Recursive Top-down Fuzzy Match: New Perspectives on Memory-based Parsing

Oliver Streiter
CKIP, Institute of Information Science, Academia Sinica
Taipei, Taiwan
oliver@iis.sinica.edu.tw

Abstract

With treebanks becoming available for more and more languages, their usage for the development of natural language parser has become a topical issue in NLP. This paper tries to give a new spin to this this stream of research, proposing a new direction in corpus-based parsing. Contrary to competitive approaches, this approach does not involve a chart parser which reassembles phrases extracted form a treebank. Instead, parsing proceeds via the extraction of example trees from the treebank using fuzzy pattern matching techniques. A set of adaptation rules modify the extracted example trees so as to produce the best possible parse given the current set of examples.

1 Introduction

With more and more computerized corpora becoming available, the development of techniques which allow to compile a corpus automatically into a running NLP-tool has been established, together with the creation of such corpora, as a central issue in NLP. Corpora with syntactic annotations are commonly referred to as *treebanks* and are developed for more and more languages. One way to benefit from these treebanks is to compile them into parsers for that language, which, as first attempts have shown, tend to outperform parsers working on hand-written rules. Nevertheless, these grammar-derived parsers share with their hand-crafted counterparts a number of drawbacks such as their inefficiency and unrecoverable generalizations. Trying to avoid these drawbacks we investigate an alternative parsing strategy, using for training and testing the Chinese CKIP-treebank (Chen et al., 1999).

2 Competitive Parsing Approaches

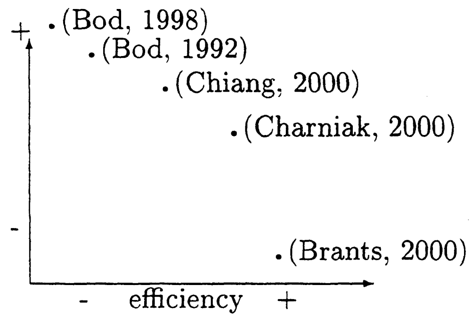
Since the first appearance of treebanks, there have been attempts to use these resources for parsing. One way to do so is to convert the subtrees represented in the treebank into Stochastic Phrase Structure Grammars (Charniak, 1996). Recent publications have shown that this approach can be improved upon with respect to the efficiency (Brants, 2000) or the degree of lexicalization (Collins, 1996; Charniak, 2000). Both directions hint to some basic problems with this approach. As for the lexicalization, the basic model in (Charniak, 1996) contains no lexemes: They are removed during the extraction of rules and parsing is performed not on a sequence of words but on a sequence of part-of-speech tags. On the one hand such parsers require a part-of-speech tagger, on the other hand, the lack of lexicalization is detrimental for the parsing accuracy whenever relevant information cannot be expressed by the part-of-speech (Bod, 1999; Streiter, 2000). Collins and Charniak however could show that lexical dependencies can be handled in SPSGs in the form of of probabilities on head-dependents relations. The limitations of this approach however are obvious as not even the head-dependents relations can be described satisfyingly since there are headless structures, structures with multiple heads and structures containing at the same time syntactic and semantic heads. In addition, there are still different lexical dependencies which are important for parsing (Doi, 1992; Streiter et al., 1999).

The absence of lexicalization is part of a more general problem, referred to as *unrecoverable generalizations*. Such generalizations are manipulations of the training data which allow to license unseen language events and are used instead of the original data. Experiments have shown that unrecoverable generalizations reduce the performance of NLP-tools as natural languages are mixtures of regular and irregular phenomena, the latter of which by no means need to be frequent (Daelemans et al., 1999; Bod, 1999; Streiter, 2000). In addition to the deletion of lexemes, breaking examples down into phrases also may introduce unrecoverable generalization if the knowledge of the functioning of the phrase in a specific example is transformed into probabilities on the functioning compiled over the whole corpus.

Corpus-derived lexicalized probabilistic Tree Adjoining Grammars (Chiang, 2000) share the same flaws, although they have the advantage that the elementary trees can span a larger part of the sentence and introduce thus less unrecoverable generalizations. It is thus not surprising that TAGs may outperform SPSG (Bikel and Chiang, 2000). However, the greater context sensitivity of TAGs may make them run slower than SPSG.

Data Oriented Parsing approaches (Bod, 1992; Bod, 1998) do not share the flaw of unrecoverable generalizations: Language examples are decomposed into all subcomponents, including trees with and without lexemes, with and without their features in all possible combinations. During parsing, all (or many) re-compositions are computed and the most frequently produced parse tree is selected as final parse. This approach has the great advantage that large lexicalized trees are retained and that each generalization is accompanied by the original data. However, as with hundreds of thousands of overlapping rules and with all (many) possible parses (even identical) computed, the approach is extremely inefficient (Manning, 1999).

Figure 1: Efficiency (x-axis) versus recoverability of generalizations (y-axis), a hypothetic classification of competitive corpus-based parsing approaches.



