in: *Proceedings of the MT Summit VII: MT in the Great Translation Era*. Asia-Pacific Association for Machine Translation. Singapore, 1999, pp. 547-555; also available from `http://www.isi.edu/~germann`
Note: This is a slightly revised version of the original paper. Some typographical errors have been corrected.
This version was produced on 5th November 1999.

A Deterministic Dependency Parser for Japanese

Ulrich Germann
USC Information Sciences Institute, Marina del Rey, CA 90292
germann@isi.edu

Abstract

We present a rule-based, deterministic dependency parser for Japanese. It was implemented in C^{++}, using object classes that reflect linguistic concepts and thus facilitate the transfer of linguistic intuitions into code. The parser first chunks morphemes into one-word phrases and then parses from the right to the left. The average parsing accuracy is 83.6%.

1 Introduction

Dependency grammars have recently received increased attention and interest from computational linguists. While there has always been active research in dependency grammars and related theories since the publication of Tesnière's *Éléments de syntaxe structurale* (1959), dependency-based approaches now also seem to become more popular for practical projects and applications such as annotating corpora (Kurohashi and Nagao, 1997; Hajic, 1998) and building parsers (e.g., Kurohashi and Nagao, 1994; Tapanainen and Järvinen, 1997; Arnola, 1998; Haruno et al., 1998; Oflazer, 1999; Sekine et al., 1999; Uchimoto et al., 1999). This interest has also manifested itself in a recent workshop on the processing of dependency-based grammars (Kahane and Polguère, 1998).

The objective of this paper is to present the results of a rule-based implementation of a deterministic parsing algorithm for Japanese. Although our parser does not perform quite as accurately as recent statistical parsers or Kurohashi and Nagao's knowledge-based parser (see section 7), we believe that our experiment provides some interesting insights. On the one hand, our results offer support for some observations made by Sekine et al. (1999) regarding the feasibility of deterministic parsing. On the other hand, we would also like see them as a counterargument to the claim made by Uchimoto et al. (1999) and Sekine et al. (1999) that rule-based systems are practically not feasible.

In the remainder of the paper, we first briefly present some core ideas of dependency grammars, and a summary of specific characteristics of Japanese that are relevant to the discussion. We then discuss and argue for parsing Japanese "backwards", i.e., from the last element of the sentence to the first. Finally, we present and discuss the results of a rule-based implementation of this algorithm.

2 Dependency Grammars

Dependency grammars assume direct, binary and asymmetric *dependency relations* between the elementary units (*nuclei*) of a sentence. For the sake of simplicity, these nuclei can be assumed to more or less correspond to words in English, although Järvinen and Tapanainen (1998:4) point out that "there is no one-to-one correspondence between nuclei and orthographic words, but the nucleus consists of one or more, possibly discontinuous, words or parts of words. The segmentation belongs to the linearisation, which obeys language-specific rules." We will later introduce the notion of *one-word-phrases* (OWPs) as an appropriate unit for the analysis of Japanese.

The relations between the nuclei of a sentence are binary in the sense that they always hold between exactly two of them, and they are asymmetric in the sense that one of these two nuclei is considered the *head* or *governor*, and the other one its *dependent*.

Dependency grammars further assume that the dependency structure of a sentence is a tree. One nucleus — often the 'main verb' or the inflected part of the predicate — has no governor and constitutes the root of the tree. It is said to be *finite* or *independent*. All other nuclei in the sentence have exactly one governor. The number of the dependents of a nucleus is not restricted. Figure 1 provides a simple example of a dependency tree.

Dependency relations can be established on the basis of morphological, syntactic, and/or semantic criteria. For the purpose of this paper, we assume that morpho-syntactic criteria are largely sufficient to determine the dependency structure of a sentence. The advantage of this approach is that it does not require extensive semantic and ontological knowledge bases. It should be noted, however, that the parser presented here does make some use of semantic cri-

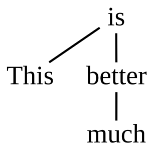

Figure 1: The dependency structure of the sentence *This is much better*.

teria, e.g. in the case of dates and quantifications, which both can be recognized by matching regular expressions.

With respect to Japanese, Rickmeyer (1983) mentions two criteria for the determination of dependency relations: morphological mark-up of the dependent and/or the linear order of dependent and governor.

