Arguments for *Parallel Distributed Parsing*
Toward the integration of lexical and sublexical (semantic) parsings

Kow KURODA

National Institute of Information and Communication Technologies (NICT), Japan

Abstract. This paper illustrates the idea of parallel distributed parsing (PDP), which allows us to integrate lexical and sublexical analyses. PDP is proposed for providing a new model of efficient, information-rich parses that can remedy the data sparseness problem.

1 Introduction

In the usual sense of syntactic parsing, the analyses of relationships among words and relationships among morphemes are separated. The former is called syntactic analysis (or syntactic parsing) and the latter is called morphological analysis (the term "morphological parsing" exists but it seems to denote a somewhat different notion). Sometimes, however, sublexical analysis is relevant. This is evident in shallow semantic parsing which is understood to be "labeling phrases of a sentence with semantic roles with respect to a target word. For example, the sentence (1) is labeled as (2):"[1)]

(1) Shaw Publishing offered Mr. Smith a reimbursement last March.

(2) [$_{\text{Agent}}$ Shaw Publishing] offered [$_{\text{Recipient}}$ Mr. Smith] [$_{\text{Theme}}$ a reimbursement] [$_{\text{Time}}$ last March] .

The target of the labeling in (2) is *offered*. Note that the same kind of labeling should be available for the argument structure of *reimbursement*, which can be illustrated in (3):

(3) [$_{\text{Payer (as Agent)}}$ Shaw Publishing] offered [$_{\text{Recipient}}$ Mr. Smith] a [$_{\text{Target}}$ reimburse]-ment last March. (No explicit mention of Payment (as Theme))

Clearly, semantic role labeling requires a high-precision predicate-argument analysis of a target predicate, whether the target is a word (e.g., *offer*) or a morpheme (e.g., *reimburse*) embedded in a word. The comparison of the two cases shows that a certain kind of parsing is necessary in effective semantic labeling, as Gildea and Palmer (2002) pointed out. But the problem is how to combine the lexical parse by which the predicate-argument structure of *offer* is recognized with the sublexical parse by which the predicate-argument structure of *reimburse* is recognized. Integrating the two kinds of parses is not a trivial task.

This paper presents the idea of *Parallel Distributed Parsing* (PDP) that is able to straightforwardly carry out the integration. The presentation, however, is theoretically oriented and the content is rather preliminary: no empirical results are presented other than a few sample parses. No parser implementation is available. The main purpose of this presentation is to illustrate a new model for parsing that integrates lexical and sublexical parsings, which I argue can be a remedy for the problem of data sparseness.

Data sparseness is a serious problem in natural language processing (NLP) even now that computers can access more raw data than the average human does. The size of textual raw data automatically acquired from the Web exceeds that which a normal human can read in a lifetime. This suggests, however, that no human seems to suffer from data sparseness. What makes this more mysterious is that humans do also employ statistical information in their language processing. The difference between humans and machines, therefore, should lie in the difference in efficiency with which they acquire knowledge, be it is syntactic, semantic or morphological, from linguistic data given. Humans are certainly, at the present, able to acquire knowledge far more efficiently than computers. The crucial question is: **How is this possible?**

I argue that data sparseness is a problem in NLP *not only because distributional data itself is sparse, but also because parses availabe today are sparse and inefficient*; otherwise, data sparseness should impact human language processing in the same way it does computers. The explanation I consider is that *data contains enough information but current technologies fail to extract it due to inefficiency of available parses*. PDP is proposed to make parses of linguistic data more efficient and less sparse. In what follows, I show how PDP can remedy the data sparseness.

[1)] The example and explanation were taken from `http://nlp.stanford.edu/projects/shallow-parsing.shtml`.

2 Efficient parsing with PDP

2.1 Preliminaries

To begin with, take the example (4):

(4) a. Ann gave Bill a headache
b. Ann gave Bill a hug.