3 Basic Outline of the Parser

In order to avoid the flaws of unrecoverable generalizations and the low efficiency, we attempted not to rely on a classic parsing algorithm, but to employ the k-nearest neighbor classifier, an approach underlying paradigms as *Example-Based Machine Translation*, *Translation Memories* and *Case-Based Reasoning*: According to this paradigm, a new problem is approached by retrieving similar problem formulation from a data-base together with their associated solutions. Retrieved old solutions are then *adapted* to the new problem formulation.
Transferring this approach to the task of parsing, the problem formulation is an input sentence to be parsed and the problem solutions are stored example trees. In the current implementation we use two main strategies in order to relate an unseen input sentence to a store example tree: *generalizations* and *fuzzy matching*. The *adaptation* consists in modifying the retrieved example trees there where they do not match the input sentence.

Generalizations are compilation-time operations on the examples which license unseen language events. Generalization may consist of inductions, i.e. replacing tokens by types (Brown, 1999a; Carl, 1999), or abductions, i.e. creating examples different from the given examples (Bod, 1992). Due to the off-line operation, the example trees can be examined and generalized quite accurately. This may cause the compilation to slow down. However, the generalizations may become part of the indexing system and, as a consequence, matches can be performed efficiently. We use generalizations to handle **unseen structures**.

Fuzzy Matches license unattested input on-line. This causes the matching process to be more complex and slower. In addition, unlike for generalizations, the internal structure of the example to be retrieved is not necessarily fully available (there may be too many of them to be checked on-line) and the decision on the licensing of unseen events has to be made on the basis of the less informative but easily accessible properties of input and example. Fuzzy Matches however offer a greater flexibility in the matching of **unseen labels**, as could be shown by comparing generalizations and fuzzy matches of syntactic categories in a parsing task (Streiter, 1999). We can expect the same advantages with respect to the handling of lexical relations: Generalizations have to specify off-line which relations have to be modeled. Fuzzy matches do not require such a model and license the input due to all occurring lexical similarities. A model however can be helpful in order to prefer one match over another.

4 The Parser

The parser consists of two main modules, a training module and a parsing module. Both are implemented as independently operating TCP/IP servers which access identical databases. Due to the nature of the parser, the training module can update the databases while simultaneously the parsing server is serving parsing clients.

4.1 Training

The training consists of two successive runs through the treebank which aim at the (*i*) collection of basic statistical data for the adaptation, (*ii*) the generalizations of the examples, (*iii*) the calculation of information weights and the (*iv*) indexing of the examples. This latter includes (a) the collection of data for the guessing of the parsing axiom, (b) the collection of data for the keywording and (c) the calculation of index positions. The working of the training module will be illustrated with the help of the example tree in Fig. 2.

Figure 2: Example tree to be trained: *he think older_brother certainly can speak.*

```
S(exper:Nep:ta1,Head:VE2:xiang3,goal:S(
  agent:Nap:ge1ge,epist:Dbaa:yi2ding4,epist:Dbaa:hui4,Head:VE2:shuo1))
```

4.1.1 Adaptation Data

Basic statistical data are compiled from the treebank aiming at the support of simple adaptation strategies (the structural adaptations referred to below). These are conditional probabilities on the semantic role and part-of-speech given a set of possibly contexts. Illustrated with the first word of the running example in Fig. 2, the following conditional probabilities are calculated: `P(exper|ta1,left_to_VE2)`, `P(Nep|ta1,left_to_VE2)`, `P(exper|Nep,left_to_VE2)`, `P(Nep|exper,left_to_VE2)`.

4.1.2 Generalization of Examples

In previously conducted experiments we have found that best results can be achieved when combining the advantages of generalizations and fuzzy matches. The generalizations we perform in the current model create examples with **unseen structures** with the help of two operation. First, the operation SUB adds all subtrees contained in the original example trees to the set of example trees, i.e. all NPs, VPs, PPs etc are stored as independent tree. The second operation DROP removes optional words and phrases from the examples in all possible combinations.

Each newly created example is added to the set of examples as illustrated in Fig. 3. As one original example may yield thousand of differently structured example trees, we restrict this generalization with a stop-value. Currently an average of 20 additional trees are generated from one example tree. Together with the trees we store their distance from the original tree in order to estimate their trustworthiness.

Figure 3: Abduction of new examples during compilation via the operations SUB and DROP. The distance Δ to the original example is used to calculate their trustworthiness.

original:S(ta xiang gege yiding hui shuo):$\Delta = 0$	SUB(0):S(gege yiding hui shuo):$\Delta = 1$
DROP(0):S(ta xiang yiding hui shuo):$\Delta = 2$	DROP(1):S(yiding hui shuo)):$\Delta = 7$
DROP(0):S(ta xiang gege yiding shuo):$\Delta = 3$	DROP(1):S(gege yiding shuo):$\Delta = 8$
DROP(0):S(ta xiang gege hui shuo):$\Delta = 4$, (...)	DROP(1):S(gege hui shuo):$\Delta = 9$, (...)

