Reducing the Complexity of Parsing by a Method of Decomposition

Caroline Lyon

Bob Dickerson

School of Information Sciences
University of Hertfordshire
Hatfield, Herts. AL10 9AB
email: C.M.Lyon@herts.ac.uk Tel: +44 (0)1707 284266

Abstract

The complexity of parsing English sentences can be reduced by decomposing the problem into three subtasks. Declarative sentences can almost always be segmented into three concatenated sections: *pre-subject* - *subject* - *predicate*. Other constituents, such as clauses, phrases, noun groups, are contained within these segments, but do not normally cross the boundaries between them. Though a constituent in one section may have dependent links to elements in other sections, such as agreement between the head of the subject and the main verb, once the three sections have been located, they can then be partially processed separately, in parallel.

An information theoretic analysis is used to support this approach. If sentences are represented as sequences of part-of-speech tags, then modelling them with the tripartite segmentation reduces the entropy. This indicates that some of the structure of the sentence has been captured.

The tripartite segmentation can be produced automatically, using the ALPINE parser, which is then described. This is a hybrid processor in which neural networks operate within a rule based framework. It has been developed using corpora from technical manuals. Performance on unseen data from the manuals on which the processor was trained are over 90%. On data from other technical manuals performance is over 85%.

1 Introduction

Now that it is technically feasible to locate the subject of a sentence automatically, we may be able to take advantage of this to reduce the complexity of parsing English text. Declarative sentences can almost always be segmented into three concatenated sections: *pre-subject* - *subject* - *predicate*. The pre-subject segment may be empty. This analysis can also be used for imperative sentences, in which case the subject section is empty. Other constituents, such as clauses, phrases, noun groups, are contained within these segments, but do not normally cross the boundaries between them. Though a constituent in one section may have dependent links to elements in other sections, such as agreement between the head of the subject and the main verb, once the three sections have been located, they can then be partially processed separately, in parallel.

This paper first demonstrates how the tripartite segmentation works in practice, drawing on examples from corpora of technical manuals. The next section (section 3) supports the argument that this decomposotion captures some of the structure of sentences by applying an information theoretic analysis. We discuss the representation that has been used: this enables us to model sequences higher than regular grammars in the Chomsky hierarchy, though not fully context free.

We then (section 4) give an outline of the ALPINE parser, which automatically locates the subject of declarative sentences [1]. Readers are invited to access a prototype via telnet, and try it on their own text [1]. Finally, we discuss the scope and limitations of this approach.

1.1 Corpora

This work has principally been developed on text of technical manuals from Perkins Engines Ltd. [2], and the ALPINE prototype was trained on 351 sentences from these manuals. Table 1 and Figure 1 show some of the statistics of the corpus.

Number of sentences	351
Average length	17.98 words
No. of subordinate clauses	
In pre-subject	65
In subject	19
In predicate	136
Co-ordinated clauses	50

Punctuation marks are counted as words, formulae as 1 word.

Table 1: Corpus statistics

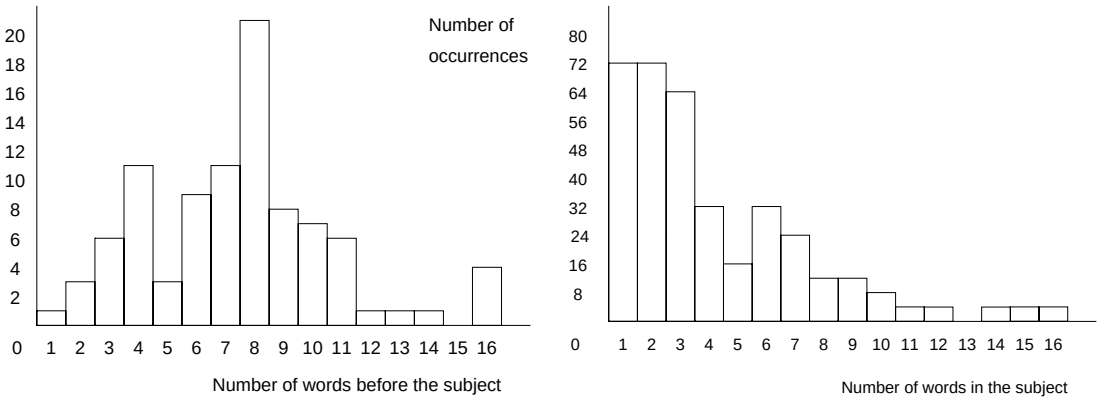

Figure 1: The frequency of constituent length for 351 sentences. 121 sentences have no pre-subject.

