A linear algorithm for chunk linking

paper #963

Abstract

Tagging, chunking, and even "clause bracketing" are now classical processes (since the early nineties) which have a linear complexity in time relatively to the number of processed words.

This article presents an original algorithm to link chunks in linear time. This algorithm has been implemented in two linear parsers : the first parser won the GRACE international contest for French taggers, and the other is used in an industrial context : in a text-to-speech system, to compute prosody, where linear complexity is a prerequisite.

Introduction

An important stake in parsing strategies today is to produce robust parsers, able to process raw linguistic material at a constant and foreseeable rate. Such parsers may be included in industrial contexts as text-to-speech systems or newsfeed processing on internet, where linear complexity is a prerequisite.

Tagging (for instance, Brill 1995, Chanod 1995) and chunking (Abney 1990) gave a new way to process raw material in linear complexity.

If we call "clausing" the two tasks : clause bracketing and computing chunk main functions inside a clause, we can say that Aït-Moktar and Chanod (1997) achieve "clausing" : it works the same way as chunking. From the functional point of view, "clausing" fills part of the gap in the way up to classical parsing functions.

But to reach classical parsing functionalities, an algorithm for linking is needed, and to fill industrial needs, it has to be of linear complexity. This is the aim of this article.

The paper is divided into three sections : the first section focuses on computing properties of tagging, chunking, and "clausing", the second presents our linking algorithm and gives examples, and the third quotes two parsers which use this algorithm : in the evaluation contest GRACE, and in the industrial context of a text-to-speech system.

1 Tagging, chunking and "clausing"

In this section, we focus on comparing computing properties of classical formal grammar parsers, to the ones of tagging, chunking, and clausing.

1.1 From parsing with formal grammars ...

Traditionally, the parsing problem is solved within the frame of the compiling model, as the parsing of a programming language, which is an exhaustively defined code (on the contrary, a natural language is only partially known). This solution is combinatorial : all the combinations of the attribute values are enumerated in the dictionary (categories, genders, numbers, persons ... of word written forms), which is supposed to be exhaustive, and all the combinations of structures are "exhaustively" enumerated in a formal grammar (inventory of phrase and sentence structures).

From these data, the combinatorial process gives a value to the attributes of the processed units : words and phrases of an input sentence. There is a valuation criterion of a whole combination (all words of a sentence have a category, and all phrases are delimited and have a structure) : a combination is evaluated in a Boolean way on the grammaticality of the sentence according to the formal grammar.

The theoretical complexity in time of such a combinatorial process is exponential according to the number of words of the parsed sentence. The claimed practical complexities in the literature are in O(n^3) to O(n^6) for Tree Adjoining Grammars.

Within the frame of this model, links are implicitly coded in the structure of clauses or recursive phrases. Furthermore, everything has to be successfully parsed between two linked words for the parser to compute the link. Thus, a failure on computing a link may be due to a lack of coverage of the grammar.

1.2 ... to contextual deductions in tagging

When tagging appeared, it was used to do shallow parsing on raw material. But, as taggers give less tags for a token than a given dictionary, they can help to reduce the combinatory aspect of formal grammar parsers while replacing morpho-lexical analysis before syntactic analysis.

But the main contribution of tagging is to focus on explicitating the deduction process by using contextual properties (new knowledge upon language is injected into the process) rather than explicitating expected structures.

The tagging process consists in applying statistical or symbolic rules to tokens, and has a linear complexity in time.

So, tagging shows a way of renewal in parsing strategies.

1.3 Tagging works better while chunking

The concept of chunk appeared in the early nineties, in Abney (1990). In the very first lines, he presented it as a prosodic segment. It is also known as "core phrase" or "non recursive phrase" in opposition to the chomskian recursive phrase.

The chunk is a quite easy segment to automatically delimit, with very little knowledge : the beginnings of chunks are often grammatical words, which are in finite number. About applications, nominal chunks are useful terms in automatic indexation and information retrieval : that is why chunkers became usual.

Vergne and Giguet (1998) have shown that tagging is easier and more accurate inside typed chunks than in a flow of words : most often, a grammatical word at the beginning of a chunk gives the type of the chunk, and the type of the chunk constraints the word categories inside it.

Aït-Moktar and Chanod (1997) have transposed this way to process words inside chunks in order to process chunks inside clauses on the same way. Their incremental finite-state parser segments a sentence into phrases and clauses and computes chunk functions inside clauses. This is a great progress toward the functions performed by a traditional formal grammar parser, while keeping a linear complexity and processing raw material.