On reading (4a), we, as human, understand at least the following:

(5) a. CAUSE(x, z, t_1, ...)
b. x=Ann, z=EXPERIENCE(y, a headache, t_2, ...), y=Bill, $t_2 \subset t_1 \subset$ PAST
c. $\nRightarrow z'$= HEADACHE(x, y, t_2, ...), $t_2 \subset t_1 \subset$ PAST

On reading (4b), we, as human, understand at least the following:

(6) a. CAUSE(x, z, t_1, ...)
b. x=Ann, z= EXPERIENCE(y, a hug, t_2, ...), y=Bill, $t_2 \subset t_1 \subset$ PAST
c. $\Rightarrow z'$= HUG(x, y, t_2, ...), $t_2 \subset t_1 \subset$ PAST

We have at least the following problems: i) Why do we have CAUSE(...) for the semantics of *give*? ii) Why do we have the different elaborations for z-term? Namely, why do we have HUG(x, y, ...) in (4b) and not have HEADACHE(x, y, ...) in (4a)? More specifically, how can we elaborate CAUSE(x, EXPERIENCE(y, a hug), ...) into CAUSE(x, HUG(x, y), ...) and not elaborate CAUSE(x, EXPERIENCE(y, a headache), ...) into *CAUSE(x, HEADACHE(x, y), ...)?[2)]

Putting aside the first problem, I limit myself to the second problem. The difference lies in the difference in the semantic structures of *headache* and *hug*, which is obvious. The question I want to address is: **Does syntactic parsing/analysis play no role in this kind of elaboration?** I raise this question based on the following contrast:

(7) a. *Ann headaches Bill. [cf. Bill aches in the head.]
b. Ann hugs Bill.

This contrast indicates that transitive use of *headache* is disallowed whereas transitive use of *hug* is allowed.[3)] I argue that the contrast in (7) is exactly the information that we need for the elaboration of CAUSE(x, EXPERIENCE(y, a hug), ...) into CAUSE(x, HUG(x, y), ...).

Many theories of parsing are happily posit that such information can be obtained by accessing the "lexicon" and syntactic parsing can (and should) be freed from it. The process by which we arrive at targeted semantic elaboration is called a series of "inferences." But nothing prevents us from doubting this position, especially when we are ready to expand the scope of syntactic parsing to include the recognition of as many semantic relations in a given input as possible. The problem of data sparseness encourages this expansion of the scope, because it is a key to overcoming the data sparseness mentioned earlier.

The greatest technical problem, of course, is how to encode semantic information in syntactic parsing without considerably increasing the complexity in parsing and incompatibility with orthodox (tree-based) parsing. The framework of parallel distributed parsing (PDP) is proposed for meeting such requirements in the most straightforward way.

2.2 Sample PDPs

An implicit assumption under the traditional view of syntactic parsing is that *not enough information is available on the surface to obtain such "inferences."* But this assumption turns out to simply be wrong if we are allowed to apply sublexical parsings of (4a) and (4b) that account for the difference in (7).

[2)] Technically, causation is nonreflexive transitive in (4a) and reflexive transitive in (4b).
[3)] This is simply because the latter is a zero-derived form of a verb *hug*; *kiss* is a similar case.

Table 1: PDP of (4a)

e=p0	*Ann*	*gave*	*Bill*	*a*	*head*	*ache*
p1	*Ann*	V				
p2	S	*gave*	O1	a		O2
p3	S	V	*Bill*			
p4	(S)	(V)	(O)	*a*	(M)	T
p4	(S)	(V)	(O)	D	*head*	T
p5			S	M1	M2*	*ache*

Table 2: PDP of (4b)

e=p0	*Ann*	*gave*	*Bill*	*a*	*hug*
p1	*Ann*	V			
p2	S	*gave*	O1		O2
p3	(S)	(V)	*Bill*		
p4	(S)	(V)	(O)	*a*	T
p5	S		O*	M	*hug*

2.2.1 Basics. For illustration, sample PDPs of (4a) and (4b) are given in Tables 1 and 2. The PDP of input *e* results in a set of *n* parses when *e* has *n* segments. This is presented in table form as in the two tables. The first row of the table represents the input under a certain type of segmentation. The other rows below represent parses of *n* segments, p1, p2, ..., p*n* which specify the parses of the 1st, 2nd, ...*n*th segment of the input. These are called "patterns."