4.1.3 Indexing of the Example Trees

All example trees (the original and the generalized trees) are indexed such that an efficient fuzzy match can be performed between the huge set of examples and the input sentence.

Inverted Indices The indexing technique we use is the so-called *Inverted List Index.* An Inverted List Index uses the content of the database/record as index. These indices point back to the list of databases/records they occurred in. Using Inverted List Indices allows to manipulate the references to the records (e.g set operations over lists, estimation of the relevance of list members etc...) without touching the records themselves. In NLP-related domains, this technique is used for full-text indexing or for the indexing of sentences in EBMT (Brown, 1999b). In our implementation, the word-related indices (these are the the lexemes as *ta1* and the parts-of-speech as *Nhaa*) point to the list of all those example trees these indices have occurred in. While the calculation of the intersection of the list of all indices of an input sentence gives the exact matching example (possibly empty), finding the list members with most matches implements a fuzzy match. It is this latter strategy we follow here.

Parsing Axiom The inverted indices (the lexeme and the part-of-speech) are further specified by the parsing axiom. The index is thus *<ta1,S>* instead of simply *ta1*. These parsing axioms can be taken easily during compilation from the example trees. This however implies for the later parsing that the axiom has to be known in order to build up correct indices. Therefor, we store during the compilation the sequences of the parts-of-speech of those example trees which are not sentences (S is the default axiom). In case of ambiguities, the last part-of-speech is iteratively replaced by the lexeme until the sequence becomes unambiguous. For a NP, we may store a sequence like `VH11,DE,Nap=>NP` or, in case of ambiguities, `VH11,DE,ge1ge=>NP`.

Keyworded Indices As we intend to build a parser which is not affected by the fact that hundreds of thousands of example trees are searched through, we implemented an indexing system that increases with the size of the training data. At this aims we employ sets of key-words which are distinctive for a small number of structures. The key-words of a sentence are those words of the sentence which have frequently co-occurred at the top-level of a number of example sentences with the same axiom. Thus, the sentence *zhe4 shi4 ta1 de ma* may obtain the keywords "=shi=ma", provided "=shi=ma" occurred with a frequence above a threshold in example sentences with the same axiom. Here, "=" refers to a position which can be filled by one or more words.

As the words of a sentence cannot be checked efficiently in all combinations in order to find the keywords, we apply the following algorithm. First, we count how often a word occurs at the top-level of structures with the axiom Y. Then we cyclically remove the least frequent word of a sentence (for the axiom Y) and check whether the remaining words occurred in this order in a frequence above the threshold for this axiom. If so, the indices are enriched by the obtained keywords, if not, one more word is removed.

The additional indexing improves the performance (not only in time) if the search-space is limited correctly, similarly to document clustering in Information Retrieval (Rijsbergen, 1979), or the parsing experiments reported in (Kim, 1995). However, this technique is not error-free. As we do not know before parsing which words of the input sentence will actually appear at the top-level and are thus legitimate candidates of keywords, only the counting of the frequencies for a given axiom during compilation is done on the top-level. Based on these frequencies, example trees (during compilation) and input sentences (during parsing) are keyworded in the same way based on their surface form, risking keywords which are not at the top-level.

Figure 4: Inverted List Indices for some words of the example tree in Fig. 2. The quadruples <*index,axiom,keywords,position*> point to the <*list of* <*tree-reference,weight*>*-tuples* >.

```
<Nep,S,=shuo1,=1> => <<347,1/6> <515,1/8> <899,1/26>>
<ta1,S,=shuo1,=1> => <<347,1/6> <699,1/14>>
(...)
<ge1ge,S,=shuo1,=1> => <<347,1/24>>
<Nap,S,=shuo1,=1> => <<347,1/24> <899,1/25>>
(...)
<shuo1,S,=shuo1,1> => <<347,1/24> <456,1/72> <457,1/80> <568,1/34> <644,1/75>>
```

Positioned Indices Inverted List Indices as we use them here to retrieve example trees are in principle position-independent and as such an good indexing mechanism for free word-order languages like Russian (although the words of different phrasal levels should not be confused). Word order in Chinese is less free and therefor Chinese may not be well suited for a position-less indexing. In addition, position-less matching requires complex adaptation strategies which have not been investigated until now. Therefor we constrain the position of the indices by reference to the keywords. Thus "=" refers to a position between the beginning and the end of a sentence when no keywords are found, "=1" refers to a position between the beginning of a sentence and the first keyword, "1=2" refers to a position between the first and second keyword and "1" refers to the position of the first keyword. With a growing number of examples, the keywording and the positioning becomes automatically more specific.