The prototype has been run on sentences from two other corpora of text from technical manuals, described in [3]. They are taken from the Dynix Automated Library Systems Searching Manual, and the Trados Translators Workbench for Windows User's Guide. 248 declarative sentences were extracted manually.

Quirk and Greenbaum's "A University Grammar of English" is used as a grammatical reference [4].

2 The Tripartite Segmentation of English Sentences

Figure 2 shows in diagrammatic form how declarative sentences are decomposed. Taking as examples two sentences from the Dynix set, the segmentation would appear like this:

[There] are two methods of accelerated searching. (1)

[1] For details contact the author at C.M.Lyon@herts.ac.uk

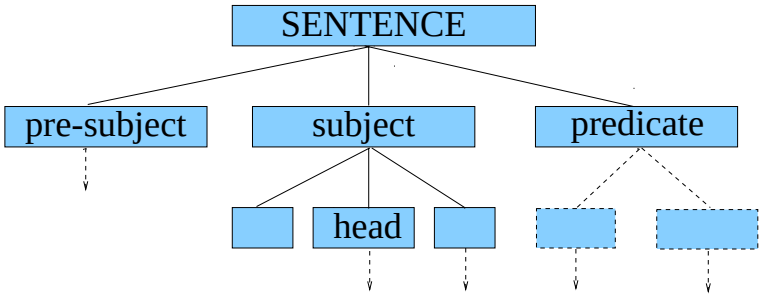

Figure 2: Decomposition of the sentence into syntactic constituents

[Your ability to use either search method] is determined by your familiarity with searching on your system. (2)

To use the first type of accelerated search, [you] must know the search menu line number for the type of search you want to perform. (3)

Sentence (1) is short, and locating the subject will not facilitate the parsing process. In sentences (2) and (3), however, we have divided the longer sentences into partially independent segments. Within each segment there can be finite and non-finite clauses, phrases, noun groups. However, these linguistic constituents are typically confined to their segment, which can thus be processed separately. Parsing complexity is reduced as longer sequences are replaced by a set of shorter sequences.

In the ALPINE prototype the next step that is taken is to find the head of the subject. At the same time that this is done the structure of the pre-subject can be analysed; simultaneously the main verb phrase in the predicate can be located, and then the structure of the rest of the predicate determined.

Though the segments are partially independent, there are of course dependencies between some of their elements. Consider a sentence from the Perkins set, with the subject and its head located.

If a cooler is fitted to the gearbox, [the pipe [connections] of the cooler] must be regularly checked for corrosion. (4)

The modal verb "must" has the same form in the singular and plural in English. If it is to be translated it is necessary to find the syntactic head of the subject to ensure number agreement. In this case, the head is the plural "connections", though the noun adjacent to "must" is the singular "cooler".

The tripartite segmentation that is described can be extended to imperatives by having an empty subject segment. Technical manuals have a prescriptive style, and imperatives are common. An example from the Dynix corpus is:

To perform an accelerated search, [ ] follow these instructions. (5)

Note that this prescriptive style is not well represented in some corpora such as Brown and LOB. Thus automatic taggers that are trained on them may have difficulty in handling imperatives - for example, sentences that open with an imperative verb that could also be tagged as a noun..

2.1 Exceptions

Declarative and imperative sentences can almost always be segmented in this way in English, but there are exceptions. Certain constructions invert word order, such as the Dynix sentence:

Only when you select an item does the system tell you who owns it. (6)

There are some idiomatic usages such as

The more you have, the more you want. (7)

And certain unusual usages could occur, such as

The student arrived who had been late before. (8)

3 Evaluating Parsing Schemes with Entropy Indicators

The decomposition of sentences as described above is not only in accordance with linguistic intuition, but can be supported by an argument from information theory. We can determine whether a representation captures some of the language structure [5]. This approach is introduced by analogy.