Each parse is a string-like object that consists of "constants" (e.g., *Ann*, *gave*, *Bill*, *a*, *head* and *ache*) and "variables" (e.g., S, O, V, P, etc). Constants in a pattern are called the pattern's "anchors." To enhance readability, anchors are usually in italics. Note that anchors usually appear on the diagonal in PDP.

Variables serve as "matching sites" or "binding sites" because they encode the information necessary to integrate a set of patterns. Patterns $p = [u_1, u_2, \ldots, u_m]$ and $q = [v_1, v_2, \ldots, v_n]$ are unified if and only if (i) they have the same number of segments (i.e. $m = n$) and (ii) either IS-A(u_i, v_i) or IS-A(v_i, u_i) holds. Variables license the IS-A relation. In some cases, variables are put in parentheses to indicate that they encode conditional information.

Take some examples. "S *gave* O1 O2" is a pattern that specifies the predicate-argument structure of *gave*. Likewise, "S V *Bill* O2" is a pattern that specifies the predicate-argument structure that accounts for the presence of *Bill* in this input. This form of encoding is called the "co-occurrence structure" of *Bill*, because *Bill*, as a noun phrase, does not have a predicate-argument structure of its own. Case assignment is involved in co-occurrence structure. Compare $\phi 1$: "*Bill* V," $\phi 2$: "S V *Bill*" and $\phi 3$: "S V P *Bill*," each of which specifies a co-occurrence structure of *Bill*. $\phi 1$ specifies the nominative form of *Bill*, and $\phi 2$ and $\phi 3$ the accusative form of it, though $\phi 2$ and $\phi 3$ are not really the same.[4)]

Relational nouns (e.g., *head*, *frend*) including derivational nouns (e.g., *hug*) have a "co-argument structure" that needs to be distinguished from the co-occurrence structure. The co-argument structure of term *w* is a specification of the prototypical predicate argument structure in which *w* serves as an argument. This is relevant only when *w* is not a genuine predicate. For relevant information, refer to Kuroda et al. (2009). It deserves a mention that co-occurrence structure and co-argument structure are identical in certain uses of relational terms, quite unfortunately.

Let me mention several confusing cases. On the one hand, p6: "S M1 M2* *ache*" in Table 1 specifies the argument structure of *ache* as a verb, rather than the co-occurrence structure of *ache* as a noun. Also, p5: "S O* M *hug*" in Table 2 specifies the argument structure of *hug* as a verb rather than the co-occurrence structure of *hug* as a noun. Their co-occurrence structures must be p6′: "(S)(V) (O1) D M *ache*" and p5′: "(S)(V)(O) D *hug*," but they do not give us desirable results. On the other hand, p5: "(S)(V)(O) D *head* T" in Table 1 specifies the co-occurrence structure of *head* rather than its co-argument structure. Its co-argument structure should be like p5′: "S *head*" with S matching *Bill*, but this does not really fit the context of (4a). In the current version of PDP, either (co-)argument structure or co-occurrence structure is specified in this priority, and not both. Admittedly, this is not a systematic solution but a compromise was made to reduce the complexity of the analysis. In the full version of PDP, a constant may have multiple parses as far as they are not incompatible. This is clearly desirable for relational nouns.

The order of variables within a pattern is important because patterns are expected to be as surface-true a specification as possible of the predicate-argument structure of lexical items. But in some cases, the arrangement of variables cannot be surface-true and generates a mismatch. For example, p5: "S O* M *hug*" in Table 2 is not surface-true in that it deviates from "S M *hug* O" which is a generalization of instances like *she once hugged him*. To encode the positional deviation of variables, "*" is used: $\alpha*$ encodes the positional deviation of α. For example, O* in "S O* M *hug*" indicates its positional deviation.

