language Processing of Basque. *6th Conf. on Implementation and Applications of Automata. CIAA'2001.*

Alegria I., X. Artola, K. Sarasola, M. Urkia, 1996. Automatic morphological analysis of Basque. *Literary & Linguistic Computing Vol. 11, No. 4*: 193-203. Oxford University Press.

Antworth E.L. 1990. *PC-KIMMO: A two-level processor for morphological analysis*. Occasional Publications in Academic Computing, No. 16, Dallas, Texas.

Arrieta B., X. Arregi, I. Alegria, 2001. An Assistant Tool For Verse-Making In Basque Based On Two-Level Morphology. *Literary and Linguistic Computing, Vol. 16, No. 1, 2001.* Oxford University press.

Black A., J. van de Plassche, B. Williams, 1991. Analysis of Unknown Words through Morphological Descomposition. *Proceedings of 5th Conference of the EACL*, pp. 101-106.

Ezeiza N., I. Aduriz, I. Alegria, J. M Arriola, R. Urizar, 1998. Combining Stochastic and Rule-Based Methods for Disambiguation in Agglutinative Languages. *Proceedings of COLING-ACL'98*.

Karttunen L., 1993. Finite-State Lexicon Compiler. Xerox ISTL-NLTT-1993-04-02.

Karttunen L., 1994. Constructing Lexical Transducers, *Proceedings of COLING´94*, pp. 406-411.

Karttunen L., 2000. Applications of Finite-State Transducers in Natural Language Processing. *Proceedings of CIAA-2000.* Lecture Notes in Computer Science. Springer Verlag.

Karttunen L. and K. R. Beesley, 1992. *Two-Level Rule Compiler.* Technical Report Xerox ISTL-NLTT-1992-2.

Karttunen L., J.P. Chanod, G. Grenfenstette, A. Schiller, 1996. Regular Expressions for Language Engineering. *Natural Language Engineering, 2(4)*: 305:328.

Koskenniemi, K., 1983. *Two-level Morphology: A general Computational Model for Word-Form Recognition and Production*, University of Helsinki, Department of General Linguistics. Publications 11.

Mohri, M., 1997. Finite-state transducers in language and speech processing. *Computational Linguistics 23(2)*:269-322.

Oflazer K, C. Guzey, 1994. Spelling Correction in Agglutinative Languages. *Proceedings of ANLP-94*.

Oflazer K. 1996. Error-tolerant Finite State Recognition with Applications to Morphological Analysis and Spelling Correction. *Computational Linguistics 22(1)*: 73-89.

Schiller A., 1996. Multilingual finite-state noun phrase extraction. In Wo*rkshop on Extended finite state models of language*, ECAI'96, Budapest, Hungary.

Sproat R., 1992. *Morphology and Computation*. The MIT Press.

Stede M., 1992. The Search of Robustness in Natural Language Understanding*. Artificial Intelligence Review 6*, 383-414.

Finding the balance between generic and domain-specific knowledge: a parser customization strategy

Myroslava O. Dzikovska, James F. Allen, Mary D. Swift

Computer Science Department
University of Rochester
Rochester, NY, USA, 14627
{myros, james, swift}@cs.rochester.edu

Abstract

Adapting spoken dialogue systems across domains presents a challenge of finding the balance between wide-coverage parsers which can be easily ported but are slow and inaccurate, and domain-specific parsers which are fast and accurate but lack portability. We propose a method for customizing a wide-coverage, domain-independent parser to specific domains. We maintain a domain-independent ontology and define a set of mappings from it into a domain-specific knowledge representation. With this method, we customize the semantic representations output by the parser for reasoning, and we specialize the lexicon for the domain, resulting in substantial improvement in parsing speed and accuracy.

1. Introduction

Developers of spoken dialogue systems for multiple domains are faced with the challenge of finding the optimal balance between domain-independent and domain-specific parsers. There are wide-coverage parsers (e.g. XTag (Doran et al., 1994), LINGO (Copestake and Flickinger, 2000)) that are domain-independent and therefore easy to port to new domains, but they are often not efficient or accurate enough. The typical approach is to hand-craft parsers specifically for each domain (see for example (Goddeau et al., 1994)), but the performance gains in accuracy and efficiency are offset by their lack of portability, requiring additional effort to adapt them to new domains. We propose an alternative approach to address this challenge with a method for customizing a wide-coverage, domain-independent parser developed for spoken dialogue applications to specific domains. We maintain two ontologies: domain-independent for the parser, and domain-specific for the knowledge representation, and define a set of mappings between domain-specific knowledge sources and the semantic representations output by the parser. This method improves upon the generic parser output by specifically tailoring the semantic representations output by the parser for use by the reasoning components in the system. We also use the mappings to specialize the lexicon to the domain, resulting in substantial improvement in parsing speed and accuracy.

The customization method described here was developed in the process of adapting the TRIPS dialogue system (Allen et al., 2001) to several different domains, such as a transportation routing system (Allen et al., 1996) and a medication scheduling adviser. We assume a generic dialogue system architecture (Allen et al., 2000) that includes a speech module, a parser, an interpretation manager (responsible for contextual processing and dialogue management), and a back-end application responsible for the general problem-solving behavior of the system.

Adapting the spoken dialogue system across domains results in tension between the representation of generic vs. specific information in the ontology. To facilitate development when porting the parser to new domains, we want to retain the syntactic and semantic information that is consistent across domains. However, each domain comes with its own semantic information relevant to the application. For example, the representation of physical objects for the transportation domain requires specifying whether an object is suitable cargo for a transportation action, such as different types of food or supplies. In this respect, the distinctions between, say, oranges and potatoes are irrelevant, since they are equally good as cargo. These distinctions become highly relevant in the medical domain, where food-medicine interactions are important. Ideally, we want to customize the ontology to the domain for the most efficient reasoning. This becomes ever more important when using specialized reasoners with pre-defined input representations, for example, a database query system that must have specific template slots filled. Thus our goal is to preserve the language information that is similar across domains, while addressing specialization issues unique to each domain as much as possible, and keeping the development time spent on custom domain adaptation to a minimum.

To reuse the syntactic information, the AUTOSEM system(Rosé, 2000) uses a syntactic lexicon COMLEX(Macleod et al., 1994) as a source of syntactic information, and manually links subcategorization frames in the lexicon to the domain-specific knowledge representation. The linking is performed directly from syntactic arguments (e.g. subject, object ...) to the slots in a frame-like domain representation output by the parser and used by the reasoners. Rosé shows that her approach speeds up the development process for developing tutoring systems in multiple domains.

Our approach introduces an intermediate layer of abstraction, a generic ontology for the parser (the **LF Ontology**) that is linked to the lexicon and preserved across domains. The parser uses this ontology to supply meaning representations of the input speech to the interpretation manager, which handles contextual processing and dialogue management and interfaces with the back-end application. The domain-specific ontology used for reasoning (the **KR ontology**) is localized in the back-end application. We then customize the communication between the parser/interpretation manager and the back-end application via a set of mappings between the LF and KR ontologies. At the same time, the domain-independent ontology preserves semantic information consistent across domains that can be used by the Interpretation Manager for reasoning or reference resolution.

This separation allows us to write mappings in semantic terms without addressing the details of the grammar and subcategorization frames, using a higher level of abstraction. The developers writing the mappings does not need to understand details pertaining to syntax such as those included in COMLEX subcategorization frames, and can instead use descriptive labels assigned to generic semantic arguments (e.g. AGENT, THEME etc.). They can also take advantage of the hierarchical structure in the domain-independent ontology and write mappings that cover large classes of words. Finally, the mappings are used to convert the generic representation into the particular form utilized by the back-end application, either a frame-like structure or a predicate logic representation.

2. The Generic Lexicon

The LF ontology is close in structure to linguistic form, so it can be easily mapped to natural language and used in multiple domains. It classifies entities (i.e., objects, events or properties) primarily in terms of argument structure. Every LF type declares a set of linguistically motivated thematic arguments, a structure inspired by FRAMENET (Johnson and Fillmore, 2000), but which covers a number of areas where FRAMENET is incomplete, such as planning. We use the LF ontology in conjunction with a generic grammar covering a wide range of syntactic structures and requiring minimal changes between domains. For example, adapting the parser from the transportation to the medical domain required adding LF types for medical terms (our generic hierarchy was incomplete in this area) and corresponding vocabulary entries, but we did not need to change the grammar or existing lexical entries, and we continue to use the same lexicon in both domains.

The LF types in the LF ontology are organized in a single-inheritance hierarchy. Obviously, some sort of multiple inheritance is required, because, for example, a person is a living being, but also a solid physical object (as opposed to a formless substance such as water). We implement multiple inheritance via semantic feature vectors associated with each LF type. The features correspond to basic meaning components and are based on the EuroWordNet (Vossen, 1997) feature system with some additional features we have found useful across domains. While the same distinctions can be represented in a multiple inheritance hierarchy, a feature-based representation makes it easy to implement an efficient type-matching algorithm based on (Miller and Schubert, 1988). More importantly, using feature vectors allows us to easily change semantic information associated with a lexical entry, a property utilized during the customization process described below.

Word senses are treated as leaves of the semantic hierarchy. For every word sense in the lexicon, we specify the following information:

- Syntactic features such as agreement, morphology, etc.;
- LF type;
- The subcategorization frame and syntax-semantics mappings.

To illustrate, consider the verb *load* in the sense *to fill the container*. The LF type definition for *LF_LOAD* is shown in Figure 1. It specifies generic type restrictions on the arguments which are then propagated in the lexical entries. Intuitively, it defines a loading event in which an intentional being (AGENT) loads a movable object (THEME) into another physical object that can serve as a container (TO-LOC). The lexicon entry for *load* is linked to LF_Load and contains two possible mappings from the syntax to the LF: one in which the THEME is realized as direct object, corresponding to *load the oranges into the truck*, and another in which the THEME is realized as prepositional complement, corresponding to *load the truck with oranges*. The restrictions from the THEME argument are propagated into the lexicon, and the parser makes use of them as follows: only objects marked as *(mobility movable)* are accepted as a direct object or prepositional *with* complement of *load*.

```
(define-type LF_LOAD
 :sem (situation (aspect dynamic)
                 (cause agentive))
 :arguments
  (AGENT (phys-obj (intentional +)))
  (THEME (phys-obj (mobility movable)))
  (TO-LOC (phys-obj (container +)))
)
```

Figure 1: The LF type definition for LF_LOAD. In the lexicon, feature vectors from LF arguments are used to generate selectional restrictions based on mappings between subcategorization frames and LF arguments

The parser produces a flattened and unscoped logical form using reified events (Davidson, 1967). A simplified representation showing the semantic content of *Load the oranges into the truck* is shown in Figure 2. [1] For every entity, the full type is written as LF-parent*LF-form, where the LF-parent is the type defined in the LF ontology, and the LF-form is the canonical form associated with the word, for example, LF_VEHICLE*truck.

3. The KR customization

To produce domain-specific KR representations from the generic LF representations, we developed a method to customize parser output. The current system supports two knowledge representation formalisms often used by reasoners: a frame-like formalism where types have named

[1]For simplicity, we ignore speech act information in our representations

```
(TYPE e LF_LOAD*load)
(AGENT e *YOU*) (THEME e v1) (TO-LOC e v2)
(TYPE v1 LF_FOOD*orange)
(TYPE v2 LF_VEHICLE*truck)
```

Figure 2: The LF representation of the sentence *load the oranges into the truck.*

```
(a)
(LF-to-frame-transform load-transform
       :pattern (LF_LOAD LOAD)
       :arguments (AGENT :ACTOR)
                  (THEME :CARGO)
                  (TO-LOC :VEHICLE))

(b) (define-class LOAD
     :isa ACTION
     :slots
       (:ACTOR AGENT)
       (:CARGO COMMODITY)
       (:VEHICLE (OR TRUCK HELICOPTER)))

(c) [LOAD
      :ACTOR [PERSON +YOU+]
      :CARGO [ORANGE V1]
      :VEHICLE [TRUCK V2]]
```

Figure 3: LF-to-frame-transform. (a) The transform for LF_LOAD type; (b) the definition of LOAD class that the transform maps into; (c) The KR frame that results from applying this transform to the load event representation in Figure 2.

```
(a)
(LF-to-pred-transform load-transform
 :pattern (LF_LOAD
           (LOAD *AGENT *THEME *TO-LOC)
          ))

(b) (define-predicate LOAD
       :isa ACTION
       :arguments
       (1:AGENT 2:COMMODITY
        3:(OR TRUCK HELICOPTER)))

(c) (AND (LOAD +YOU+ V1 V2)
         (COMMODITY V1) (TRUCK V2))
```

Figure 4: LF-to-predicate-transform. (a) The transform for LF_LOAD type; (b) the definition of LOAD predicate that the transform maps into; (c) The KR formula that results from applying this transform to the load event representation in Figure 2.

slots, and a representation that has predicates with positional arguments. The KR ontology must have subtype support, and for the lexicon specialization process described in the next section, type restrictions on the arguments of frames/predicates, though it need not be so in the most general case.

We use two basic transform types to map generic representations produced by the parser into the KR representation: LF-to-frame-transforms, shown in Figure 3, and LF-to-predicate-transforms, shown in Figure 4.

The LF-to-frame transforms convert LF types into KR frame structures by specifying the KR frame that the LF type maps into, and how the arguments are transformed into the frame slots. These transforms can be simple and name the slot into which the value is placed, or more elaborate and specify the operator expression that is applied to the value. The LF-to-predicate transforms are used to convert the frame-like LF structures into predicates with positional arguments. They specify a KR predicate that an LF type maps into and the expression that is formed.

After the parser produces the logical form, the Interpretation Manager decides which transform to apply to a given LF with the following algorithm:

- Find all transforms that are consistent with the LF or its ancestors;
- Select the most specific transform that applies, that is, the transform that uses only the roles realized in this particular LF representation, that has all obligatory mappings filled, and for which the types of the LF arguments are consistent with the type restrictions on the class arguments;
- If there are several candidates, choose the transform that uses the most specific LF, and, if there are several for the same LF, the transform that maps into the most specific KR class;
- Apply the transform to the LF type and all its arguments to produce the new representation.

For example, the parser produces the logical form in Figure 2 for *load the oranges into the truck*. The Interpretation Manager determines that the most specific transform consistent with the arguments is the `load-transform`. If the back-end reasoners use the frame representation, then we use an LF-to-frame transform and obtain the frame shown in Figure 3. Alternatively, a system using predicates with positional arguments as its representation uses an LF-to-predicate transform and obtains the (simplified) representation shown in Figure 4.

Our examples show the simplest versions of the transforms for exposition purposes. The actual implementation permits a variety of constructs that we cannot illustrate due to space limitations, including the application of operators to arguments, default transforms that apply to LF arguments if no mapping is specified in LF-to-frame transform, and the use of the lexical forms in transforms when the KR uses similar terms. For example, from the point of view of the language ontology, medication names have similar distributions across syntactic contexts, and therefore are represented as leaves under the LF_DRUG type, e.g. LF_DRUG*prozac, LF_DRUG*aspirin. The KR ontology makes pragmatic distinctions between them (e.g. prescription vs. over-the-counter medicines), but uses the names as leaf types in the hierarchy. We can write a single template mapping for all LF_DRUG children that does the conversion based on the lexical form specified in the entry. This allows us to convert the generic representation produced by the parser to a representation that uses the concepts and formalism suited to the domain.

4. Specializing the lexicon

The KR customization described above can be implemented as a two-stage process with a generic grammar and lexicon and a post-processing stage. We also use the mappings to speed up the parsing and improve semantic disambiguation accuracy by integrating the domain-specific semantic information into the lexicon and grammar.

We pre-process every entry in the lexicon by determining all possible transforms that apply to its LF. For each transform, we create a new sense definition identical to the old generic definition plus a new feature *KR-TYPE* in the semantic vector. The value of *KR-type* is the KR ontology class that results from applying this transform to the entry. Thus, we obtain a (possibly larger) set of entries which specify the KR class to which they belong. We then propagate type information into the syntactic arguments, making tighter selectional restrictions in the lexicon. We also increase the preference values for the senses for which mappings were found. This allows us to control the parser search space better and obtain greater parsing speed and accuracy.

Consider the following example. Given the definition of the verb *load* and LF_Load in Figure 1, and the definitions in Figure 3, the algorithm proceeds as follows:

- As part of generating the lexical entry for the verb *load*, the system fetches the definition of *LF_load* and the semantic vectors for it and its arguments;

- Next, the system determines the applicable LF-to-frame-transform, `load-transform`;

- Based on the transform, *KR-type load* is added to the feature vector of *load*;

- Since the mapping specifies that the LF argument THEME maps to KR slot *CARGO*, and the class definition contains the restriction that cargo should be of class *COMMODITY*, *KR-type commodity* is added to the feature vector of the THEME argument. Similar transforms are applied to the rest of the arguments.

As a result, in the lexicon we obtain a new definition of *load* with 2 entries corresponding to the same two usages described in section 2., but with stricter selectional restrictions. Now suitable objects or prepositional complements of *load* must be not only movable, but also identified as belonging to class COMMODITY in our domain. Since similar transforms were applied to nouns, oranges, people and other cargoes will have a *KR-type* value that is a subtype of COMMODITY inserted in their semantic feature vectors.

As a result of the specialization process, the number of distinct lexical entries will increase because there is not a one-to-one correspondence between the LF and KR ontologies, and several transforms may apply to the same LF depending on the syntactic arguments that are filled. A new entry is created for every possible transform, but during parsing the selectional restrictions propagated into the entries will effectively select the correct definitions. The Interpretation Manager thus knows the correct KR types assigned to all entities in the logical form output by the parser and the corresponding transforms, and only needs to apply them to convert the LF expression into the form used by the back-end reasoners.

	Generic	Transportation	Medical
# of senses	1947	2028	1954
# of KR classes	-	228	182
# of mappings	-	113	95

Table 1: Some lexicon statistics in our system

	Transportation	Medical
# of sentences	200	34
Time specialized (sec)	4.35 (870)	2.5 (84)
Time generic (sec)	9.7(1944)	4.3 (146)
Errors specialized	24%(47)	24% (8)
Errors generic	32% (65)	47% (16)

Table 2: Average time per lattice and the sentence error rate for the grammar specialized by our method compared to our generic grammar. Numbers in parentheses denote the total time and error count for the test set.

Lexicon specialization considerably speeds up the parsing process. We conducted an evaluation comparing parsing speed and accuracy on two sets of 50-best speech lattices produced by our speech recognizer: 34 sentences in the medical domain and 200 sentences in the transportation domain. Table 1 describes the ontologies used in these domains. The results presented in Table 2 show that lexicon specialization considerably increases parsing speed and improves disambiguation accuracy. The times represent the average parsing time per lattice, and the errors are the number of cases in which the parser selected the incorrect word sequence out of the alternatives in the lattice [2].

At the same time, the amount of work involved in domain customization is relatively small. The generic lexicon and grammar stay essentially the same across domains, and a KR ontology must be defined for the use of back-end reasoners anyway. We need to write the transforms to connect the LF and KR ontologies, but as their number is small compared to the total number of sense entries in the lexicon and the number of words needed in every domain,

[2]We considered correct the choices where a different pronoun, an article or a tense form were substituted. For example *can I tell my doctor* and *could I tell my doctor* were considered equivalent for purposes of this evaluation. However, we counted as errors the equally grammatical substitutions that selected a different word sense, e.g. *drive the people* vs. *get the people*

this represents an improvement over hand-crafting custom lexicons for every domain.

5. Conclusion

The customization method presented here allows the use of a lexicon and grammar with generic syntactic and semantic representations for improved domain coverage and portability, while facilitating the specialization of the lexicon and the representation produced by the parser to the needs of a particular domain. With this method we can produce specialized grammars for more effi cient and accurate parsing, and allow the parser, in cooperation with Interpretation Manager, to produce semantic representations optimally suited for specifi c reasoners within the domain.

6. Acknowledgments

We would like to thank Carolyn Rosé for her feedback on this article.

This material is based upon work supported by the Offi ce of Naval Research under grant number N00014-01-1-1015 and the Defense Advanced Research Projects Agency under grant number F30602-98-2-0133. Any opinions, fi ndings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of ONR or DARPA.

7. References

J. F. Allen, B. W. Miller, E. K. Ringger, and T. Sikorski. 1996. A robust system for natural spoken dialogue. In *Proceedings of the 1996 Annual Meeting of the Association for Computational Linguistics (ACL'96)*.

Allen, Byron, Dzikovska, Ferguson, Galescu, and Stent. 2000. An architecture for a generic dialogue shell. *NLENG: Natural Language Engineering, Cambridge University Press*, 6.

J. Allen, D. Byron, M. Dzikovska, G. Ferguson, L. Galescu, and A. Stent. 2001. Towards conversational human-computer interaction. *AI Magazine*.

Ann Copestake and Dan Flickinger. 2000. An open source grammar development environment and broad-coverage English grammar using HPSG. In *Proceedings of the 2nd International Conference on Language Resources and Evaluation*, Athens, Greece.

Donald Davidson. 1967. The logical form of action sentences. In Nicholas Rescher, editor, *The Logic of Decision and Action*, pages 81–95. University of Pittsburgh Press, Pittsburgh. Republished in Donald Davidson, *Essays on Actions and Events*, Oxford University Press, Oxford, 1980.

Christy Doran, Dania Egedi, Beth Ann Hockey, B. Srinivas, and Martin Zaidel. 1994. XTAG system – a wide coverage grammar for English. In *Proceedings of the 15th. International Conference on Computational Linguistics (COLING 94)*, volume II, pages 922–928, Kyoto, Japan.

D. Goddeau, E. Brill, J. Glass, C. Pao, M. Phillips, J. Polifroni, S. Seneff, and V. Zue. 1994. Galaxy: A human-language interface to on-line travel information. In *Proc. ICSLP '94*, pages 707–710, Yokohama, Japan, September. URL http://www.sls.lcs.mit.edu/ps/SLSps/icslp94/galaxy.ps.

Christopher Johnson and Charles J Fillmore. 2000. The framenet tagset for frame-semantic and syntactic coding of predicate-argument structure. In *Proceedings ANLP-NAACL 2000*, Seattle, WA.

C. Macleod, R. Grishman, and A. Meyers. 1994. Creating a common syntactic dictionary of English. In *SNLR: International Workshop on Sharable Natural Language Resources*, Nara, August.

Stephanie A. Miller and Lenhart K. Schubert. 1988. Using specialists to accelerate general reasoning. In Tom M. Smith, Reid G.; Mitchell, editor, *Proceedings of the 7th National Conference on Artificial Intelligence*, pages 161–165, St. Paul, MN, August. Morgan Kaufmann.

Carolyn Rosé. 2000. A framework for robust semantic interpretation. In *Proceedings 1st Meeting of the North American Chapter of the Association for Computational Linguistics*.

P. Vossen. 1997. Eurowordnet: a multilingual database for information retrieval. In *Proceedings of the Delos workshop on Cross-language Information Retrieval*, March.

Data-driven application configuration

David M. de Matos and Nuno J. Mamede

L^{2}F/INESC-ID/IST - Spoken Language Systems Laboratory
Rua Alves Redol 9, 1000-029 Lisboa, Portugal
david.matos@inesc-id.pt, nuno.mamede@inesc-id.pt

Abstract

Constructing modular applications from existing parts is difficult if there are mismatches due to input or output semantic differences during module interconnection. In order to minimize the effort of building such applications, and also as a guideline for designing modular applications from scratch, we propose an architecture in which modules are able to interface with each other without having to be reprogrammed. The architecture can be completely described using a small number of concepts. These factors allow rapid application building and reconfiguration with minimal manual intervention, potentiating module reuse and reducing the effort invested in building new applications.

1. Introduction

When building modular applications, it is possible to use parts that have been constructed by third parties, that solve part of the global problem. While this way of work is desirable because it promotes reuse, reducing the global development effort, it is all but straightforward: in fact, integrating foreign modules is almost never a simple task. The integration effort may become so expensive that it may seem better to build everything from scratch.

Managing these architectures is, thus, a challenging task and their complexity can be a serious hurdle when trying to bring together different components. Although not limited to the group, this problem also occurs when building natural language processing (NLP) applications and on various levels: from file-format handling or network-level communication to interaction between modules in a large application.

Here, we are concerned primarily with the latter aspect, even though the discussion could be applied to other levels, e.g. communication issues in a distributed application. We consider such lower-level aspects transport issues, though, that may be dealt with separately. Thus, CORBA (OMG, nda) and similar architectures are not an issue, since what we are concerned with is the way modules within an application exchange data and how to describe the way they do it.