3 The Structure of Japanese

3.1 Morphology

Japanese is a primarily agglutinative language, i.e., lexical items are morphologically marked by means of affixes (pre- and suffixes). In agglutinative languages, each affix typically contributes only one grammatical feature, as opposed to inflecting languages, where one morpheme may carry several features. Figure 2 illustrates this with an analysis of the Japanese word form 食べられなかった (*tabe.rare.na.kat.ta* - ‘*s.o.* could not eat *s.th.* / *s.o.* or *s.th.* was not eaten (by *s.o.* or *s.th.* else)’[1].

tabe
食べ – ‘eat’ (*V* – lexeme verb),
rare
られ – passive/potential (*-v* – suffix verb),
na
な – negation (*-a* – suffix adjective),
kat
かっ – a suffix verb (*-v*) needed for derivation, so that
the following flexive can be attached
ta
た – flexive (*f*) marking perfect

Figure 2: Morphological analysis for the Japanese word form V+v+a+v+f 食べられなかった (*tabe.rare.na.kat.ta* – ‘*s.o.* could not eat *s.th.* / *s.o.* or *s.th.* was not eaten’), following Rickmeyer (1995)

3.2 *Bunsetsu* (文節) and One-Word-Phrases — the Units of Analysis

Dependency-based analyses of Japanese usually rely upon a concept called *bunsetsu*, which Fujio and Matsumoto (1998:88) describe as follows: “A *bunsetsu* basically consists of one (or a sequence of) content word(s) and its succeeding function words (that forms the smallest phrase, such as a simple noun phrase).”

Table 1: Morpheme classes according to Rickmeyer (1995). The part of speech of a word or one-word-phrase is determined by the part of speech of its last *derivational* morpheme.

part of speech	**derivational**			**other**
	lexeme (L)	**suffix (s)**	**particle (p)**	
Verb	V	-v	=v	
Noun	N	-n	=n	
Adjective	A	-a	=a	
Nominal Adj.	K	-k	=k	
Adverb	M	-m		
Adnominal	D	-d		
Interjection	I			
	non-derivational			
Particle			=p	
Prefix				q
Flexive				f

Rickmeyer (1995) provides a much more precise concept which he calls *one-word-phrase* (*Einwortphrase*). In his analysis of Japanese, he distinguishes five morpheme classes: prefixes, lexemes, suffixes, particles, and flexives. The distribution over the various parts-of-speech is shown in Tab. 1. Of these morpheme classes, only lexemes can occur in isolation, all other morphemes occur only when affixed to a lexeme. A *word* is defined as a lexeme or lexical compound with up to two prefixes, possibly suffixes and at most one flexive. A *one-word-phrase* (OWP) is a word plus any number of particles, whereby certain particles can affix suffixes and flexives. The structure of a Japanese one-word-phrase can be described schematically by the regular expression

(q{0,2})(L|C)(s*)(f?)(p(s*)(f?))*

where

‘q’	= prefix
‘L’	= lexeme
‘C’	= compound (= /(((L)(s*))+)*L/)
‘s’	= suffix
‘f’	= flexive
‘p’	= particle
‘{0,2}’	= ‘occurs 0 to 2 times’
‘*’	= ‘occurs any number of times’
‘?’	= ‘occurs at most once’
‘+’	= ‘occurs at least once’

All morpheme classes except lexemes are closed classes and can be enumerated.

Examples:

- (新)$_1$ (民主)$_2$ (連合)$_3$ (所属)$_4$ (議員)$_5$ (の)$_6$
  siñ miñsyu reñgoo syozoku giiñ no
  q + N + N + N + N = p
  ‘(of)$_6$ (members of parliament)$_5$ (belonging to)$_4$ the (New)$_1$ (Democratic)$_2$ (Union)$_3$’

- (食べ)$_1$ (られ)$_2$ (な)$_3$ (かっ)$_4$ (た)$_5$ (の)$_6$ (で)$_7$
  tabe rare na kat ta no de
  V + v + a + v + f = n = p
  ‘(due to)$_7$ (the fact that)$_6$ [= because] [*s.o.*] (was)$_5$ (not)$_3$ (able to)$_2$ (eat)$_1$ [*s.th.*]’

It should be noted that Rickmeyer’s concept of one-word-phrases is not equivalent to the more widely used

[1] The morphological form ’passive’ can express both passive and potential in Japanese

concept of *bunsetsu*.The main difference is that Rickmeyer's classification is based strictly on syntactical and morphological criteria, whereas the notion of *bunsetsu* also takes recourse to semantics and often considers syntagmas **one** *bunsetsu* rather than two or more. Consider, for example the sequence

(努力)$_1$		(し)$_2$		(な)$_3$		(ければ)$_4$	│	(なら)$_5$		(な)$_6$		(い)$_7$
doryoku		si		na		kereba	│	nar-		-a)na		i
Nv	+	V	+	a	+	f	│	V	+	a	+	f

'(If)$_4$ [*s.o.*] does (not)$_3$ (make)$_2$ an (effort)$_1$, it will (not)$_6$ (become)$_5$ [anything] = [*s.o.*] must make an effort'

Since なる (*nar.u* – 'become') can also occur independently, it is a lexeme. Therefore, according to Rickmeyer's analysis, the syntagma consists of **two** *one-word-phrases* (OWPs), whereas it is considered **one** *bunsetsu* in most *bunsetsu*-based approaches, due to the "functional" character of *nar.u* in this syntagma. As a rule of thumb, OWPs tend to provide a slightly more fine-grained text segmentation than *bunsetsu*. In the following sections, we will assume that the analysis of Japanese is based on OWPs. In practice, the differences between the two notions do not seriously affect the performance of the parser.