[4)] In many languages, case-marking systems are not elaborated enough to reflect the distinction between cases like $\phi 2$ and $\phi 3$.

2.2.2 Interpretation of PDPs. Under the brief explanation above, the two PDPs read as follows: In the PDP in Table 1, p1 says that *Ann* is the subject of a certain verb, symbolized by V, which is either transtive or intranstive. V is to be unified with *gave*. p2 says that *gave* is a ditrantive verb: its subject, direct and indirect objects are to be unified with *Ann*, *Bill* and *ache*, respectively. p3 says that *Bill* is the direct object of a ditrantive verb: its subject is unified with *Ann*, its verb with *gave* and its indirect object with *ache*. p4 says that *a* is the determiner for a theme/target, which is to be unified with *ache*. p5 says that *head* is a prenominal modifer to a theme/target to be unified with *ache*. p6 says that *hug* is a transitive verb: its subject is to be unified with *Ann*, its object with *Bill* and its two modifiers are bound to *a* and *head*. If M1 means anything, it would mean *once* when it is bound to *a*. If M2 means anything, it would mean *(in the) head*. Note that p6 allows us to state that *Bill aches in the head* is embedded in the PDP of (4a).

In the PDP in Table 2, p1 says that *Ann* is the subject of a certain verb. p2 says that *gave* is a ditrantive verb: its subject, direct and indirect objects are to be unified with *Ann*, *Bill* and *hug*, respectively. p3 says that *Bill* is the direct object of a ditrantive verb: its subject, verb and indirect object are to be unified with *Ann*, *gave* and *hug*. p4 says that *a* is the determiner for *hug*, its theme/target. p5 says that *hug* is the verb: its subject and object are to be unified with *Ann* and *Bill*. If *a* is bound to its modifier M, it can imply *S hug O once*. Note that p5 allows us to state that *Ann hug(ged) Bill (once)* is embeded in the PDP of (4b).

2.3 Procedure

In the simplet form, the PDP of input e is peformed in the following procedure:

(8) a. **Step of segmentation:** Segment e into a list of units. Most simply, we have $[u_1, u_2, \ldots, u_n]$ when $e = u_1 \cdot u_2 \cdots u_n$.

b. **Step of pattern identification:** For u_i in $[u_1, u_2, \ldots, u_n]$, find out a "pattern" p_i that specifies the predicate-argument structure of u_i in the optimal granularity.

where specifications of p_i and p_j can be independent.

In the following, the explanation of segmentation follows the explanation of pattern identification.

Table 3: Initial state of PDP of (4b)

e=p0	*Ann*	*gave*	*Bill*	*a*	*hug*
p1	*Ann*	gave	Bill	a	hug
p2	Ann	*gave*	Bill	a	hug
p3	Ann	gave	*Bill*	a	hug
p4	Ann	gave	Bill	*a*	hug
p5	Ann	gave	Bill	a	*hug*

Table 4: Phase 1 of PDP of (4b)

e=p0	*Ann*	*gave*	*Bill*	*a*	*hug*
p1	*Ann*	V	O1	D	O2
p2	S	*gave*	O1	D	O2
p3	S	V	*Bill*	D	O2
p4	(S)	(V)	Bill	*a*	T
p5	S	V	O1	D	*hug*

2.3.1 Essence of pattern identification. PDP is still under construction because no implementation is available for pattern identification, but I present a rough sketch of it here by taking (4b) for example. I expect that implementation is feasible using a method of supervised machine learning such as SVM.[5)]

PDP starts with the initial state like the one in Table 3. This undergoes the process of abstraction in the following sense. For every parse, all constants except for the anchor (usually on the diagonal) are abstracted by replacing them by labels such as S, O, V, and P. Replacement of constants $c_1, c_2, \ldots, c_n$ in parse $p = c_1 \cdot c_2 \cdots a \cdots c_n$ with anchor a is carried out in such a way that constant c_i is replaced by a grammatical role/function that is defined relative to a. Thus, p1: "*Ann* gave Bill a hug" is rendered into the sequence "*Ann* V O1 D O2" because *gave*, *Bill*, *a* and *hug* bear the roles of verb (V), direct object (O1), determiner (D) for O2, and indirect object (O2) relative to *Ann*. The same holds for other parses. This results in the specification in Table 4.[6)]

The next thing to do is to take care of the informational asymmetry between the past and future. The presence of any constants after the achor is less certain. This requires variables to be less specific after the anchor than before. If this asymmetry is taken care of, we finally have a PDP like the one in Table 2.