We have two goals: to define a uniform way for modules to produce/consume data; and to define a uniform module interoperability model. We intend for these aspects to be realized complementarily: the latter will be a consequence of the former. In aiming at reaching these goals we are also promoting reuse and easy construction/configuration, since we provide a way for describing module interfaces for use with existing resources.

This document is organized as follows: the data model is defined in section 2.; a working example is presented in section 3.; and, finally, some concluding remarks and directions for evolution are presented.

2. Model

This section presents the architectural model. The first part presents structural aspects; the second part details the data model; the third part deals with semantics; and the fourth part details the implied application specification.

2.1. Structural aspects

We consider modular applications in which the modules exchange data through connections between their ports. These objects as well as their properties and relationships are presented here.

Definition 1 (portsets) *Let $\mathbb{M}$ be the set of all modules in an application. For a module m, the following portsets are defined: $\mathbb{O}^m$ (all output ports); $\mathbb{I}^m$ (all input ports); and $\mathbb{P}^m = \mathbb{I}^m \cup \mathbb{O}^m$ (all ports). In addition, $\mathbb{I}^m \cap \mathbb{O}^m = \varnothing$. We use p_i^m to denote the i-th port of m (i ranges over the corresponding portset).*

The definition for connection, while still a structural aspect, is better presented below (see def. 6).

2.2. Data model

Definition 2 (unrestricted grammar) *Unrestricted grammars (Lewis and Papadimitriou, 1981, def. 5.2.1.) are quadruples $G = (V, \Sigma, R, S)$, where V is an alphabet; Σ is the set of terminal symbols ($\Sigma \subseteq V$); $(V - \Sigma)$ is the set of nonterminal symbols; S is the start symbol; and R is the set of rules (finite subset of $(V^*(V-\Sigma)V^*) \times V^*$). Direct derivation (eq. 1), derivation (eq. 2), and generated language (eq. 3) are defined as follows:*

$$\begin{aligned} u \underset{G}{\Rightarrow} v \text{ iff } w_1, w_2 \in V^*, (u', v') \in R, \\ u = w_1 u' w_2 \wedge v = w_1 v' w_2 \end{aligned} \tag{1}$$

$$w_0 \underset{G}{\Rightarrow} w_1 \underset{G}{\Rightarrow} \cdots \underset{G}{\Rightarrow} w_n \Leftrightarrow w_0 \overset{*}{\underset{G}{\Rightarrow}} w_n \tag{2}$$

$$L(G) = \{w \mid w \in \Sigma^* \wedge S \overset{*}{\underset{G}{\Rightarrow}} w\} \tag{3}$$

Σ is the union of three disjoint sets (eq. 4): Σ_k, the keyword set – the vocabulary for data description; Σ_d, used for writing data items; and Σ_i, used for writing intrinsic syntactic elements.

$$\Sigma = \Sigma_k \cup \Sigma_d \cup \Sigma_i \tag{4}$$

Definition 3 (data grammar; type grammar) *Consider port p and two grammars (as in def. 2): $\breve{G}(p)$ – for* *writing data (the down-turned mark refers to data grammar entities); and $\hat{G}(p)$ – for writing datatypes (the upturned mark refers to datatype grammar entities).*

These grammars must share the keyword set (denoted by $\Xi(p)$ (eq. 5)) and must be such that entities belonging to $L(\hat{G}(p))$ describe the datatypes of the entities belonging to $L(\check{G}(p))$. Each of the former entities works as a third grammar restricting $\check{G}(p)$: it is used to validate data written according to the lowermost-level grammar.

$$\check{\Sigma}_k(p) = \hat{\Sigma}_k(p) = \Xi(p) \tag{5}$$

Definition 4 (data; datatype; correctness; validity) *Consider port p: $dat(p) \in L(\check{G}(p))$ denotes the data at p. We define port datatype, $typ(p) \in L(\hat{G}(p))$, as a data type specification according to $L(\check{G}(p))$ and $L(\hat{G}(p))$ (the third-level entities mentioned before). The following relation exists between a datastream and its associated datatype:*

$$typ(p) \rightsquigarrow dat(p) \tag{6}$$

Data is correct if it belongs to the language generated by the associated grammar: $dat(p) \in L(\check{G}(p))$, by definition; but it may happen that $dat(q) \notin L(\check{G}(p))$ (for some other port q) – in this case, $dat(q)$ would be incorrect according to $\check{G}(p)$.

Data is valid if $typ(p) \rightsquigarrow dat(p)$, i.e., the data stream follows the datatype definition (besides respecting the underlying grammar's rules).

The complete discussion of $typ(p)$ would only be complete taking into account the semantics of $L(\hat{G})$, but that is out of the scope of this document.

Taking into account the definitions in this section, we now give an example. Consider port p and a data representation containing the following XML (W3C, 2001a) fragment:

$dat(p) =$

```
[...]
<class name="nc">dog</class>
[...]
```

Then the terminal symbol sets would be (at least):

$$\check{\Sigma}_i(p) = \{\texttt{<}, \texttt{>}, \texttt{=}, \texttt{/}, \texttt{"}\}$$
$$\check{\Sigma}_k(p) = \Xi(p) = \{\texttt{class}, \texttt{name}\}$$
$$\check{\Sigma}_d(p) = \{w \mid w \notin \check{\Sigma}_i(p) \cup \Xi(p)\}$$

Consider a datatype description, for the data representation above, of which the following XML Document Type Definition (DTD) fragment is a part:

$typ(p) =$

```
[...]
<!ELEMENT class (#PCDATA)>
<!ATTLIST class name CDATA #REQUIRED>
[...]
```

Then the terminal symbol sets would be (at least):

$$\hat{\Sigma}_i(p) = \{\texttt{<}, \texttt{>}, \texttt{!}, \texttt{\#}, \texttt{ELEMENT}, \texttt{PCDATA}, \texttt{ATTLIST}, \texttt{CDATA}, \texttt{REQUIRED}\}$$
$$\hat{\Sigma}_k(p) = \Xi(p) = \{\texttt{class}, \texttt{name}\}$$
$$\hat{\Sigma}_d(p) = \{w \mid w \notin \hat{\Sigma}_i(p) \cup \Xi(p)\}$$

Thus verifying the grammar pair selection conditions (def. 3 and eq. 5).

2.3. Semantic aspects

This section deals with semantic aspects and restrictions that have to be observed when handling connections.

Each module has sole control over its internal semantics, in particular, in what concerns data semantics (defined by the receiver).

Definition 5 (semantics) *Consider port p and some interpretation function I (defined by the module's inner semantics): $sem(p)$ denotes the semantics required at p for normal processing behavior; $sem(dat(p))$ represents the data stream's semantics at p: computed by I (eq. 7). The data stream's semantics must subsume the port's semantics (eq. 8).*

$$sem(dat(p)) = I(dat(p)) \tag{7}$$
$$sem(p) \sqsubseteq sem(dat(p)) \tag{8}$$

Although we have no way of knowing how a module will interpret a piece of data, we can still write the following relations if we consider $\mathbb{D}$, the function denoting its argument's domain, and $\equiv$ the usual identity operator:

$$[\, typ(p) \rightsquigarrow dat(p) \,] \Leftrightarrow [\, I(typ(p)) \equiv \mathbb{D}(I(dat(p))) \,] \tag{9}$$

and thus (from 7, 9, and $\mathbb{D}$'s definition):

$$sem(dat(p)) \in I(typ(p)) \tag{10}$$

Definition 6 (connection) *Consider modules m and n and ports $p_i^m \in \mathbb{O}^m$ and $q_j^n \in \mathbb{I}^n$. Let predicate $con(p_i^m, q_j^n)$ be true if a connection exists between the pair. In the semantics domain, the output port's semantics must subsume the input's, i.e., condition 11 must be met.*

$$sem(q_j^n) \sqsubseteq sem(p_i^m) \tag{11}$$

Definition 7 (semantics mapping function) *When establishing a connection between two ports, p_i^m and q_j^n, if $typ(p_i^m) \neq typ(q_j^n)$, we need a semantics mapping function, $\theta_{i,j}^{m,n}$, for translating semantics across the connection (eq. 12, but also eq. 13). Furthermore, for the ports to be connectable, the receiving port's semantics must be subsumed by a transformation of the semantics of the previous module's output (cond. 14).*

$$\theta_{i,j}^{m,n} : typ(p_i^m) \to typ(q_j^n) \tag{12}$$
$$\theta_{i,j}^{m,n} : L(\check{G}(p_i^m)) \to L(\check{G}(q_j^n)) \tag{13}$$
$$sem(dat(q_j^n)) \sqsubseteq sem(\theta_{i,j}^{m,n}(dat(p_i^m))) \tag{14}$$

It is impossible, however, to guarantee a correct translation in the semantics domain, since, ultimately, input semantics is defined by the data consumer: we approach semantics conversion through datatype-directed data conversion. Since this conversion uses outside information about the ontologies of both sender and receiver, $\theta_{i,j}^{m,n}$ cannot be automatically generated solely from the information available at each end. Nevertheless, $\theta_{i,j}^{m,n}$ can be defined extensionally for each $typ(p_i^m)$.

We assume that it is always the receiver's responsibility to convert the data, since the data producer may be unable to determine how its results will be used. In the current discussion, we will also assume that condition 14 always holds, either because $\theta_{i,j}^{m,n}$ can satisfy it or, if that is not the case, because missing data parts can be supplemented by defaults when computing $sem(q_j^n)$.

2.4. Specifying the application

The model above gives rise to a data-oriented module interconnection architecture in which modules send/receive information to/from each other through typed channels that are uniquely defined by the datatypes at each end-point and by the corresponding translation function.

Since the architecture is not concerned with the modules' inner semantics, all that is needed to describe it completely are the collections of port datatypes and translation functions associated with connected ports. These collections are represented, respectively, by T, the datatype matrix, and by Θ, the translation matrix.

The datatype matrix is defined for all modules and their ports. Entries that do not correspond to actual ports are empty.

$$\underset{m_i \in \mathbb{M}}{T} = \begin{bmatrix} typ(p_1^{m_1}) & \cdots & typ(p_1^{m_\mathcal{M}}) \\ \vdots & & \vdots \\ typ(p_\mathcal{P}^{m_1}) & \cdots & typ(p_\mathcal{P}^{m_\mathcal{M}}) \end{bmatrix} \tag{15}$$

$$\mathcal{M} \equiv \#\mathbb{M} \qquad \mathcal{P} \equiv \max_{m \in \mathbb{M}}(\#\mathbb{P}^m)$$
$$\forall_{m \in \mathbb{M}} \forall_{1 \le v \le \mathcal{P}},\ p_v^m \notin \mathbb{P}^m \Rightarrow typ(p_v^m) = \varnothing \tag{16}$$

The translation matrix is defined for all connected ports: one function for each connection. In all other cases, Θ is undefined.

$$\Theta = \begin{cases} \theta_{i,j}^{m,n} & con(p_i^m, q_j^n) \quad \text{(see def. 6)} \\ \textit{undefined} & \text{otherwise} \end{cases} \tag{17}$$

3. A small example

This example simplifies the model in important ways: all data flowing between ports is represented in XML and all datatypes can be specified either using DTDs or XML Schemas (XSD) (W3C, 2001d). Thus, in principle, all mismatches are due to variations in the XML data type definitions.

$$\forall_{m \in \mathbb{M}} \forall_{p,q \in \mathbb{P}^m}, \check{G}(p) \equiv \check{G}(q), \hat{G}(p) \equiv \hat{G}(q) \tag{18}$$

unless (only keywords are different)

$$\forall_{m \in \mathbb{M}} \forall_{p,q \in \mathbb{P}^m}, typ(p) \neq typ(q) \Rightarrow \Xi(p) \neq \Xi(q) \tag{19}$$

In our example, all datatypes have been described using DTDs and all necessary $\theta_{i,j}^{m,n}$ functions have been specified by Extensible Style Sheet (XSL) (W3C, 2001b) templates. By specifying all DTDs and XSL templates, the application becomes completely defined from the point of view of its data exchange paths.

The rest of this section will particularize further each of these aspects.

3.1. The application

The example application performs syntactic analysis of natural language sentences (fig. 1).

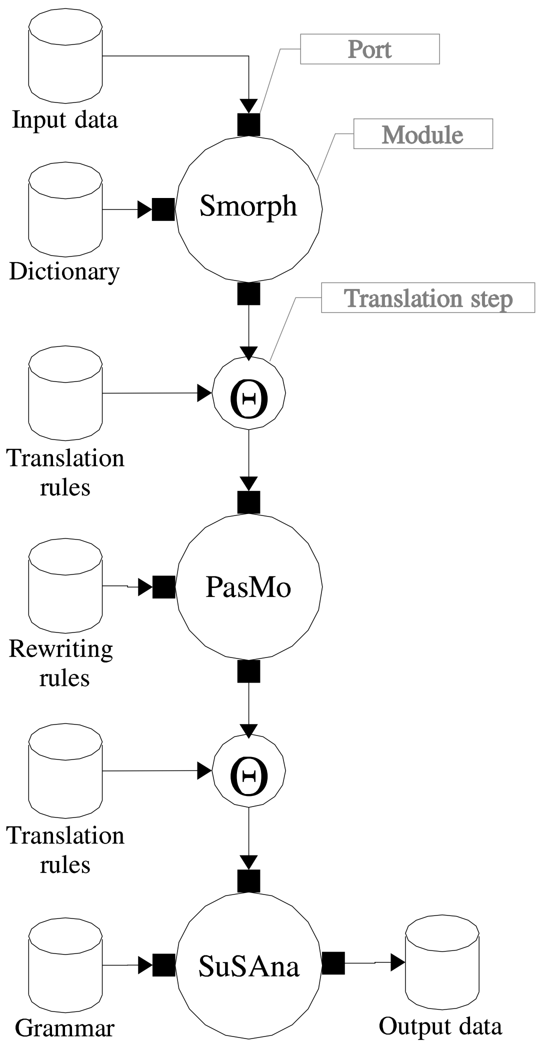


Figure 1: The example application.

The application consists of three modules: Smorph (Aït-Mokhtar, 1998) (morphological analyzer); PAsMo (Paulo, 2001) (rule-based rewriter); and SuSAna (Hagège, 2000; Batista, nd) (syntactic analyzer).

We consider only ports dealing with the data stream to be processed, thus ignoring those used for reading static data (such as dictionaries). Furthermore, in the following we will focus on the connection between Smorph and PAsMo, since the other relevant connection (that between PAsMo and SuSAna) is analogous.

3.2. The application ports

The relevant ports are Smorph's output (s) and PAsMo's input (p). To describe the data flowing through them, we need to specify just $typ(s)$ and $typ(p)$ (eq. 15 and figures 2 and 3). Smorph's output will be translated before being used by PAsMo. Note that Smorph's is a more expressive description (thus obeying condition 11), and that some information will be lost in the conversion (not a problem as long as condition 14 remains true).

```
<?xml version="1.0" encoding="iso-8859-15"?>
<!ELEMENT pasmo-in (word)*>
<!ELEMENT word     (class)*>
<!ATTLIST word  text CDATA #REQUIRED>
<!ELEMENT class    (flag)*>
<!ATTLIST class root CDATA #REQUIRED>
<!ELEMENT flag     EMPTY>
<!ATTLIST flag  name  NMTOKEN #REQUIRED>
<!ATTLIST flag  value CDATA   #REQUIRED>
```

Figure 2: DTD for PAsMo's input port, corresponding to $typ(p)$.

```
<?xml version='1.0' encoding='iso-8859-1' ?>
<!ELEMENT smorph (item)*>
<!ELEMENT item (root)*>
<!ATTLIST item value CDATA #REQUIRED>
<!ELEMENT root (class)*>
<!ATTLIST root value CDATA #REQUIRED>
<!ELEMENT class (flags,flags)>
<!ATTLIST class type (0|mi) "0">
<!ELEMENT flags (flag)*>
<!ATTLIST flags level (1|2) #REQUIRED>
<!ELEMENT flag EMPTY>
<!ATTLIST flag name  NMTOKEN #REQUIRED>
<!ATTLIST flag value CDATA   #REQUIRED>
```

Figure 3: DTD for Smorph's output port, corresponding to $typ(s)$.

3.3. The translation step

The only relevant transformation, $\theta^{s,p}$, is the one mapping Smorph's output to PAsMo's input. It is implemented as a XSL transformation step and is completely specified by the set of XSL templates (figure 4) that map between data described according to Smorph's DTD and PAsMo's.

```
<?xml version='1.0' encoding='iso-8859-1' ?>
<xsl:stylesheet
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  version="1.0">
<xsl:output method="xml"
  encoding="iso-8859-15"
  doctype-system="pasmo-in.dtd"/>
<xsl:template match="/smorph">
  <pasmo-in>
    <xsl:apply-templates/>
  </pasmo-in>
</xsl:template>
<xsl:template match="item">
  <word text="{@value}">
    <xsl:apply-templates/>
  </word>
</xsl:template>
<xsl:template match="root">
  <class root="{@value}">
    <xsl:apply-templates
      select="class/flags/flag"/>
  </class>
</xsl:template>
<xsl:template match="flag">
  <flag name="{@name}" value="{@value}"/>
</xsl:template>
</xsl:stylesheet>
```

Figure 4: The translation specification in XSL, corresponding to $\theta^{s,p}$.

4. Related work

This work is related with several fields. The first is the field of data modeling, especially in what concerns very high-level modeling, such as the one done using UML (OMG, ndb). Specifications done in UML can be described using the XML Metadata Interchange (XMI) (OMG, 2002) specification that can then be used to specify the XSDs for the data being sent/received on a module's ports. This is useful because it allows us to describe graphically each module and its interconnections and, by extension, an entire application.

Since we plan on evolving in the direction of service specification(see sec. 5.), we have considered work in this area. One such is IBM's Web Services Flow Language (Leymann, 2001) which can be used for specifying multiple aspects of web services. This language is also layered on top of others: Web Services Description Language (WSDL) (W3C, 2001c) and Web Services Endpoint Language (WSEL) (Leymann, 2001). Although this structure closely parallels what we intend in our work, it has a different focus and does not invalidate our proposal.

The third area is that of communication systems, which typically define module interconnection architectures. An example is CORBA. Another, of particular interest for NLP, is the Galaxy Communicator (MIT, 2001; DAR, nd). This architecture is a distributed, message-based, hub-and-spoke infrastructure optimized for constructing spoken dialogue systems. It uses a plug-and-play approach that enables the combination of commercial and research components. It supports the creation of speech-enabled interfaces that scale gracefully across modalities. In this context, our proposal enables easy specification of Galaxy applications. At a different level, our specifications can be gracefully translated into hub scripts and server interface definitions.

In the context of reference architectures, such as the ones proposed by the TIPSTER (TIP, nd) or RAGS (ITRI, nd) projects, our model may prove useful in facilitating integration of external modules into the frameworks defined by those architectures. Note that, unlike most software infrastructures for Language Enginnering research and development, e.g. GATE (Cunningham et al., 1996), our model does not say anything about any module's function or impose any restrictions on their interfaces and is, thus, application- and domain-independent. This is so because the model is exclusively concerned with the data streams flowing between modules and the relations between their semantics at each end and not with the way each stream is used, i.e., the model is not directly concerned with application-related issues. In this sense, the model could be used to describe a kind of “smart glue” for use with other architectures, e.g. in integration efforts of existing modules into GATE's CREOLE sets, or in datatype management.

Other application-development or intercommunication infrastructures may benefit from using a high-level specification such as the one we propose here.

5. Conclusions and future directions

Our approach is useful for application development, since it focuses exclusively on the inputs and outputs of each module, without regard for module internals. This contributes to significant dependency reductions, for the modules can be almost anything and run almost anywhere, as long as a communications channel (according to our restrictions) can be established between them.

We envision various directions for future work.

The first is to provide higher-level service specifications on top of port descriptions. This would allow services to be defined using the descriptions of its inputs and outputs and, rather than exhaustively describing each port and its data, we would be able, at that higher abstraction level, to simply specify the name of the service. The rest would follow from lower-level descriptions.

Also, along the lines of higher-level abstractions and services, it would be interesting to try and specify automatic translation functions ($\theta_{i,j}^{m,n}$) based on service semantics. Of course, this would mean that semantics would have to be specified in some way as well.

Both these approaches would help to integrate user-developed modules and help integrators to develop transformation steps that cannot be wholly automatically generated.

Another direction worth considering is the construction of module and application servers: modules or pre-built applications would be presented, e.g. via a web browser, enabling users to specify custom applications.

6. References

S. Aït-Mokhtar. 1998. *L'analyse présyntaxique en une seule étape*. Thèse de doctorat, Université Blaise Pascal, GRIL, Clermont-Ferrand.

Fernando Batista. n.d. Análise sintáctica de superfície e coerência de regras. Master's thesis, Instituto Superior Técnico, UTL, Lisboa.

Hamish Cunningham, Yorick Wilks, and Robert J. Gaizauskas. 1996. Gate – a general architecture for text engineering. In *Proceedings of the 16th Conference on Computational Linguistics (COLING96)*, Copenhagen.

DARPA, n.d. *DARPA Communicator*. See: www.darpa.mil/ito/research/com/index.html.

Caroline Hagège. 2000. *Analyse syntaxique automatique du portugais*. Thèse de doctorat, Université Blaise Pascal, GRIL, Clermont-Ferrand.

ITRI, nd. *RAGS – A Reference Architecture for Generation Systems*. Information Technology Research Institute, University of Brighton. See: http://www.itri.brighton.ac.uk/projects/rags/.

Harry R. Lewis and Christos H. Papadimitriou. 1981. *Elements of the Theory of Computation*. Prentice-Hall, Englewood Cliffs, NJ. ISBN 0-13-273426-5.

Frank Leymann, 2001. *Web Services Flow Language (WSFL 1.0)*. IBM Software Group, May. See also: xml.coverpages.org/wsfl.html.

MITRE Corporation, 2001. *DARPA Communicator*, May. See: http://fofoca.mitre.org/.

Object Management Group (OMG), 2002. *XML Metadata Interchange (XMI) Specification, v1.2*, January. See: www.omg.org/technology/documents/formal/xmi.htm.

Object Management Group (OMG), n.d.a. *Common Object Request Broker Architecture (CORBA)*. See: www.corba.org.

Object Management Group (OMG), n.d.b. *Unified Modelling Language*. See: www.uml.org.

Joana Lúcio Paulo. 2001. Aquisição automática de termos. Master's thesis, Instituto Superior Técnico, UTL, Lisboa.

NIST, nd. *TIPSTER Text Program*. See: http://www.itl.nist.gov/iaui/894.02/related_projects/tipster/.

World Wide Web Consortium (W3C), 2001a. *Extensible Markup Language*. See: www.w3c.org/XML.

World Wide Web Consortium (W3C), 2001b. *The Extensible Stylesheet Language*. See: www.w3.org/Style/XSL.

World Wide Web Consortium (W3C), 2001c. *Web Services Description Language (WSDL) 1.1*, March. See: www.w3.org/TR/wsdl.

World Wide Web Consortium (W3C), 2001d. *XML Schema*. See: www.w3c.org/XML/Schema and www.oasis-open.org/cover/schemas.html.

Towards Saving On Software Customization

Svetlana SHEREMETYVA
LanA Consulting
Madvigs Alle, 9, 2tv
1829DK Copenhagen Denmark
lanaconsult@mail.dk

Alexsei PERVUCHIN, Vladislav TROTSENKO,
Alexej TKACHEV
Southern Ural State University, 76, Lenin av.
454080Chelyabinsk, Russia,
{perv,inter,tam @inf.susu.ac.ru}

Abstract

The paper suggests some ways to save on software customization when developing a family of NLP applications meeting specific domain and task requirements. The emphasis is made on the application architecture modularity and reusability of system components. A particular focus is set on an easy-to-use environment for linguist developers integrated with the architecture to facilitate the reuse and customization phase.