4.1.4 Deriving Weights

As during parsing many similar example trees may be retrieved (only the minimum is limited by the threshold defined above), criteria for the selection of the best example tree are required. We therefor assign weights to the lexemes and part-of-speech of each word of an example tree which indicate their importance for this structure. The weights we calculated are first added to the inverted indices (see Fig. 4) and secondly compiled into the example trees (for a more fine-graded evaluation during the alignment described in Section 4.2.2). Due to the keywording which already identifies the most crucial words of a sentence, the further assignment of weights can remain very simple: We assume every sentence to have approximately the same total weight close to 1. An additional weight is given to longer sentences as short sentence are easily matched. The distance Δ from the original tree is used in order to calculate the trustworthiness of the examples (cf. Fig. 3). The weight of one level is distributed equally over the daughters of that level with lexemes and parts-of-speech obtaining the same weight.

$$weight\_of\_tree = \frac{1 + log(sentence\_length)}{1 + log(\Delta)} \tag{1}$$

Figure 5: The weight of a tree is included in a stored example trees. The 3 daughters of the main phrase obtain each $\frac{1}{3}$ which has to be distributed equally among lexeme and the part-of-speech. The weight $\frac{1}{3}$ of a phrasal daughter is distributed equally over the grand-children.

```
S(exper:Nep:1/6:ta1:1/6,Head:VE2:1/6:xiang3:1/6,goal:S(
  agent:Nap:1/24:ge1ge:1/24,epist:Dbaa:1/24:yi2ding4:1/24,
  epist:Dbaa:1/24:hui4:1/24,Head:VE2:1/24:shuo1:1/24))
```

4.2 Parsing

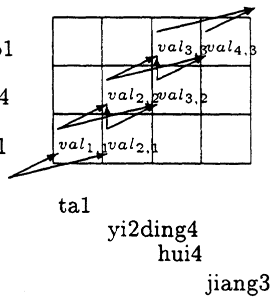
4.2.1 Fuzzy Nearest Neighbor Retrieval

Parsing starts with a lexicon look-up. The sequence of lexemes and possible parts-of-speech allows to look up the axiom (cf. Section 4.1.3). Co-occurrences of lexemes for this axiom above the threshold identify the keywords in the same way as during compilation. The keywords allow for a positioning of the indices as during the compilation phase (the left side of Fig. 4). With the help of the resulting index the database is accessed and lists of tuples of *<tree-reference,weight>* retrieved. The weights are summed up for every tree-reference. k tree-references which accumulate the highest sum of weights are retained for further processing. The retained k tree-references are then used to retrieve the example trees.

4.2.2 Alignment

After the NN-Retrieval the k example trees and the input sentence are aligned. This can be conceived of as a refinement of the retrieval as with this limited set of retrieved examples a more precise evaluation is possible. As the input sentence may have more words than the example sentence (we face a *deletion*), the best mapping of the words of the input sentence and the positions in the example tree has to be found. At this aims dynamic programming strategies can be used: Imagine the words of the input sentence to be plotted on the x-axis and the slots in the example tree to be plotted on the y-axis. We first determine the "envelope", i.e. all possible combinations of x and y. Within this envelop every cell is filled with a score.

Figure 6: The alignment of input sentence and example tree using dynamic programming. The score is more precise than that obtained from the NN-retrieval.



This score combines the similarity of the indices of the example tree and the input sentence, and the weight assigned to them. The alignment of the input sentence and the example tree in Fig. 6 consists of finding the path through this lattice which accumulates the highest sum of scores. By storing in a hash table partial best paths (e.g. the best path starting from (3,2) to the end), not all possible paths have to be run through, but can be calculated by summing up partial results (Viterbi-Algorithm). The last step of the alignment consists of the replacement of the words of the best example tree by the words of the input sentence. This is the first *adaptation* step. The parts-of-speech of the example tree are retained for the next processing steps, as the new coupling of the words and the part-of-speech allows to identify mismatches.

4.2.3 Insertion of Unaligned Words

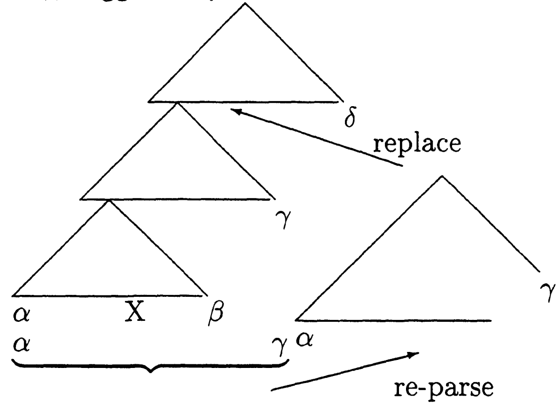
Unaligned words are inserted in a second adaptation step. By default deleted words are inserted into the deepest embedded structure, unless there is strong statistical relation to the head-word of the highest level. If the word is correctly inserted at the deepest level, the derivational adaptation described in 4.2.4 can correct the position and encoding of this word.

