Parsing and reading times: Testing Hale's Theory of Chunking in Human Language Parsing

Author: Davy Chen (5739772)
Supervisor and first assessor: Jakub Dotlačil
Second assessor: Frans Adriaans

Bachelor's Thesis Kunstmatige Intelligentie
Utrecht University
7.5 ECTS

Contents

Abstract

In the field of human language comprehension, Hale (2014) introduced a novel theory, which incorporates the chunking theory of learning into human language parsing. According to the chunking theory of learning, oft-repeated subtasks in cognitive processes are sped up by being combined into a single operation. Given that parsing language is an oft-practiced skill, it would make sense that the chunking theory of learning also applies to language parsing. To test this theory, Hale implements a left-corner parser, which previous research claims suitable for parsing natural language. He then creates a metric called cohesion, the log-likelihood ratio of parser-operations. Hale found a significant effect of cohesion on reading time. Chunks that appear often are read faster than chunks that are rarer. In this thesis, Hale's experiment has been repeated, with some improvements. The found results are similar to Hale's results, providing additional evidence for the relevance of the chunking theory of learning in human language parsing.

1 Introduction

Natural language is a complex system of rules and axioms. Most people, however, can easily understand at least one language fluently: no second thought is needed to comprehend most sentences, whether it is written or verbal. Despite the ease at which language can be understood, a great number of mechanisms underlie the process of reading sentences, breaking them down into appropriate grammar constructions, et cetera. This process is called parsing. One intuitive mechanism that could steer parsing is experience and memory: language is acquired through a long learning process, and memory naturally plays a role in this process.

The effect of memory on parsing is well-researched. Research on the topic has shown significant memory-related effects on parsing difficulty, expressed in the amount of time it takes to read a certain word or sentence. Some metrics that have been studied include word probability (Smith & Levy, 2013), and n-gram surprisal, the logarithm of the reciprocal of the probability of an n-gram (Levy, 2008). These studies have delved into effects on the word-level, as well as the grammar-level.

Hale (2014), in his book *Automaton Theories of Human Sentence Comprehension*, proposes a novel theory in this field of research. Instead of focusing on effects on the word or grammar-level, he theorises that memory affects parsing on the most basal level, namely, the parser-level. Any given parser uses several types of operations to build a parse of a sentence. Hale states that the strength of memory-embeddings of parser-operations could affect parsing decisions. Operations which happen frequently, and which are thus well-embedded in memory, will be processed more readily than rare counterparts. Implementing a parsing strategy called the *left-corner* strategy that previous research has forwarded as a plausible human parsing strategy, Hale carries out an experiment to confirm his theory. He creates a new metric called "cohesion" that is based on the rarity of parser-operations, and a significant effect of this metric on reading time was found.

Given that Hale's theory is still fairly novel, no further research has been carried out to confirm his findings. In this thesis, Hale's experiment will be replicated in an attempt to find additional evidence that the frequency of parser-operations indeed affect reading times. Some theoretical background and additional information must be given first, however. In chapter 2, it will be explained how natural languages can be parsed by a computer, including a description of the left-corner parser that Hale utilised. In chapter 3, the cohesion metric that Hale created, and the theory behind it, will be described. In chapter 4, the specifics of both Hale's experiment, and current replication will be laid out. Finally, chapters 5, 6, and 7 pertain to the implementation of the experiment and the results of the experiment.

A confirmation of Hale's would explain in part what mechanisms underlie human

parsing. If human language processing could be accurately modelled using left-corner parsers, further research could be performed with this model in mind. This theory could furthermore have implications for AI that are based on human cognition, as it furthers the understanding of human cognition. If Hale's theory holds true, text-generating-AI could also be adjusted to create easy to read text using his chunk-cohesion metric. Lastly, if cohesion is found to be a good predictor for human parsing performance, computer-implementations of parsers could use this metric to determine which operation to select for successful parsing.

2 Parsing natural language

2.1 Context-free grammars

While natural languages are easy to understand for humans, these languages must first be formalised before a computer can properly parse them. The syntax and components of any given formal grammar date back to at least Chomsky (1956), and there exist several formal grammars to choose from. For the purposes of this study, the *context-free grammar* is chosen as the computer-representation of a natural grammar. A context-free grammar consists of the following components:

N: A set of *non-terminal* symbols, represented by capital letters, which contains at least S, the start symbol.

Σ: A set of *terminal* symbols, represented by lower-case letters.

Production rules: A set of rules in the form $A \rightarrow \alpha$, where A is any single non-terminal, and α is a combination of terminals and non-terminals of arbitrary length. The symbol(s) to the left of the arrow is called the left-hand side, while the symbol(s) to the right are called the right-hand side.

The language L belonging to this grammar is then all the combinations of terminals that can be made by starting with the start symbol S, iteratively matching non-terminals with left-hand sides of productions and replacing the used non-terminal with the right-hand side of the found production, until no non-terminals remain.