3.3 Syntax

In addition to the general constraints on dependency structures mentioned above, namely that the dependency structure of a sentence is a tree, Japanese is furthermore characterized by the following two principles:

1. the dependent always precedes its governor;
2. dependency relations do not cross each other. For example, in the sequence *a b c d e* in Fig. 3, *a* cannot be governed by *c*, because this dependency would cross the dependency between *b* and *e*. This constraint applies to many languages (often with systematic exceptions such as wh-movement in questions, e.g. in English) and is also known as the principle of *projectivity* (cf. Mel'čuk, 1988:35ff.).

4 An Argument for Parsing Japanese Backwards

Various researchers have suggested and used backwards parsing as an adequate and efficient approach to parsing head-final languages such as Japanese (Fujita, 1988; Rickmeyer, 1995; Sekine et al., 1999) and Korean (Kim et al., 1994). Indeed, given the fact that, in Japanese, the dependent always precedes its governor, and given the assumption of dependency grammar that the governor determines the form or occurrence of its dependent but not vice versa, it seems reasonable to analyze the dependency structure of Japanese in a backwards fashion for the following reasons:

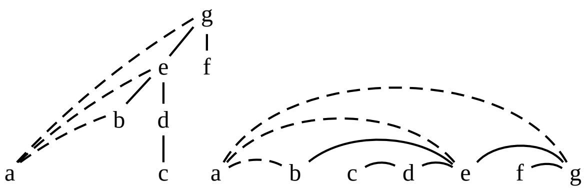

Figure 3: No-crossings or *projectivity* constraint: a can only be attached to elements on the left edge of the parse tree, i.e., b, e, or g. All other attachments would lead to crossing dependencies and therefore violate the constraint. The linear order of the elements is a, b, c, d, e, f, g.

- Since the dependent always precedes its governor, it follows that the last OWP in each sentence must be the root of the corresponding dependency tree.
- Since it is assumed that the dependent does not affect the behavior of its governor, no information preceding ('to the left of') any given OWP should be needed in order to determine the governor of this OWP.
- When parsing head-final languages such as Japanese from the left to the right ('forward'), we face the dilemma that we have to decide how many of the elements already in the parse space are governed by the most recent addition to it. It may be all of them, none, or any number in between. At the time we have to make the decision, we do not even know what other 'candidates for governor' there are. This may be acceptable for non-deterministic parsing techniques, which keep track of alternatives, but it is a serious problem for a deterministic parser that commits itself to one single interpretation early on.

  Parsing 'backwards' avoids this problem. Since we always add a new dependent, not a new (potential) governor to the parse space, and since every dependent has exactly one governor, we know that we have to establish exactly *one* dependency relation. In addition, the search space is closed: we must select the governor from the OWPs already in the parse space.

Working from the end of the sentence, we get a safe start: the last OWP in the sentence necessarily governs its immediate predecessor. We can then turn to the third-but-last OWP and try to determine which of the last two OWPs is its governor. Next, we examine the fourth-but-last OWP, and so on.

Figure 4 outlines the parsing process. In the outer *for* loop, the OWPs are introduced into the parse space one by one, working backwards from the end of the sentence to its beginning. In order to determine their respective governors, the parse space is now traversed in the opposite direction (left-to-right; cf. the inner loop in Fig. 4).

The left-to-right approach in the inner loop is motivated by the following consideration: Since dependency

```
Declarations

        current, tmp        are variables for one-word-phrases (OWPs)
        first               is the first OWP in the sentence
        last                is the last OWP in the sentence
                            last - 1 refers to the second-but-last OWP, last -2 to the third-but-last, etc.
        next(OWP)           is a function that returns the immediate successor of a OWP
        gov(OWP)            is a function that returns the governor of a OWP

for each current from (last -1) to first do:
        REM Step 1: collect potential governors (candidates)
        tmp = next(current)
        begin loop
            if tmp could be the governor of current (based on morphological, lexical and/or semantic
               criteria), add tmp to the list of candidates
            if tmp = last then
                    exit loop
            else
                    tmp = gov(tmp)
            end if
        end loop
        REM Step 2: selection of governor
        select one of the candidates as the actual governor of current; if the list is empty, select the immediate
           successor of current as its governor. Governor selection is based on configurational criteria.
end for
```

Figure 4: The parsing algorithm. The outer *for* loop traverses the sentence from right to left, the inner loop traverses the parse space from left to right.

relations do not cross each other in Japanese[2], it is sufficient to examine only OWPs on the 'left edge' of the parse tree as potential governors. Consider, for example, the situation in Fig. 3. Since b is governed by e, and e by g, the only candidates for the attachment of a are b, e, and g. The projectivity constraint reduces the number of attachments that have to be considered in the parsing process. In the worst case (all OWPs depend on their immediate neighbor), the time complexity of the parsing algorithm is $O(n \cdot \log n)$, where n is the number of OWPs in the sentence. The left edge of the parse tree can be easily detected by following the dependency chain from the leftmost element in the parse tree (i.e., the immediate neighbor of the OWP under consideration) to the root.