4.2.4 Derivational Adaptations

This adaptation step is triggered by a mismatch, defined as a word of the input sentence which is aligned to a part-of-speech in the example tree which has not be found with this word in the learned corpus. Subtrees in which such mismatches occur are replaced with trees obtained by the recursive application of the whole parsing procedure to the words of the subtree.

As the phrase to be re-parsed is shorter than the input sentence, we are likely to obtain a better match, unless the initial chunking is wrong. The largest possible sub-tree is chosen for re-parsing in order to have a large context for the unmatched word and, secondly, to correct possible errors in the surrounding of the mismatch. Mismatches at the sentence top level cannot be corrected by this adaptation strategy. The last adaptation, the *structural adaptation*, is applied to these words.

Figure 7: Re-parsing of the subtree from α to γ, triggered by a mismatch in X.



4.2.5 Structural Adaptations

The final *structural adaptations* operate on single words. They handle erroneous word mismatches, unknown words and phenomena of type shifting. Structural adaptations are triggered by the same unattested relations of input word and part-of-speech which also trigger the re-parsing. If the word is unknown, the aligned part-of-speech is maintained as a kind of top-down unknown word guessing. In the case of a mismatch, either the mismatch is maintained (the shifting of a verb to a noun) or the mismatch is attempted to be corrected by assigning the words most likely part-of-speech and semantic role (given the part-of-speech and the position of the head-word, see Section 4.2.). If the aligned part-of-speech and the parts-of-speech found in the learned corpus are similar, the example part-of-speech is replaced by the attested part-of-speech. The semantic role of the example tree is maintained in that case (assuming that it is compatible with the new part-of-speech). If the example part-of-speech is very different, the most probable new part-of-speech and semantic role is searched for, using the probabilities collected during compilation (Section 4.2).

5 Implementation

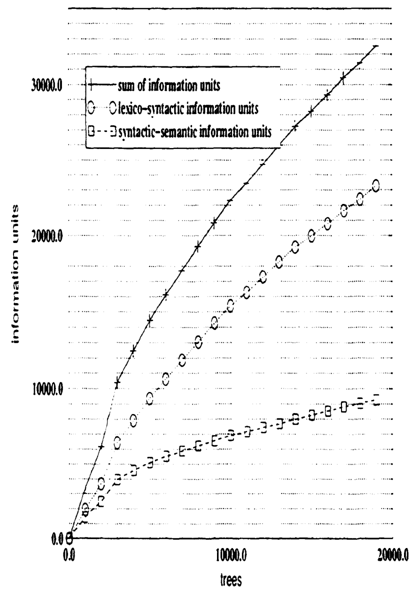
The outlined parsing approach has been been implemented in a Chinese parser called OCTOPUS, using for training and testing the treebank developed at the Academia Sinica in Taiwan for Modern Mandarin Chinese (Chen et al., 1999). The annotation guidelines for the treebank and a sample of 1000 trees can be found at `http://godel.iis.sinica.edu.tw/CKIP/`. While the semantic role labels are almost self-explaining, part-of-speech tags are more complex: Tags starting with `N` refer to nouns, starting with `V` refer to verbs, starting with `P` refer to prepositions etc. Additional characters develop a finer classification e.g. VK1 is a subset of VK which is a subset of V. The current specifications comprise almost 200 part-of-speech tags, 45 phrasal labels and 46 semantic role labels.

The parser is written in Perl and has been developed under Linux. With minor changes the parser may run also under commercial operating systems. The parsing module and the learning module are a multi-tasking servers which can be accessed via TCP/IP. A demo-system and a download of a parsing-client can be found under http://rockey.iis.sinica.edu.tw/oliver/parser.

6 Evaluation

The evaluation of NLP-tools should serve two aims. First, it should help the researcher to develop better NLP-tools. Secondly, the evaluation should allow for the comparison among tools and approaches. However, the current standard evaluation approach which consist in testing on unseen 10% of a training corpus may serve neither of these aims, as it does not distinguish the system quality from the representativeness of the training data for the testing data. That this representativeness cannot be taken for granted can be seen, for example, from the CKIP treebank we use for training and testing (Chen et al., 1999). When plotting the occurrence of new lexical-syntactic information units (a word and it syntactic category) and syntactic-semantic information units (stating that category X at the left/right of the syntactic head-category Y has the semantic relation Z) we can see easily that, even though the sum of these information units is a very poor language model for Chinese, at no time the training data can be representative for the testing data. Therefor, additional evaluation techniques are required which allow to abstract away from the representativeness of the training data. By this, the strong and weak sides of the NLP model can be identified more clearly and the comparison of NLP tools across different training data should become possible.