2 Linking chunks

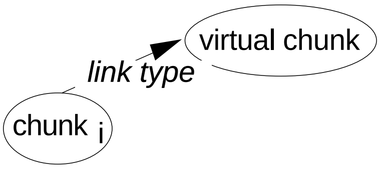
The algorithm we present is able to link any kind of linguistic unit, but we used it first to link chunks in sentences represented as a chain of chunks.

2.1 Principle of the linking algorithm

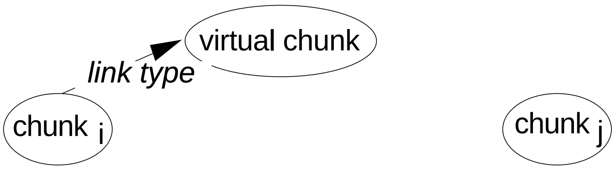
The algorithm links 2 chunks into 2 successive steps by means of a virtual chunk, and we present it as a process on a graph in which nodes are chunks and arcs are links between chunks :

- the flow of chunks, is processed in their arrival order (in the diagram, first the chunk $_{i}$, then the chunk $_{j}$).

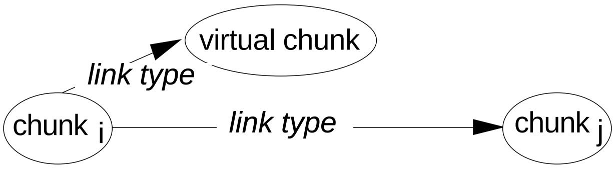
• at **step 1**, when chunk $_{i}$ arrives, it is linked to the virtual chunk, which can be invoked at any time in the rules; for instance, if chunk $_{i}$ is a nominal chunk, it is placed into a waiting position for a verbal chunk (subject-verb link) :



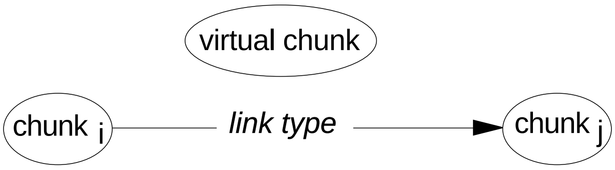
• at **step 2**, when chunk $_{j}$ arrives (for instance a verbal chunk), a rule condition verifies whether a chunk (here chunk $_{i}$) is linked to the virtual chunk by a link of a specified type (in our example, subject-verb) :



- then, still in step 2, an action of this rule links both chunks with a link of the same type :



- at the end of step 2, the link between chunk $_{i}$ and the virtual chunk is discarded, and that means in our example that this subject has found its verb and does not wait for a verb any more :



We may notice this algorithm is very general : it works whatever the length and the structure of the segment which separates the two linked chunks; and it does not depend on the processed natural language, because the process only works with chunks types, and does not depend on written forms any more.

2.2 Two examples

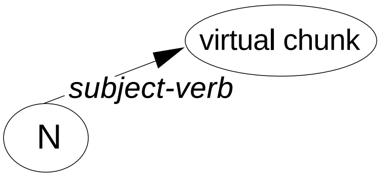
2.2.1 Subordinated chunks

Here is the beginning of a sentence to be parsed, where subject and verb are not contiguous :

> *Technology stocks, which had been among the most volatile sectors last week, were also very active ...*

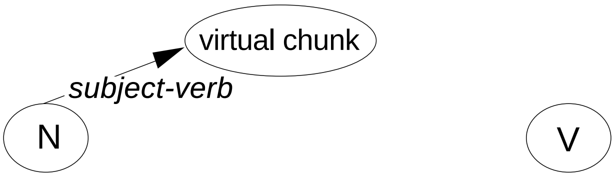
At this level, chunking is already done :

• at **step 1**, chunk $_i$ arrives : it is the first Nominal chunk of the sentence (*Technology stocks*), coded N in the diagram; it is linked to the virtual chunk; it is placed into a waiting position for a verbal chunk, which may arrive or not :

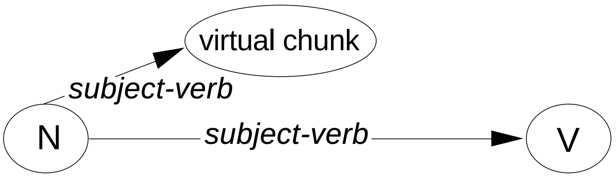


- then, between the two steps, the following chunks arrive : they are analysed as an embedded relative clause (*which had been among the most volatile sectors last week*);