[5)] Theoretically, all we need is a high-precision training corpus with enough coverage, but the means of preparing it is a different matter.

[6)] I omitted the details in constructing "(S)(V)(O1) *a* T," which requires the definition of theme/target T.

As mentioned above, variables like S, O, V, P, etc encode what grammatical roles/functions constants bear against an achor. For example, *ache* is a verb inside p6 in Table 2 and *hug* is a verb inside p5 in Table 2 But they are indirect objects of *gave* in p2 in Tables 1 and 2. Note that uniqueness is not required on the role assignment across the set of patterns. In this sense, identification of grammatical relations is relativized to each constant. Parallelism would be impossible without it.

2.3.2 Essence of segmentation. Segmentation need not be lexically based. It can be done sublexically or superlexically. If every *u* is a word, it gives lexical PDP. If a *u* is a morpheme, this gives sublexical PDP. Specifications of p5: "(S) (V) D *head* T" and p6: "S M1 M2* *ache*" in Table 1 for *head* and *ache* are cases of sublexical parsing.

Table 5: PDP of (1) where constants appear on the diagonal in italics, and variables in normal face.

p0	*S. Publishing*	*offered*	*Mr.*	*Smith*	*a*	*reimburse*	*-ment*	*last*	*March*
p1	*S. Publishing*	V							
p2	S	*offered*		O1			O2		
p3	(S)	(V)	*Mr.*	T					
p4	S	V	D	*Smith*					
p5	(S)	(V)		(O)	*a*	(M)	T		
p6	S			O*		*reimburse*			(M)
p7	S					V	*-ment*		
p8	(S)	(V)						*last*	T
p9		T						M	*March*

2.4 More details of PDP

2.4.1 Composition by superposition. PDP is compositional but in a different way. In most traditional theories of syntactic structure, composition of substructures $s_1, s_2, \ldots s_n$ into a whole structure t is achieved by means of **substitution** of certain "variables" in t by substructures. For example, *Ann gave Bill a headache* results when *Ann*, *gave*, *Bill*, and *a headache* are substituted for S, V, O1 and O2 in the host structure "S V O1 O2" (or for NP1, V, NP2 and NP3 in the host structure "NP1 V NP2 NP3").[7)]. This is not true of PDP, where **superposition** is used instead. As Table 1 indicates, p1, p2, ..., p6 are superposed on each other to produce p0 = (4a). Superposition of p1, p2, ..., p*n* into p0 is column-wise (feature-based) unification. Thus, substitution plays no role in composition of p0 out of p1, 2, ..., p*n* in PDP. This is expected to reduce the computational complexity.

2.4.2 Constraints on patterns. A pattern is a sequence of either lexical items, called constants, such as *Ann*, *gave*, *Bill*, *a*, *headache*, ..., or some of the variables listed in (9):

(9) a. Predicate types: **V** (verb), **U** (auxiliary verb), **P** (preposition, particle and postposition), **R** (underspecified type between **V** and **P**), **J** (junction)[8)], and **A** (adjective).[9)]

b. Argument types: **S** (subject), **O** (object) [**O1** direct object and **O2** indirect object. In general, **O***n* for the *n*th object of a predicate], **C** (nominal complement of *be*-type verb)

c. Functional types: **D** (determiner), and **Q** (quantifier)

d. Other types: **T** (theme/target of a determiner), **M** (modifier), and **X** (unknown type: use needs to be avoided whenever possible).

e. Hybrid types: types like $\alpha + \beta$ (e.g., S+V, P+O2) and $\alpha = \beta = \gamma = \cdots$. The former encodes the amalgamation of type α and β. The latter encodes the multiplicity of labels.