3.1 Analogy from Letter Sequences

The principle on which the evaluator is based is derived from Shannon's original work with letter sequences [6]. His ideas can be extended to other linguistic entities. The metric used is based on the entropy of a sequence, which is a measure, in a certain sense, of the degree of unpredictability. If the grammar captures some of the structure of language, then the relative entropy of the text should decline after parsing.

We describe a method of representation that enables the entropy of sentences to be measured under different parsing schemes. We can thus objectively assess whether parsers that accord with linguistic intuition do indeed capture some regularity in natural language.

Shannon examined the entropy of letter sequences, and produced a series of approximations to the entropy H of written English, which successively take more of the statistics of the language into account. A sequence can be represented as a series of single letters, or a series of letters taken n at a time, an n-gram representation. If the sequence is represented by an n-gram, then information over the preceding $n-1$ letters is taken into account, and the entropy is represented as H_n, as shown in Table 2 [2].

H_0	entropy with no information on letter frequencies
H_1	entropy with information on single letter frequencies
H_2	entropy with information on bigram frequencies
H_3	entropy with information on trigram frequencies

Table 2: Notation for representing the entropy of a sequence

As n increases the entropy declines: the degree of predictability is increased as information from more adjacent letters is taken into account. A formal explanation can be found in [5].

Now, the entropy can also be reduced if an extra character representing a space between words is introduced, producing a 27 letter alphabet, with associated entropies H'. By introducing an extra element, the number of choices has increased, so $H'_0 > H_0$. But for higher order entropies, $H'_n < H_n$. This is partly because the space will be more common than other characters, but also because "a word is a cohesive group of letters with strong internal statistical influences" [6]. The introduction of the space has captured some of the structure of the letter sequence.

3.2 Representing the entropy of a sentence

In order to assess the entropy of text we have to address the problem that there are an indefinite number of words. However, we can partition the vocabulary into a limited number of part-of-speech tag classes, and map sentences onto tag sequences. By taking this step we lose much

[2] This notation is derived from that used by Shannon, but differs from that used by Bell et al. [7]

information: the process is not reversible. However, we aim to retain the information that is needed for one stage of processing, and return to the actual words at a later stage.

Now, language can be represented at a primary level as a regular grammar, and we can apply Shannon's analysis to tag sequences. However, this is an inadequate representation. Its shortcomings are apparent as the the statistical patterns of tag sequences may be disrupted at clause and phrase boundaries. Consider the probability of part-of-speech tags following each other: some combinations are "unlikely", such as *noun - pronoun* and *verb - auxiliary verb* but they may occur at constituent boundaries in sentences like

The shirt he wants is in the wash. (9)

The tag string representation can be extended to accommodate this. In a similar manner to the insertion of a space between words, the embedded clause is delimited by inserting boundary markers, or hypertags, like virtual punctuation marks. The sentence is represented as

The shirt [he wants] is in the wash. (9a)

The part-of-speech tags have probabilistic relationships with the hypertags in the same way that they do with each other. The pairs generated by this string would exclude *noun - pronoun*, but include, for instance, *noun - hypertag.left*. Using this representation, one level of embedding has been modelled. We can thus represent sequences higher in the Chomsky hierarchy than regular grammars, though not fully context free.

Since the boundaries of clauses and phrases often coincide with the boundary of the subject, we expect that the insertion of hypertags to demarcate the subject will lower the entropy. If their insertion has captured some of the structure of language the bipos and tripos entropy should be reduced. This is indeed what was found on the data from the Perkins corpus. A tagset of 32 was used, including the hypertags.

Entropy measures have been used to determine the most effective representation for a formal language [8]. We suggest that they can also be used to evaluate representations for natural language.

4 Automatic subject location

In this section we give an overview of the ALPINE parser, designed to locate the subject of a declarative sentence automatically. For more details see [1, 9, 10, 11]. The principle on which this hybrid system is based is to generate all possible parses, prune them, then pick the right one out of the remainder with a neural network. Currently there are arbitrary limits on the length of the subject (12 words) and the length of the pre-subject (15 words).

The processor is trained in supervised mode on correctly marked up data. Then the parameters are frozen for use on unseen data. The initial processes are the same in both training and testing mode.