1 Introduction

In this paper we address the problem of rapid and low cost deployment of NLP systems by suggesting some ways to save on software customization. A wide range of literature can be found in the area of R&D of language processing software, whose goal is to facilitate future development efforts thus reducing customization cost. The issue of customization is closely related to reuse strategies and integration during development process (Prieto-Diaz, 1993; Thomas and Nejmeh, 1992). Constructing general-purpose tools that can be shared by the community is a popular topic of interest nowadays. Such tools are developed both for language acquisition and language processing. To name just a few one can mention GATE, - a tool for locating, loading and initializing components from local and non-local machines (see, e.g. Cunningham, 1999), an abstract model of thesauri and terminology maintenance OO framework (Fischer et al., 1996), grammar development environments integrated with sophisticated text-processing interfaces such as Boas acquisition tool for a quick ramp up of MT systems (see e.g., Sheremetyeva and Nirenburg, 2000), and the Advanced Language Engineering Platform,- a grammar development tool for high-level linguistic processing, (see, e.g. Bredenkamp and Henzte, 1995). It is also recognized that though many increasingly convivial, more widely distributed and hardware-independent applications softwares are currently available it is difficult to find the system that matches exactly the end-user requirements (Degoulet et al., 1994). It seems highly problematic to identify once and forever a particular locus to the dilemma of genericity versus specificity when speaking of genericity "in general" as applied to all kinds of applications. Indeed, the locus can be anything, - the system architecture, the application components, the language resources, etc. If, however, the concept of genericity is considered as applied to a family of applications, i.e., applications interleavingly sharing tasks and domains, one can probably suggest particular approaches to solve the problem. In this paper we attempt just that. The problem of customization can be considered from two perspectives: internal and external. Internal customization is responsible for improvements in a current application and for tailoring this application to the profile of a particular user. External customization refers to the effort and cost to "turn" a current application to a new one. We address these two aspects of customization by describing a cost-effective development of a specific application called AutoPat, - an application for authoring patent claims describing apparatuses in the English Language. A prototype of this application has been developed many years ago (see, e.g., Sheremetyeva and Nirenburg, 1996) so that we shall not deal with specifications of the application but rather with a re-engineering issue. We focus here on the problem of components reusability and integration of a developer's toolkit into the application architecture. Our objective is

?? to describe a cost-effective migration from the old experimental version of AutoPat, that did not support a lot of functionalities to the AutoPat product;

?? to suggest ways to make improvements (and tailoring) in the current version of the application without extra programming effort;

?? to discuss effectiveness of AutoPat external customization to conceive and realize other specific applications of the same family.

The AutoPat "closest" family includes applications with any combination of the values of the following features:

?? Application type < authoring, machine translation, information retrieval, etc.>

?? Domain < patents with different subject matters <apparatus, method, process, etc.>

?? Document type <patent disclosure, claims>

?? Languages < English, Danish, Swedish, Norwegian, German, French, etc.>

In what follows we first present the context of the AutoPat application and the development environment. Then we shall overview reusable components and detail developers' (customization) tools. Our discussion will mainly address the improvement of the application as well as advantages of distributed environment to develop these kinds of applications.

2 System overview

AutoPat is an NLP application that consists of an interactive technical knowledge elicitation module with a sophisticated but easy-to-use interface at the user end, analysis module and fully automatic text generation module. The architecture of AutoPat with integrated development environment is given in Figure 1.

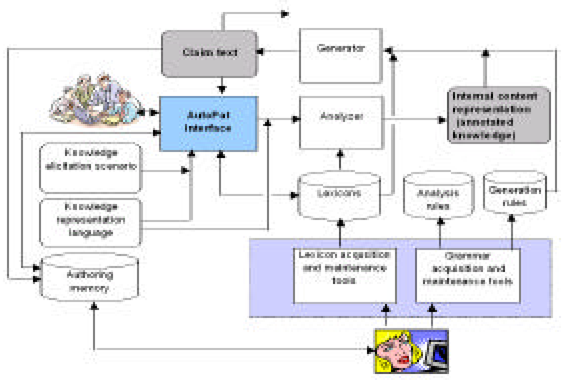
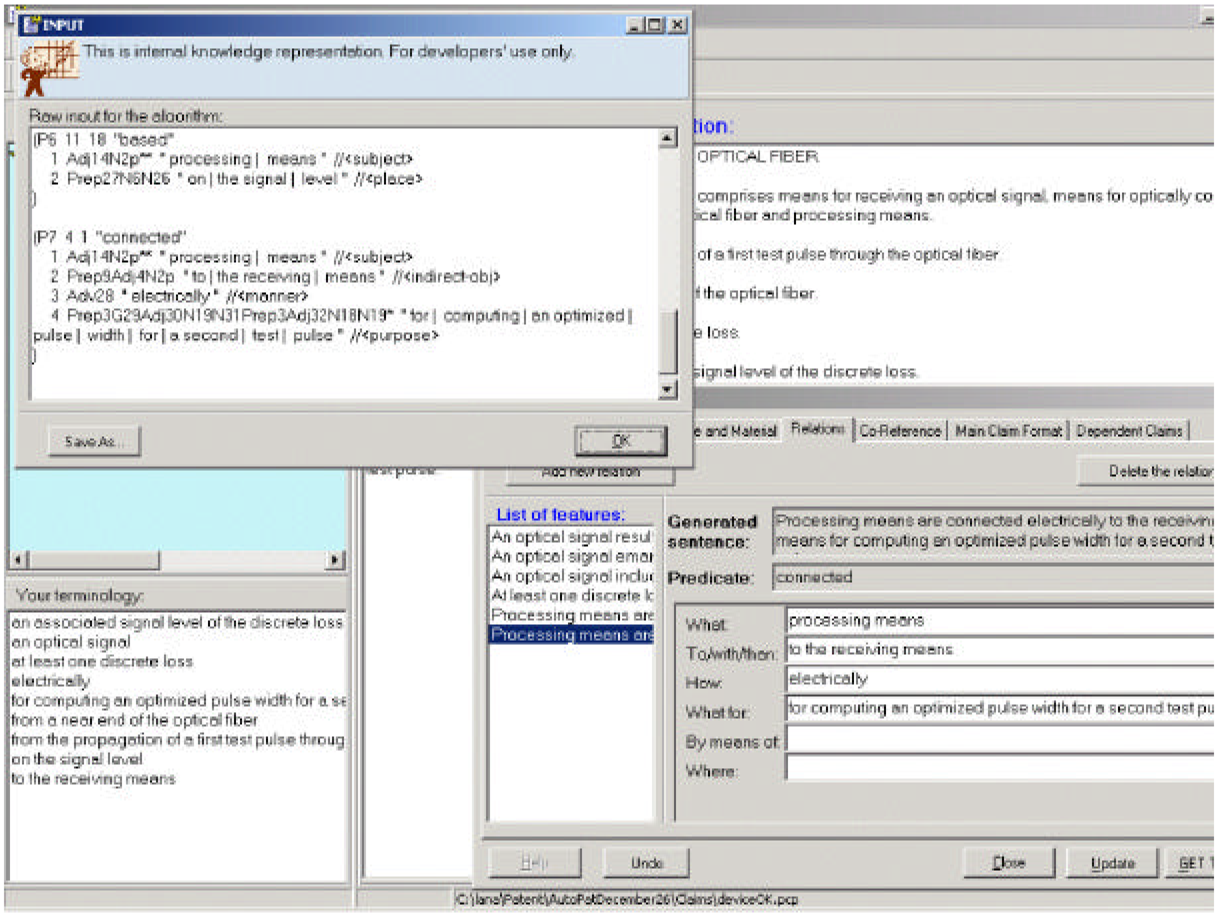

Claim text
Generator
AutoPat Interface
Analyzer
Internal content representation (annotated knowledge)
Knowledge elicitation scenario
Knowledge representation language
Lexicons
Analysis rules
Generation rules
Authoring memory
Lexicon acquisition and maintenance tools
Grammar acquisition and maintenance tools

Figure 1. AutoPat overall architecture

Superficially, the architecture of our system conforms to the standard emerged in natural language generation, in that it includes the stages of content specification, text planning and surface generation (realization), as expressed, for instance in Reiter, (1994). However, there are some important differences. Unlike the typical content specification modules, our system relies on an authoring workstation environment equipped with a knowledge elicitation scenario for joint human-computer content specification. The latter starts with the user supplying natural language phrases into the system in the process of computer interview and results in the production of a content representation of a nascent claim.

A wide range of complex problems which are considered to be specific for generation may lead one to believe that generation is completely independent of analysis. This is not, however, the case in practice. The input to generation systems that is fed into the system directly by a user must first be somehow analyzed. This problem becomes especially important in those applications in which input to generation (as it is in our case) is in textual form. The stages of AutoPat processing are not strictly pipelined.The content specification interleaves with interactive semantico-syntactic analysis in that it assigns case-roles status to input phrases and memorizes their boundaries. The values of case-roles is then



This is internal knowledge representation. For developers' use only.

Raw input for the algorithm:

```
(P6 11 18 "based"
   1 Adj14N2p** " processing | means " //<subject>
   2 Prep27N6N26 " on | the signal | level " //<place>
)

(P7 4 1 "connected"
   1 Adj14N2p** " processing | means " //<subject>
   2 Prep9Adj4N2p " to | the receiving | means " //<indirect-obj>
   3 Adv28 " electrically " //<manner>
   4 Prep3G29Adj30N19N31Prep3Adj32N18N19* " for | computing | an optimized |
pulse | width | for | a second | test | pulse " //<purpose>
)
```

tion:

OPTICAL FIBER

comprises means for receiving an optical signal, means for optically co
ical fiber and processing means.

of a first test pulse through the optical fiber

f the optical fiber

e loss

signal level of the discrete loss.

e and Material | Relations | Co-Reference | Main Claim Format | Dependent Claims

Your terminology:

an associated signal level of the discrete loss
an optical signal
at least one discrete loss
electrically
for computing an optimized pulse width for a se
from a near end of the optical fiber
from the propagation of a first test pulse throug
on the signal level
to the receiving means

test pulse

List of features:

An optical signal resul
An optical signal emar
An optical signal inclu
At least one discrete lo
Processing means are
Processing means are

Generated sentence: Processing means are connected electrically to the receivin
means for computing an optimized pulse width for a second t

Predicate: connected

What: processing means
To/with/than: to the receiving means
How: electrically
What for: for computing an optimized pulse width for a second test pu
By means of:
Where:

C:\lans\Patent\AutoPatDecember26\Claims\deviceOK.pcp

Figure 2. A screenshot of the AutoPat developer's interface window (top left corner) overlapping the user's interface for knowledge elicitation. It displays internal representation of a quantum of knowledge supplied by the user and processed by the application analyzer.

automatically unambiguously POS-tagged, assigned agreement features and interactively marked for coreference thus converting "raw" input into a shallow content representation, a claim draft. The draft is then submitted to an automatic text planner, which outputs a hierarchical structure of templates that is then input into linearizer and grammaticalizer to be converted into a legal claim text.

3 Development process: migration from experimental system to product.

3.1 Design

The first step in developing AutoPat was to define a subset of the experimental model for authoring patent claims that will be the basis of the application and the extension it will need to be turned into a product. The different functionalities of AutoPat application require, besides kernel components (such as the knowledge elicitation scenario, the knowledge representation language, lexicons, grammars and processing algorithms), a user-adaptive interface and linguistic knowledge acquisition tools to fight the well known problem of NLP applications, that of knowledge bottleneck. Developer's tools were integrated into the system architecture to facilitate the customization process and to make it cheaper.

3.2 Reuse and customization of existing components

The second step in the development process is the reuse of already developed components and their customization if they do not fit developer's needs.

3.2.1 Knowledge elicitation scenario

The knowledge elicitation scenario was almost fully reused in AutoPat, only one more step was added that of eliciting knowledge about dependent claims.

3.2.2 Knowledge representation language

Internal knowledge representation language was completely reused.

3.2.3 User Interface

The interface of the old system supported its main functions to model a professional behavior of a patent expert working with an inventor, - a knowledge elicitation interview, and to build internal content representation.

The content of the interview was almost fully reused in AutoPat, only one more step was added that of eliciting knowledge about dependent claims. Major customization dealt with lexicon acquisition functionalities and what might seem minor issues that in reality are very time consuming and thus affect the cost of application. The AutoPat interface is customized so as to support automation of tedious tasks such as typing, revising texts, making sure terminology is consistent, propagating changes through document, spell checking and lexicon acquisition. It has two different acquisition functionalities for predicate lexicon and for lexical units that fill predicate case-roles thus supporting two frames of their description (see Sections 3.2.5 and 4.2). The interface was also customized so as to better suite the user profile in terms of proficiency: the beginner has a chance to work in the Wizard Guide mode. It strictly guides the user through a step-by step procedure of describing essential features of invention and reuses experimental system interview procedure. An experienced user can work in the Professional mode that allows for more speed and flexibility when authoring a claim - the user may freely navigate among the stages of claim composition. Another new functionality allows the user to quit the program at any moment of elicitation session so that next time the user starts it s/he can resume the work where s/he left off.

3.2.3 Analyser

AutoPat reuses the architecture of the old system analyser, which consists of a submodule of interactive semantico-syntactic analysis and a submodule of automatic morphological analysis applied to the case-role fillers. The first analyser submodule is reused.

The submodule of morphological analysis is completely redone, as the old one was just a "toy" for feasibility study. Unfortunately we failed to incorporate to our system any of the described analysers as they proved to be either unavailable or not tuned to our domain. On the one hand, we tried to build our morphological analyser it in the most effort- and time-saving way. On the other hand, in view of other extensions of our AutoPat, such as multilingual generation, machine translation, information retrieval, etc., we decided to develop a reusable, possibly "extendable" morphological tool. As different types of applications need different depth of analysis our morphological submodule of the AutoPat analyzer features flexible sets of tags so that developer could vary the depth of analysis (see Sections 3.2.5 and 4.1). For example, for MT one might think of tags marking semantic information in addition to morphological, which for generation (i.e., for our immediate needs) is only necessary for predicates, but not for case-role fillers. (see Sections 3.2.5, 4.1 and 4.2). The AutoPat morphological analyser applies two levels of disambiguation procedure, one relies on context constraints, and another involves knowledge about case-roles. With one level of disambiguation it can be used as a stand-alone tool for any text from patent domain.

3.2.4 Generator

The upper level generation algorithm is reused. It consists of the same procedures: building a forest of predicate templates, linearization of this forest in a bracketed string of characters, and grammaticalization. The difference between the old version of the system and

AutoPat is that in the latter it is possible to customize algorithms at every generation step to improve the system output without extra programming effort (see Section 4.3).

3.2.5 Lexicons and grammar rules

The AutoPat lexicons are corpus-based and draw heavily on the sublanguage and on the needs of application. They include

a shallow lexicon of lexical units tagged with their class membership, which conveys morphological information (such as POS, number and inflection type) and semantic information, a concept, defining a word membership in a certain semantic class (such as object, process, substance, etc.). For example, the tag Nf shows that a word is a noun in singular (N), means a process (f), and does not end in *–ing*. This tag will be assigned, for example, to such words as *activation* or *alignment.* At present we use 23 tags that are combinations of 1 to 4 features out of a set of 19 semantic, morphological and syntactic features for 14 parts of speech. For example, the feature structure of noun tags is as follows:

```
Tag [ POS[Noun
        [object   [plural, singular]
         process [-ing, other[plural, singular]]
         substance [plural, singular]
         other      [plural, singular]]]]]
```

a deep (information-rich) lexicon of predicates. This lexicon is the main part of the AutoPat static knowledge and covers both the lexical and, crucially for our system, the syntactic and semantic knowledge. The structure of the entries of this lexicon was reused, the vocabulary was greatly enlarged. *Grammar rules* are updated (see section 4.3).

4 AutoPat Development tools

All developer's tools, including lexicon acquisition tools, have interfaces, which, on the one hand prompt acquirers to encode all the necessary features and, on the other hand, do not let them to add anything that is not relevant for the system.

4.1 Shallow lexicon acquisition tools

Shallow lexicon acquisition environment consists of several programs, including *User Interface*, for different stages of lexicon acquisition. *Web Spider* creates lists of words from a particular domain web site (5 million word corpus of US patents, in our case) in text format. *Word Sorter* sorts input wordlists in alphabetical, reverse or frequency order. *Pre-POS-Tagger* creates "dirty" lists of parts-of-speech. The human further cleans these lists. *Word Format Converter* converts lists in .txt formats into a .wdl format, - a special format used by AutoPat programs. *Word List Creator* takes unsorted files, in .wdl format sorts them in any order, merges or subtracts lists of words. *Word List Editor* maintains tagged lists of words allowing for editing, adding, deletion and search of the words. *Tag Editor* edits number and content of tags assigning them to certain groups of lexemes in the final morphological lexicon.

User Interface is used for shallow lexicon acquisition in the course of automatic spell checking. A word typed in by the user is highlighted as misspelled in two cases: when it is really misspelled or when it is not found in the underlying lexicon. The main distinguishing feature of the AutoPat spell checker is that in addition to providing hints to correct a word, it also provides for a pop-up menu of features for a word (in case the user considers it is correct) to be put into the AutoPat shallow lexicon with proper description.

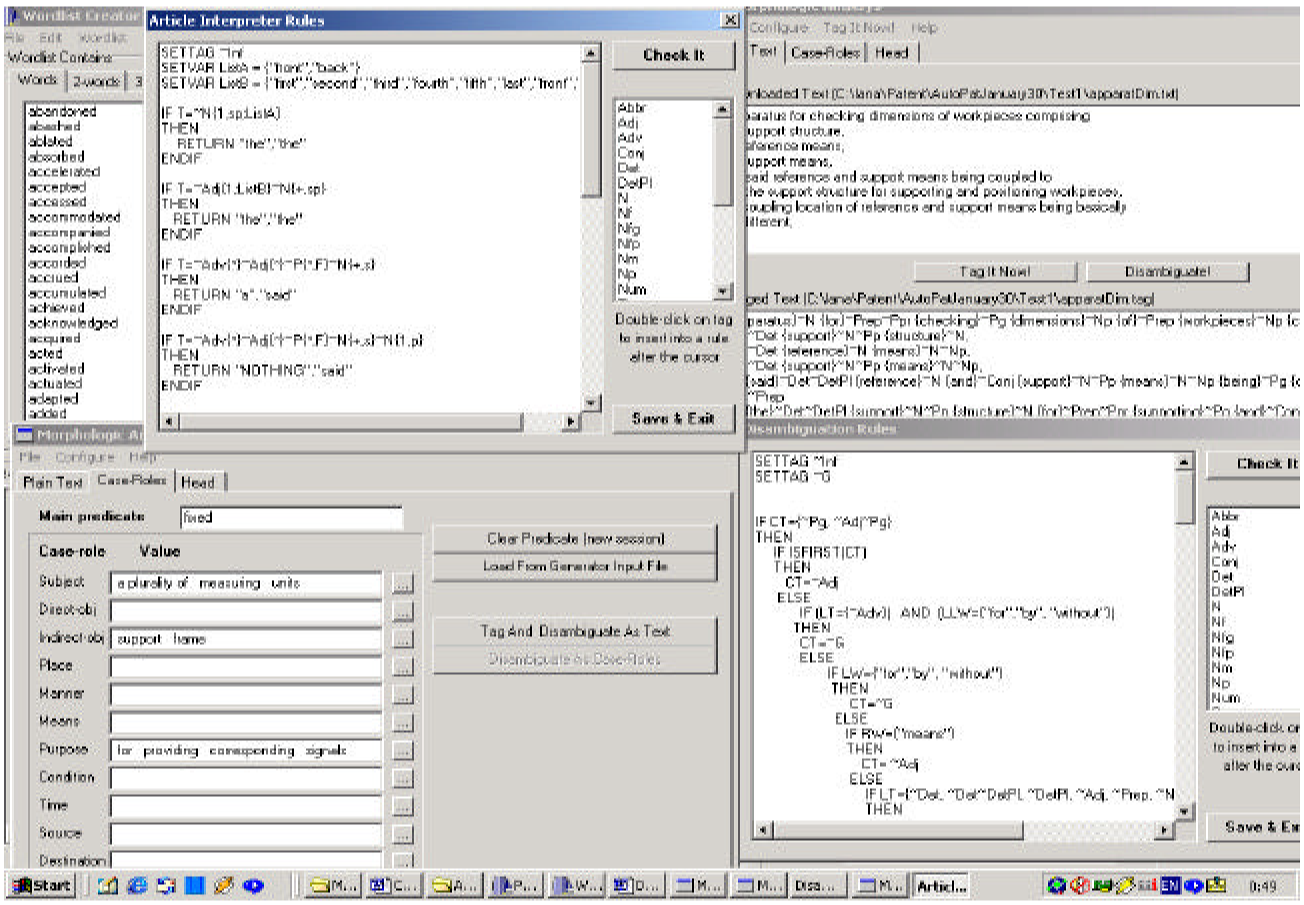
4.2 Predicate lexicon acquisition tools

The main tool for predicate acquisition is graphical Predicate Lexicon Interface. It is directly linked to the main application engine, which relies on linguistic knowledge contained in the lexicon. The interface allows for editing any of lexicon fields, search any word by its prefix or semantic class, propagate changes from one field to another. The interface program has a built-in morphological generator that automatically generates all the word forms of the predicate necessary for generation. The interface has a standing menu of semantic classes and case-roles to select from when acquiring a new word. The acquirer can customize the menu of semantic classes. Most of the fields of a new predicate entry are automatically filled with default fillers after the semantic class is acquired. The interface is programmed so as to keep acquirer "on the right road" by means of different hints and waning messages. The user can also acquire a predicate through the *User Interface* by simply typing it in a pop-up box and selecting its semantic class in the menu. The grammar forms of a new word are automatically generated in a word box for the user to check and edit if necessary, other information is assigned to a new predicate automatically by default depending upon a semantic class selected by the user. Every new word thus introduced by the user is flagged so that later a linguist could check its entry through the predicate dictionary interface closed for the user.

4.3 Grammar acquisition tools

Grammar acquisition tools include 7 compilers. Compilers 1-4 belong to the AutoPat analyzer, while compilers 5-7 compile rules for the AutoPat generator. All compilers have front-end interfaces providing rule writing help. The formal language for writing rules is very simple and has an IF-THEN-ELSE-ENDIF structure (see Figure 3). Every compiler has another interface to test the rules. Compilers for the analysis rules allow downloading any text files, not necessarily the user's input into AutoPat. That means that these compilers can be used as stand alone programs. In fact the whole morphology analysis module can be used as off-the-shelf-tool separately from AutoPat. *Compiler-1* is used to create tag disambiguation rules, which are applied to the Tagger output. These rules only use context information, which might be a tag or a lexeme within a 5-word window with a tag in question in the middle. The output of this compiler together with the output from the *Tagger* is fed to *Disambiguator* and used for the first disambiguation pass. *Compiler-2* is used to create or edit the second set of tag disambiguation rules that use syntactic knowledge about the case-roles filled by the analyzed strings. *Disambiguator* uses the output of this compiler at the second pass. *Compiler-3* creates rules, which determine whether singular and plural forms of the nouns belong to the same lexeme and can be considered as coreference candidates. *Compiler-4* creates rules for determining agreement features between the predicate and its first case-role. *Compiler-5* is used to write rules for linerazation of the claim plan tree of predicate templates. They specify the order of the words in every predicate template and the location of the templates relative one another in the nascent claim. These rules are more often subject to changes than any other rules. They are fed into *Linearizer* that substitutes the tree of templates with a bracketed string of tags. *Compiler-6* is for writing cohesion rules that delete some of the tagged strings from *Linearizer* output, insert commas and assign morphological features to predicates. Condition part of the rules uses specific knowledge provided by the *Linearizer* and by the predicate lexicon. *Compiler-7* is used for writing rules for inserting determiners before noun phrases in the final claim text. These rules should recognize coreferential phrases, which may be parts of other phrases or worded differently.

In AutoPat three types of rules are not directly linked to any compiler for updating but are "welded in" the programs. They are tagging and semantico-syntactic rules in the analysis module, and text planning rules in the generation module. Tagging rules are very simple and only suggest look-ups in the morphological lexicon. These rules can indirectly be updated through editing tag sets and morphological lexicon. Results of this knowledge update can be displayed in a. special developers' interface. Syntactico-semantic rules rely on interactive knowledge elicitation procedure and consist in looking up a predicate (selected by the user at the knowledge elicitation stage) at the predicate lexicon, presenting the user with a selected predicate template, assigning a case-role status (place, manner, etc.) to every phrase put by the user into a corresponding slot of the template and registering the boundaries of these phrases. Though these types of rules are not editable, the output of syntactico-semantic analyzer can still be checked through the developers interface built into the users' interface (see Figure 2) and its output can be edited indirectly by editing predicate lexicon. Text planning rules are very complex. They include algorithms of grouping and sorting conceptually close predicate templates into a forest of trees relying on semantic, legal, stylistic and rhetoric domain knowledge built into the system.