The list of pattern variables here is not meant to be exhaustive.

To make PDP descriptively adequate, patterns need to be well constrained. In respect to the formulation of such constraints, PDP is still under develpment. It is still unclear what makes patterns valid or invalid. But we can state a few requirements on good patterns:

7) Notably, grammar is responsible for generation of structures like S V O1 O2 or NP1 V NP2 NP3.
8) This is a generalized class of conjunction and disjunction.
9) There is no type for adverbials because they are usually not referred to by other lexical units.

(10) a. A pattern needs to be a description of an argument structure within a minimal span.
b. A pattern is valid when it expresses a generalization as surface-true as possible of the predicate argument structures of a lexical item or a series of lexical items.

3 Parallel relational parsing with PDP

3.1 Further issues

After the brief introduction to PDP above, let us now turn to the analysis of (1), which is a case more complicated than two cases in (4). The PDP of (1) is given in Table 5, which illustrates distinctive features of PDP.

3.1.1 Effects of parallelism. In the first place, PDP allows for sublexical parses without introducing contradiction with lexical parses. As mentioned, the problem is how to integrate the parses for *offer* and *reimburse* in *S. Publishing offered Mr. Smith reimbursement last March* [=(1)]. The PDP solution is given in Table 5. More specifically, the peaceful coexistence of p2 and p6 in Table 5 shows that the integration is successful.

There is a subtle point related to the interpretation of *last March*.[10] It is possible to read it so that reimbursement was made in a month referred to as *last March*. But this is only a suggestion because the interpretation is no longer valid in cases like: *The company offered Mr. Smith reimbursement last March but he declined.* This contrast suggests that the completion of the reimbursement by Mr. Smith is implied only when the completion of the company's offer of it is factive. This confirms that the modification by *last March* to *offer* is direct and its modification to *reimburse* is indirect and probably conditional. This effect is more or less predictable from the information encoded by p9: "T M *March*" in Table 5.[11]

There are still some subtleties in PDP, however. First of all, I have to admit that it is not true that everything in PDP has a precise interpretation. This is actually false. There are several cases in which patterns fail to receive straightforward interpretations. For example, it is not clear what *-ment* means in p7 in Table 5. My best guess is that it serves as a kind of auxiliary here. With this problem remaining, PDP should turn out be useful for limited purposes.

3.1.2 Effects of distributed representation. Another interesting characteristics of PDP is that it allows us to assign multiple grammatical roles to a lexical item. In fact, it allows us to directly encode "hidden" roles which are usually indirectly encoded using transformations or lexical redundancy rules. For example, "subjects" are assigned to implict verbs like *ache* in (4a) and *hug* in (4b). This means that PDP is able to encode grammatical relations in term of distributed representation.

But subjects are not exclusively assigned to verbs and adjectives: they are assigned to prepositions, particles (e.g., *up*), adverbs and adverbials, too. This point is illustrated in the PDP of (11) presented in Table 6.

(11) On this note, C-in-C gave up the idea of retaining Ben in the front.

In PDP, all relational kinds of constants are expected to have subjects of their own.[12]

3.1.3 Sparseness somehow remedied. Sparseness of the label distribution in Table 6 suggests that syntactic structure consists mainly of local, short-distance dependencies. But there are special constructions like support verb constructions (e.g., S *give up the idea of* V-*ing* = { p5, p6, p7, p8, p9, p11 }) that can extend localities to some extent.[13]

From the PDP in Table 6, we can get at least the following predications:

(12) a. C-in-C gave up the idea [presupposed by the semantics of *On this note*].
b. C-in-C (wanted to) retain Ben [presupposed by the semantics of *give up the idea*].
c. Ben (was) in the front [presupposed by the semantics of *retain*].