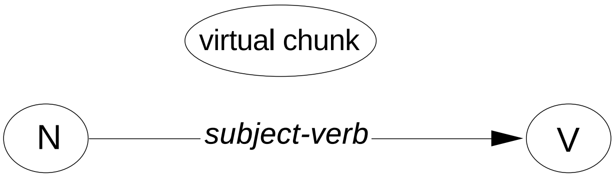
• at **step 2**, chunk $_j$ arrives : it is a Verbal chunk (*were also very active*), coded V in the diagram; a rule condition verifies whether a nominal chunk is linked to the virtual chunk by a subject-verb link :



- then, still in step 2, an action of this rule links subject and verb with a subject-verb link :



- at the end of step 2, the link between the subject and the virtual chunk is discarded, and that means here that this subject has found its verb and does not wait for a verb any more :



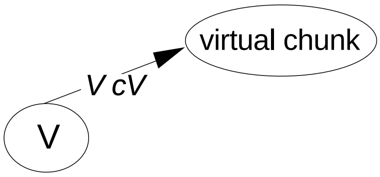
2.2.2 *Co-ordinated chunks*

Here is the beginning of a sentence to be parsed, which contains co-ordinated verbal chunks :

> *The original Pan Am World Airways began flying in 1927 and grew into one of the world's largest airlines ...*

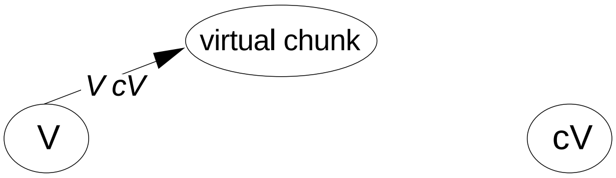
At this level, chunking is already done :

• at **step 1**, chunk $_i$ arrives : it is a Verbal conjugated chunk (*began flying*), coded V in the diagram; it is linked to the virtual chunk; it is placed into a waiting position for a co-ordinated verbal chunk, which may arrive or not :

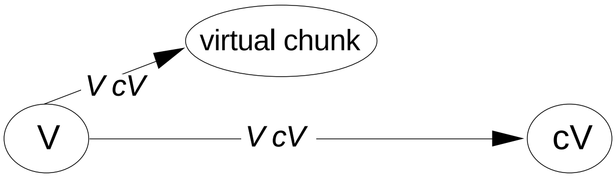


- then, between the two steps, the following chunk arrives : it is analysed as a prepositional nominal chunk (*in 1927*);

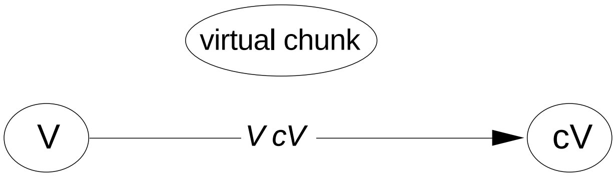
• at **step 2**, chunk $_j$ arrives : it is a co-ordinated Verbal chunk (*and grew*), coded cV in the diagram; a rule condition verifies whether a verbal conjugated chunk is linked to the virtual chunk by a V cV link :



- then, still in step 2, an action of this rule links both verbal conjugated chunks with a V cV link :



- at the end of step 2, the link between the first verbal chunk and the virtual chunk is discarded, and that means here that this verb has found its co-ordinated verb and does not wait for one any more :



2.3 Implementation in declarative rules and complexity

This process is implemented into 2 rules : the first is triggered while processing chunk $_i$ at step

1, the second is triggered while process chunk j at step 2. It is generalised for any type of link : for instance subject-verb, verb-object, co-ordination of any type of chunk, nominal chunk-relative clause.

The process consists in passing a constant number of rules of the form "conditions actions" once on every chunk and therefore has a linear complexity.

Conditions are on attributes of chunks and on links between chunks, and actions may give values to attributes, set or discard links. Chunks are processed in a flow, that is at a constant rate (a constant number of chunks per second). A chunk in the flow is completely processed, in a unique pass.

This linking algorithm allows to build parsers of linear complexity (without a formal grammar), which have the same main functions as formal grammar parsers : segmenting the flow into tokens, chunks, clauses and sentences, and linking these chunks.

3 Two parsers using this algorithm

This algorithm has been implemented in two linear parsers : the first won the GRACE contest for French taggers, and the other is used in an industrial context : in a text-to-speech system, to compute prosody.