Wordlist Creator
File Edit Wordlist
Wordlist Contains
Words 2-words 3
Article Interpreter Rules
Check It
Abbr Adj Adv Conj Det DetPl N Nf Nfg Nfp Nm Np Num
Double-click on tag to insert into a rule after the cursor
Save & Exit
Configure Tag It Now! Help
Text Case-Roles Head
Tag It Now!
Disambiguate!
Disambiguation Rules
Morphologic A
File Configure Help
Plain Text Case-Roles Head
Clear Predicate (new session)
Load From Generator Input File
Tag And Disambiguate As Text
Disambiguate As Case-Roles
Start

```
abandoned
abashed
ablated
absorbed
accelerated
accepted
accessed
accommodated
accompanied
accomplished
accorded
accrued
accumulated
achieved
acknowledged
acquired
acted
activated
actuated
adapted
added
```

```
SETTAG ~Inf
SETVAR ListA = {"front","back"}
SETVAR ListB = {"first","second","third","fourth","fifth","last","front",

IF T=^N{1,sp,ListA}
THEN
  RETURN "the","the"
ENDIF

IF T=~Adj{1,ListB}~N{+,sp}
THEN
  RETURN "the","the"
ENDIF

IF T=~Adv{*}~Adj{*}~P{*,F}~N{+,s}
THEN
  RETURN "a","said"
ENDIF

IF T=~Adv{*}~Adj{*}~P{*,F}~N{+,s}~N{1,p}
THEN
  RETURN "NOTHING","said"
ENDIF
```

```
nloaded Text [C:\Iana\Patent\AutoPatJanuary30\Text1\apparatDim.txt]
paratus for checking dimensions of workpieces comprising
upport structure,
eference means;
upport means,
said reference and support means being coupled to
the support structure for supporting and positioning workpieces,
coupling location of reference and support means being basically
different,
```

```
ged Text [C:\Iana\Patent\AutoPatJanuary30\Text1\apparatDim.tag]
paratus}~N {for}~Prep~Pp {checking}~Pg {dimensions}~Np {of}~Prep {workpieces}~Np {c
^Det {support}^N^Pp {structure}^N,
~Det {reference}~N {means}~N~Np,
^Det {support}^N^Pp {means}^N^Np,
{said}~Det~DetPl {reference}~N {and}~Conj {support}~N~Pp {means}~N~Np {being}~Pg {c
^Prep
{the}^Det^DetPl {support}^N^Pp {structure}~N {for}^Prep^Pp {supporting}^Pg {and}^Con
```

```
SETTAG ^Inf
SETTAG ~G


IF CT={^Pg, ^Adj^Pg}
THEN
   IF ISFIRST(CT)
   THEN
     CT=~Adj
    ELSE
       IF (LT={~Adv}) AND (LLW=("for","by","without"))
      THEN
       CT=~G
       ELSE
            IF LW={"for","by","without"}
             THEN
                CT=^G
             ELSE
               IF RW=("means")
               THEN
                  CT=^Adj
                ELSE
                   IF LT={^Det, ^Det^DetPl, ^DetPl, ^Adj, ^Prep, ^N
                   THEN
```

Main predicate	fixed
Case-role	**Value**
Subject	a plurality of measuring units
Direct-obj	
Indirect-obj	support frame
Place	
Manner	
Means	
Purpose	for providing corresponding signals
Condition	
Time	
Source	
Destination	

Figure 3. A screen shot of different compilers interfaces

These rules are not editable. But the developer can still update the structure of this tree by updating the predicate lexicon. A special interface was built for the developer to follow the stages of construction of the internal meaning representation and intermediate outputs of every generating procedure. In fact the rules for building a text plan is language independent, they depend only on semantic properties of predicates which could be treated as universal for different languages.

5. Discussion and conclusion

In this paper we addressed the problem of saving on software customization when developing a family of NLP applications sharing domain and task requirements or when updating applications once created. We illustrated the approach on the example of migrating from a prototype system for authoring patent claims to an AutoPat product. The migration was performed in two steps. The first step was the analysis of the aplication, the improvement of old components, such as generator, and the realization of new

components, such as morphological analyzer and a new user interface. The second step of migration which was described in this paper was to create developers tools for customization of application and integrate them into the system.
We were unable to compare the effectiveness of our development tools to other such tools due to their unavailability. Most of developer's tools are components of commercial products and are presented as black boxes, only used internally. This makes them unsuitable for research purposes (see, for example, a similar complain in (Lezius, 1998)).

The application development process described in the paper and targeted at saving on software customization emphasizes reuse and integration. At the level of each component, the AutoPat developer can access specific tools to perform reuse and customization. Integration is about the extent of compatibility of these tools and how seamlessly they can facilitate the development of applications. The development process of AutoPat-product validated the effectiveness of both the tools and their integration into the system. Programmers' work on AutoPat was finished long before the system could be considered a product. After manual acquisition of a training amount of knowledge for programming work the linguist completed the task of creating product-size and -quality knowledge without extra programming effort. We are planning to reuse the same tools for other applications of the same family (see Introduction) including syntax parsing, machine translation and automatic indexing. For example, we have already started the work on machine translation of patent claims where all the English lexicons and tools (e.g. interfaces) for their acquisition will be reused though augmented with new relevant for MT functionalities.

Acknowledgements

This research and development has been supported by Zacco A/S, - international Intellectual Property Rights consulting firm, Denmark, Sweden, and Norway.

References

Bredenkamp and Henzte, 1995. Some aspects of HPSG implementation in the ALEP formalism. Working Papers in Language Processing No 46.

Cunningham, H 1999. JAPE: a Java Annotation Pattern Engine. Research Memorandum CS-99-06, Department of Computer Science, University of Sheffield

Degoulet P, Jean FC, Engelmann U, Meinzer HP, Baud R, Sandblad B, Wigertz O, Le Meur R, Jargermann CA. 1994. The component-based architecture of the HELIOUS medical software engineering environment. *Comp. Methods and Programs Biomed. 45, Suppl.*

Fischer,D., W.Mohr, and L.Rostek, 1996. A Modular, Object-Oriented and Generic Approach for Building Terminology Maintenance Systems. In TKE'96: *Terminology and Knowledge Engineering.* Frankfurt.

Lezius W., Rapp R., and Wettler M. 1998. A freely Available Morphological Analyzer, Disambiguator and Context Sensitive Lemmatizer for German. *Proceedings of the COLING-ACL'98 conference.* Monreal, Canada, August

Prieto-Diaz R. 1993 Status report: software reusability. *IEEE Software 10(3).*

Reiter, E.B. 1994. Has consensus natural language generation architecture appeared and is it psycholinguistically plausible? In *Proceedings of the 7th International Workshop on Natural Language Generation*

Sheremetyeva, S and S. Nirenburg. 2000. Towards A Universal Tool For NLP Resource Acquisition. *Proceedings of LREC-2000 (Second International Conference on Language Resources and Evaluation)* Athens, Greece., June

Sheremetyeva S. and S. Nirenburg. 1996. Interactive Knowledge Elicitation in a Patent Expert's Workstation. *IEEE Computer. Vol.7.*

Thomas I., and Nejmeh B.1992. Definitions of tool integration for environments. IEEE Software. 9(2).

Resource integration and customization for automatic hypertext information retrieval in a corporate setting

Maria Nava

Université de Paris-Sorbonne
Institut des Sciences Humaines Appliquées
96 boulevard Raspail,
F-75006 Paris, France

Electricité de France R&D
Dept. SINETICS/TAIC
1, avenue du Général de Gaulle
F-92141 Clamart, France

maria.nava@edf.fr

Abstract

In this paper, we describe our experience in reusing and customizing existing tools to meet new information retrieval needs in a corporate setting.
The problem was to supply an authoring aid to handle customers enquiry letters by exploiting a textual case base.
We decided to integrate, and go as far as possible with, a terminology extractor and a context exploration platform. They were previously developed through an academic and industrial collaborative research.
We have found a method to generate an information retrieval hypertext structure on a large collection of homogeneous documents by creating links between noun phrases that are pertinent for navigation. Noun phrases are selected by automatic extraction and filtered on the basis of the linguistic context class where they appear, also determined automatically.
We have tried to point out the peculiar features that made possible the reuse and integration of existing resources, to produce a relatively new solution to a fairly constrained real-world problem.

1. Introduction

Our work is motivated by an novel information retrieval (IR) need formulated in a corporate setting, at *Electricité de France R&D* (EDF R&D, the research and development department of the French national electricity board). The general problem was to supply an authoring aid to help EDF employees handle customers enquiry letters.

Starting from a textual case base and software available, we were invited to study a flexible and cost effective solution that would respect the employees savoir-faire and experience, and add value to existing tools.

Our approach aims at identifying the context where interesting NPs occur in the enquiry letters, in order to enhance the selection of pertinent cross-document links. Context identification is based on spotting linguistic markers of the expression of enquiries and on the exploitation of a structured lexicon that we can extract automatically from the textual case base.

2. The starting point

The initial scenario presented a number of constraints to be respected, concerning both the nature of the IR solution and the technical implementation.

2.1. Two corporate memory corpora

Two corpora were used to carry out a linguistic analysis, train our system for marker identification and test processing performance.

2.1.1. A large corpus of stored letters

A corpus of about 2000 customer letters, in French, was first made available by *EDF R&D*. The collection contains inquiries, intervention requests and complaints. Even when a complaint is not formulated explicitly, generally the writer's intention is to point at some sort of problem that needs fixing.

The corpus is homogenous from the point of view of the general subject matter and purpose of the letters. On the other hand, the variety of speech acts performed by the writers lends a challenging heterogeneity to the texts, interesting but problematic for automatic processing.

The corpus can be introduced in the corporate memory as a case base, and connected to customer profile and commercial strategy databases for global IR about a single customer case.

Unfortunately, letters in this corpus were not associated to the answers they had actually received.

2.1.2. A smaller corpus of letters and related answers

A second corpus of about 200 question-answer pairs is used for testing and discussing evaluation issues. It is a collection of letters that were sent directly to EDF branch managers to solicit special treatment on peculiar issues.

This gives the letters a somewhat special status, which is reflected in their style, vocabulary and structure.

We have used this smaller, more personal collection to put our system to test, point out its limitations, and try to explain them.

2.2. A terminology extraction tool: Lexter

The acquisition and exploitation of a structured lexicon are carried out automatically by the Lexter[1] system (Bourigault *et al*., 1996), developed at EDF R&D in the framework of a PhD research project. Lexter was designed to extract noun phrases (NPs) from a corpus of texts (in French). Extraction is based on the hypothesis that eligible NPs must exhibit the syntactic pattern of candidate terms, as established by terminology theory. For example : *definite article* + *noun* + *preposition* + *noun* is an observed candidate term pattern. NPs are not extracted by direct pattern matching, but they are isolated by spotting their syntactic boundaries, like, for instance, verbs. The "terminological hypothesis" is not without consequence for our work, as it will be pointed out in the conclusive section.

Extracted NPs are then automatically organized in a structured network of head-expansion relations.

Lexter accounts for morphological variants and head-modifier relations of nouns and NPs, that are grouped into families. It also supplies simple distributional figures, such as frequency of a candidate term in the corpus or candidate term head-modifier productivity within the structured network.

Lexter also stores the whole corpus divided into paragraphs, along with a pointer to the location of each candidate term in the text. This feature was initially designed to supply the terminologist with a linguistic context for validation.

Extraction and corpus-related information is stored in a relational database. We have taken advantage of all Lexter features and results for the generation of hypertext links, as described below.

2.3. A context exploration tool: ContextO

The identification of context classes where candidate terms appear is based on the contextual exploration method (Desclés *et al*., 1997) implemented in the ContextO platform (Ben Hazez & Minel 2000). The system was designed and is still developed at the LaLICC laboratory (*Langage, Logique, Informatique, Cognition et Communication*) of the Sorbonne University in Paris .

The exploration engine deployed by ContextO is based on the identification of markers of a large number of linguistic functions, as observed in the general language. Markers are acquired through a "manual" linguistic analysis of a corpus of texts, to model the expressions of linguistic functions, depending on the application. They are subsequently organized in semantic classes with object-model relations. Markers are stored in a knowledge base (a relational database, the same as Lexter' s), whichsi accessed by the contextual exploration engine of ContextO, a Java application. Markers are exploited by specialized agents, performing specific tasks. A number of tasks were already available; for example, the identification of static relations (*is-a*, *has*, etc.), causal relations, thematic focus, citations, definitions, etc. For the time being, ContextO exploits markers from French and Spanish, but could easily be adapted to other languages.

The study of our own corpus of letters has helped us find a number of linguistic structures regularly associated to the expression of complaints, justifications or requests. Each letter contains linguistic markers indicating a focus on certain speech acts that help the writer organize argumentative discourse.

For the first tests, the database contained about 200 markers organized into 24 functional classes ("complaint", "demand", "justification", etc.).

3. HyTEC, a new tool born from customization

Hypertext generation based on automatically extracted key-words usually produces an overwhelming number of non-pertinent links. Any NP can actually constitute an anchor for too large a set of heterogeneous links, a serious limitation to the effectiveness of IR.

By exploiting the features of the existing tools, we have designed a system, HyTEC (*Hypertext from TErms in Context*), capable of generating a IR hypertext structure on a large collection of homogeneous documents by selecting only those NPs that are pertinent for navigation.

Our work can be placed in the domain of IR automatic hypertext (Agosti *et al*., 1997; Allan, 1997), where paragraph (and document) linking is based on IR similarity measures, and is typed.

The specification of our IR hypertext system is based on a real-world application, that is, browsing a large textual case base made of customer enquiry letters, along with the associated reply letters. The aim of the navigation in the document base is to help finding consistent answers to any new incoming letter.

As our document base is liable to frequent updating, we found it interesting that the hypertext structure be generated at each IR session. Therefore, the document base is dynamically indexed by a short content-sample text at the beginning of the session.

A new browsing session is booted by the content of the incoming letter, which supplies content elements to compute a thematic similarity with enquiry letters stored in the corporate memory.

Navigation allows to gather information on similar cases that have already been solved and reuse written material to compose a response to handle the problem.

3.1. Identifying the context of lexical expressions

Textual similarity is computed from what we call the “pragmatic profile” of an input letter. We want to identify the discursive context of NPs in order to select only the most interesting ones and create links to similar NPs appearing in the case base, in comparable discursive contexts.

Our research is based on the articulation of two principles:

1. The exploitation of a lexicon structured by grammatical relations, extracted automatically from the whole text collection;

1 © EDF R&D.

2. The identification of linguistic markers indicating the expression of requests, complaints, justifications and other discourse acts that are relevant in our working context.

These two principles are implemented in the two different NLP systems, that offer complementary functions and results, that we have integrated.

3.2. Computing lexical links between texts

Our hypothesis is that the co-occurrence of a candidate term and a focusing structure selects a portion of text interesting for our similarity search in the case base.

The search for pertinent markers is a means to refine link generation on a number of texts already selected by their lexical components, extracted by Lexter.

In order to reduce the number and, at the same time, to keep only the most pertinent links, we have decided to maintain only the links between NPs. NPs represent a form of mutual contextualization of lexical elements and allow a more precise automatic indexing than simple nouns (Evans & Zhai, 1996). For example, instead of retaining the simple word *electricity*, we will first choose expressions like *electricity bill* or *electricity meter* (as translated from French) as content carriers, because we feel they are thematically more precise.

We have then integrated this domain-specific lexical information, extracted automatically by Lexter, and semantic and pragmatic context information supplied by markers of the general language, identified by ContextO. The lexical information triggers context analysis to create a "signature", a context-tag / NP relation, that is used for indexing and filtering.

Even if the actual language we use is French, the same principle may as well be illustrated with an example in English, like

> Due to *temporary money problems*, I'd be happy if I could *pay the bill* by installments.

Context analysis is triggered by the phrases in italics (*pay the bill* would be a nominal form in French). As markers like *I'd be happy if* (demand) or *Due to* (justification) would be found in a particular context (by context exploration rules), the sentences would be tagged as belonging to a pertinent context class.

Links between portions of texts are computed by matching signatures formed by NPs that are flagged with a semantic tag indicating a context class.

4. Similarity search results

The results obtained by testing the system on three sample entry letter are summarized in Table 1.

The performance of our system on sample texts shows that the simple association of NPs and their conditions of use can effectively improve retrieval precision, when compared to results obtained by generating links between NPs alone.

	Before context analysis		**After context analysis**	
Samples	Initial number of links	Non pertinent	Non pertinent links eliminated	Pertinent links eliminated
1	158	81	71	8
2	93	23	16	11
3	78	32	24	5

Table 1: Results for three sample input letters on the main corpus

For instance, consider the following input text (as translated from French), where extracted NPs are in italics and context markers are in bold:

> Dear Sirs,
>
> Earlier this month, I have received an invoice from you, concerning the **use of gas and electricity**, whose amount *I do not agree* with. As the **big amount** you are asking for apparently concerns only a 2-month period, I have taken down the numbers shown on my **gas meter**. The meter indicated 00613, but your invoice reports 00878. I know this number represents an estimation.
>
> On the other hand, if we consider the **huge difference** between **your estimation** and my **actual gas consumption**, *I refuse* to **pay the amount** you are asking for and *invite you* to send me a **new invoice**, reporting figures closer to reality.

Context identification allowed to retain a target text like :

> Dear Sir,
>
> I am the tenant of the apartment located X Street in TheTown, belonging to Mr and Ms Y. Since I have taken up the place in 1996, I have only received invoices reporting estimations of my electricity consumption.
>
> Before I was here, the place was unoccupied. I'll take the liberty to tell you that at present, the electricity meter indicates 36,637.
>
> ***I'd be grateful if*** you could send me an invoice corresponding to **the actual consumption**.

Notice that the NP *real … consumption* was also included in a focused sentence in the input letter (*On the other hand …, I refuse ...*). Notice also that the target text focuses on the NP *electricity meter* (*I'll take the liberty* *to…*), that could also be found in the input letter under the form *gas meter*.

We have found that it is not necessary to carry out context indexing for the input letter to improve precision: it is enough to search the context of NPs in the candidate target texts. On the other hand, to execute a relational indexing (NP + context tag) both for the input and the target texts allows precise link typing, which makes navigation easier.

The same search session as above allowed to eliminate a number of target texts, that had been retrieved be found on the basis of simple NP matching, but were not pertinent, like:

> Sirs ;
>
> I take the liberty to draw your attention to the dangerous situation menacing all the families living in our building.
>
> We experienced important damages due to water overflow last summer. To day, the leakage, which has not been stopped yet, affects the wall bearing our **electricity meters** and wiring.
>
> [*long descriptive text snipped*]
>
> If you consider that there actually is no danger, *I'd be grateful* if you could send us an official written declaration about this, *etc*.

In this target text, there is no co-occurrence of context markers and extracted NPs, as compared to the input letter. Therefore it was not retained by HyTEC, which improves retrieval precision.

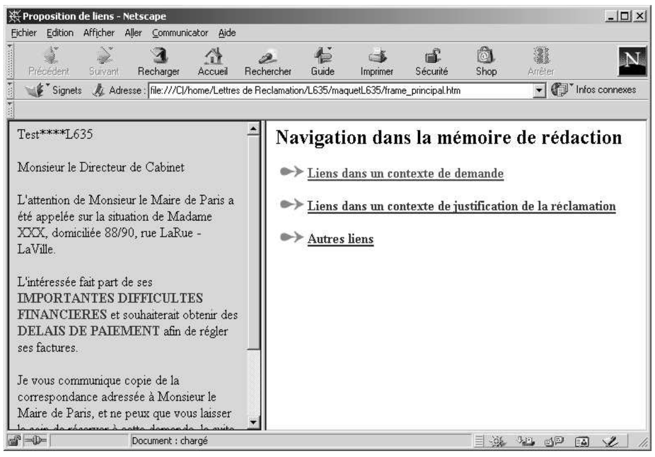
Eventually, we are left with 1) non-pertinent targets that have been retained, but also 2) pertinent candidates that have not been retrieved.

In the first case, co-occurrence of markers and NP has been identified in a sentence or paragraph, yet the rest of the letter relates events or circumstances that are different from those found in the input letter. However, the noise caused by uninteresting letters is very low, considering the number of searched texts. In this case, retrieval precision would probably improve if we could rely on a global text model, accounting for lexical and discursive chaining.

In the second case, in spite of global similarity between the input letter and a possible candidate target, content proximity has not been identified. The most frequent cause of this kind of failure is that pertinent markers focus on *synonyms* of extracted NPs. Improved recall rates should be attained cost-effectively by adding a relatively small number (the corpus is homogeneous) of synonymic relations to the NP network. We are planning to test the integration of a tool (SynoTerm) that automatically supplies Lexter with candidate synonyms from general language resources (digital dictionaries) (Hamon, 2000).

5. Generation of a hypertext structure

The results of link computation are presented in the form of a hypertext structure generated on-the-fly, directly exploiting the data structure in Lexter's relational database.


Proposition de liens - Netscape
Fichier Edition Afficher Aller Communicator Aide
Précédent Suivant Recharger Accueil Rechercher Guide Imprimer Sécurité Shop Arrêter
Signets Adresse : file:///C|/home/Lettres de Reclamation/L635/maquetL635/frame_principal.htm
Infos connexes
Document : chargé

Test****L635

Monsieur le Directeur de Cabinet

L'attention de Monsieur le Maire de Paris a été appelée sur la situation de Madame XXX, domiciliée 88/90, rue LaRue - LaVille.

L'intéressée fait part de ses IMPORTANTES DIFFICULTES FINANCIERES et souhaiterait obtenir des DELAIS DE PAIEMENT afin de régler ses factures.

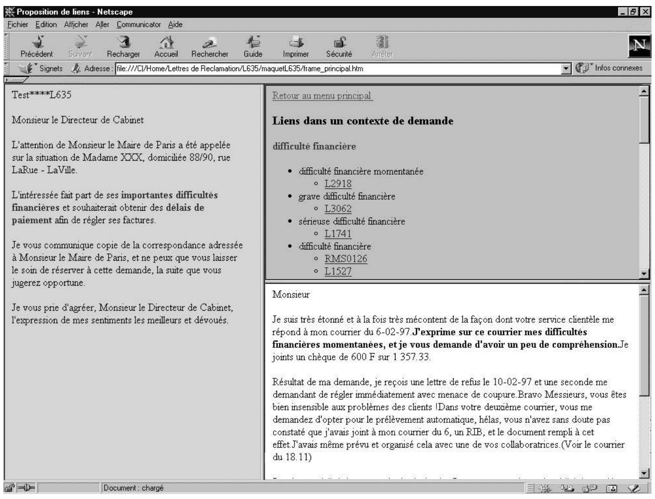
Je vous communique copie de la correspondance adressée à Monsieur le Maire de Paris, et ne peux que vous laisser

Navigation dans la mémoire de rédaction

- Liens dans un contexte de demande
- Liens dans un contexte de justification de la réclamation
- Autres liens

Figure 1: Navigating in context classes

The demonstration window (Figure 1) shows the text of the input letter (left) with salient NPs highlighted and (right) a choice of links to pertinent context classes (complaint formulation, enquiry, justification, etc.).


Proposition de liens - Netscape
Fichier Edition Afficher Aller Communicator Aide
Précédent Suivant Recharger Accueil Rechercher Guide Imprimer Sécurité Arrêter
Signets Adresse : file:///C|/Home/Lettres de Reclamation/L635/maquetL635/frame_principal.htm
Infos connexes
Document : chargé

Test****L635

Monsieur le Directeur de Cabinet

L'attention de Monsieur le Maire de Paris a été appelée sur la situation de Madame XXX, domiciliée 88/90, rue LaRue - LaVille.

L'intéressée fait part de ses **importantes difficultés financières** et souhaiterait obtenir des **délais de paiement** afin de régler ses factures.

Je vous communique copie de la correspondance adressée à Monsieur le Maire de Paris, et ne peux que vous laisser le soin de réserver à cette demande, la suite que vous jugerez opportune.

Je vous prie d'agréer, Monsieur le Directeur de Cabinet, l'expression de mes sentiments les meilleurs et dévoués.

Retour au menu principal

Liens dans un contexte de demande

difficulté financière

- difficulté financière momentanée
  - L2918
- grave difficulté financière
  - L3062
- sérieuse difficulté financière
  - L1741
- difficulté financière
  - RMS0126
  - L1527

Monsieur

Je suis très étonné et à la fois très mécontent de la façon dont votre service clientèle me répond à mon courrier du 6-02-97.**J'exprime sur ce courrier mes difficultés financières momentanées, et je vous demande d'avoir un peu de compréhension.**Je joints un chèque de 600 F sur 1 357.33.

Résultat de ma demande, je reçois une lettre de refus le 10-02-97 et une seconde me demandant de régler immédiatement avec menace de coupure.Bravo Messieurs, vous êtes bien insensible aux problèmes des clients !Dans votre deuxième courrier, vous me demandez d'opter pour le prélèvement automatique, hélas, vous n'avez sans doute pas constaté que j'avais joint à mon courrier du 6, un RIB, et le document rempli à cet effet.J'avais même prévu et organisé cela avec une de vos collaboratrices.(Voir le courrier du 18.11)

Figure 2: From typed links to target paragraphs

Once a context class has been selected, the links to target texts appear (Figure 2).