Of course, an erroneous 'long' attachment in the parsing process may block certain attachments later on. In order to assess the effect of such errors, we compared the results of 'regular' parsing with the results of an experiment in which we adjusted such erroneous 'long' attachments before we continued to parse. The improvement in accuracy was not significant (from 70789/84704 = 83.57% to 70866/84704 = 83.66%; tested on the January 1-10 sections of the Kyoto Treebank).

5 Implementation

The parsing algorithm described here was implemented in a rule-based fashion in C^{++}. The parser operates on the output of the JUMAN text segmentation and tagging utility developed at the University of Kyoto (Kurohashi et al., 1994; Kyoto University, 1997a). JUMAN segments texts into morphemes and annotates them with information on their part of speech and inflectional form. Prior to the actual parsing process, the JUMAN classifications are mapped to our own classification scheme, which is based mostly on Rickmeyer (1995), and the morphemes are chunked into OWPs. Morphemes, OWPs, and sentences are each wrapped in corresponding C^{++} classes that provide member functions to check for various linguistic properties of the respective entity.

The general idea behind this approach is to provide programmatic equivalences to concepts and notions that linguists use when talking about the structure of Japanese, and to thereby allow code writing that is closer to the way that linguists 'speak' than previous approaches.

The properties provided by the C^{++} object classes range from rather simple ones such as the part of speech or the inflectional form of the respective entity to more complex and abstract properties such as 'does this OWP have a verbal regimen?', which is true not only for verbs and adjectives, but also, for example, for verbal nouns (Nv) under certain conditions (see section 5.2 below), and for OWPs that are nouns by means of derivation but contain a verb or adjective. For instance, the OWP V+f=n=p 探るの は

[2] There are a few rare exceptions that can be treated on an individual basis, cf. Rickmeyer (1995: 58f.).

```
if (   (current->adnominal() && candidate->nominal())

       ...  // other conditions that allow an attachment of ‘current’ to ‘candidate’

    || (  (candidate == current->next)       // ‘current’ immediately precedes ‘candidate’
        &&(current->casemarker() == "を")    // and ‘current’ is marked with を
        &&(candidate->casemarker() == "に")  // and ‘candidate is marked with に
        &&((*candidate->governor)->bform != "する")
          // and ‘candidate’ is not governed by a form of する (s.uru)
        &&is_any_of(candidate->bform,
                    "もと|めど|中心|目当て|契機|前提|機会|合言葉|合い言葉|軸|基|対象|機|基礎|基点"))
       )
   )
{
    current->candidates.push_back(candidate);
    // then add ‘candidate’ to the list of potential governors
}
```

Figure 5: Sample code: This piece of code handles exceptions that allow adnominal attachment of nouns marked with =p を (*wo*; usually exclusively adverbal) to certain other nouns marked with =p に (*ni*). The code has been changed slightly from the original.

(*sagur.u=no=wa* – ‘Finding’) is a derivational noun morphologically, i.e., its morphological behavior and its behavior as a dependent is that of a noun. It was derived from the verb 探る (V+f *sagur.u* – ‘find’) by affixing the the particle noun の (=n *no*). The focus particle は (=p *wa*) is non-derivational, i.e., it has no effect on the part of speech of the OWP. In spite of the derivation, the verb contained in the OWP maintains its verbal regimen, so that the OWP as a whole may also govern adverbal constituents.[3] For example, in [妥協点を]$_1$ [探るのは]$_2$ [難しそうだ]$_3$ ([*dakyooten=wo*]$_1$ [*sagur.u=no=wa*]$_2$ [*muzukasi.soo=da*]$_3$ – ‘[Finding]$_2$ [a common ground]$_1$ [seems to be difficult]$_3$’), the dependency attachment of []$_1$ to []$_2$ is adverbal, which is clearly marked by the particle を (*wo*). On the other hand, as a dependent, []$_2$ behaves like a noun, filling a valency slot of []$_3$.

Correspondingly, verbal OWPs derived from nouns by suffixation of the suffix verb だ (=v *da* – ‘be’) allow **both** an adverbal complements marked with =p が (*ga*; nominative), **and** adnominal modifiers such as relative clauses or nouns marked with =p の (*no*; genitive). Sometimes certain morphological forms, such as N=で (*de*), or positions, such as the finite (= final) position in the sentence, can also affect the valency.