3.1 The parser which competed in GRACE

From 1995 to 1998, the GRACE[1] contest had the aim to compare French taggers in a unique protocol. It was organised by two CNRS[2] laboratories : LIMSI and INaLF. The competitors where 14 laboratories (France : 8, Swiss : 2, Germany : 3, Canada : 1), and 8 companies (including IBM France, Xerox Grenoble, and AT&T Bell). The tagset came from the MULTEXT[3] tagset : 11 main categories, and 311 different tags.

We have presented our parser at this contest. Chunking allowed us to tag tokens inside chunks (it is more accurate). Linking chunks allowed us to give the correct tag to an extra 3% of tokens, the ones which are impossible to tag solely by the context inside a chunk (as "*de*", "*des*", "*du*", "*que*"), and ambiguous verbs which are alone in their chunk (as "*ferme*", "*montre*"), and it allowed us also to compute genders et numbers from subject-verb links and agreements. It was an important factor which gave us the first place.

The practical linear complexity of this parser is illustrated by the diagram in appendix : for a corpus of 2,500 sentences from the newspaper Le Monde, every sentence is represented by a point according to its number of words and its parsing time in seconds. The parsing rate is constant and foreseeable : about 0.3 seconds per word. The linear complexity is a property of the algorithm, and then, the constant parsing rate depends on the processor speed.

3.2 Parsing to compute prosody in a text-to-speech system

As Abney wrote (1990 and 1995), chunks are prosodic groups, and we made the hypothesis that pauses are proportional to the link length.

That is why a parser is necessary to compute prosody from chunks and chunk links, in a text-to-speech system which has to say long texts as papers and novels in an understandable way.

On the other hand, such a parser must have a linear complexity to work in the flow, faster than the speech rate.

Our parser with its linking algorithm has been included into such a system, named KALI[4], which is already marketed for French synthesis. The English synthesis is now in progress[5].

Conclusion

We have presented in this paper an algorithm for linking chunks, and this algorithm is of linear complexity in time. Thanks to this linking algorithm, the parser that we presented to the GRACE contest has been successfully evaluated comparatively to other parsers and taggers. This linking algorithm has been included into an industrial text-to-speech project which needed a parser to compute prosody in linear time.

Our team now participates to a new industrial project concerning linguistic processing on newsfeed on internet. A new parser has been developed, and a new rule formalism has been designed. It includes our linking algorithm, with a generalisation into two directions (as announced above at the end of the subsection 2.1) : 1) the level direction : it can link other types of constituents (clauses, sentences and paragraphs), and 2) the language direction : it parses French and English newsfeed. The results of this new parser are still to be evaluated.

[1] GRACE : **G**rammaires et **R**essources pour les **A**nalyseurs de **C**orpus et leur **E**valuation
`http://www.limsi.fr/TLP/grace/`

[2] CNRS : Centre National de la RechercheScientifique - France.

[3] `http://www.lpl.univ-aix.fr/projects/multext`

[4] Some wav files are available on :
`http://elsap1.unicaen.fr/demokali.html`

[5] This project has been partially financed with the FEDER European funds.

References

Abney S. (1991). *Parsing By Chunks*. In: Robert Berwick, Steven Abney and Carol Tenny (eds.), Principle-Based Parsing. Kluwer Academic Publishers, Dordrecht.

Abney S. (1995). *Chunks and Dependencies: Bringing Processing Evidence to Bear on Syntax*. In: Computational Linguistics and the Foundations of Linguistic Theory. CSLI. pp. 145-164.

Abney S. (1996). *Tagging and Partial Parsing.* In: Ken Church, Steve Young, and Gerrit Bloothooft (eds.), Corpus-Based Methods in Language and Speech. An ELSNET volume. Kluwer Academic Publishers, Dordrecht.

Aït-Mokhtar S. and Chanod J.-P. (1997) *Incremental Finite-State Parsing*. Proceedings of ANLP'97, Washington, pp.72-79.

Brill E. (1995) *Transformation-BasedError-Driven Learning and Natural Language Processing: A Case Study in Part of Speech Tagging*. Computational Linguistics.

Chanod J.-P. and Tapanainen P. (1995) Creating a tagset, lexicon and guesser for a French tagger. *ACL SIGDAT workshop on "From Texts To Tags: Issues In Multilingual Language Analysis".* Dublin, pp. 58-64.