6. Evaluating task performance

The results of the first experiments are encouraging in terms of precision/recall ratio (Nava & Garcia, 2001). However, we feel that traditional evaluation measures are not completely adapted to the task, as it is often a delicate matter to decide whether two letters are even loosely connected.

As we are currently testing the system on a more extensive input letter set, a more flexible evaluation protocol is under study. It will possibly include an improved link type taxonomy and link weighing.

7. Methodological issues about customization

In this paper, we have shown how we have reused, customized and integrated two different NLP tools.

Lexter and ContextO belong to two different paradigms, which are, we believe, complementary.

Lexter and ContextO were not designed to be integrated. Lexter is a corpus-based extraction tool, ContextO is a knowledge-rich filtering system. However, we found that their results are complementary, and their coupling has provided benefits that reach beyond the simple application of cascading NLP processes.

In our case, we have observed that facility of integration and customization are related to a number of features, ranging from modularity and separation of linguistic resources and procedures, to implementation by off-the-shelf technology.

7.1. Lexter

We have taken advantage of the following:

1. Corpus-based extraction without domain-specific resources (dictionary, thesaurus, etc.);
2. Shallow morpho-syntactic structuring;
3. Access to the full-text source;
4. Simple distributional data (frequency, head-modifier productivity in the morpho-syntactic network);
5. Extraction results stored and organized in an off-the-shelf relational database (Microsoft Access).

On the other hand, considering our particular application, we have experienced an important limitation due to the fact that Lexter is basically an extractor of candidate terms. This is certainly well suited for technical, domain-specific text processing; but for our collection (customers letters), this constraint is rather restrictive. Given the source, purpose and style of the texts, we would have been happier with additional lexical information, like, for example, verbal phrases (which are generally ignored by the classical terminology theory and applications). In informal writing, an expression like *pay the bill* is often preferred to *bill payment*.

7.2. ContextO

ContextO was *designed* to facilitate the acquisition and reuse of linguistic knowledge, based on the following:

1. Separation of linguistic knowledge and search engine;
2. State-of-the-art object model of text, linguistic data and tasks;
3. Independent specialized agents exploiting the knowledge base;
4. Portable Java engine implementation accessing an off-the-shelf relational database (Microsoft Access).
5. Exploitation of markers related to structures of the general language.

It must be noted, however, that if the marker collection is largely domain-independent, it is sensitive to style and textual genre. Moreover, certain semantic classes (for example, thematic markers) are more generally reusable than others (for example, static relation markers).

Prospective work includes the adaptation of our approach to automatic, corpus-based terminology structuring.

8. Acknowledgements

Our PhD research is financed by *EDF R&D.*

9. References

Agosti, M., Crestani, F. and Melucci; M., 1997. On the use of information retrieval techniques for the automatic construction of hypertext. *Information Processing and Management*, 33(2):133-144.

Allan, J., 1997. Building hypertext using information retrieval. *Information Processing and Management*, 33(2):145-159.

Ben-Hazez, S. and Minel, J.L., 2000. Designing tasks of identification of complex linguistic patterns used for semantic text filtering. *Proceedings of RIAO 2000*, Paris, France, 1560-1568.

Bourigault, D., Gonzalez-Mullierand, I. and. Gros C., 1996. Lexter, a Natural Language Tool for Terminology Extraction. *Proceedings of the 7th EURALEX International Congress*, Göteborg, Sweden, 771-779.

Desclés, J.P., Cartier, E., Jackiewicz; A. and Minel, J.L., 1997. Textual processing and contextual exploration method. Proceedings of *CONTEXT '97*, Rio de Janeiro, Brazil, 189-197.

Evans, D.A. and Zhai C., 1996. Noun-phrase analysis in unrestricted text for information retrieval. Proceedings of the *34th Annual Meeting of the Association for Computational Linguistics*, Santa Cruz, California.

Hamon, T., 2000. *Variation sémantique en corpus spécialisé : Acquisition de relations de synonymie à partir de ressources lexicale*s. PhD Thesis, Université de Paris Sorbonne.

Nava, M. and Garcia, D., 2001. Automatic hypertext information retrieval in a corporate memory using noun phrases in context. In R. Mitkov (ed.), *Proceedings of Recent Advances in Natural Language Processing 2001 (RANLP '01)*, Tzigov Chark, Bulgaria.

Lexicalized Grammar Specialization for Restricted Applicative Languages

Patrice Lopez, Christine Fay-Varnier, Azim Roussanaly

LORIA, INRIA Lorraine and Universities of Nancy
BP 239, 54506 Vandœuvre-lès-Nancy, France {lopez, fay, azim}@loria.fr

Abstract

In the context of spoken interfaces, we present a practical methodology and an implemented workbench called EGAL (Lexicalized Tree Grammar Extraction) dedicated to design and test restricted languages used in specific task-oriented applications. A complementary methodology is proposed to process the extraction of these applicative languages from a general LTAG grammar and a training corpus. Additional results allow us to estimate the representativeness of the training corpus. An application of the system is presented for the tuning of a LTAG grammar dedicated to a spoken interface on the basis of a Wizard of Oz corpus.

1. Introduction

1.1. Motivations

In the case of a spoken dialogue system, the quality of the human computer interaction largely depends on the ability of the computer to understand spontaneous utterances normaly used by humans. The practical development of a spoken interface for a restricted domains implies that we perform the tunning of existing lexicon and grammar to a particular application. This paper proposes a methodology and an implemented workbench called EGAL (Lexicalized Tree Grammar Extraction) dedicated to design and test restricted natural languages used in specific task-oriented applications. This workbench is a sub-component of a general platform for designing spoken language systems and addresses software designers who are non-experts in natural language processing.

Specializing a grammar for restricted domains supposes at least the two following tasks:

- Cutting down the existing lexicon and grammar.
- Adding new words and new syntactic constructions.

In recent years, the development of large covering lexicalized grammars could be observed. Complementary, studies about the use of this kind of formalism for parsing spoken language have been performed. To address spoken disfluencies and robustness constraints in the context of human computer interaction, additional mechanisms have been proposed which often depend on the application domain. At the lexical and syntactic level, the following adaptations are required:

- Model spoken phenomena that could be considered agrammatical or rare in written language but frequent in spontaneous speech such as ellipsis or interpolated clauses (Price et al., 1989).
- Use robust parsing techniques to take into account the variability of the input.
- Specialize a lexicon and a grammar dedicated to text to a specific kind of dialogue and a specialized domain.

This paper addresses the last point. The specialization of a general hand written grammar to a specific domain is not a trivial task. Probabilistic methods and grammar inference as (Bod, 1995) can be seen as an alternative to this problem. Still a linguistically motivated hand written grammar provides a precise understanding of the occuring phenomena and reusability. In particular, this kind of grammar allows us to take into account the important ambiguity of the syntactic level. This ambiguity is one of the main differences between natural language that we want to process and regular languages which are just an approximation of natural language. Moreover, probabilist methods need very large annotated training corpora. Their development can require the same amount of effort as the writing of a wide-covering grammar.

We present in this paper a methodology and an implemented system called EGAL (Lexicalized Tree Grammar Extraction), able to perform an assisted specialization of a general grammar in order to obtain an applicative sublanguage from a corpus. When the specialized grammar has been obtained, a parsing module allows the evaluation of the grammar on a test corpus and the choice between various parsing algorithms and strategies. The partial and complete derivations can be visualized and compared following different criteria. The methodology also allows us to obtain information about the representativeness of the initial training corpus. Finally, the lexicalized grammar and the parser can be integrated in concrete HCI systems.

The proposed workbench can be applied to various domains. Our main goal is to design generic and portable spoken systems that can process spontaneous language. To illustrate our methodology and system, we have chosen a target application and collected an experimental Wizard of Oz corpus from which we have extracted a lexicon and a specialized grammar. We have finally evaluated the representativeness of the resulting grammar.

1.2. Lexicalized Tree Adjoining Grammars

The lexicalization of a syntactic formalism consists of the association of a set of appropriate syntactic contexts to each entry of the lexicon. Lexicon and grammar are merged in a single entity called *syntactic lexicon.* Lexicalization provides at least two main advantages: First the ability to describe syntactically each specific lexical entry allows us to choose the required complexity of the syntactic structures with flexibility. Even for restricted domains, too much generalization in syntactic descriptions generally results in unexpected border effects. Secondly the lexicalization allows parsing heuristics since a lot of syntactic ambiguity problems become lexical ambiguities which are easier to process (Abeillé, 1991).

The choice of the formalism is essential for the representation and the understanding of linguistic phenomena. It is also important to consider its applicability for NLP applications. The Lexicalized Tree Adjoining Grammars (LTAG) (Joshi and Schabes, 1992) is interesting for parsing and generation thanks to the lexicalization property and extended domain of locality. Linguistic studies and large-covering grammar developments for example in English and French have shown the practical interest of these properties. Moreover probabilistic models based on LTAG as stochastic TAG (Srinivas, 1997) or supertagging (Srinivas, 1997), allow optimizations for the processing of lexical and syntactic ambiguities on the basis of preferential choices. These properties make the LTAG formalism interesting for spoken utterances understanding (Halber, 1998) and generation in spoken systems (Becker et al., 2000).

Still the lexicalization has some drawbacks, in particular the task of designing of the grammar. Still work in progress, the English grammar of the XTAG system (Doran et al., 1994) already took ten years of development, the French grammar (Abeillé et al., 1994) more than seven years. A large covering grammar can include several thousand of elementary tree patterns called *schemata* (Candito, 1999) and a syntactic database that gives for each lemma the set of corresponding trees or tree families. Considering a given application, the use of the whole general grammar would lead to a prohibitive number of hypotheses. Moreover our goal is to avoid the development of a new grammar for each new application.

Work on the use of LTAG for dialogue systems for both parsing and generation of a sublanguage has been done recently, but the tuning of a general grammar to a specific application and domain remains a problem for the practical application of such a lexicalized formalism. The extraction of sublanguage grammars for LTAG has been discussed in (Doran et al., July 1997). But the proposed solution was based on successive manual approximations by experts. No practical methodology was proposed. No significant features have been identified that could help to perform more efficiently this task or that could lead to a software engineering solution.

2. Collection methodology

2.1. Restricted language

A restricted language can be defined as a set of utterances linked by a restricted domain, used for a particular function and generated by a specific grammar and vocabulary (Deville, 1989). Two factors limit the general language: The kind of discourse or dialogue which is realized and the application domain of the system. A restricted language is not only a subset of the whole language since an application can use technical terms which are only relevant for the domain. Moreover even in limited domains, the size of the vocabulary and the syntactic constructions change as the application evolves. Consequently a system has to propose a methodology to add new words and new syntactic contexts for the structures that would not be covered by a general grammar.

The practical advantages of the restricted language definition are a reduction of the combinatoric complexity of the processing and the ability to use a hand-written grammar (which is for example not realistic for dictation systems).

In the case of spoken dialogue systems, we claim that the systems should not understand words out of the corresponding restricted language because such words do not belong to the competence of the system. The lexicalized grammar defines here the norm of the applicative language, i.e. what is acceptable or not. Since domain restricted applications should not understand every user's request, they eventually have to lead additional dialogues with the user in the case of out of domain words.

2.2. Wizard of Oz experiments

The Wizard of Oz experiments are now widely used as a first step of the design of a spoken dialogue system. This experiment consists in the simulation of a spoken dialogue system in order to get a set of possible user interactions for a given application. The resulting corpus (which has a subjective representativeness) becomes a reference for the linguistic modeling. In other restricted domain applications, such as automatic thematic classification of e-mail in e-commerce, a similar step is necessary.

One of the main problems related to this kind of corpus is its representiveness for the application sublanguage we want to model. If the principle of restricted language is relevant, we can expect that by increasing the size of the training corpus, we will reach a size such that any addition will not result in a significant increase in the vocabulary or the size of the grammar.

Our approach consists first of obtaining a corpus which is classically divided in two parts. The first part is used to design the grammar of the restricted language (*training corpus*). The second one is dedicated to test (*test corpus*).

We have presented the different aspects which are essential for the kind of system we want to build: WoZ Experimental approach in order to obtain a corpus, specialization/designing of a lexicalized grammar dedicated to spoken language understanding, test of the resulting grammar and representativeness evaluation of the training corpus.

We have not found any existing workbench for lexicalized grammar which would combine all these aspects.

3. Presentation of the workbench

The general organization of a lexicalized tree grammar dedicated to parsing relies on three main knowledge sources:

- A morpho-syntactic database which associates an inflected form, a syntactic category and a set of morphological features.
- A syntactic database which associates a given lemma to a set of elementary trees representing the valid syntactic context for this lemma.
- A set of schemata (Candito, 1999).

The grammar designing/tuning module of the system is based on these three kinds of databases (see figure 1).

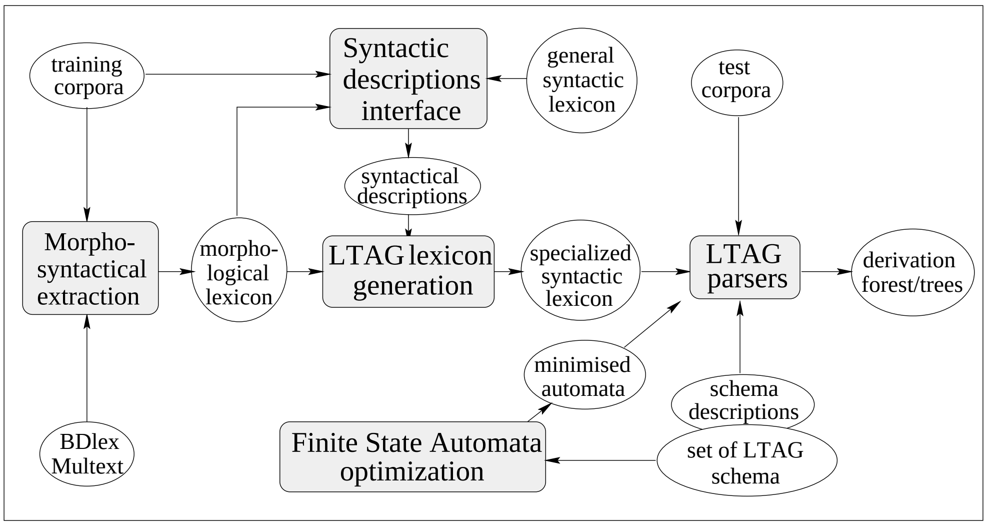


Figure 1: Overall presentation of the EGAL workbench.

3.1. Assisted generation of the lexicons

Morpho-syntactic extraction Given a training corpus, this step just corresponds to the exploitation of existing morpho-syntactic databases, Multext and BDlex (Ide and Véronis, 1994), by extracting the required information for all the words used in the corpus. This process has been implemented with an automaton-based compilation of the morpho-syntactic databases.

Set of schemata We assume that we already have a set of schemata (non-lexicalized elementary trees). For instance this schema can come from an existing hand-written grammar or from an automatic tree generation system as proposed by (Candito, 1999). A graphical editor allows the design of new schemata or the modification of existing ones.

Syntactic descriptions The goal of this module is to identify the syntactic properties associated with a lemma in order to select its correct syntactic structures. This identification is not an automatic process since resources able to enumerate all the possible predicative structures for a given lemma are not available. This result is obtained on the basis of a graphical interface dedicated to non-grammarian users.

The main idea is to associate a term of syntactic features to characterize (i) the various possible syntactic contexts covered by the general grammar (i.e. the various LTAG schemata), (ii) each lemma of a given corpus on the basis of a linguistic test suite illustrated by examples. The unification of these two structures characterizes then the precise subset of the acceptable syntactic constructions for each lemma.

The definition of our syntactic feature set is based on linguistic studies of French (mainly (Abeillé, 1991)). The current system uses nineteen syntactic features for the characterization of a verbal context (for example arity, passive, subject-verb inversion, support verb, equi-verb, reflexive, auxiliary,...) and a frame of possible prepositions. An alternative would be to use the syntactic features corresponding to the metagrammar described in (Candito, 1999) and the corresponding grammar generation system: In this case the description term corresponding to the schema that would be obtained automatically with the generation of the schemata.

For each syntactic feature we create a linguistic test composed by a question labeling the set of possible values and a set of examples. The tests are stored in a declarative way in a XML document. This XML document is then used by a generic test interface that allows a user to fill the frame for each lemma in a friendly way. The result of these tests consists of a feature term which is the syntactic description of the lemma.

For example the two following questions begin the French linguistic tests for verbs:

- Which auxiliary is used with the verb? (one between *être* and *avoir*)
- Can the verb be used in an intransitive/transitive/ditransitive context?

The tests continue until the complete frame of syntactic features and the preposition frame are specified.

The unification of the terms associated to the different schemata and the term obtained for a given lemma gives the correspondence between an entry of the lexicon and the subset of schemata that can be anchored by this entry. For instance on figure 2, the tree schema can be used with the lemma *enlever* since the two syntactic descriptions can be unified. This lexicalization process is uniform with the lexicalization performed on the basis of morphological features (for instance infinite verbs only lexicalize infinitive contexts).

This module can be used in two different ways:

- Completion of the whole list of linguistic tests in order to characterize completely a lemma for all its possible uses.
- Characterization of the syntactic contexts observed in the training corpus.

For the proposed methodology, the second possibility must be chosen. The list of utterances (in the training corpus)

enlever

```
auxiliary="avoir"
arity=transitive
nominalization="enlèvement"
argument=nominal
reflexive=true
passive=true
reflexive-lex=false
ergative=false
permutation-subjet-argument=false
inversion-subject=false
support=flase
symetrical=false
ellipse-obj1=true
preposition1=no|"de"
```

Canonical Transitive Tree Schema

```
arity=transitive
argument=nominal
preposition1=no
```

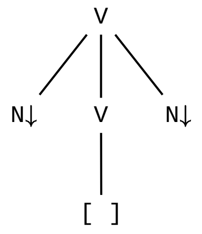


Figure 2: Two examples of syntactic descriptions: one for the French lemma *enlever*, one for a transitive tree schema.

which contain the lemma and the linguistic tests are proposed simultaneously to the user by the graphical description tool. In our methodology, contrary to the classical approach for cutting down the grammar, we specify each entry of the lexicon in terms of its category and also in terms of its correct syntactic contexts. The resulting grammar is really a *lexicalized* subgrammar.

We do not use the principle of tree family used by the XTAG system because of the small size of the lexicon and for reasons of computational efficiency. With tree families, the final selection of trees associated to an entry of the lexicon is obtained dynamically by unification at the time of instantiation. Here the correct trees are already predefined and listed in the syntactic lexicon.

A complementary tool for linguists allows the design of linguistic tests. We note that:

- The descriptions obtained by filling the features frame are independent from the lexicalized formalism. For instance, one could use HPSG lexical types.
- This module allows us to integrate easily new words to a system by characterizing the inflected forms which are not recognized during the morphological extraction. Moreover a very important point is that adding new words with this tool can be done by a non-linguist user if the linguistic tests are correctly written.

Automatic generation of the specialized LTAG syntactic lexicon This step produces the syntactic lexicon by exploiting information from the three databases described before. We add to each entry of the morphological lexicon the list of LTAG schemata which can be lexicalized. This list is obtain by

- The unification of the morphological features of the flexed form with the morphological features of the node to be anchored.
- The unification of the syntactic feature term that describes the corresponding lemma with all the syntactic feature terms of the schemata.

The links to schema are simply noted with external references using the XML links mechanisms. The final anchoring is classically done as a pre-parsing process.

3.2. Parsing test workbench

After the generation of a grammar for an applicative sublanguage given a training corpus, this module aims to test the results on a second test corpus. It allows us:

- To visualize the parsing results (both partial and complete ones).
- To check the generated grammar and possibly change manually some data in the syntactic lexicon or the set of schemata.
- To test and to compare various parsing heuristics and strategies.
- To study out of grammar phenomena.

This workbench implements two chart parsing algorithms and several parsing heuristics:

- A bottom-up connection driven algorithm that delivers extended partial results (Lopez, 23 25 February 2000).
- An implementation of the top-down Earley-like algorithm of (Schabes, 1994).

The bottom-up parser gives complete and partial parses with or without unification of features structures used in Feature Based LTAG. These different kinds of results aim to test the grammar by identifying the step involved in the failure of the parsing.

3.3. Technical choices

The implementation have been made in Java for portability reasons. All the involved data are encoded in the highly portable formalism XML. A specific application of XML dedicated to resources used with LTAG has been developed called TagML (Tree adjoining grammar Markup Language) (Lopez and Roussel, 2000). TagML allows an efficient representation of these data in term of redundancy. For instance it is possible to encode only one time substructures that are redundant in several schemata. Similarly it is also possible to share feature equations occuring in several schemata. All these redundancies imply redundant computation that could be avoided. This standard representation allows easy resource exchanges with our research partners and allows the sharing and the comparison of tools. The DTD allows us to check the consistency of the whole grammar. Every parser that respects this encoding norm can be integrated to the parsing workbench very easily.

The Java sources, classes and documentation of the parsing test workbench, including editors, are freely available on request. The other modules should also be packaged and available at the time of the conference.

4. Grammar of the GOCAD corpus

4.1. A target application: GOCAD

The GOCAD application aims to model geological surfaces. The protocol and the Wizard of Oz experiment used with this application are presented in (Chapelier et al., 1995). This experiment allowed us to obtain a corpus which has been encoded following the TEI specifications[1]. This corpus of transcribed French spoken utterances is presented in Table 1.

4.2. LTAG for the applicative restricted language

The corpus has been divided in a training corpus (80% of the utterances) and a test corpus (20%). The size of the LTAG grammar obtained with the EGAL system is presented Table 2. The total number of links to schema is a good metric for the whole size of the syntactic lexicon.

Given this specialized lexicalized grammar, the average time for parsing is 167 ms per utterance with an average lenth of utterances of 6.42 words per utterance on Sun Ultra 1. It is difficult to compare with results obtained with the complete French LTAG grammar because first the covering of this complete grammar is really limited for this corpus (124 unknown words). Moreover, for technical reasons, this grammar has been designed for the XTAG system which is very difficult to install (SunOS 4 only for instance) and use. For indication, the parsing of sentences of 10 to 15 words can take more than ten minutes.

4.3. Representativeness of the training corpus

The morphological extraction phase and the generation of the syntactic lexicon for GOCAD are fast (less than one second for the first one, less than ten seconds for the second on an average workstation). Consequently it is possible to realize systematic tests to study the evolution of the generated data. The method consists of first randomly selecting utterances from the whole corpus and then generating the corresponding LTAG grammar. This allows us to study the evolution of the size of the grammar given the number of links to a schema in function of the number of utterances taken into account. A decrease of the slope of the curve indicates an improvement of the coverage. A horizontal asymptote would mean that the coverage of the grammar is perfect for the target sublanguage. The Figure 3 gives the evolution observed for the GOCAD corpus: The number of new structures obtained by considering the last two hundred utterances is very low and we can conclude that the final generated grammar is a good approximation of the GOCAD sublanguage.

[1]This corpus is available on the Silfide server (`http://www.loria.fr/projets/Silfide/`)

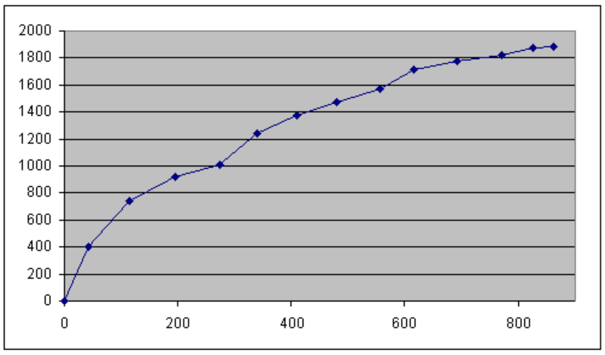


Figure 3: Evolution of the size of the generated LTAG grammar (number of links to schema) as a function of the size of the training corpus (number of utterances)

Such a result can be very useful to estimate the size of the corpus needed to reach a satisfactory covering rate. Covering 100% of the utterances is not our objective since in our approach only utterances corresponding to the competence of the spoken system need to be understood.

5. Future direction

We plan to see how the workbench scales up to other corpora and applications different than spoken interfaces. Our second goal is to extend the specialization workbench to cover multilinguality. One difficulty that arises is that the syntactic features used for the description of tree schemata and lemmas can be different from one language to another. It would mean that only a subset of these features has a real multilingual validity and could be used for parallel specialization of multilingual syntactic ressources. Syntactic features depending on the language might be limited if we only restrict them to pairs of languages, i.e. not considering all the languages at the same time.