Since dependency relations are often marked morphologically on the dependent in Japanese, properties such as *adverbal* or *adnominal* indicate whether a OWP expects governor with a verbal or adnominal regimen. Some sample code is given in Fig. 5.

5.1 Detecting Potential Dependency Relations

As mentioned above, governor selection takes place in two stages. First, a list of all *potential* governors is compiled, and then one *actual* governor is chosen among them. The criteria used for determining whether or not a dependency relation might hold between a pair of OWPs can be divided into three categories: morphological/syntactic (including punctuation), pattern-based semantics, and idiosyncratic.

- Morphological and syntactic criteria rely on such features such as part of speech, inflection, position in the sentence, and punctuation.

- Pattern-based criteria use regular expressions to detect certain semantic properties that can be inferred from the spelling of the words. For instance, it is possible to use regular expressions to fairly reliably detect numbers, dates, quantities, distances, etc.

- Idiosyncratic criteria are based on individual words or word forms. Nouns marked with =p を (*wo*; accusative), for example, generally cannot depend on another noun. However, there are exceptions. Certain nouns such as もと (*moto* – ‘basis’) or 中心 (*tyuusin* – ‘focus’) can govern other nouns marked with を (*wo*) when they themselves are marked with に (*ni*; dative), e.g. Xを もとに (X=*wo moto=ni* – ‘based on X’). These phenomena could be explained transformationally as elliptic structures derived from X=*wo* Y=*ni si.te* (‘making X Y’). However, since dependency grammar does not assume ‘hidden’ or ‘emtpy’ nodes, X is considered dependent on Y in these cases. Note the explicit listing of certain entry forms (“`bform`”) in Fig. 5.

[3] Note the similarity to Turkish, also an agglutinative language, where the behavior of a word as a dependent is determined only by its last inflectional group, while dependency relations emanating from dependents may generally ‘land’ on any inflectional group within that word (Oflazer, 1999:252–255).

```
if (   current->checkpos("Nv")
    && current->has_particle("none")
    && current->ends_with_comma
   )
{
   if (current->prev->adnominal() || current->prev->ends_with_comma )
   {
      // this case is handled further down in the code ...
   }
   else
   {
   while (   !(   (   (*current->governor)->checkpos("V|A")
                   || (current->checkpos("Nv") && current->has_particle("none"))
                  )
               && (*current->governor)->ends_with_comma
              )
          && current->advance_governor()
         ){}
   }
}
```

Figure 6: Sample code handling the attachment of verbal nouns without particles but marked with a comma. The method `advance_governor()` returns true unless the last candidate has been reached.

5.2 Determinism: Choosing One Governor

Detecting *potential* governors is a task that can be performed fairly easily with a constraint-based approach, determining for each pair of nuclei individually whether or not one nucleus *could* be dependent on the other. The deterministic step of the parsing process, namely to select one of the candidates as the best or correct one, on the other hand, requires a weighting scheme or some other heuristic procedure. Designing such a scheme or procedure by hand may seem to be a daunting task at first glance. Indeed, our parser is currently being outperformed by several statistical parsers (see section 7 below). However, we found that even with hand-coded rules, it is possible to achieve acceptable parsing results. First of all, the nearest candidate is the correct choice in about 77% of all non-trivial cases. We consider as non-trivial all dependency relations except the obvious one between the last two OWPs of the sentence.

Using the nearest candidate as a starting point, we tried to identify indicators for longer attachments. Our approach was to use these indicators as a trigger conditions to 'advance' a pointer that points to the governor of the OWP under consideration along the list of candidates (ordered by precedence) until we encounter certain stop conditions.

For example, commas often are a good indicator that a longer attachment should be preferred. Figure 6 shows a section from the code that handles the attachment of so-called verbal nouns (Rickmeyer, 1995:249f.; the traditional term is サヘン名詞; *sahen-meishi*) that have no particles and are marked with comma. Verbal nouns (Nv) are a subclass of nouns that can compound with the verb する (*s.uru* – '*make*') to form verbs. Especially in newspaper texts, they often are used in their verbal meaning without compounding with *s.uru*. These occurrences are ambiguous in the sense that the occurrence may be **either** verbal **or** nominal but not both in the particular context. Based on the intuition that such an occurrence is verbal only if there is an adverbal dependent, we first look at the immediate predecessor to determine if it is marked as adverbal or adnominal. If it is adnominal, we conclude that the verbal noun under consideration is nominal (because it has either no dependent or an adnominal one). The same applies if we find an adverbal predecessor marked with a comma, which indicates a dependency that reaches over the verbal noun in question. Note, by the way, that this is a case where we do make use of the left context, even though in a very local manner.