First, words are mapped onto one or more possible part-of-speech (POS) tags. Many words have more than one tag. It was found that for the first task of locating the subject the tagset could be reduced from 32 to 22. This was mainly achieved by omitting number information, which contributed very little. At later processing stages, number information is needed.

Then the boundary markers of the subject, the hypertags, are placed in all possible positions, with all possible tag combinations. Tag disambiguation is currently an integral part of this first

	H_0	H_1	H_2	H_3
26 letter	4.70	4.14	3.56	3.3
27 letter	4.76	4.03	3.32	3.1

Table 3: Comparison of entropy for different n-grams, with and without representing the space between words

	H_0	H_1	H_2	H_3
plain tag string	5.0	3.962	2.659	2.132
tags + subject markers	5.0	4.135	2.472	1.997

Table 4: Entropy measures for text with and without subject boundary markers

stage. This produces for each sentence one correct and many incorrect strings. However, the generation of strings is limited by (i) the grammatical framework and (ii) local and semi-local constraints.

The grammatical framework first *asserts* that the sentence can be decomposed into consecutive segments. It then expands each segment in a very shallow analysis, based on an EBNF formalism [11].

The local and semi-local constraints are derived from the ideas of Barton et al. [12]. Applying these constraints the generation of any string is zapped if a prohibited feature is produced. An example of a local prohibition is that the adjacent pair *(verb, verb)* is not allowed. Of course *(auxiliary verb, verb)* is permissible, as is *(verb,] , verb)*.

An example of a semi-local constraint is that, if you enter into a subordinate clause, (a) you cannot exit until a verb has occurred (b) you must exit before entering the next segment.

The pruning process leaves a set of candidate strings for each sentence. With the Perkins' data, the upper limit was about 25, average about 4. With other test data much larger sets may be generated.

Each of these candidate strings will be presented to the neural net, which will be trained to give a positive score to the desired string, a negative score to the wrong strings. In testing mode, the string with the highest grammaticality score Γ is taken as the correct one. The placement of the hypertags and the disambiguated ordinary tags in the winning string are accepted.

To code the input for the neural network, create a binary vector, whose elements are POS pairs and triples, bipos and tripos. If a pair or triple occurs in a string, that element is flagged to 1.

4.1 Coding the Input

As an example of input coding consider a short sentence:

All papers published in this journal are protected by copyright. (10)

(A) Map each word onto 1 or more tags:

all	predeterminer
papers	noun or verb
published	past-part-verb
in	preposition or adverb
this	pronomial determiner
journal	noun
are	auxiliary-verb
protected	past-part-verb
by	preposition
copyright	noun
.	endpoint

(B) Generate strings with possible placement of subject boundary markers, and possible tag allocations (pruned).

string no. 1
strt [pred] verb pastp prep prod noun aux pastp prep noun end
. . .

string no. 4
strt [pred noun] pastp adv prod noun aux pastp prep noun end

string no. 5
strt [pred noun pastp] adv prod noun aux pastp prep noun end

string no. 6 *** target ***
strt [pred noun pastp prep prod noun] aux pastp prep noun end

string no. 7
strt [pred noun pastp adv prod noun] aux pastp prep noun end

(C) Transform strings into sets of tuples

string no. 1
(strt, [ ) ([, pred) (pred,]) .....(noun, end)
(strt, [, pred) ([, pred,]) (pred,], verb)..... (prep, noun, end)

and similarly for other strings.
(D) The elements of the binary input vector represent all tuples, initialized to 0. If a tuple is present in a string the element that represents it is changed from 0 to 1.

4.2 Data Representation and Neural Network Design

The method of higher order representation, taking the input data as pairs and triples, has two advantages.

1. It partially captures the sequential nature of language, with a set of bipos and tripos sequences. Alternative methods of capturing the sequential character of the data are to use a moving window technique, or else to use recurrent networks.

2. The higher order transformation produces data that is (almost) linearly separable, so we can use single layer networks. These have the advantage of functional transparency and operational speed.

It is always theoretically possible to solve supervised learning problems with a single layer, feed forward network, providing the input data is enhanced in an appropriate way. A good explanation is given by Pao [13, chapter 8] or Widrow [14, page 1420]. Whether this is desirable in any particular case must be investigated. The enhancement, a non-linear transformation, will map the input data onto a space, usually of higher dimensionality, where it will be linearly separable. In our processor this function is an ordered 'AND'. A similar function is used in the grammatical inference work of Giles et al. [15]; it is also used in DNA sequence analysis [16].