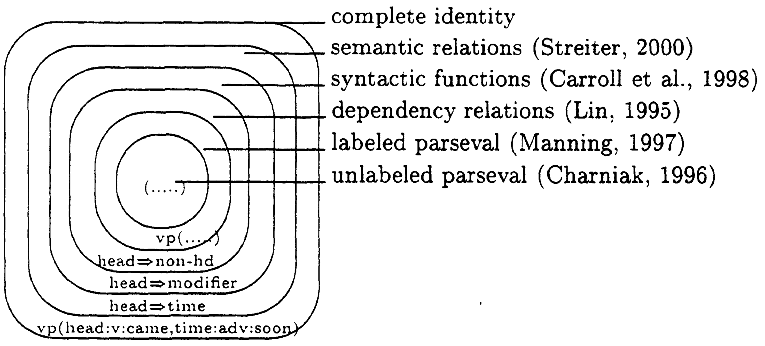
Figure 8: The closure of the CKIP-treebank.



6.1 The Evaluated Units

In order to have an as informative as possible evaluation which allows for the comparison among parsers which reproduce parses with a different degree of annotation richness, we opt for a multiple evaluation which applies to most annotation levels represented in Fig. 9. As syntactic functions are not annotated in the treebank we use for training, we evaluate the parser for the unlabeled parseval, the labeled parseval, the dependency relations and the semantic role relations. The scores for

Figure 9: A hypothetical hierarchy of increasingly difficult evaluation units for parser evaluation.


complete identity
semantic relations (Streiter, 2000)
syntactic functions (Carroll et al., 1998)
dependency relations (Lin, 1995)
labeled parseval (Manning, 1997)
unlabeled parseval (Charniak, 1996)
(.....)
vp(.....)
head⇒non-hd
head⇒modifier
head⇒time
vp(head:v:came,time:adv:soon)

syntactic functions may be interpolated using the above hierarchy. The evaluated units for the example tree in Fig.2 are given below.

Figure 10: Examples of evaluated units for $Tree_2$.

unlabeled parseval	(ta...shuo), (gege...shuo)
labeled parseval	S(ta...shuo), S(gege...shuo)
dependency relations	xiang->ta, xiang->shuo, shuo->gege, shuo->yiding, shuo- hui
semantic role relations	xiang-exper->ta, xiang-goal->shuo, shuo-agent->gege, shuo-epist->yiding, shuo-epist->hui
identity	S(exper:Nep:ta,Head:VE2:xiang,goal:S(agent:Nap:gege,= epist:Dbaa:yiding,epist:Dbaa:hui,Head:VE2:shuo))

6.2 The Basic Measures

Dividing the number of correctly identified evaluated units by the number of evaluated units in the reference corpus, we obtain the *recall.* Dividing the number of correctly identified evaluated units by the number of evaluated units in the parsing output we obtain the *precision.* Both scores are combined into the *f-score* via the following formula.

$$f\text{-}score = 2\cdot\frac{precision\cdot recall}{recall + precision} \tag{2}$$

6.3 Secondary Measures

From the basic measures (recall, precision and f-score) we derive secondary measures by running the parser on different sorts of test corpora. These secondary measures are the *coverage*, the *generality* and the *reliability.*

6.4 Coverage

NLP-tools derived from corpora share a number of properties which may make them preferable to their rule-based counterparts. Among them, their low development costs, their maintainability, their tunability to a subject domain, their robustness and often, but not necessarily, their quality in output. These advantages may be traded for the *coverage*, defined either as *"the range of texts or text types a (...) system can handle without deteriorating its performance"* (Carl et al., 2000) or as *"f-score with (...) unlearned test corpora"* (Streiter, 2000). This limitation in coverage is due to the fact that for all corpus-derived tools the training corpus S^l_{ex} is only a subset of all possible realizations S^l_{tot} in language l. As has been pointed out, this evaluation technique does not allow to determine the qualities of the corpus-based system as it includes the representativeness of S^l_{ex} for S^l_{tot}. Nevertheless we shall report on the 'traditional' coverage, defined as *f-score on* $S^l_{tot} - S^l_{ex}$. At this aims we trained 10.000 sentences and parsed the following 2.000 sentences.

6.5 Reliability

Beside the *coverage* of a system we evaluate the *reliability*, defined as *f-score on* S^l_{ex}. The reliability determines the extend to which instances of S^l_{ex} may be reproduced successfully. This is not a trivial task, even for corpus-based systems, due to retrieval strategies (Carl, 1999b) or statistical biases (Bod, 2000). In fact we hypothesize that most SPSGs perform poorly on this task and followers of this approach may therefor contest the usefulness of this evaluation. The necessity for a high reliability in a closed domain setting however cannot be denied. But even in an open domain setting, the reliability directly influences the coverage, as experimental

Figure 11: The coverage, reliability and generality of the parser expressed in recall, precision and f-score. The input sentences are not tagged for the part-of-speech.