If the occurrence of the verbal noun is verbal (and, in this particular case, is therefore also considered adverbal), we advance the pointer to the governor until we encounter (a) a verb or adjective marked with a comma, or (b) another verbal noun without particles and marked with a comma (cf. Fig. 6).

The parser applies a similar strategy for OWPs marked with the focus marker は (*wa*). In other cases, we decided to assume an attachment to the very last candidate directly. For example, a sentence may start with a preliminary remark that provides some background information for the following statement. The root of the subtree representing this preliminary remark is usually a verb or adjective in present or past tense, marked with the particle が (*ga*) and a comma. If we encounter such a configuration, we directly assume attachment to the last candidate. The same holds for certain sentence-initial discourse markers such as しかし (*shikashi* – 'but') or だから (*dakara* – 'therefore').

Table 2: Performance of the parser for the different sections of the Kyoto Treebank. Parts of the treebank, in particular from sections 01/01 and 01/03 were also used as a testbed during the development of the parser. There was no issue of the *Mainichi Shinbun* on 01/02/95.

method	**total**	**01/01**	**01/03**	**01/04**	**01/05**	**01/06**	**01/07**	**01/08**	**01/09**	**01/10**
immediate neighbor	**59.5%**	59.9%	59.5%	59.4%	59.8%	59.5%	59.2%	58.4%	59.7%	60.1%
nearest candidate	**77.4%**	79.0%	78.2%	77.5%	77.7%	77.0%	77.7%	76.6%	76.4%	78.4%
with heuristics	**83.6%**	84.0%	84.2%	82.9%	83.6%	83.3%	83.8%	82.9%	82.9%	84.6%

6 Evaluation

In order to assess to performance of our parser, we evaluated it against the Kyoto Treebank (Kurohashi and Nagao, 1997; Kyoto University, 1997b), which provides dependency structures for about 10,000 sentences from the newspaper *Mainichi Shinbun* from early January, 1995. The average number of OWPs per sentence is 9.94, ignoring one-OWP sentences. In order to keep the analyses comparable, we accepted the chunking of the sentences into *bunsetsu* provided by the treebank.

As a measure for accuracy, we computed the the ratio of correct connections to the total number of connections, excepting the obvious connection between the last two OWPs in the sentence from counting. For example, if the parser correctly identifies 9 dependency relations in a sentence with 12 OWPs (i.e., there are 11 dependency relations within the sentence), we will consider this as 80% accurate ((9-1)/(11-1)). This measure is also used by Mitsuishi et al. (1998), Shirai et al. (1998), and Sekine et al. (1999).

Table 2 compares the accuracy of the parser to two base line algorithms, one postulating the dependency relation always between immediate neighbors, the other one always selecting the nearest candidate.

7 Related Work

Recently, numerous parsers and parsing strategies have been proposed and developed for Japanese.

Mitsuishi et al. (1998) implemented an underspecified, HPSG-Style grammar in LiLFeS (Makino et al., 1998). They report an accuracy of 72.6% at a coverage of 91.9%, tested on 10,000 sentences from the EDR corpus. This is below the performance of our *first candidate* parsing strategy at 100% coverage (cf. Tab. 2).

Shirai et al. (1998) focus on the use of lexical association statistics for parsing. They developed two statistical models for Japanese, one based on syntactic criteria, the other one based on lexical association statistics. These models were then used to disambiguate the output of a probabilistic Generalized LR-parser that was trained on ca. 10,000 sentences from the Kyoto Treebank. Their results showed that by combining lexical and syntactical statistics, parsing accuracy can be improved from 72.1% for the syntactic model and 76.5% for the lexical model to 82.8% for the combined model.

Hermjakob (p.c.) reports an accuracy of 84.5% for a trainable shift-reduce parser, which he originally developed for English (Hermjakob and Mooney, 1997) and recently adapted to Japanese. This parser was trained on the Kyoto Treebank and parses left to right.

In contrast to the work mentioned above, which relies on concatenative grammars for parsing, Fujio and Matsumoto (1998) calculate the dependency probability of each potential dependency relation directly using a maximum likelihood model. They report a recall of 83.5% at a precision of 86.1% (Fujio and Matsumoto, 1998:Table 3). However, since the evaluation scheme differs, these number cannot be compared directly to the other numbers reported here.

Kurohashi and Nagao (1994) built a dependency parser (*KN-parser*) that relies upon linguistic knowledge bases such as a thesaurus, a "surface case dictionary" and a case frame dictionary. Their dependency parser first identifies coordinate structures and then uses valency information from the surface case dictionary as well as simple heuristics to determine dependency relations within these coordinate chunks. According to their own evaluation, they achieve an overall accuracy of 96% on a test corpus of 150 sentences from technical and scientific articles. In a comparative evaluation carried out by Sekine et al. (1999), the KN-Parser achieved an accuracy of 90.8% on a test corpus of 279 sentences of 7-9 *bunsetsu* each. These sentences were taken from the January 9 section of the Kyoto Treebank (newspaper text).