[10] This point was brought to my attention by one of the anonymous reviewers, for which I'm grateful.
[11] This is also related to a subtle point that it is discouraged to have (M) in p6: "S O* *reimburse* (M)" because specification of unseen elements, which occurr in the future, needs to be less specific.
[12] By this, it is safe to state that PDP captures the effects of "trace" without positing movement and those of pro and PRO without positing specific configuration of phrase structure.
[13] Note incidentally that S *give up the idea of* V-*ing* can be a paraphrase of "S *stop thinking of* V-*ing*." They are both cases of subject-to-subject raising.

Table 6: PDP of (11) [constants in italics, and variables in normal face]

p0	*on*	*this*	*note*	*X*	*gave*	*up*	*the*	*idea*	*of*	*retain*	*ing*	*B.*	*in*	*the*	*front*
p1	*on*		O	(S)	V=T										
p2	P	*this*	T												
p3	P	D	*note*	(S)	V=T										
p4				*X*	V										
p5				S	*gave*	P+O2		O1							
p6					(M)	*up*		S							
p7				(S)	(V)	(M)	*the*	T							
p8				S	V	(M)	D	*idea*							
p9								S	*of*		O				
p10				S						*retain*		O			
p11				S						V	*ing*				
p12				S						V		*B.*			
p13												S	*in*		O
p14												(S)	(P)	*the*	T
p15												S	P	D	*front*

They justify, in combination, that *C-in-C* are subjects of *gave up* and *retain* and *Ben* is the subject of *(be) in the front*.

What *of* does in (11) is, as specified by p9, bridges the two propositions encoded by *idea* and *retain*. In this respect, it is desirable to detect that "S *think of* V-*ing* and "S *want to* V" are in the relation of a paraphrase. Relating to this, it should be added that *C-in-C* is identified as the subject of *think* if we can somehow identify that the relation of "S *idea of* V-*ing*" and "S *think of* V-*ing*" is of a paraphrase, but this is not specified in the PDP in Table 6. PDP is not responsible for the detection of paraphasability.

As pointed out in §1, it is often the case that inefficient parses increases the severity of data sparseness. Inefficiency comes from the sparseness of parses. Parallel distributed sublexical parses provided by PDP would be useful for remedying this.

3.2 Related work

Table 7: Simplified from of MST Parse of (11)

	Lemma	TAG	Target	Function
1	On	IN	7	VMOD
2	this	DT	3	NMOD
3	note	NN	1	PMOD
4	,	,	7	P
5	the	DT	6	NMOD
6	C-in-C	NN	7	SUB
7	gave	VB	0	ROOT
8	up	RP	7	VMOD
9	the	DT	10	NMOD
10	idea	NN	7	OBJ
11	of	IN	10	NMOD
12	retaining	VB	11	PMOD
13	Ben	NN	12	OBJ
14	in	IN	12	VMOD
15	the	DT	16	NMOD
16	front	NN	14	PMOD

Table 8: PMA format of parse in Table 7

	on	*this*	*note*	,	*the*	*C-in-C*	*gave*	*up*	*the*	*idea*	*of*	*retaining*	*Ben*	*in*	*the*	*front*
p1	*on*															
p2		*this*														
p3	PMOD	NMOD	*note*													
p4				,												
p5					*the*											
p6					NMOD	*C-in-C*										
p7	VMOD			P		SUB	*gave*	VMOD		OBJ						
p8								*up*								
p9									*the*							
p10									NMOD	*idea*	NMOD					
p11											*of*	PMOD				
p12												*retaining*	OBJ	VMOD		
p13													*Ben*			
p14														*in*		
p15															*the*	
p16														PMOD	NMOD	*front*

3.2.1 Dependency parsing All parses in PDP can be seen as distinct runs of dependency parsing that run in parallel. What distinguishes it from other formalisms is that it tries to make use of parallelism. For one, PDF does not avoid crossing-links (McDonald et al., 2005). To make this point clear, take the analysis of (11) for example. MST Parser (v0.4.3) produces the dependency parse in Table 7. Its comparison with Table 6 reveals that the dependency parse in Table 7 is a subset of the PDP in Table 6. This point is made clear in Table 8. In other words, PDP describes whatever dependency parsing describes.