The context-free grammar is not an ideal grammar for natural language. There exist several languages that cannot be properly expressed using these simple rules alone, including Dutch (Huybregts, 1984), Swiss German (Shieber, 1985), and Bambara (Culy, 1985). However, for most of the major human languages, context-free grammars seem to hold. The simplicity of this grammar also has an upside: the rest of the model is not unnecessarily complicated due to the complexity of the grammar.

2.2 Pushdown automata

Now that a formal notion of grammar has been described, a form of computation that could parse this grammar must be provided. The computational model that will be used is called the automaton. For the class of context-free grammars, there exists a type of automaton called the pushdown automaton. For any given context-free grammar and its language L, there exists some pushdown automata that accepts the language L. A pushdown automaton is identical in basic setup to a finite-state machine: an input string starts at the start state, and at the first character in the input string. If a transition is found that requires a symbol(s) equivalent to the current character(s) in the input string, then the automaton progresses to the state specified in the transition, shifting the input string ahead exactly the symbol(s) found. This is repeated until an accepting state is reached and the input string is accepted, or until no further progress can be made and the string is rejected. Note that upon reaching an accepting state, the machine does not necessarily have to halt, further transitions can be taken. The pushdown automaton is further enhanced with a stack memory, and transitions that push and pop elements from this stack.

Semi-Formally a pushdown automaton consists of:

S: A set of states, containing at least a start state. A subset of S must be the set of accepting states.

Σ: The set of input symbols, the input itself is a string of arbitrary length containing only these input symbols.

Γ: The set of symbols that can go onto the stack memory

A stack: containing zero or more initial symbols, symbols are put on the stack in a LIFO manner. Putting an item on the stack is called a "push", taking an item from the stack is called a "pop".

δ: A set of transition rules, of the form (s, σ,γ, s', γ*), where s is the required state before transitioning, σ is the required current symbol at the top of the input string, γ the required element popped from the stack, s' the state after transitioning, and γ* the symbols to put on the stack after transitioning.

The relation between a context-free grammar and a pushdown automaton might not directly be apparent, however, it becomes clearer once one imagines that the set of transitions can be productions as seen in the context-free grammar, with σ being lexical elements, γ being the left-hand sides of productions, and $\gamma*$ being the right-hand sides of productions.

While pushdown automata can be created that accept the same input strings as a context-free grammar G, the specific parsing-strategy to implement still has not

been described, and it is this strategy that determines what contents several of the component sets of a certain pushdown automaton receives. Multiple strategies can be considered, two well-known ones are the bottom-up, and top-down strategies of parsing. A less-common strategy called the left-corner strategy exists, which some consider a strategy that mimics human language parsing. They will be summarised, and a case for the left-corner parser will be made afterwards.

2.3 Bottom-up parsing

The bottom-up parser, as the name suggests, builds a parse from the *bottom* of a syntax tree. Parsing starts with leaves, terminals, of this parse tree until a sequence of symbols has been found which forms the right-hand side of a production in the grammar. The left-hand side of this production then becomes a symbol of its own, being able to form sequences with other terminals and non-terminals to find matching right-hand sides again. This continues until the list of input symbols is exhausted. The exact implementation of bottom-up parsers may differ, but the key element stays consistent. One starts with the evidence (words, lexical elements), and builds productions by matching strings of evidence with right-hand sides of productions.

2.4 Top-down parsing

The top-down parser is the opposite of the bottom-up parser. In this strategy, parse trees are not built from the bottom, but from the top. Parsing starts by taking a production, and recursively evaluating non-terminals on the right-hand side, until a sequence has been found which matches a given input. Again, the exact implementation of top-down parsers may differ, but all have in common that they parse by deducing what *should* be seen in the future.

2.5 Left-corner parsing

The left-corner parser is named after the way it searches for evidence for the next production to consider. The left-corner parser only looks for *partial* matches of productions. The first symbol on the right-hand side of any production is called the left-corner. The left-corner of any production must be found first before that production may be used. If such a production is considered, then the symbols that come after the left-corner must be evaluated, or expected. Since it is this strategy that will be used in this research, a description of the parser implementation will be given.

Given: a stack of parser-symbols S , initially only containing an "*expectation*" of the start symbol of the language.
And a queue Q, a FIFO data type, containing the lexical elements belonging to a certain sentence.

The following operators are available to the parser:

Complete: If a "*symbol*" Y is seen on top of the stack, and the element below it is an "*expectation*" X, and there exists a production X $\rightarrow$ Yα, then remove both X and Y from the stack. Push each right-hand side symbol in $\alpha = Z_1, Z_2...$, onto the stack as an "*expectation*".

Project: If a "*symbol*" Y is seen on top of the stack, and there exists a production X $\rightarrow$ Yα, then remove Y from the stack. First, push X onto the stack as a "*symbol*", then, push each remaining right-hand side symbol in $\alpha = Z_1, Z_2...$, onto the stack as an "*expectation*".

Shift: The default step. If neither of the other two actions can be performed, then dequeue the next lexical element from the queue Q, and push this as a "*symbol*" on top of the parser stack S.