6. References

Anne Abeillé, Béatrice Daille, and A. Husson. 1994. FTAG : An implemented Tree Adjoining grammar for parsing French sentences. In *TAG+3*, Paris.

Anne Abeillé. 1991. *Une grammaire lexicalisée d'arbres adjoints pour le français*. Ph.D. thesis, Université Paris 7.

Tilman Becker, Anne Kilger, Patrice Lopez, and Peter Poller. 2000. Multilingual generation for translation in speech-to-speech dialogues and its realization in verbmobil. In *ECAI'2000, Berlin, Germany*.

Rens Bod. 1995. *Enriching Linguistics with Statistics : Performance Models of Natural Language*. Ph.D. thesis, University of Amsterdam.

Marie-Hélène Candito. 1999. *Structuration d'une grammaire LTAG : application au français et à l'italien*. Ph.D. thesis, University of Paris 7.

Laurent Chapelier, Christine Fay-Varnier, and Azim Roussanaly. 1995. Modelling an Intelligent Help System from

Number of user utterances	number of words	average number of words/utterance.
862	5535	6,42

Table 1: GOCAD corpus

number of inflected forms	number of schemata	number of links to schema
526	71	1776

Table 2: Size of the LTAG grammar corresponding to the training GOCAD corpus.

a Wizard of Oz Experiment. In *ESCA Workshop on Spoken Dialogue Systems*, Vigso, Danemark.

Guy Deville. 1989. *Modelization of task-Oriented Utterances in a Man-Machine Dialogue System*. Ph.D. thesis, University of Antwerpen, Belgique.

Christy Doran, Dania Egedi, Beth Ann Hockey, B. Srinivas, and Martin Zaidel. 1994. XTAG System - A Wide Coverage Grammar for English. In *COLING*, Kyoto, Japan.

C. Doran, B. Hockey, P. Hopely, J. Rosenzweig, A. Sarkar, F. Xia, A. Nasr, O. Rambow, and B. Srinivas. July 1997. Maintaining the forest and burning out the underbrush in XTAG. In *Workshop on Computational Environments for Practical Grammar Development (ENVGRAM '97)*, Madrid.

Ariane Halber. 1998. Grammatical factor and spoken sentence recognition. In *Workshop on Text, Speech and Dialog, Brno*.

Nancy Ide and Jean Véronis. 1994. Multext (multilingual tools and corpora). In *14th Conference on Computational Linguistics (COLING'94), Kyoto, Japan*.

Aravind K. Joshi and Yves Schabes. 1992. Tree Adjoining Grammars and lexicalized grammars. In Maurice Nivat and Andreas Podelski, editors, *Tree automata and languages*. Elsevier Science.

Patrice Lopez and David Roussel. 2000. Predicative LTAG grammars for Term Analysis. In *TAG+5*, Paris, France.

Patrice Lopez. 23-25 February, 2000. Extended Partial Parsing for Lexicalized Tree Grammars. In *International Workshop on Parsing Technology, IWPT 2000*, Trento, Italy.

Patti Price, Robert Moore, Hy Murveit, Fernando Pereira, Jared Bernstein, and Mary Dalrymple. 1989. The integration of speech and natural language in interactive spoken language systems. In *Proceeding of Eurospeech*, Paris, France.

Yves Schabes. 1994. Left to Right Parsing of Lexicalized Tree Adjoining Grammars. *Computational Intelligence*, 10:506–524.

Bangalore Srinivas. 1997. *Complexity of lexical descriptions and its relevance to partial parsing*. Ph.D. thesis, University of Pennsylvania, Philadelphia.

A Versatile Knowledge Management Package

Hurskainen Arvi
University of Helsinki, Box 59
FIN-00014 University of Helsinki, Finland
Arvi.Hurskainen@helsinki.fi

Abstract

It is suggested that for the knowledge management system to be accurate, flexible and have wide coverage, the customer should have access to the whole chain of analysis modules, whether general or language-specific. The customer should not be made dependent on pre-coded texts. The analysis package should provide possibilities to perform various levels of encoding, disambiguation, etc. A knowledge management environment, tested with text of 8 million words, is briefly described.

1. Introduction

Sophisticated and powerful systems for knowledge management are not necessarily easy to use, while user-friendly systems are found among such applications that perform fairly simple tasks. Can these expectations be met within one single system? And if not, which of these two should be given preference? There are big differences among end users in the preparedness to use time for learning the system.

I personally, coming from the research community and serving primarily specialised communities outside the academia, am interested in maximizing the usability of a knowledge management package, which consists of language resources as well as of various language-specific and general-purpose applications. I shall discuss a system that analyses the language cumulatively, starting from morphology (Koskenniemi 1983; Hurskainen 1992) and extending to disambiguation (morphological and semantic) and syntactic analysis (Karlsson 1990, 1995a, b; Karlsson et al.; Tapanainen 1996, 1999; Tapanainen and Järvinen 1994; Voutilainen et al. 1992; Voutilainen and Tapanainen 1993). Maximum versatility of the system is achieved by including into the package all components of analysis, so that the user can perform a large number of different tasks within one single system (Hurskainen 1999a). The cumulative language analysis package can be conceived as a chain of phases, where each phase takes the output of the previous phase as input and performs a defined operation. Therefore, each phase of analysis is a potential cutting point, which can be a basis for a customized application. The result of a morphological analyser, for example, is the phase, on which a spelling checker can be built. A further phase in the chain provides a basis for more advanced language proofing tools, such as phrase structure checkers. These are examples of such customisation where the end user has very few choices.

We may, however, conceive of a system where the end user has access to each phase of the analysis chain and to all its intermediate analysis results. Such possibly useful intermediate phases are the list of tokens (tokeniser), morphological analysis with all possible grammatically correct interpretations of each word-form (useful for studying homonymy), heuristic morphological guesser for assigning analysis for unrecognised words, morphological disambiguation (non-ambiguous morphological interpretation of word-forms), semantic disambiguation (non-ambiguous word sense interpretation of word-forms), syntactic analysis (shallow syntactic parsing or full dependency trees [Tapanainen and Järvinen 1997; Järvinen and Tapanainen 1997; Tapanainen 1999]), etc.

If the user has access to the whole package, he or she is able to make maximum use of the power of each of the programs, and can use texts of one's own choice as input. In order to utilize the versatility of the system, the user environment should facilitate processing in pipe, such as found in Unix/Linux environments. Also several general-purpose utilities, filters and programming languages of Unix add to the usefulness of the system. In other words, a comprehensive analysis system working in a powerful environment facilitates the maximum number of applications.

It becomes obvious from the above that this is not a hit-a-button system, and my impression is that really useful systems cannot be made very simple. But they can be made manageable and fairly easy to learn and use, provided that correct and detailed information is given about the properties and function of each module. Below I shall explicate the ideas given above by describing a language-specific knowledge management system applied to Swahili.

2. Components of the system

The knowledge management system has the following components:

(1) A program that cuts and marks the text into suitable pieces for further processing.

(2) Text normaliser that formalises the text suitable for computational analysis.

(3) Tokeniser that identifies each token in text and verticalises the text.

(4) Morphological analyser that gives each token one or more analyses.

(5) Heuristic guesser that utilizes morphological features and assigns an analysis to unrecognised words.

(6) Morphological disambiguator that resolves morphological ambiguity on the basis of context.

(7) Semantic tagger that tags the word-forms semantically with the help of a look-up dictionary.

(8) Semantic disambiguator that resolves semantic ambiguity with the help of (a) context-sensitive rules, and (b) by utilizing the technique of Self-Ordering Map (SOM).

(9) Syntactic analyser. Two versions are provided, one for shallow syntactic parsing (Constraint Grammar), and another one for constructing full syntactic trees (Dependency Grammar).

(10) General lexical database manager for transforming the result suitable for general dictionary compilation.

(11) Domain-specific database manager for preparing domain-specific dictionaries.

(12) Information extraction based on linguistic analysis.

The system is so constructed that each module is built on top of the previous one. For example, the heuristic guesser (5) can be applied only after the phases 1-4 have been processed. Because all modules, except for (1) and (3), are language-specific, access to the whole package has to be provided for the user.

The system has been under development since 1985, and currently it has major components for morphological analysis (Hurskainen 1992), morphological and semantic disambiguation (Hurskainen 1996), as well as for shallow syntactic mapping (Hurskainen 1999a).

3. Where is the bottleneck?

Although it is generally accepted that knowledge-based systems enhance greatly the performance of knowledge management, very few such systems exist. The bottleneck might not be only the lack of sufficiently advanced modules in the system, but rather the fact that the system is necessarily fairly complex, and that rarely the designer(s) of the system have access to the source code of all the components of the system. It often turns out that the developer of a module finds himself struggling with problems that actually should have been solved by the developer of the previous module in the chain. And if the previous module cannot be corrected, the developers are left struggling with problems in the wrong environment. Work with rule-based disambiguation and syntactic parsing is an example of such work, where the developers should have a possibility to correct or change the morphological parser. There are always bugs in the morphological parser, and such bugs are often found in testing disambiguation rules. Also the coding system might need improvement, and it would be best to fix it in the morphological parser and not patch it up after the analysis has already been performed.

4. Shell scripts vs. user-defined processing

The system sketched above makes it possible to construct the processing chain in several ways, depending on needs. As it is well known, command calls of varying complexity may be stored into shell script files. For example, shell scripts for performing various phases in the process may be constructed so that one script performs phases 1-4, another 1-5, another one again 1-6, etc. It is important to note that the system allows the processing of raw text, and with each call the processing is taken that far as is defined in the shell script.

What is defined in shell scripts may be called also directly from the command prompt. Command calls are, however, usually so complex that the use of shell scripts has become a common practice. However, the shell script does not necessarily always give the desired result. For that reason it is often useful to combine shell scripts and command calls as one consequtive chain of commands. What is important is that the user may build one's own working environment by using the possibilities offered by the operating system (Unix/Linux). This does not exclude the possibility that the basic package distributed to the customers contains a model environment for those who do not have interest or time to develop their own more flexible environment.

5. What format should the source text have?

Corpus texts available to the customers are often encoded in some way. If they do not have part-of-speech tagging, or even more sophisticated encoding, they are encoded at least on the level of document structure, by using SGML, TEI, or XML encoding. It depends very much on the application which type of encoding is useful. If, for instance, the application is aimed at retrieving whole documents according to selected criteria, the document structure encoding is useful, and often sufficient. On the other hand, if the application aims at fine-grained knowledge management, the encoding of the document structure provides only a kind of background level, and each token has to be encoded in respect to several levels of linguistic analysis. Let us look in more detail the usefulness of various formats of source text.

(a) SGML, TEI, and XML encoding

One of these encoding systems, increasingly XML, is used in encoding corpora. Although such encoding is not very useful for some applications, it is not harmful either, because it helps in including or excluding sections of the documents in processing.

(b) Corpora with part-of-speech tagging

For a long time corpora tagged with POS codes have been principal sources of corpus-based linguistic investigation. Tagged corpora have the advantage of transparency and fairly good reliability, because the code set as well as the actual encoding of text is visible, and the encoding has been checked. Its disadvantage is that it is a frozen document. Both the text itself and the encoding are in fixed format, and it allows very little calibration.

(c) Corpora with analysis programs

If the aim is to establish an accurate and efficient knowledge management environment, it is hard to see any other possibility than to provide the customer with a package that contains both the source texts and the analysis programs. And in fact the programs are even more important, because they give the user a possibility to use any texts. In order to make the system user-friendly, the analysis system should be able to accept text in the format where texts are usually available.

When the user has access to all programs in the package, it makes it possible to maintain the same texts in various formats. It is obvious that the ordinary text format is useful in retrieving context for keywords. It is also very likely that the user would like to have the text also in some kind of pre-processed format, so that there is no need to process all phases each time when information is searched. At least a tokenised format, or sentence-per-line format, is a useful format to preserve. In that case, processing could be started from phase 4. The heaviest part of processing is the section composed of phases 5-9, because they contain the actual linguistic analysis and disambiguation. Because this takes some time, depending on the size of the corpus, there is a temptation to analyse the corpus and preserve the analysed version of corpus as a source for further processing. The disadvantage is that the analysed files tend to become fairly large, in Swahili for example 14 times the size of the original file. The disambiguation process downsizes the result by about 50 per cent. However, if the disambiguated text is saved in zipped format, its size is about the same as the original unzipped text. Therefore, there are practical possibilities to maintain even large texts in analysed format.

6. The optimal compromise

It depends on the skills of the user what kinds of use can be made of the language management system. Testing the performance and accuracy of dictionaries (Hurskainen 1994, 1999b) requires much more from the user than the production of concordances, for example. Taking into account the fact that most users value the ease of use, the various phases of analysis described in (2) above can be packed to the following components:

(a) **Tokeniser** includes the phases 1-3
(b) **Morphological** analyser includes the phases 4-5
(c) **Disambiguator** includes the phases 6-8
(d) **Syntactic analyser** includes the phase 9

These components can be compiled as runtime versions, in which case they cannot be modified by the user but their use is easy and fast. This solution includes the most important language analysis components but still leaves the user freedom to process the result of each phase for one's own needs.

7. Summary

A full language analysis package adapted to a suitable platform enhances an accurate, powerful and versatile working environment for a number of applications and user-defined tasks. The system described here relies heavily on language-specific and rule-based components. Although the basic components are generic, domain-specific applications can be built on the basis of features in the analysis. In Appendix are some extracts from text in various formats.

Appendix

Original format

Nyakati hizi kila mtafiti analazimika kupambana na wafadhili ili apate mbinu za kufanya kazi ya utafiti. (*These times every researcher is forced to fight with sponsors in order to get means to do research work.*)

Morphologically analysed format without disambiguation

```
"<*nyakati>"
        "wakati"  N 11/10-PL AR ' time '
"<hizi>"
        "hizi"  V IMP  AR SV ' put to shame '
        "hizi"  V <kwisha  AR SV ' put to shame '
        "hi-i"  PRON DEM :hV 9/10-PL ' this '
"<kila>"
        "kila"  N 7/8-SG IND A [C] [X]
        "kila"  ADJ A-UNINFL AR ' every '
"<mtafiti>"
        "tafiti"   V  SBJN  VFIN  1/2-SG3-OBJ   AR  SV  SVO  '  do
research '
        "tafiti"   V  SBJN  VFIN  1/2-PL2-SP   AR  SV  SVO  '  do
research '
        "mtafiti"  N 1/2-SG AR ' researcher '
"<analazimika>"
        "lazimika"   V  1/2-SG3-SP  VFIN  PR:na   AR  SV  '  be
necessary , be forced to ' STAT
"<kupambana>"
        "pambana"  V INF  SV SVO ' decorate ; fight with ' REC
"<na>"
        "na"  CC @CC
        "na"  AG-PART
        "na"  NA-POSS
        "na"  PREP
"<wafadhili>"
        "fadhili"   V  SBJN  VFIN  1/2-PL2-OBJ   AR  SV  SVO  '  be
generous , donate '
        "fadhili"   V  SBJN  VFIN  1/2-PL3-OBJ   AR  SV  SVO  '  be
generous , donate '
        "fadhili"   V  SBJN  VFIN  1/2-PL3-SP   AR  SV  SVO  '  be
generous , donate '
        "fadhili"   V  1/2-SG2-SP  VFIN  PR:a   AR  SV  SVO  '  be
generous , donate '
        "fadhili"   V  3/4-SG-SP  VFIN  PR:a   AR  SV  SVO  '  be
generous , donate '
        "fadhili"   V  11-SG-SP  VFIN  PR:a   AR  SV  SVO  '  be
generous , donate '
        "fadhili"   V  1/2-PL3-SP  VFIN  PR:a   AR  SV  SVO  '  be
generous , donate '
        "mfadhili"  N 1/2-PL AR ' donor , patron'
"<ili>"
        "ili"  CONJ **CLB AR ' so that , in order to '
"<apate>"
        "pata"  V SBJN VFIN 1/2-SG3-SP  SV SVO ' get '
"<mbinu>"
        "mbinu"  N 9/10-NI-SG ' means '
        "mbinu"  N 9/10-NI-PL ' means '
"<za>"
        "a"  GEN-CON 9/10-PL
"<kufanya>"
        "fanya"  V INF  SV SVO ' do, make '
"<kazi>"
        "kazi"  N 9/10-0-SG ' work '
        "kazi"  N 9/10-0-PL ' work '
"<ya>"
        "a"  GEN-CON 3/4-PL
        "a"  GEN-CON 9/10-SG
        "a"  GEN-CON 5/6-PL
        "a"  5/6-PL-SP
"<utafiti>"
        "tafiti"   V  SBJN  VFIN  3/4-SG-OBJ   AR  SV  SVO  '  do
research '
        "tafiti"   V  SBJN  VFIN  11-SG-OBJ   AR  SV  SVO  '  do
research '
        "tafiti"   V  SBJN  VFIN  1/2-SG2-SP   AR  SV  SVO  '  do
research '
        "tafiti"  V SBJN VFIN 3/4-SG-SP  AR SV SVO ' do research
'

        "tafiti"  V SBJN VFIN 11-SG-SP  AR SV SVO ' do research '
        "utafiti"  N 11-SG AR HC ' research '
"<.$>"
```

Constraint Grammar parsing, including morphological and semantic disambiguation

```
"<*nyakati>"
        "wakati"  N 11/10-PL AR ' time ' @TIME
"<hizi>"
        "hi-i"  PRON DEM :hV 9/10-PL ' this ' @<ND
"<kila>"
        "kila"  ADJ A-UNINFL AR ' every ' @AD-A>
"<mtafiti>"
        "mtafiti"  N 1/2-SG AR ' researcher ' @SUBJ
"<analazimika>"
        "lazimika"  V 1/2-SG3-SP VFIN PR:na  AR SV '  be forced
to ' STAT @FMAINV
"<kupambana>"
        "pambana"  V INF  SV SVO ' fight with ' REC
"<na>" @-FMAINV-n
        "na"  PREP @PREP>
"<wafadhili>"
        "mfadhili"  N 1/2-PL AR ' donor ' @I-OBJ
"<ili>"
        "ili"  CONJ **CLB AR ' in order to ' @CS
"<apate>"
        "pata"   V  SBJN  VFIN  1/2-SG3-SP   SV  SVO  '  get  '
@FMAINVtr>
"<mbinu>"
        "mbinu"  N 9/10-NI-PL ' means ' @OBJ
"<za>"
        "a"  GEN-CON 9/10-PL @<NOM
"<kufanya>"
        "fanya"  V INF  SV SVO ' do ' @-FMAINV-n
"<kazi>"
        "kazi"  N 9/10-0-SG ' work ' @OBJ
"<ya>"
        "a"  GEN-CON 9/10-SG @<NOM
"<utafiti>"
        "utafiti"  N 11-SG AR HC ' research ' @<P
"<.$>"
```

Syntactically analyzed format (FDG)

1	Nyakati	wakati	tmp:>5	@TIME N 11/10-PL AR ' time '
2	hizi	hizi	det:>1	@<ND PRON DEM :hV 9/10-PL ' this '
3	kila	kila	det:>4	@ADJ> ADJ A-UNINFL AR ' every '
4	mtafiti	mtafiti	subj:>5	@SUBJ N 2-SG AR ' researcher '
5	analazimika	lazimika	main:>0	@FMAINV V 2-SG3-SP VFIN PR:na AR SV ' be forced to ' STAT
6	kupambana	pambana	mod:>5	@-FMAINV-n V INF SV SVO ' fight with ' REC
7	na	na	ha:>8	@ADVL PREP ' with '
8	wafadhili	mfadhili	obj:>6	@I-OBJ N 2-PL AR ' sponsor '
9	ili	ili	pm:>10	@CS CONJ AR**CLB ' in order to '
10	apate	pata	cnt:>6	@FMAINVtr> V SBJN VFIN 2-SG3-SP SV SVO ' get '
11	mbinu	mbinu	obj:>10	@OBJ N 9/10-NI-PL ' means '
12	za	za	mod:>11	@<NOM GEN-CON 9/10-PL
13	kufanya	fanya	pcomp:>11	@-FMAINV-n V INF SV SVO ' do '
14	kazi	kazi	obj:>13	@OBJ N 9/10-0-SG ' work '
15	ya	ya	**:>16	@<NOM GEN-CON 9/10-SG
16	utafiti	utafiti	attr:>14	@<P N 11-SG AR HC ' research '
17	.			

Disambiguated format, sorted in order of frequency

1006	"<wakati>" "wakati" N 11-SG AR ' time '
1050	"<kwamba>" "kwamba" CONJ **CLB ' that '
1057	"<*mungu>" "*mungu" PROPNAME AN HUM ' God '
1086	"<vya>" "a" GEN-CON 7/8-PL
1117	"<ya>" "a" GEN-CON 3/4-PL
1272	"<ili>" "ili" CONJ **CLB AR ' so that '
1504	"<watu>" "mtu" N 1/2-PL ' man '
1563	"<alisema>" "sema" V 1/2-SG3-SP VFIN PAST SV SVO ' say '
1759	"<kama>" "kama" ADV AR ' like , such as '
1821	"<na>" "na" PREP ' with , by '
1827	"<wa>" "a" GEN-CON 3/4-SG
1879	"<cha>" "a" GEN-CON 7/8-SG
1896	"<na>" "na" AG-PART ' by '
2177	"<hiyo>" "hi-o" PRON DEM :hV ASS-OBJ 9/10-SG ' this '
2708	"<wa>" "a" GEN-CON 1/2-SG
2923	"<wa>" "a" GEN-CON 1/2-PL
3063	"<kuwa>" "kuwa" CONJ **CLB ' that '
3252	"<za>" "a" GEN-CON 9/10-PL
3343	"<wa>" "a" GEN-CON 11-SG
3533	"<ya>" "a" GEN-CON 5/6-PL
3641	"<la>" "a" GEN-CON 5/6-SG
4819	"<ni>" "ni" DEF-V:ni ' be '
4917	"<katika>" "katika" PREP ' in, at '
5798	"<kwa>" "kwa" PREP ' at, to, for '
7812	"<ya>" "a" GEN-CON 9/10-SG
12835	"<na>" "na" CC ' and '

Disambiguated format, sorted according to word-form

17	"<kinywaji>" "kinywaji" N 7/8-SG DER:ji ' drink '
7	"<kinywa>" "kinywa" N 7/8-SG HC ' throat '
8	"<kinywani>" "kinywa" N 7/8-SG HC LOC LOC ' throat '
1	"<kiofisi>" "ofisi" ADV ADV:ki 9/10-0-SG DER:i ' office '
163	"<*kiongozi>" "kiongozi" N 7/8-SG DER:zi AN ' leader '
189	"<kiongozi>" "kiongozi" N 7/8-SG DER:zi AN ' leader '
12	"<kioo>" "kioo" N 7/8-SG 'mirror'
12	"<kiota>" "kiota" N 7/8-SG ' nest '
6	"<kipaji>" "kipaji" N 7/8-SG ' talent , gift'
12	"<kipande>" "kipande" N 7/8-SG ' piece '
21	"<kipato>" "kipato" N 7/8-SG DER:o ' income '
2	"<kipato>" "pato" ADV ADV:ki 5a/6-SG DER:o ' income '
30	"<kipaumbele>" "kipaumbele" N 7/8-SG ' priority '
2	"<kipawa>" "kipawa" N 7/8-SG ' income '
5	"<kipengele>" "kipengele" N 7/8-SG ' point , aspect '

Disambiguated format, sorted according to lemma

12	"<kiota>" "kiota" N 7/8-SG ' nest '
23	"<vipaji>" "kipaji" N 7/8-PL ' gift '
6	"<kipaji>" "kipaji" N 7/8-SG ' gift '
8	"<vipande>" "kipande" N 7/8-PL ' piece '
12	"<kipande>" "kipande" N 7/8-SG ' piece '
3	"<vipandikizi>" "kipandikizi" N 7/8-PL DER:zi HC ' graft '
6	"<vipato>" "kipato" N 7/8-PL DER:o ' income '
21	"<kipato>" "kipato" N 7/8-SG DER:o ' income '
2	"<vipaumbele>" "kipaumbele" N 7/8-PL ' priority '
30	"<kipaumbele>" "kipaumbele" N 7/8-SG ' priority '
3	"<vipawa>" "kipawa" N 7/8-PL ' gift '
2	"<kipawa>" "kipawa" N 7/8-SG ' gift '
11	"<vipengele>" "kipengele" N 7/8-PL 'point , aspect '
5	"<kipengele>" "kipengele" N 7/8-SG ' point , aspect '

References

Hurskainen, A. (1992). A Two-Level Computer Formalism for the Analysis of Bantu Morphology: An Application to Swahili. Nordic Journal of African Studies 1(1): 87-122.