The function can be arithmetic: for instance, for polynomial discriminant functions the elements of the input vectors are combined as products [17, page 135]. Successful uses of this approach include the discrimination of different vowel sounds [18] and the automated interpretation of telephone company data in tabular form [19].

Radial basis function (RBF) networks also come into the class of Generalized Single Layer Networks (GSLNs) [20, 21]. In their two stage training procedure the parameters governing the basis functions are determined first. Then a single layer net is used for the second stage of processing.

4.3 Performance

In analysing the performance we will consider whether the subject boundary markers are correctly placed. Other metrics include an analysis of tag disambiguation, and measures of whether other constituents are correctly found.

Ratio test set / training set	Hodyne % tags + hypertags correct	Hodyne % hypertags correct
0.10	92.9	100
0.20	91.4	100
0.23	89.2	100
0.26	84.4	95.5

Table 5: Best results on Perkins data, after 2% sentences had been edited. This used both bipos and tripos representation together

Text	Number of sentences	% hypertags correctly placed
Dynix	114	92.1
Trados	134	85.1
Total	248	88.3

Table 6:

All declarative sentences were extracted from some of the Perkins manuals. 2% were edited so that they they fell within the restrictions on subject and pre-subject length. The 351 sentences that made up the Perkins data was divided into four parts. The neural networks were trained on three of the parts and tested on the fourth. The performance of three different neural networks were compared, and the results are given below in Table 5 for the best of them. This was developed from the Hodyne higher order, single layer neural net, which originated at British Telecom Research Laboratories [22].

The Hodyne network was then trained on all the Perkins data, augmented with a small amount of other straightforward English sentences. This is the prototype, on which users can try their own text.

Using this network, the data from the other technical manuals was processed. The declarative sentences were extracted, and we only looked to see if the hypertags were correctly placed. Results are in Table 6, and an example of the output is in the Appendix. Before we could run this data through ALPINE, we had to write a short pre-editing program, so that the system would accommodate printing styles not previously encountered. Examples are: all capital acronyms, or sentences ending a full stop inside quote marks, such as “Asimov.”.

5 Scope and Limitations

We suggest that this type of system could be used as a pre-processor to facilitate the processing of longer sentences by other NLP methods. Though there are arbitrary limits on the length of constituents that can be processed by this method (15 words in the pre-subject, 12 words in the subject), these bounds are comparatively wide, and of course we plan to extend them. In the Trados data 9 sentences out of 134 fell outside these limits.

We have only tested this system on declarative sentences, but can see no problem in extending it to imperatives. We will examine ways in which it can be used for questions. The problem of handling non-sentential data is more difficult. In the first instance titles, lists, captions must be identified as such. We expect to do this using other sorts of information, such as change of font and formatting.

The Perkins manuals were written in a deliberately careful way, so that they would be as clear as possible. The authors were expected (not always successfully) to follow certain rules. For instance, sentences were not meant to be more than 23 words long, and the use of present participles was discouraged. Considering the narrowness of this domain, it is surprising how

the characteristics of the language are captured. This is demonstrated by *ad hoc* use of the prototype on other text.

One of the advantages of this approach to parsing is that it lends itself to the extraction of predicate/argument structure. After the subject has been located the main verb will be found in the predicate, and then the object or complement. With the head of the subject found, we then have the raw material from which we can begin to extract the predicate/argument structure.

Appendix

Examples from the Trados corpus processed by ALPINE

The prototype locates both the subject and its head.

> In Word, { [you] } have several possibilities to view non-textual data, such as carriage returns or tabs .
>
> { Another important [category] of non-textual data } is what is referred to as "hidden text."
>
> { The [Workbench] } makes massive use of hidden text in order to perform several tasks .
>
> After opening a new translation unit with the TWB1 or TWB2 buttons,
> { the [Workbench] } inserts hidden tags into the Word document .
>
> { These [tags] } delimit the current translation segments, that is, the source segment in the blue source field, and the target segment in the yellow target field as follows .
>
> Later on, { these delimiting [tags] } play a crucial role for recognizing "Source Matches" which will be described further down .
>
> After translating, { the source [segment] } is kept in your document as "hidden text", together with all delimiting tags .
>
> { The target [text] , that is, the translation you enter, } is of course formatted as normal text, with all formattings intact .
>
> { A quick [way] to toggle between visualizing and hiding nonprinting characters } is again the button in WinWord's standard toolbar .