	coverage (10K/2K)			reliability (3K/4K)			generality (10K/2K)		
	recall	prec.	f-sc.	recall	prec.	f-sc.	recall	prec.	f-sc.
identity	0.27	0.27	0.27	0.994	0.994	0.994	0.51	0.51	0.51
semantic roles	0.41	0.41	0.41	0.999	0.999	0.999	0.54	0.54	0.54
dependencies	0.48	0.48	0.48	0.999	0.999	0.999	0.60	0.60	0.60
labeled parseval	0.57	0.60	0.59	0.999	0.999	0.999	0.72	0.73	0.73
unlabeled parseval	0.63	0.67	0.65	0.999	1	0.999	0.78	0.79	0.78

data in (Daelemans et al., 1999; Streiter, 2000; Bod, 2000) show. While (Daelemans et al., 1999; Streiter, 2000) argue for a causal relation between the performance on S^l_{ex} (cause) and S^l_{tot} (result), (Bod, 2000) notices this relation only en passant. An additional advantage of the testing of the reliability relies in the possibility for the system developer to check the well-functioning of the parser, as it may be otherwise difficult to distinguish bad parses resulting from bad or missing examples from bad parses resulting from a system error.

In order to test the reliability of the parser, we trained 4.000 sentences and parsed the first 3.000 sentences of the training corpus. As the results show, the parser has almost 100% reliability. Among the sentences which have been parsed incorrectly, about 10% are due to different annotations of sentences occurring more than once. 90% of the errors are accounted for by the fact that the treebank contains unary projections. In such cases, the parser has no means to make a well-motivated choice among the different projections, e.g. chose between `v(v,v)` or `vp(v(v,v))`.

6.6 Generality

As a third system property we define the *generality* of a system as *f-scores on* $S^l_{mod} - S^l_{ex}$. S^l_{mod} is is a subset of S^l_{tot} and a superset of S^l_{ex}. S^l_{mod} contains those unseen language events which are conform to a language model instantiated by the data of S^l_{ex}. The language model we use here is the same we used to define the closure of the treebank above in Sec. 5. S^l_{exe} thus contains those language events we can reasonably expect a system to handle. However even on this task parser do not perform well for two reasons. Either a parser is lazy (as for example OCTOPUS) and enriches the internal language model not with all possible combinatorial power. OCTOPUS for example can not necessarily analyze the sentence *he came home* after having seen *he came* and *came home*. Lazy parsers may handle this and other sentences, but not necessarily the information unit. Eager parsers transform every information unit in the corpus into a highly potential information unit in the internal language model. As a consequence they have to handle the ambiguities which arise during the analysis and may fail for this reason (Charniak, 1996). When analyzing parsing errors on $S^l_{mod} - S^l_{ex}$ it is possible to identify whether these errors are due to the eagerness (the structure is there but not selected) or the laziness (the structure is not there) and to optimize the parser accordingly.

In order to show the generality of the parser, we trained 10.000 sentences and extracted from the following 10.000 sentences 2.000 sentences which are covered by the simple language model instantiated the the data from S^l_{exe}.

7 Summary and Conclusion

We summarized current attempts of corpus-based parsing and criticized statistically-based parsers for their inefficiency and unrecoverable generalizations. An alternative parsing approach has been developed for the parsing of Chinese, using recursive top-down fuzzy matches in order to extracts example trees from a treebank.

The parsing results differ from those of statistically-based parsers. The high scores on reliability and identity show that whenever there is an identical or similar example, the parser performs almost perfectly. If not, the performance values drop rapidly. What may seem a disadvantage includes the possibility of infinite trainability (that is what the reliability actually expresses). Given that 30.000 trees are currently available for training and an amount of 80.000 trees is aimed at, this property seems of utmost importance. Furthermore, as corpus-derived parsers should support further treebank development, parsing errors may be corrected by the training of a single example. A detailed study of this subject can be found in (Streiter, 2001).

For known or close to known language examples, the recall and precision are balanced. Only for unseen language events, there is a bias which reveals a tendency to (incorrectly) prefer flat structures. Currently, it is is not clear whether this is a result of the retrieval algorithm or of the way deleted words are inserted.

We further hypothesize that the evaluation in terms of *reliability* and *generality* as conducted here allows for a much better evaluation and tuning of a corpus-derived tool than the black-box evaluation of in terms of *coverage*. This conforms to our experience made during the development of OCTOPUS: Using the *coverage* to guide the system development leaves one with blind guesses how to improve the parser. The other evaluation techniques point directly to mistakes or weak points in the parser. For a related discussion cf. (Reinke, 2000).

Finally we express the conviction that the performance of the parser can be still improved upon in a modular way on many points such as the axiom-guessing, the keyword-finding, the assignment of weights, the conditions for re-parsing and, most important, the insertion of deleted words. At the same time the whole approach may benefit from integrating new insights from data mining, information retrieval and example-based Machine Translation.

References

Bikel, Daniel M. and David Chiang. 2000. Two statistical parsing models applied the the Chinese treebank. In M. Palmer, M. Marcus, A. Joshi, and F. Xia, *Second Chinese Language Processing Workshop*, Hong Kong, October. ACL.

Bod, Rens. 1992. Data oriented parsing (dop). In *COLING'92*.