3.2.2 Word Expert Parsing. There is an important conceptual precursor of PDP. Word Expert Parsing (WEP) was developed by Small (1979; 1983; 1988) to implement the idea of "parsing as cooperative

distributed inference," and extended to Parallel Word Expert Parsing (PEP) by researchers like (Hahn, 1986; Devos et al., 1988).

WEP/PEP has many things in common with PDP. Simply speaking, PDP could be seen as merely adding linguistic sophistication to WEP/PEP. There is, however, an essential difference. First, WEP/PEP only targets the construction of semantic interpretation, or more precisely it only targets word sense disambiguation tasks, and accordingly does not really "parse" the text, though it is called a framework for text parsing. Second, WEP/PEP embodies a very simplistic view of the lexicon in the sense that it defines "words" as elementary units of syntactic representation and tries to directly associate them to semantic/conceptual structures. This means that there is no place where context-sensitive encodings like p3 = "S *offered* O1 O2," p4 = "S V D *Smith*," and p5 = "S V O1 *a* O2" play any role (some familiar examples of such patterns are "multi-word expressions" (Sag et al., 2002) and "constructions" (Fillmore, 1988; Goldberg, 1995)). In contrast, patterns are fundamental units of linguistic representations in PDP, and more importantly, parallelism is required to handle them in the most natural way.

4 Conclusion

In this short, far from a complete paper, I presented arguments for parallel distributed parsing (PDP). It is motivated for the integration of lexical and sublexical parses. I must admit that PDP is still underdeveloped, as many technical details required for serious parsing are missing. But this does not mean, I hope, that it cannot be a new model of syntactic description. I say this because it may give us a clue for overcoming the data sparseness problem from which many NLP researchers suffer.

References

G. Adriaens and Steven L. Small. 1988. Word expert parsing revisited in a cognitive science perspective. In *Lexical Amibiguity Resolution: Perspectives from Psycholinguistics, Neuropsychology, and Artificial Intelligence*, pages 13–43. Morgan Kauffmann Publishers.

M. Devos, G. Adrieans, and Y. D. Willems. 1988. The parallel expert parser (pep): A throughly revised descendent of the word expert parser (wep). In *Proceedings of the 12th Conference on Computational Linguistics, Vol. 1*, pages 142–147.

C. J. Fillmore. 1988. The mechanisms of 'Construction Grammar'. In *BLS*, volume 14, pages 35–55. BLS.

D. Gildea and M. Palmer. 2002. The necessity of parsing for predicate argument recognition. In *Proceedings of ACL 2002*, pages 239–246.

A. D. Goldberg. 1995. *Constructions: A Construction Grammar Approach to Argument Structure*. University of Chicago Press, Chicago, IL.

U. Hahn. 1986. A generalized word expert model of lexically distributed text parsing. In *Proceedings of the 7th ECAI (Brighton, UK), Vol. 1.*, pages 203–211.

K. Kuroda, M. Murata, and K. Torisawa. 2009. When nouns need co-arguments: A case study of semantically unsaturated nouns. In *Proceedings of the 5th International Workshop on Generative Approaches to the Lexicon, Sep. 17-19, 2009, Pisa, Italy*, pages 193–200.

R. McDonald, F. Pereira, K. Ribarov, and J. Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of HLT/EMNLP 2005*.

C. Rieger and Steven L. Small. 1979. Word expert parsing. In *Proceedings of the 6th International Joint Conference on Artificial Intelligence, 1979*.

I. Sag, T. Baldwin, F. Bond, A. Copestake, and D. Flinckinger. 2002. Multiword expressions: A pain in the neck for NLP. In *Proceedings of the 3rd International Conference on Intelligent Text Processing and Computational Linguistics (Mexico City)*, pages 1–15.

S. L. Small. 1983. Parsing as co-operative distributional inference: Understanding through memory interaction. In M. King, editor, *Parsing Natural Language*. Academic Press.