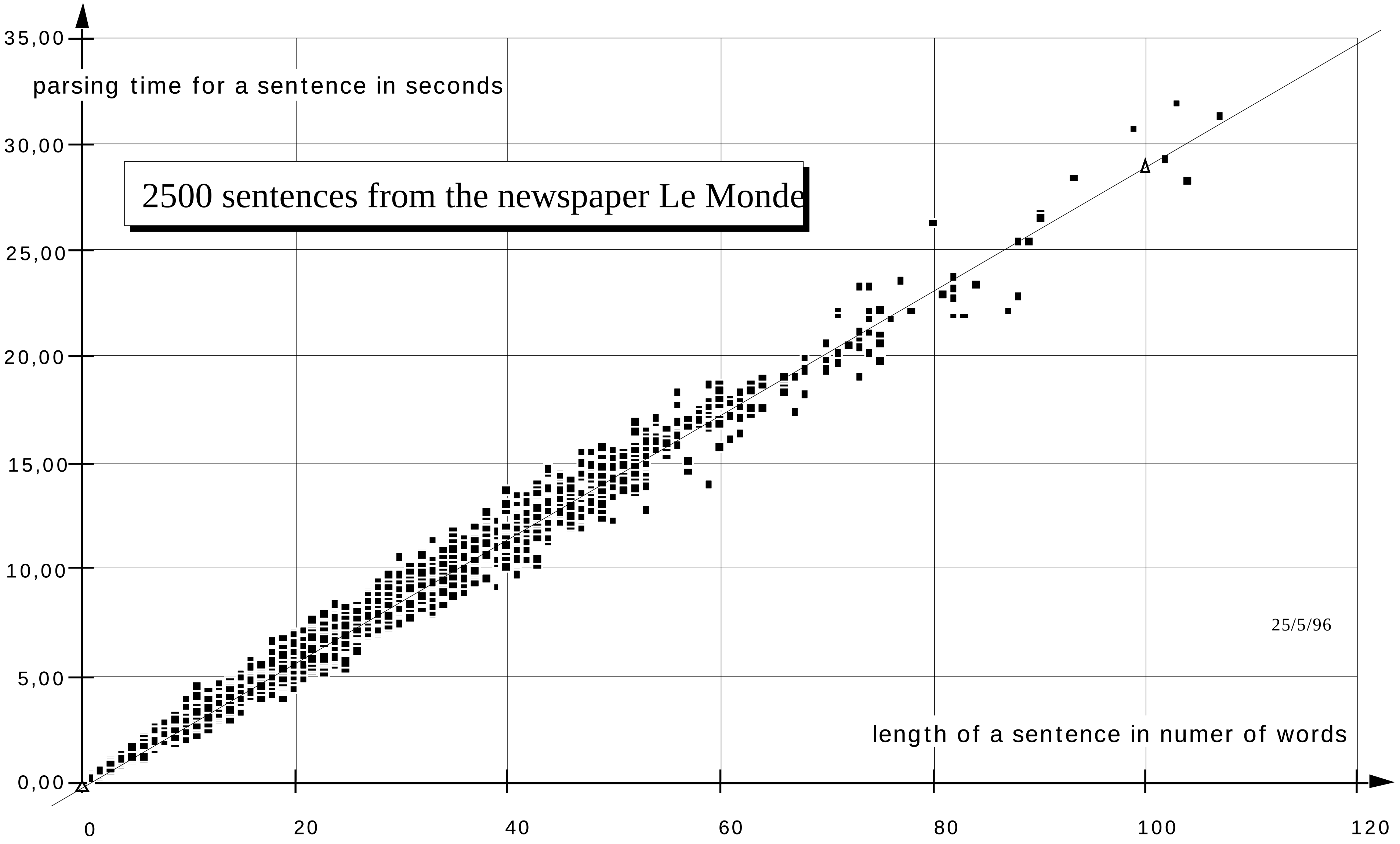
Chanod J.-P. and Tapanainen P. (1995) Tagging French - Comparing a Statistical and a Constraint-Based Method. Seventh Conference of the European Chapter of the ACL (EACL'95), Dublin, pp. 149-156. .

Giguet E. (1998) *Méthode pour l'analyse automatique de structures formelle sur document multilingues*. Ph.D thesis, l'Université de Caen.

Vergne J. and Giguet E. (1998) Regards Théoriques sur le “Tagging”. *Cinquième conférence annuelle : Le Traitement Automatique des Langues Naturelles*, TALN'98, Paris, pp. 22-31.

Appendix

(see last page)


35,00
30,00
25,00
20,00
15,00
10,00
5,00
0,00
parsing time for a sentence in seconds
2500 sentences from the newspaper Le Monde
25/5/96
length of a sentence in numer of words
0
20
40
60
80
100
120