Haruno et al. (1998) use decision trees to generate probabilistic dependency matrices for the *bunsetsu* in a sentence. These dependency matrices allow a probabilistic ranking of alternative parses. The authors report an accuracy of 85.0%.

Finally, Sekine et al. (1999) developed a statistical dependency parser that uses a backwards parsing strategy very similar to the parsing algorithm employed in our parser. However, instead of restricting the search space by a set of constraints and then selecting the governor by means of hand-coded heuristics, they estimate the probability of each potential dependency relation based on a a maximum entropy model of Japanese. The model was trained on sections January 1–8 of the Kyoto treebank and tested on the January 9 section. The authors report an accuracy of 85.5%[4]. Among other things, their research also shows

[4] In the same paper and elsewhere (Uchimoto et al., 1999), the authors also report an accuracy of 87.1% for deterministic

that keeping track of more than one option at each parsing step does not improve accuracy significantly. From this they conclude that the contribution of the left context of each nucleus to the disambiguation of dependency ambiguities is negligible.

8 Discussion

From the work reported in the previous section we can conclude the following:

- Current state-of-the-art statistical parsers can achieve accuracies of up to 85.5% without the use of extensive linguistic knowledge beyond information on the part of speech and inflection as provided by state-of-the-art segmenters and taggers. Making use of additional knowledge bases, Kurohashi and Nagao (1994) were able to achieve an accuracy of between 90.8% (evaluation by Sekine et al., 1999) and 96% (Kurohashi and Nagao, 1994).

- Parsing Japanese backwards in a deterministic fashion does not seem to lead to less accurate parses. It can therefore serve as a mechanism to improve parsing efficiency by systematically restricting the search space.

The rule-based parser presented in this paper does not (yet) achieve the accuracy reported for parsers using statistical methods or extensive knowledge bases. While we were able to reach an accuracy of about 82% fairly easily, improving the parser beyond that turned out to be a tedious and slow process that would have been hardly possible without an annotated corpus (treebank) against which hypotheses could be tested. Nevertheless we believe that our approach may be useful in certain situations and for a variety of applications.

First of all, our approach does not, unlike statistical parsers, *necessarily* require a training corpus. On early stages of development, the developer can evaluate the output of the parser directly. On later stages, however, the use of a test corpus may be mandatory in order to assess the overall effect of changes to more fine-grained rules. If such a corpus does not exist, it may be feasible to develop a hand-coded parser and annotate a corpus in parallel. Depending on the annotation interface, providing the annotator with a pre-parsed structure that is 80% correct and only needs to be corrected may be much more efficient than putting 100% of the annotation burden on the annotator. The annotated parts of the corpus could serve as a testbed for improvements to the hand-coded parser, and as training and test data for statistical approaches. As soon as there is sufficient data for statistical parsers to outperform the hand-coded parser, the hand-coded parser could be replaced.

Secondly, we expect that much of the code we developed for this project can be re-used for other purposes. In particular, the object classes reflecting linguistic intuitions and concepts, such as *morpheme*, *OWP*, *sentence*, etc., and their respective linguistic properties should prove useful for developing applications that deal with tasks such as semantic interpretation or discourse parsing. These are areas in which we are to date not aware of sufficiently annotated corpora that are large enough to serve as a basis for statistical methods.

9 Conclusion

We presented a deterministic ‘backwards’ parser for Japanese. The parser was implemented in C^{++} using object classes that were designed to reflect linguistic intuitions. Using a constraint-based approach to limit the range of potential governors for each nucleus and simple heuristics to select the right one among them, we achieved an overall accuracy of 83.6%. Even though the parser is currently outperformed by other parsers, we consider our approach an option especially for cases where annotated corpora or linguistic knowledge bases are not available. We also expect the code we developed to be useful in other applications.

10 Acknowledgments

I am very grateful to my reviewers of the *MT Summit VII* program committee for various very helpful pointers to related work.

References

Arnola, H. (1998). “On parsing binary dependency structures deterministically in linear time”. In Kahane and Polguère (1998), pages 68–77.

Fujio, M. and Matsumoto, Y. (1998). “Japanese dependency structure analysis based on lexicalized statistics”. In *Proceedings of the 3rd Conference on Empirical Methods in Natural Language Processing (EMNLP ’98)*, pages 88–96.

Fujita, K. (1988). “A deterministic parser based on kakari-uke grammar”. In *Proceedings of the 2nd National Meeting of the (Japanese) Association for Artificial Intelligence*, pages 399–402. [藤田克彦. 1988. 決定的係り受け解析に関する試み. 昭和６３年度人工知能学会全国大会 (第２回)].