Bod, Rens. 1999. Extracting stochastic grammars from treebanks. In *Journées ATALA sur les Corpus annotés pour la syntaxe*. Talana, Paris VII.

Bod, Rens. 2000. An empirical evaluation of LFG-DOP. In *COLING'2000*. pages 62–68.

Bod, Rens and Ronald M. Kaplan. 1998. A probabilistic corpus-driven model for lexical-functional analysis. In *COLING-ACL'98*. http://www/lfg.stanford.edu/lfg/lfg-dop.

Brants, Thorsten and Crocker Matthew. 2000. Probabilistic parsing and psychological plausibility. In *COLING'2000*. pages 111–117.

Brown, Ralf D. 1999a. Adding linguistic knowledge to a lexical example-based translation system. In *Theoretical and Methodological Issues in Machine Translation-99*.

Brown, Ralf D. 1999b. Generalized example-based machine translation. http://www.cs.cmu.edu/ April.

Carl, Michael. 1999. Inducing translation templates for example-based machine translation. In *MT-Summit'99*, Singapore. http://www.iai.uni-sb.de/.

Carl, Michael and Silvia Hansen. 1999. Linking translation memories with example-based machine translation. In *MT-Summit'99*, Singapore.

Carl, Michael, Leonid L. Iomdin, Catherine Pease, and Oliver Streiter. 2000. Towards a dynamic linkage of example-based and rule-based machine translation. *MT-Journal (to appear End 2000).*

Carroll, John, Ted Briscoe, and Antonio Sanfilippo. 1998. Parser evaluation: a survey and a new proposal. In *First International Conference on Language Resources & Evaluation, Granada, Spain.*

Charniak, Eugene. 1996. Tree-bank grammars. In *13th National Conference on Artificial Intelligence, AAAI-96*, pages 1031–1036.

Charniak, Eugene. 2000. A maximum entropy-inspired parser. In *1st Meeting o the North American Chapter of ACL*, pages 132–139, Seattle, Washington.

Chen, Keh-Jiann, Chi-Ching Luo, Zhao-Ming Gao, Ming-Chung Chang, Feng-Yi Chen, and Chao-Jan Chen. 1999. The CKIP Chinese Treebank. In *Journées ATALA sur les Corpus annotés pour la syntaxe.* Talana, Paris VII.

Chiang, David. 2000. Statistical parsing with automatically-extracted Tree Adjoining Grammar. In *38th Annual Meeting of the Association for Computational Linguistics*, Hong Kong, October.

Collins, Michael. 1996. A new statistical parser based on bigram lexical dependencies. In *34th Annual Meeting of the ACL.*

Daelemans, Walter, Antal Van den Bosch, and Jakub Zavrel. 1999. Forgetting exceptions is harmful in language learning. *Machine Learning*, special issue on natural language learning(34):11–43.

Doi, Shinichi and Kazunori Maraki. 1992. Translation ambiguity resolution based on text corpora of source and target language. In *COLING'92.*

Kim, Sung Dong and Yung Taek Kim. 1995. Sentence analysis using pattern matching in English-Korean Machine Translation. In *ICCPOL*, pages 199–206, Honolulu.

Lin, D. 1995. A dependency-based method for evaluating broad-coverage parsers. In *IJCAI-95*, pages 1420–1425, Montreal, Canada.

Manning, Christopher D. 1997. Probabilistic models of language structure. URL cf. below.

Manning, Christopher D. and Schütz Hinrich. 1999. *Foundations of Statistical Natural Language Processing.* The MIT Press, Cambridge, London. http://www.sultry.arts.usyd.edu.au/

Reinke, Uwe. 2000. Evaluierung der linguistischen Leistungsfähigkeit von Translation Memory-Systemen - Ein Erfahrungsbericht-. In *IAI Working Paper 36. Hybrid Approaches to Machine Translation.* O. Streiter, M. Carl and J. Haller (editors). http://iaisun.iai.uni-sb.de/~carl/iaiwp/.

Rijsbergen, C.J. 1979. *Information Retrieval.* Butterworth, London. http://www.dcs.ac.uk/Keith/.

Streiter, Oliver. 1999. Parsing Chinese with randomly generalized examples. In *NLPRS'99 Workshop on Multi-lingual Information Processing and Asian Language Processing (Natural Language Processing Pacific Rim Symposium)*, pages 114–119, Beijing, November.

Streiter, Oliver. 2000. Reliability in example-based parsing. In *TAG+5, Fifth International Workshop on Tree Adjoining Grammars and Related Formalism*, pages 257–260, Paris.

Streiter, Oliver. 2001. Corpus-Based Parsing and Treebank Development. Submitted.

Streiter, Oliver, Leonid L. Iomdin, Munpyo Hong, and Ute Hauck. 1999. Learning, forgetting and remembering: Statistical support for rule-based MT. In *TMI'99.* http://rockey.iis.sinica.edu.tw/.