Hurskainen, A. (1994). Kamusi ya Kiswahili Sanifu in test: A computer system for analyzing dictionaries and for retrieving lexical data. Afrikanistische Arbeitspapiere 37 (Swahili Forum I): 169-179.

Hurskainen, A. (1996). Disambiguation of morphological analysis in Bantu languages. COLING-96, Proceedings of the 16th International Conference on Computational Linguistics, Copenhagen, August 5-9, 1996. Pp. 568-573.

Hurskainen, A. (1999a). SALAMA: Swahili language manager. Nordic Journal of African Studies 8(2): 139-157.

Hurskainen, A. (1999b). Salim K. Bakhressa, Kamusi ya Maana na Matumizi. Nairobi: Oxford University Press. Review article. Journal of African Languages and Linguistics 20.

Järvinen, T. & Tapanainen, P. (1997). Timo Järvinen and Pasi Tapanainen. A Dependency Parser for English. Technical Reports, No. TR-1. Department of General Linguistics. University of Helsinki, 1997.

Karlsson, F. (1990). Constraint Grammar as a framework for parsing running text. In Hans Karlgern (ed.), COLING-90. Papers presented to the 13th International Conference on Computational Linguistics. Vol. 3, pp. 168-173, Helsinki, 1990.

Karlsson, F. (1995a). Designing a parser for unrestricted text. In Karlsson et al (eds.), Constraint Grammar: A Language?Independent System for Parsing Unrestricted Text. Mouton de Gruyter, Berlin. Pp. 1-40.

Karlsson, F. (1995b). The formalism and environment of Constraint Grammar Parsing. In Karlsson et al (eds.), Constraint Grammar: A Language?Independent System for Parsing Unrestricted Text. Mouton de Gruyter, Berlin. Pp. 41-88.

Karlsson, F., A. Voutilainen, J. Heikkilä & A. Anttila (eds.) (1994). Constraint Grammar: A Language-Independent System for Parsing Unrestricted Text. Mouton de Gruyter, Berlin, 1994.

Koskenniemi, K. (1983). Two-level morphology: A general computational model for word-form recognition and production. Publications No. 11. Department of General Linguistics, University of Helsinki, 1983.

Tapanainen, P. (1996). The Constraint Grammar Parser CG-2. Publications No. 27. Department of General Linguistics, University of Helsinki, 1996.

Tapanainen, P. (1999). Parsing in two frameworks: finite state and functional dependency grammar. PhD dissertation. Language technology, Department of General Linguistics, University of Helsinki. http://ethesis.helsinki.fi/julkaisut/hum/yleis/vk/tapanainen/

Tapanainen, P. & Järvinen, T. (1994). Syntactic analysis of natural language using linguistic rules and corpus-based patterns. COLING-94. Papers presented to the 15th International Conference on Computational Linguistics. Vol. 1, pp. 629-634, Kyoto, 1994.

Tapanainen, P. & Järvinen, T. (1997). A non-projective dependency parser. In Proceedings of the 5th Conference of Applied Natural Language Processing, March 31st - April 3rd, Washington D.C., USA. Pp. 64-71.

Voutilainen, A., J. Heikkilä & A. Anttila, (1992). Constraint Grammar of English - A Performance-Oriented Introduction. Publications No. 21. Department of General Linguistics, University of Helsinki, 1992.

Voutilainen, A. & Tapanainen, P. (1993). Ambiguity resolution in a reductionistic parser. In Proceedings of the Sixth Conference of the European Chapter of the Association for Computational Linguistics. EACL-93. pp. 394-403, Utrecht, Netherlands, 1993.

Challenges in MT customization on closed and open text styles

Rémi Zajac
SYSTRAN Software, Inc.
zajac@systransoft.com

Abstract

This paper reports work in progress on two on-going customization projects at Systran. One project targets on-line technical support documentation. This project falls in a domain that has been (and still is) a favorite target for high-quality MT applications. The second project targets open style (on-line) texts on a large set of small domains. We outline and contrast customization issues for these two projects, and present the customization process based on an automated analysis of monolingual corpora.

1. Introduction

Manual versus automatic customization

Customization of MT systems is a problem that has not received much attention. Typically, customization is reduced to the (manual) development of a simple domain-specific dictionary. Complex lexical entries, involving complex subcategorization patterns for example, are excluded; a fortiori, syntactic customization is excluded too.

Most previous work on automated customization make use of a parallel corpus, for example Yamada et als. (1995) and Su et als. (1995, 1999). Of course, example-based systems may be considered fully customized systems (Richardson et als. 2001, Pinkham et als. 2001).

Yamada et als. (1995) present a method to adapt a rule-based MT system to a new domain by using aligned sets of sentences. The method involves the comparison of the MT parse tree (presumably after transfer) with the parse tree of the manually produced translation. A side effect of the comparison is the automatic generation of either bilingual dictionary entries or transfer rules. The interest of the method is not clear since the technical description is rather sketchy and since there is no discussion on the influence of the bilingual corpus on quality improvement. The method seems to be implemented only for simple bilingual lexical equivalences.

Su et als. (1995, 1999) suggest that customizing an MT system can be reduced to learning probabilistic parsing parameters. They use probabilistic learning techniques to select the best parse of a non-deterministic parser. The best parse is the one that gives a translation that is closest to the manually translated sentence (or the one which produces a parse tree that is closest to the parse tree for the manually translated sentence, the paper is unclear on this point). The method does not seem to be implemented.

The best current approaches, providing highest quality results, to fully automatic customization are using example-based techniques built on a substrate of a comprehensive rule-based system as in the MSR-MT project (Richardson et als. 2001, Pinkham et als. 2001). In this approach, there is no distinction between lexical and syntactic customization. What is learned is, in essence, a set of lexicalized transfer rules that may cover entire sentences.

Customization projects at Systran

Systran has recently started several customization projects, for example for on-line technical support documentation as in the Autodesk project (Senellart et als. 2001b). In all these projects, there is no bilingual corpus available. An essential part of the effort is the development of an automated methodology and tools to speed-up customization and to lower costs. A parallel effort that also supports customization projects is directed at the restructuration of the MT architecture towards better modularity and declarativity, and improved performances (Senellart 2001a).

The paper is organized as follows. The next section gives an overview of two on-going customization projects at Systran and outlines specific customization issues. One project falls in a domain that has been (and still is) a favorite target for high-quality MT applications: (on-line) technical support documentation. The second project targets open style (on-line) texts on a large set of small domains. The next two sections present the customization process. Section 3 describes the assessment of customization needs for a given application. This assessment translated into a customization plan, and the customization process itself is described in Section 4. We conclude on several open issues.

2. Two customization projects at Systran

MT applications have been traditionally divided into dissemination and assimilation applications. An assimilation scenario is an analyst working of foreign documents in open domains and open styles (technical documents as well as web postings and email). These kinds of applications use a generic MT engine with a very large lexical coverage and implementing a non-specific language model that will succeed in providing an average quality on most texts and fail to provide a high quality on most texts. Dissemination applications target the translation of technical documents, typically technical user manuals, for publication. Technical documents cover closed domains and closed styles. Dissemination applications use specific MT engines with targeted lexical coverage and implementing a tailored language model that succeeds in providing a good quality on most texts.

It should be clear that, at present, manual customization could only be envisaged in the dissemination scenario, on a closed domain and a closed text style. For this kind of scenario, some rule-based MT systems have demonstrated high-quality translations. However, no system has ever been able to provide high-quality translation on open domains and open style documents. The consensus in the MT community seems to be that on these kinds of texts, MT can only be used for assimilation scenarios. However, this consensus is based on past experience with rule-based systems: it is still an open question whether example-based or statistical-based systems may achieve high-quality on open domains and open style.

Scenario 1: on-line technical support documentation

The source corpus for this project is medium size (tens of thousands documents, about 2M words). The document style is technical and homogeneous and documents are written by technical writers following specific style guidelines and using an in-house glossary. Sub-types of documents are well defined, for example, FAQs and procedures. The domain is homogeneous covering a single family of products, but very complex, with a high number of concepts and a high number of relationships between concepts. The corpus evolves slowly over time as new documents are added for new versions of existing products. There may be completely new products (still within the same product family) with new concepts and new terminology to cover. Therefore, a continuous monitoring of the document database is necessary to keep track of the emergence of new concepts and terms. At that point, a focalized customization effort is needed for these new documents.

This type of application has been a favorite target for high-quality MT in the past and still remains a favorite (See e.g., Richarson et als. 2001, Pinkham et als. 2001). Translation problems may be reduced to some extend by using a writing style guide. For example, the lexicon can be limited to common words (new words for new concepts only). Grammar and style variation can also limited by the use of technical writing guidelines. Such guidelines could be implemented in a controlled language checker. The idea is the same as for other controlled languages, but with a more modest aim: not solving all MT problems but limiting MT problems to a narrower range.

The main challenge is how to describe the terminology of a large and complex domain: the ontology of the domain is narrow but deep, complex and very specific. For this project, we use a mix of terminology extraction tools (see e.g., Jacquemin 2001). Another issue will be tracking emergence of new concepts as the document database changes over time, and update the terminology accordingly.

Scenario 2: fast changing on-line postings

This customization project blurs the distinction between MT for assimilation and MT for dissemination. In this project, the corpus is very large, millions of postings, with several thousands of new postings every day. The style is very relaxed and makes use of a large range of colorful expressions, with plenty of misspellings and grammar errors, highly variable punctuation usage, as well as uncommon abbreviations. The corpus can be divided into a large number of unrelated sub-domains. Within each domain, the terminology is relatively restricted with a limited number of concepts and a limited number of relationships between concepts. However, there is a very large number of proper names referring to specific products and entities.

In a posting, we can identify several sections that can be categorized into different styles. For example, a description of an object, contractual sections dealing e.g., with payment options or shipping, etc. However, there are always sections that cannot be fitted into any well-defined slot and must be considered free open text. These sections are the most challenging since they are typically argumentative in nature, conveying opinions and trying to convince the reader to adhere to these opinions. These sections are also the ones exhibiting free informal syntax and creative use of language. A specific issue here is to automatically segment a posting into sections that correspond to homogeneous styles and select most appropriate translation parameters for each style.

Each posting addresses a specific domain, and each domain is shallow and relatively simple. Each domain can be managed using simple thesaurus-like management tools à la Wordnet. There are however two main challenges. One is the number of domains (hundreds). Another is the novelty factor that requires constant tracking and customization to match changes in language. In particular, we need to track the emergence of new words (neologisms as well as new names and abbreviations) and new expressions.

3. Evaluation of Customization Needs

Customization assumes a base MT system. The first step in a customization project is to measure the gap between the quality of this base system and the quality of the targeted customized system in order to evaluate as precisely as possible the customization needs, and to develop a customization plan.

If a bilingual corpus is available, the customization needs could be estimated by evaluating the performance of the base system against the corpus. The translation of the source corpus produces a baseline translation that can be compared and evaluated against the manual translation (target side of the bilingual corpus). An in-depth evaluation of mismatches provides a detailed catalog of customization requirements for both lexical customization and grammatical customization. This evaluation can also assign mismatches to specific sub-grammars of the MT system: NP analysis, verb transfer, relative clause generation, etc. This of course can only be done manually on a small set of documents only.

An indirect but more economical way of evaluating the customization needs is to:

- Measure the performance of the system on a known source baseline corpus, and to
- Evaluate the distance between the baseline corpus and the source corpus.

By using a set of quantitative linguistic indicators, it is possible to estimate the amount of customization needed to achieve a pre-set quality target. The following paragraphs give an overview of an automated customization evaluation process that includes the establishment of a baseline and the construction of a terminological (domain) profile, a lexical profile and a syntactic profile for the source corpus. These profiles are compared to the baseline in order to provide a quantitative estimate of the customization needs (Underwood & Longejan 1999).

A terminological profile of a corpus provides an estimate of the closure of the vocabulary of the corpus as well as the complexity of the domain. The vocabulary closure is measured by counting the number of new terms that appear when a new text is seen (term growth curve). If this curve flattens out rapidly (few new terms appear in newly seen documents), the vocabulary is essentially closed. In such a case, the customized system will probably require little lexical maintenance after delivery.

The complexity of the domain is estimated using the number of technical terms belonging to the domain and the number of interconnections between these terms. The number of interconnections between terms can be estimated by counting the number of syntactic relations between technical terms occurring in the same sentence: predicate-arguments relationships (predicate-object, but also predicate-subjects), and head-modifiers relationships. New usages of existing words can be detected only as a failure in parsing or translation: parsing and translation failures are collected and sorted by shared lexical units: any lexical unit that occurs in several parsing or translation failures is a potential source of failure and should be investigated.

The syntactic profile of a corpus provides an estimate of the customization work needed on grammars for parsing, transfer and generation. The base system is evaluated on a standardized test suite where test items are categorized by linguistic classes of phenomena. This evaluation provides a detailed account of the strengths and weaknesses of the base system in terms of linguistic categories. We then run the system on the source corpus and extract a frequency profile of morphosyntactic phenomena. This frequency profile is matched to the baseline profile in order to build a customization plan, and to estimate the level of quality that can be achieved for a given level of effort.

4. Customization Process

The customization loop

The source corpus is segmented into translation units (sentences) and the translation units are translated, sorted and stored in the development database. Customization then proceeds along two parallel lines: one customization for terminology and lexical elements, and another for grammar and style. Customization plans are directly derived from terminological, lexical and syntactic profiles. Since any change in a component of the system may have unforeseen impact on other components, and in order to ensure constant progress and test for regression, testing is done continuously and in parallel to development. Continuous testing uses the development database, allows to focus on the main customization issues, to deal rapidly with any potential regression, and to measure progress.

Words that are not in the system dictionaries are extracted during the initial assessment. This initial step also produces a list of lexical units that may be sources of parsing or translation failures, and are therefore candidates for revision. Terminology lists are built using terminology extraction tools. Initial customization proceeds using these lists. As the systems dictionaries are updated, the test database is translated with the updated dictionaries, and new translations are compared with the initial ones. Any translation that shows a difference is added into the review set.

The initial assessment produces a frequency list of morphosyntactic structures that appear in the source corpus. Given that the baseline evaluation identifies the weak areas of the system, this list is converted into a customization plan where the most frequent weak areas are dealt with first (modulo dependencies between grammar modules).

Testing

Testers review new translations as the system is updated. Two different kinds of testing are done, one for terminological and lexical customization, and another for grammar and style. New translations are sorted according to various criteria, including coverage of terminology and difference in translation. For example, new terms added in the dictionaries should be matched and translated for all translation units containing these terms. Two lists are built: one containing matched terms (for simple checking) and one containing unmatched terms (to identify potential dictionary coding problems). A similar testing process is used for structural customization. For example, after working on relative clauses, all sentences containing relative clauses are extracted and divided into a list of changed translations and a list of unchanged translations.

When a translation is changed, it should show an improvement in quality: progress is tracked for any changed translation and quality of new translations is evaluated and recorded. Tracking the quality improvement rate allows us to estimate the cost-effectiveness of the customization effort.

5. Conclusion

The customization method presented in this paper is directly challenged by fully automatic methods using example-based techniques, including for example MSR methods (Richardson et als. 2001, Pinkham et als. 2001). Although manual customization is obviously feasible and can reach acceptable quality, one important issue is the cost-effectiveness of the method: a manual method should be cheaper than using automated customization with a bilingual corpus. Therefore, it should be cheaper or equivalent to the cost of translating a bilingual training corpus (this obviously depends on the minimal size required by the training algorithm). We assume that it may be cheaper when there are multiple target languages as the initial work of analyzing the source corpus and extracting terminology and other specific linguistic pattern can be shared among all target languages. Another important issue is the evolution of the source document database: we need to develop specific methods for tracking changes in language and for updating the language resources at a minimal cost.

To evaluate the accuracy of the estimates of customization effort, and to evaluate the speed and cost-effectiveness of the customization methodology, we are recording a set of quantitative indicators to help us provide accurate estimations. During a customization project, we are tracking cost of creating/customizing lexical entries together with the quality impact of these new of customized entries on the whole corpus. We do the same for grammar customization. Finally, the quality of the MT system is evaluated before customization, and a post-customization evaluation provides a measure of the improvement in quality that has been achieved. Experience over several projects should help us find the most relevant indicators, and obtain accurate estimates from detailed corpus analyses.

6. References

Jacquemin, Christian. 2001. *Spotting and Discovering Terms trough Natural Language Processing.* The MIT Press.

Lalaude, Myriam, Veronika Lux, Sylvie Regnier-Prost. 1998. “Modular controlled language design”. CLAW-98, Pittsburgh, PA. Pp103-113.

Pinkham, Jessie, Monica Corston-Oliver, Martine Smets, Martine Petterano. 2001. “Rapid assembly of a large-scale French-English MT system”. MT Summit VIII, Santiago de Compostela, Spain. Pp277-282.

Richardson, Stephen, William Dolan, Arul Mezenes, Jessie Pinkham. 2001. “Achieving commercial-quality translation with example-based methods”. MT Summit VIII, Santiago de Compostela, Spain. Pp293-298.

Senellart, Jean, Peter Dienes, Tamas Varadi. 2001a. “New generation Systran translation system”. MT Summit VIII, Santiago de Compostela, Spain. Pp311-316.

Senellart, Jean, Mirko Plitt, Christophe Bailly, Francoise Cardoso. 2001b. “Resource alignment and implicit transfer”. MT Summit VIII, Santiago de Compostela, Spain. Pp317-324.

Su, Keh-Yih, Jing-Shin Chang. 1999. “A customizable, self-learnable parameterized MT system: the next generation”. MT Summit VII, Singapore. Pp182-190.

Underwood, Nancy L., Bart Jongejan. 1999. “Profiling Translation Projects”. TMI-99, Chester, England. Pp139-149.

Yamada, Setsuo, Hiromi Nakaiwa, Kentaro Ogura, Satoru Ikehara. “A method for automatically adapting an MT system to different domain”. TMI-95, Leuven, Belgium. Pp303-310.

Locating and Reusing Sundry NLP Flotsam in an e-Learning Application

Anju Saxena and Lars Borin

Department of Linguistics, Uppsala University,
Box 527, SE-751 20 Uppsala, Sweden
and
Computational Linguistics, Department of Linguistics,
Stockholm University, SE-106 91 Stockholm, Sweden

anju.saxena@ling.uu.se, lars.borin@ling.su.se

Abstract

We describe the background and motivation for an e-learning project—*IT-based Collaborative Learning in Grammar*—where NLP resource reuse has become an important issue. The resources are of several kinds: POS-tagged and syntactically annotated corpora (treebanks), parsing systems and grammar writer's workbenches, and visulization and manipulation tools for linguistically annotated corpora. Our experience thus far has been that although there are a number of such resources available e.g. on the Web, as a rule, numerous incompatibilities and lack of standardization at all levels—markup formats, linguistic annotation schemes, grammatical framework, software APIs, etc.—make the reuse of these resources into a non-trivial endeavor.

0. Preamble: the Setting

It is generally acknowledged that the goal of teaching grammar—especially at the university level—should not primarily be that students memorize definitions of concepts and grammatical constructions, but rather that they understand and learn to recognize different structural patterns. This can hardly be achieved without giving students practical training in the skill of grammatical analysis. Research has shown that hands-on problem-solving is more stimulating and thought-provoking than when the information and results are handed down to the pupils during lectures. Further, our experience has been that students learn about grammatical constructions and phenomena more actively when these constructions are discussed by comparing the system found in their native language with that of another language. An added factor contributing to an active student participation is the choice of the material forming the basis for exercises and group activities, which should preferably be as natural as possible.

With these pedagogical considerations in mind, we formulated a project for realizing a new format for teaching courses in grammar in Linguistics and Computational Linguistics (the ability to reason about grammar and to carry out grammatical analyses of language utterances being necessary prerequisites for all linguistic studies of language and thereby part of the core curriculum of these subjects). In the proposed format interactive practical training and corpus-based exercises comprise an integral part of the students' learning process, giving them the opportunity and incentive to participate more actively in their own learning process. Using IT as a tool for collaborative work allows the students to choose the problem-solving strategy which suits them best, as well as the time and place to work on the problem. A corpus of natural language material for grammatical analysis contributes to a more active participation, as it not only presents the grammatical constructions in their context, but also gives students a greater freedom to approach the material and conduct the investigation from a perspective which suits their individual learning styles. A text corpus consists of naturally occurring language in its natural physical context, since it is made up of complete texts or large text fragments, as opposed to the made-up or isolated single sentences or phrases often used to illustrate grammatical points in linguistics textbooks. This accompanying physical context makes it possible to investigate the textual, discourse-level, functions of the grammatical phenomena.

An outline of the proposed training material is presented below. It has a modular architecture, composed of four types of modules (see Figure 1, below):

1. 'Encyclopedia' module, containing descriptions of grammatical concepts and constructions. Its content will be attuned to the contents of the course and the interactive exercises (as, in their turn, the exercises will be adapted to the 'encyclopedia' contents), and at appropriate places, there will be hyperlinks to interactive exercises dealing with the current topic.

2. 'Text corpus' module, containing at least (a) POS-tagged and syntactically annotated corpora of Swedish, and (b) an annotated corpus of a foreign language. For (a), we will use the SUC and Talbanken annotated Swedish corpora (see below); for (b), we will use a corpus of Kinnauri (a Tibeto-Burman language spoken in India) narratives available on the web (`http://www.ling.uu.se/anjusaxena/corpus.html`; see figure 2), which is hyperlinked to a morpheme dictionary. Further, with the help of a graphic interface students will be able to see a 'map' of how and where one particular morpheme or a word occurs in the corpus (see Olsson and Borin (2000)), providing support in their work on the functions of grammar. The students will work with the

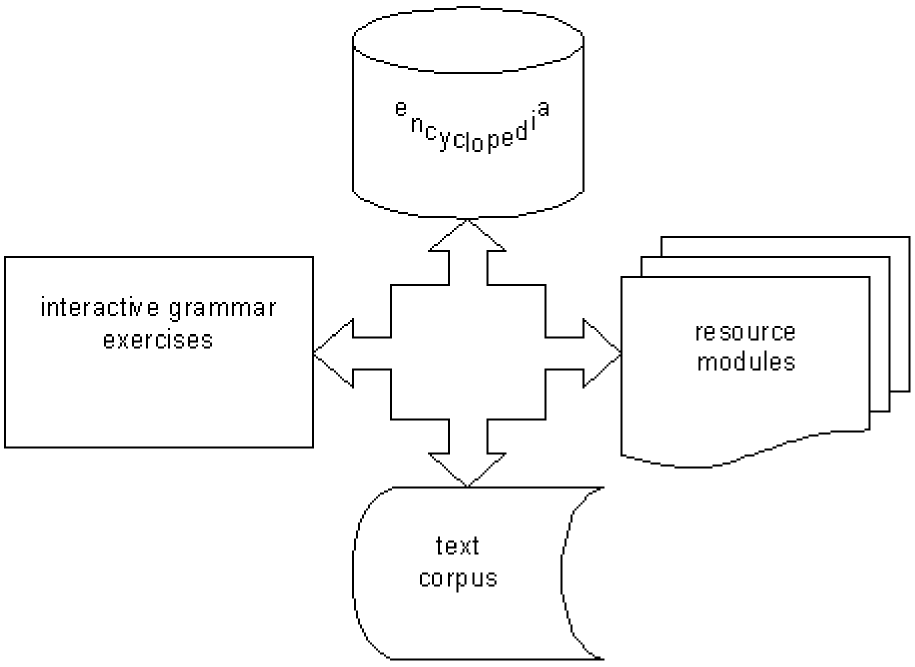


Figure 1: Organization of the proposed IT-supported grammar training application

same corpus as part of their group activities and as part of their examination.

3. ‘Interactive exercise’ module. Our aim here will be to provide students with a set of exercises, with basic tools for computer-mediated student cooperation in virtual work- groups (a ‘spreadsheet’ for problem-solving; optional ‘step-by-step questions’ for the grammatical topic covered; grammar rule writing exercises to be discussed in more detail below), with hyperlinks to the ‘encyclopedia’, to the ‘resources’ (see below) and to the annotated corpus of a foreign language (which, in turn, will be hyperlinked to the dictionary; see Saxena (2000)). As part of each theme, students will first discuss the construction during the lecture session, then again while examining the construction in the corpus, and finally also while comparing the results of the corpus-based analysis with the Swedish system and then discussing it in the group. This learning method where the same construction is examined from a number of mutually reinforcing practical and theoretical viewpoints will, hopefully, provide the students with support and incentive in their learning process. Further, the same corpus will be used in grammar courses in first and second semesters, providing grounds for deeper analyses in the second semester than would have been the case.