Hajic, J. (1998). “Building a syntactically annotated corpus: The Prague dependency treebank”. In E. Hajičová (ed.), *Issues of Valency and Meaning*, pages 106–132. Karolinum, Prague.

Haruno, M., Shirai, S., and Ooyama, Y. (1998). “Using decision trees to construct a practical parser”. In *Proceedings of the Joint 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (COLING-ACL ’98)*, pages 505–511.

parsing. The difference is due to the fact that this higher number includes the last – obvious – dependency in each sentence.

Hermjakob, U. and Mooney, R. J. (1997). “Learning parse and translation decisions from examples with rich context”. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL ’97)*, pages 482–489.

Järvinen, T. and Tapanainen, P. (1998). “Towards an implementable dependency grammar”. In Kahane and Polguère (1998), pages 1–10.

Kahane, S. and Polguère, A. (eds.). (1998). *Processing of Dependency-based Grammars: Proceedings of the Workshop. COLING-ACL ’98*.

Kim, C., Kim, J.-H., Seo, J., and Kim, G. C. (1994). “A right-to-left chart parsing for dependency grammar using headable paths”. In *Proceeding of the 1994 International Conference on Computer Processing of Oriental Languages (ICC-POL ’94)*, pages 175–180.

Kurohashi, S. and Nagao, M. (1994). “KN parser: Japanese dependency / case structure analyzer”. In *Proceedings of the International Workshop on Sharable Natural Language Resources*, pages 48–55, Nara, Japan. Nara Institute of Science and Technology.

Kurohashi, S. and Nagao, M. (1997). “The Kyoto University text corpus project”. In *Third Annual Meeting of the Association for Natural Language Processing*. [黒橋貞夫、長尾眞. 1997. 京都大学テキストコーパス・プロジェクト. 言語処理学会 第３回年次大会].

Kurohashi, S., Nakamura, T., Matsumoto, Y., and Nagao, M. (1994). “Improvements of Japanese morphological analyser JUMAN”. In *Proceedings of the International Workshop on Sharable Natural Language Resources*, pages 20–28, Nara, Japan. Nara Institute of Science and Technology.

Kyoto University. (1997a). *JUMAN*. http://www-lab25.kuee.kyoto-u.ac.jp/nl-resource/juman.html. As of 05/22/1997. URL valid on 06/11/1998.

Kyoto University. (1997b). *Kyoto University Text Corpus, Version 1.0*. http://www-lab25.kuee.kyoto-u.ac.jp/nl-resource/corpus.html. As of 09/23/1997. URL valid on 06/11/1998.

Makino, T., Yoshida, M., Torisawa, K., and Tsujii, J. (1998). “LiLFeS - towards a practical HPSG parser”. In *Proceedings of the Joint 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (COLING-ACL ’98)*, pages 807–811.

Mel’čuk, I. (1988). *Dependency Syntax: Theory and Practice*. State University of New York Press, Albany.

Mitsuishi, Y., Torisawa, K., and Tsujii, J. (1998). “Hpsg-style underspecified Japanese grammar with wide coverage”. In *Proceedings of the Joint 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (COLING-ACL ’98)*, pages 876–880.

Oflazer, K. (1999). “Dependency parsing with an extended finite state approach”. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL ’99)*, pages 254–260.

Rickmeyer, J. (1983). “Wie schwierig ist die Japanische Sprache. Ein Vergleich mit dem Deutschen”. In *Bochumer Jahrbuch zur Ostasienforschung*, pages 187–202.

Rickmeyer, J. (1995). *Japanische Morphosyntax*. Groos, Heidelberg.

Sekine, S., Uchimoto, K., and Isahara, H. (1999). “Statistical dependency analysis using backward beam search”. *Natural Language Processing*, 6(3):59–73. [関根聡、内元清貴、井佐原均. 1999. 文末から解析する統計的係り受けアルゴリズム. 自然言語処理. 6 (3)].

Shirai, K., Inui, K., Tokunaga, T., and Tanaka, H. (1998). “An empirical evaluation on statistical parsing of Japanese sentences using lexical association statistics”. In *Proceedings of the 3rd Conference on Empirical Methods in Natural Language Processing (EMNLP ’98)*, pages 80–87.

Tapanainen, P. and Järvinen, T. (1997). “A non-projective dependency parser”. In *Proceedings of the 5th Conference on Applied Natural Language Processing*, pages 64–71.

Tesnière, L. (1959). *Éléments de syntaxe structurale*. Klincksieck, Paris, 2nd edition.

Uchimoto, K., Sekine, S., and Isahara, H. (1999). “Japanese dependency structure analysis based on maximum entropy models”. In *Proceedings of the 9th Conference of the European Chapter of the ACL (EACL ’99)*.