Examples on which the system will fail include the following.
(i) Test data not well enough modelled by training data

> { A 100% match [means] that exactly this sentence } was already translated, and the suggested translation can therefore be accepted as is .

(ii) Idiomatic usage not recognized

> { [That] } is, these words make the source sentence longer or shorter than the TM sentence .

(iii) Subject or pre-subject too long

> For instance, if a different product name is used in the source sentence than in the fuzzy-match equivalent from TM, both product names will be shown in yellow .

References

[1] C Lyon and R Frank. Using single layer networks for discrete, sequential data: an example from natural language processing. *Neural Computing Applications*, 5 (4), 1997, to appear.

[2] P. Pym. Perkins engines and publications. In *PROCEEDINGS of Technology and Language in Europe 2000*. DGXIII-E of the European Commission, 1993.

[3] R Sutcliffe, H-D Koch, and A McElligott, editors. *Industrial Parsing of Technical Manuals*. Rodopi, 1996.

[4] R Quirk and S Greenbaum. *A University Grammar of English*. Longman, 1976.

[5] C Lyon. Evaluating parsing schemes with entropy indicators. In *MOL5, 5th Meeting on the Mathematics of Language*, August 1997. To appear.

[6] C E Shannon. Prediction and Entropy of Printed English. *Bell System Technical Journal*, pages 50–64, 1951.

[7] T C Bell, J G Cleary, and I H Witten. *Text Compression*. Prentice Hall, 1990.

[8] K Lari and S Young. Estimation of stochastic context free grammars. *Computer Speech and Language*, 1990.

[9] C Lyon and R Frank. Neural network design for a natural language parser. In *International Conference on Artificial Neural Networks (ICANN)*, pages 105–110, 1995.

[10] C Lyon and R Dickerson. A fast partial parse of natural language sentences using a connectionist method. In *7th Conference of the European Chapter of the Association of Computational Linguistics*, pages 215–222, 1995.

[11] C Lyon. *The representation of natural language to enable neural networks to detect syntactic structures*. PhD thesis, University of Hertfordshire, 1994.

[12] G E Barton, R C Berwick, and E S Ristad. *Computational Complexity and Natural Language*. MIT Press, 1987.

[13] Yoh-Han Pao. *Adaptive pattern recognition and neural networks*. Addison Wesley, 1989.

[14] B Widrow and M Lehr. 30 years of adaptive neural networks. *Proceedings of IEEE*, 78:1415–1442, September 1990.

[15] C L Giles et al. Higher order recurrent networks and grammatical inference. In D S Touretzky, editor, *Advances in neural information processing systems*, pages 380–388. Morgan Kaufmann, 1990.

[16] A Lapedes, C Barnes, C Burks, R Farber, and K Sirotkin. Applications of neural networks and other machine learning algorithms to DNA sequence analysis. In *Computers and DNA*, pages 157–182. Addison Wesley, 1992.

[17] R Duda and P Hart. *Pattern Classification and Scene Analysis*. John Wiley, 1973.

[18] S Holden and M Niranjan. On the practical applicability of VC dimension bounds. *Neural Computation*, 7(6):1265–1288, 1995.

[19] A K Chhabra, S Chandran, and R Kasturi. Table structure interpretation and neural network based text recognition for conversion of telephone company tabular drawings. In *International Workshop on Applications of Neural Networks to Telecommunications*, pages 92–98, Princeton, 1993.

[20] M Anthony and S B Holden. On the power of polynomial discriminators and radial basis function networks. In *Proc. of 6th ACM Conference on Computational Learning Theory*, 1993.

[21] S Holden and P Rayner. Generalization and PAC learning: Some new results for the class of generalized single layer networks. *IEEE Trans. on Neural Networks*, pages 368–380, 1995.

[22] P J Wyard and C Nightingale. A single layer higher order neural net and its application to context free grammar recognition. In N Sharkey, editor, *Connectionist Natural Language Processing*. Intellect, 1992.