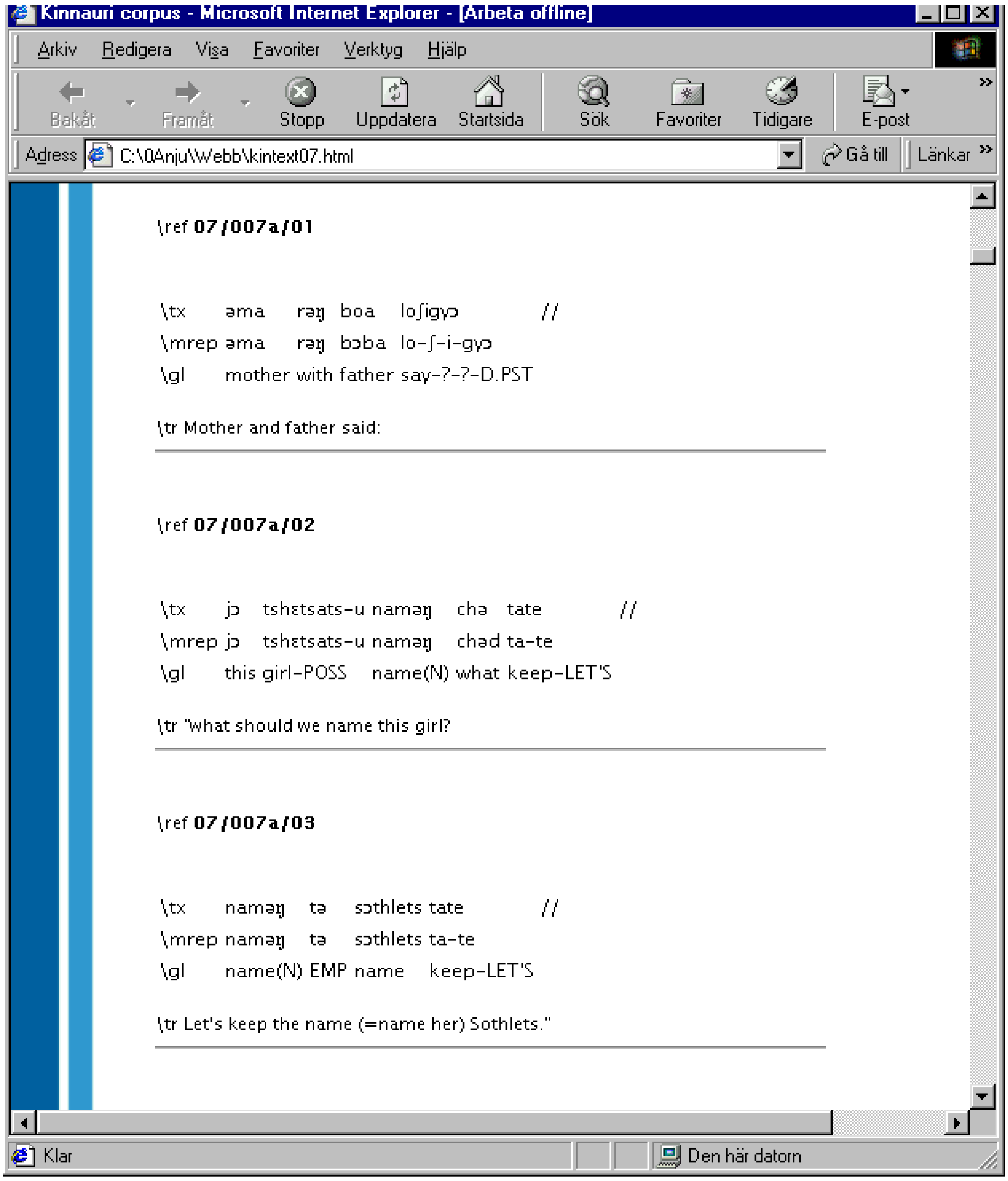
4. ‘Resource’ modules will provide a pool of resources for further reading and relevant links to other sites.

The architectural organization of the software proposed here has several advantages, the two most significant ones being extensibility and ‘conceptual decentralization’. Extensibility means that new functions can be easily integrated in the application. ‘Conceptual decentralization’ is especially significant as it allows the possibility of adjusting to individual learning styles. For example, if the student prefers to start out with the ‘encyclopedia’ material and go from there to the appropriate exercises, when she feels the need to do so, she has that choice. At the same time, the application allows the possibility of starting out at other entry points, e.g., ‘interactive exercises’, with the option of calling up the relevant ‘encyclopedia’ material at each instant.

1. The NLP Resource Customization Problem

NLP resource customization has become an issue in this project mainly in connection with module 3 (interactive grammar exercises). It has been our aim from the conception of the project to rely mostly on standard WWW and open-source software—i.e., software which is generally free and where the source code is freely available and modifiable by the user—for implementing the modules. This design philosophy has the advantage of making the application maximally platform-independent, as well as providing a familiar interface—a standard web browser—for students and faculty.

One of the exercises that we have planned for module 3 builds upon a combination of a syntactically annotated corpus (a treebank) and a grammar writer’s workbench. The basic premise of the exercises is a further refinement of the idea presented by Borin and Dahllöf (1999). We propose to use grammar rules written by students (using an existing grammar development tool) as search expressions in the


Kinnauri corpus - Microsoft Internet Explorer - [Arbeta offline]
Arkiv Redigera Visa Favoriter Verktyg Hjälp
Bakåt Framåt Stopp Uppdatera Startsida Sök Favoriter Tidigare E-post
Adress C:\0Anju\Webb\kintext07.html
Gå till Länkar
Klar
Den här datorn

```
\ref 07/007a/01

\tx    əma   rəŋ  boa   loʃigyɔ          //
\mrep əma   rəŋ  bɔba  lo-ʃ-i-gyɔ
\gl    mother with father say-?-?-D.PST

\tr Mother and father said:

\ref 07/007a/02

\tx    jɔ   tshɛtsats-u naməŋ   cha   tate          //
\mrep jɔ   tshɛtsats-u naməŋ   chəd ta-te
\gl    this girl-POSS    name(N) what keep-LET'S

\tr "what should we name this girl?

\ref 07/007a/03

\tx    naməŋ   tə    sɔthlets tate          //
\mrep naməŋ   tə    sɔthlets ta-te
\gl    name(N) EMP name     keep-LET'S

\tr Let's keep the name (=name her) Sothlets."
```

Figure 2: The Kinnauri corpus – Web format

treebank. In its simplest form, the result of the search would be expressed as precision and recall. Given an NP rule formulated by a student, we could automatically tell how many of the (maximal) treebank POS sequences matching the rule actually make up NPs, how many are not NPs, and how many NPs in the treebank are not described by the rule. There are all kinds of conceivable elaborations of this basic scheme, which could be seen as a more linguistically sophisticated parallel to the use of (unannotated) text corpora and concordancing software in so-called data-driven language learning (Flowerdew, 1996).[1] For the Computational Linguistics students, there is the additional advantage of being able to work from the very beginning of their studies with the same kind of tools and resources that they will be using 'for real' after graduating, in their professional life.

What we have found already in this beginning stage of the project, however, is that there are some serious obstacles to using available NLP resources.[2] Mostly, the issues that have arisen in this connection concern (lack of) compatibility and standardization of NLP resources. Some of

[1]The basic idea here is similar to the ICECUP FTF (Fuzzy Tree Fragment) grammatical query system for parsed corpora (Wallis and Nelson, 2000), but with a diffent use and target audience in mind.

[2]Here, we use "NLP resources" as a cover term for both *language resources* and *processing resources* in the terminology adopted by Cunningham (2002).

```
<text id=kl01>
<body>
<p>
<s id=kl01-001>
<c lem='-' msd='FI' n=1>-</c>
<w lem='vilken' msd='DH@0P@S' n=2>Vilka</w>
<w lem='djävla' msd='AQP00N0S' n=3>djävla</w>
<w lem='optimist' msd='NCUPN@IS' n=4>optimister</w>
<c lem=',' msd='FI' n=5>,</c>
<w lem='frusta' msd='V@IIAS' n=6>frustade</w>
<name type=person>
<w lem='Lasse' msd='NP00N@0S' n=7>Lasse</w>
</name>
<c lem='.' msd='FE' n=8>.</c>
</s>

<suctext id=kl01>
<p>
<s id=kl01-001>
<d n=1>-<ana><ps>MID<b>-</d>
<w n=2>Vilka<ana><ps>HD<m>UTR/NEU PLU IND<b>vilken</w>
<w n=3>djävla<ana><ps>JJ<m>POS UTR/NEU SIN/PLU IND/DEF NOM<b>djävla</w>
<w n=4>optimister<ana><ps>NN<m>UTR PLU IND NOM<b>optimist</w>
<d n=5>,<ana><ps>MID<b>,</d>
<w n=6>frustade<ana><ps>VB<m>PRT AKT<b>frusta</w>
<name type=person>
<w n=7>Lasse<ana><ps>PM<m>NOM<b>Lasse</w>
</name>
<d n=8>.<ana><ps>MAD<b>.</d>
</s>
```

Figure 3: Alternative SUC annotation formats

the issues are:

- Differences in fundamental storage and text markup formats. The three corpora that we are considering for use in the project have three different storage formats: (1) The basic format of Saxena's Kinnauri narrative corpus is as a Shoebox database (Buseman and Buseman, 1998) (see figure 4), from which a web version in HTML hyperlinked to a morpheme lexicon was semiautomatically derived (see figure 2); (2) The Stockholm Umeå Corpus (SUC; Ejerhed and Källgren (1997)) comes in an SGML corpus format as specified by the Text Encoding Initiative (TEI; `http://www.tei-c.org/`), and further, there are two different grammatical annotation formats, Parole/EAGLES format (see Monachini and Calzolari (1996)) and SUC format (see figure 3); (3) The Talbanken syntactically annotated corpus of Swedish (Einarsson, 1976a; Einarsson, 1976b; Teleman, 1974) is in an 80-column punch card format with only capital letters (see figure 5).

```
\ref 07/007a/01
\tx @ma r@N boa loshigyO //
\mrep @ma r@N bOba lo-sh-i-gyO
\gl mother with father say-?-?-D.PST
\tr Mother and father said:

\ref 07/007a/02
\tx jO tshEtsats-u nam@N ch@ tate //
\mrep jO tshEtsats-u nam@N ch@d ta-te
\gl this girl-POSS name(N) what keep-LET'S
\tr "what should we name this girl?

\ref 07/007a/03
\tx nam@N t@ sOthlets tate //
\mrep nam@N t@ sOthlets ta -te
\gl name(N) EMP name keep-LET'S
\tr Let's keep the name (=name her) Sothlets."
```

Figure 4: The Kinnauri corpus – Shoebox format

- Differences in POS tagging and syntactic annotations between corpora. The SUC and Talbanken Swedish corpora, although both are POS tagged, use different tagsets, with e.g. SUC having two and Talbanken three subclasses of nouns, and SUC, but not Talbanken, marking number in nouns, etc. Tagset incompatibilities, even within a language is a problem that has been noted in the literature (e.g. by Atwell et al. (2000)), and there has been some work on tools for automatic tagset mapping (e.g. Teufel (1995)). The problems are compounded when several languages are involved,[3] which would be desirable in our setting, where the linguistic subdisciplines of Contrastive Linguistics and Language Typology rely on explicit comparisons between languages at various linguistic levels. As stated above, we know from experience that students learn about grammatical constructions and phenomena more actively when these constructions are discussed by comparing the system found in their native language with that of another language. Preferably, the other language should be one that the students do not know already, as they then will be better able to concentrate on the analysis of 'pure' form. This is why we intend to use the Swedish and Kinnauri corpora together in our first application.

- Differences in POS categories, syntactic categories and grammatical framework between the corpora on

[3]The problem of crosslinguistic mapping of part-of-speech tags has not been extensively discussed in the computational linguistics literature (see Borin (2000); Borin (Forthcoming 2002); Borin and Prütz (2001)), but in general linguistics, there is an extensive literature on the issue of crosslinguistic properties of part-of-speech systems and the universality of proposed parts of speech, which is very relevant in this context (e.g., Anward et al. (1996); Itkonen (2001); Pawley (1993)).

```
P21803012001     0000                    <<      GM          010
P21803012002     *DET                    POOP    SS          010
P21803012003     RÖR                     VVPS    FV          010
P21803012004     SIG                     POXP  AAOO          010
P21803012005     ALLTSÅ                  ABKS    +A          010
P21803012006     OM                      PR      OAPR        010
P21803012007     FALL                    NN      OA          010
P21803012008     1000                    RC      OAET        010
P2180301200910002DÄR                     ABRA    RA          010
P2180301201010002ORSAKEN                 NNDD    SS          010
P2180301201110002TILL                    PR      SSETPR      010
P2180301201210002PATIENTENS              NNDDHHGGSSETDT      010
P2180301201310002SYMTOM                  NN      SSET        010
P2180301201410002INTE                    ABNA    NA          010
P2180301201510002PRIMÄRT                 AJ      AA          010
P2180301201610002ÄR                      AVPS    FV          010
P2180301201710002ÅDERFÖRKALKNING         VN  SS  SP          010
P21803012018100021100                    +F      +F          010
P2180301201911002UTAN                    ++MN    ++          010
P2180301202011002I                       ABMN    +A          010
P2180301202111002STÄLLET                 ID      +A          010
P2180301202211002BEROR                   VVPS    FV          010
P2180301202311002PÅ                      PR      OAPR        010
P2180301202411002EN                      EN      OADT        010
P2180301202511002SANNOLIK                AJ      OAAT        010
P2180301202611002STÖRNING                VN      OA          010
P2180301202711002I                       PR      OAETPR      010
P2180301202811002CIRKULATIONEN           VNDD    OAET        010
P2180301202911002AV                      PR      OAETETPR    010
P2180301203011002DEN                     PODP    OAETETDT    010
P2180301203111002VÄTSKA                  NN      OAETET      010
P21803012032110021110                    RC      OAETETET    010
P2180301203311106SOM                     PORP    SS          010
P2180301203411106OMGER                   VVPSSM  FV          010
P2180301203511106HJÄRNAN                 NNDD    OO          010
P21803012036     .                       IP      IP          010
```

Figure 5: The annotation format in the Talbanken treebank

the one hand and the grammar writing tools and parsers on the other. Thus, the Talbanken corpus uses a fairly traditional Swedish functional grammatical framework, where e.g. NPs are not directly recoverable, but only indirectly, through a combination of syntactic function and lexical category of the head word, while it seems that many, perhaps the majority, of the grammar writing tools freely available on the Web presuppose a phrase structure framework.

- Differences in implementation language, storage model, API, documentation and source code availability, etc. of potentially suitable software. For an excellent overview of these issues, see Olsson (2002).

Thus, we have been forced from the outset to discuss seriously how we are to integrate existing NLP resources in our application, as well as how to make the application itself extensible, so that e.g. new language corpora or new annotations can be added.[4]

2. Taking Stock and Looking Ahead

We are attempting to reuse NLP resources originally meant for NLP research—both *language resources* (notably annotated text corpora) and *processing resources* (the most important being parsers and grammar writing tools)—in an e-learning application for IT-based collaborative learning in grammar courses for Linguistics and Computational Linguistics university students. At the moment, we are locating and evaluating[5] NLP resources, mainly on the web, for the corpus-based interactive grammar exercises. As the corpora are in place already, we are now evaluating tools for the manipulation and visualization of corpus data, parsing systems, and grammar writing environments (workbenches), which raises a number of compatibility/standardization issues that need to be resolved. These compatibility/standardization issues point in two directions simultaneously, as it were:

1. backwards: How can we integrate in our application, with the least amount of effort, existing NLP resources of the kind that we need?

2. forwards: How can we ensure that we ourselves, as well as others, will be able in the future to modify the existing NLP resources, or add new ones, in the framework that we define?

The preliminary answers to these two questions are as follows.

There does not seem to be a simple answer to the first question. Generally, we think that it is more desirable to be able to reuse existing language resources—i.e., texts and corpora, lexicons, and the like—than processing

[4]Courses in Hindi and Turkish at Uppsala University will be used as testbeds during the third year of the project, based on relevant Hindi and Turkish corpus resources.

[5]The evaluation is to be mainly pedagogical, i.e. we will ask ourselves whether a particular resource will be suitable for the pedagogical framework that we have adopted for teaching grammar. However, usability—as the term is used in Human–Computer Interaction research—will also be an important evaluation criterion, as well as the the estimated effort needed to adapt the resource for our needs. See Hammarström (Forthcoming 2002) for details.

resources—in our case first and foremost grammar writing and processing environments—for the pragmatic reasons that

- constructing an annotated corpus from scratch is likely to be a much larger effort than building a grammar writing environment;

- standardization efforts have progressed further particularly in the realm of POS tagged language corpus resources than in the case of language processing resources (Monachini and Calzolari, 1996; Bird et al., 2000; Ide et al., 2000; Cotton and Bird, 2002) (and treebank formats; see Atwell et al. (2000)), although, as a rule, their use in computer-assisted language learning applications has not been considered in this connection (Borin, 2002).

Hence, we aim at being able to handle at least POS-tagged corpora using the EAGLES/Parole tag scheme and marked-up according to the TEI/CES SGML or TEI/XCES XML language corpus formats (thus recognizing, e.g., the SUC Parole format without special preprocessing).

As for the second question, it too, is easier to answer for language resources. Here, we will harmonize the underlying corpus formats with other ongoing projects in our departments,[6] while simultaneously endeavoring to conform to standards that are being worked out in the NLP community. This means that we will undertake the conversion of the Kinnauri and Talbanken corpora into this format, and that in due course we plan to make the corpora generally available in the new format.

As far as 'grammar writer's workbenches' are concerned, we have not yet been able to find a ready-made environment user-friendly enough (for our Linguistics students) and bug-free enough to be immediately useful for our purposes. Thus, it seems likely that we will have to put in some development effort in this area. If this turns out to be the case, the most likely kind of workbench that we will modify or build, will be one within the general paradigm of unification-based feature structure grammar. The evaluation of these systems is still ongoing, however (Hammarström, Forthcoming 2002).

3. Acknowledgements

The work described here forms part of the project *IT-based Collaborative Learning in Grammar*, a collaboration between the universities in Uppsala and Stockholm, funded by the Swedish Agency for Distance Education (DISTUM), for the three years 2002–2004. Anju Saxena is the principal investigator for the project. See also `http://www.ling.uu.se/anjusaxena/distum.html`.

[6]We will strive to be compatible with the corpus format developed in the CrossCheck (`http://www.nada.kth.se/theory/projects/xcheck/`), Svante (`http://www.ling.uu.se/lars/SVANTE/`) and *ASU availability* projects, in all of which formats and tools for Swedish *learner corpora* (see Granger (1998)) are being developed. The basic corpus format will adhere closely to XCES, with 'standoff' linguistic annotation (Ide et al., 2000).

4. References

Jan Anward, Edith Moravcsik, and Leon Stassen. 1996. Parts of speech: a challenge for typology. *Linguistic Typology*, 1(2):167–183.

Eric Atwell, George Demetriou, John Hughes, Amanda Schiffrin, Clive Souter, and Sean Wilcock. 2000. Comparing linguistic annotation schemes for English corpora. In Anne Abeille, Torsten Brants, and Hans Uszkoreit, editors, *Proceedings of the Workshop on Linguistically Interpreted Corpora. LINC-2000*, pages 1–10. Held at the Centre Universitaire, Luxembourg, August 6, 2000.

Steven Bird, David Day, John Garofolo, John Henderson, Christophe Laprun, and Mark Liberman. 2000. ATLAS: a flexible and extensible architecture for linguistic annotation. In *Proceedings of LREC 2000*, pages 1699–1706, Athens. ELRA.

Lars Borin and Mats Dahllöf. 1999. A corpus-based grammar tutor for Education in Language and Speech Technology. In *EACL'99. Computer and Internet Supported Education in Language and Speech Technology. Proceedings of a Workshop Sponsored by ELSNET and The Association for Computational Linguistics*, pages 36–43, Bergen. University of Bergen.

Lars Borin and Klas Prütz. 2001. Through a glass darkly: Part of speech distribution in original and translated text. In Walter Daelemans, Khalil Sima'an, Jorn Veenstra, and Jakub Zavrel, editors, *Computational Linguistics in the Netherlands 2000*, pages 30–44. Rodopi, Amsterdam.

Lars Borin. 2000. Enhancing tagging performance by combining knowledge sources. In Gunilla Byrman, Hans Lindquist, and Magnus Levin, editors, *Korpusar i forskning och undervisning. Corpora in Research and Teaching*, pages 19–31, Växjö Universitet, Växjö. ASLA, ASLA.

Lars Borin. 2002. Where will the standards for intelligent computer-assisted language learning come from? In *Proceedings of LREC 2002 workshop on International Standards of Terminology and Language Resource Management*. To appear.

Lars Borin. Forthcoming 2002. Alignment and tagging. In Lars Borin, editor, *Parallel Corpora, Parallel Worlds. Selected Papers from a Symposium on Parallel and Comparable Corpora at Uppsala University, Sweden, 22–23 April, 1999*. Rodopi, Amsterdam.

Alan Buseman and Karen Buseman, 1998. *The Linguist's Shoebox for Windows and Macintosh.* Summer Institute of Linguistics, Waxhaw, North Carolina:.

Scott Cotton and Steven Bird. 2002. An integrated framework for treebanks and multilayer annotations. In *Proceedings of LREC 2002*, Las Palmas. ELRA. To appear.

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

Jan Einarsson. 1976a. Talbankens skriftspråkskonkordans. Corpus on CD-ROM.

Jan Einarsson. 1976b. Talbankens talspråkskonkordans. Corpus on CD-ROM.

Eva Ejerhed and Gunnel Källgren. 1997. Stockholm Umeå

Corpus version 1.0, SUC 1.0. Department of Linguistics, Umeå University.

John Flowerdew. 1996. Concordancing in language learning. In Martha C. Pennington, editor, *The Power of CALL*, pages 97–113. Athelstan, Houston, Texas.

Sylviane Granger, editor. 1998. *Learner English on Computer*. Longman, London.

Harald Hammarström. Forthcoming 2002. Overview of IT-based tools for learning and training grammar. Project report, IT-based Collaborative Learning in Grammar. Department of Linguistics, Uppsala University.

Nancy Ide, Patrice Bonhomme, and Laurent Romary. 2000. XCES: an XML-based encoding standard for linguistic corpora. In *Proceedings of the 2nd International Conference on Language Resources and Evaluation (LREC2000)*, pages 825–830, Athens. ELRA.

Esa Itkonen. 2001. Concerning the universality of the noun vs. verb distinction. *SKY Journal of Linguistics*, 14:75–86.

Monica Monachini and Nicoletta Calzolari. 1996. Synopsis and comparison of morphosyntactic phenomena encoded in lexicons and corpora. a common proposal and applications to European languages. EAGLES Document EAG-CLWG-MORPHOSYN/R.

Leif-Jöran Olsson and Lars Borin. 2000. A web-based tool for exploring translation equivalents on word and sentence level in multilingual parallel corpora. In *Erikoiskielet ja kännösteoria – Fackspråk och översättningsteori – LSP and Theory of Translation. 20th VAKKI Symposium*, pages 76–84, Vaasa, Finland. University of Vaasa.

Fredrik Olsson. 2002. *Requirements and Design Considerations for an Open and General Architecture for Information Refinement*. Number 35 in Reports from Uppsala University, Department of Linguistics, RUUL. Uppsala University, Department of Linguistics.

Andrew Pawley. 1993. A language which defies description by ordinary means. In W. A. Foley, editor, *The Role of Theory in Language Description*, pages 87–129. Mouton de Gruyter, Berlin.

Anju Saxena. 2000. Corpora of lesser-known languages on the internet: A pedagogical tool for the teaching of syntax. Paper presented at the workshop on IT inom språkundervisningen. Uppsala University. `http://www.ling.uu.se/anjusaxena/symposium0303.html`.

Ulf Teleman. 1974. *Manual för grammatisk beskrivning av talad och skriven svenska*. Liber, Lund.

Simone Teufel. 1995. A support tool for tagset mapping. In *Proceedings of SIGDAT 1995. Workshop in connection with EACL 95*, Dublin. Association for Computational Linguistics.

Sean Wallis and Gerry Nelson. 2000. The FTF home pages. WWW: `http://www.ucl.ac.uk/english-usage/ftfs/faqs.htm`. Accessed on 10 April 2002.