If, after using a combination of these operators, both the lexical queue Q and the parser stack S are empty, then the parse is succesful.

This parsing strategy combines elements from both the bottom-up, and top-down approaches. It is bottom-up, due to the fact that it looks for current evidence in the right-hand side of productions: the left-corner must match a symbol on top of the parse stack. And it is top-down due to the fact that it attempts to evaluate the remaining symbols that come after the left-corner: symbols that come after the left-corner are expected to be seen in the future.

2.6 The case for left-corner parsers

Previous research (Resnik, 1992; Johnson-Laird, 1983) on this topic concludes that neither bottom-up parsing, nor top-down parsing accurately represents human sentence parsing. Resnik and Johnson-Laird state that the memory-usage patterns of both top-down and bottom-up parsers are not consistent with human performance. Each of these parser strategies struggle given certain deeply-nested syntactic structures. If humans were to employ top-down parsing, they would find it difficult to parse left-branching structures. Vice versa, if humans were to employ bottom-up parsing, they would find it difficult to parse right-branching structures. However, neither left-branching, nor right-branching structures have been found to be particularly troublesome for humans to parse. Examples of both structures can be seen below by Resnik (1992, p.1), with an additional *centre-embedded* construction.

Example left-branching contruction: [[[John's] brother's] cat] despises rats.

Example right-branching construction: This is [the dog that chased [the cat that bit [the rat that ate the cheese]]]

Example centre-embedded construction: #[The rat that [the cat that [the dog] chased] bit] ate the cheese.

While neither left-branching, nor right-branching constructions form a problem for human parsing, centre-embedded structures have been found to be troublesome (Chomsky & Miller 1963), since too many sub-structures must be remembered before parsing may conclude. Resnik shows that left-corner parsers mimic these characteristics. Both left-branching and right-branching structures only require O(1) memory complexity in left-corner parsers, however, centre-embedded structures require O(N) complexity. Hale chooses the left-corner parser as the parser of choice for his later experiment, stating that a parser incorporating the left-corner strategy might be one that mimics human language processing. The theory that left-corner parsers are related in some capacity to human cognition dates back at least several decades (Crocker, 1999).

3 Cohesion

Both the computational model, and the implementation of the parser have been described. However, to test whether the left-corner parser can accurately model human parsing, a metric must be devised that can somehow tie parser-operations to human research data. Hale builds on earlier research, applying information-theoretical principles to parser-operations to create a metric that is based on the chunking theory of learning.

3.1 Chunking Theory of Learning

Language, as a frequently used skill with a wide scope, would naturally fit the chunking theory of learning. This theory is built upon the observation that the time required to perform a task decreases as the frequency of performing the task increases. (Rosenbloom & Newell, 1987) The authors state that this effect could be explained by the *chunking* effect. Subtasks of oft-repeated cognitive processes are combined to form a new operation which is more efficient than carrying out the subtasks on their own. This chunking effect becomes stronger the more frequent a certain task is carried out.

3.2 Cohesion

Hale points out that chunking would be a natural fit for parser-operations in humans. The application of chunking to these operators would not change the ways sentences are parsed, but they do change which operations are bundled together as a cognitive process to speed up parsing. Sequences of operations that are often seen in sequence (for example, sequences containing an operation on a determiner, followed by an operation to a noun) should be candidates to be chunked, while sequences that are rarely seen will not be chunked. In other words, text that is parsed with operations that appear in sequence often should be read faster than text that has to go through abundant rare operations.

To test this theory Hale creates the metric known as cohesion. Cohesion is the name for log-likelihood ratios of parser-operation n-grams. Log-likelihood ratios are a form of hypothesis testing, in this case, where one hypothesis is that all parser-operations in an n-gram of operations are independent, and where the second hypothesis states that these operations are not independent. The bigger the log-likelihood ratio, the more likely it is that the operations in the n-gram are *not* independent. Henceforth, the word "chunk" will be used as a synonym for parser-operation n-gram.

In the following formula, the formula for log-likelihood ratios for lexical bigrams given by Manning and Schütze (1999) is adapted to parser-operations. The symbol α_i will be used to denote parser-operations. Given a chunk of size two denoted by $\alpha_1\alpha_2$, where α_1 is the first parser-operation in this chunk, and α_2, the second parser-operation in this chunk. The formula for chunks of size two is as follows, with c_1, c_2, c_{12}, the frequencies for parser-operations α_1, α_2, $\alpha_1\alpha_2$ respectively and N, the combined count of all operations[1]:

Given: $p = \frac{c_2}{N} \quad p_1 = \frac{c_{12}}{c_1} \quad p_2 = \frac{c_2 - c_{12}}{N - c_1}$

Assuming a binomial distribution:

$$b(k; n, x) = \binom{n}{k} x^k (1-x)^{(n-k)}$$

Then the formula for cohesion is as follows:

$$\begin{aligned} &= -2\ log \frac{b(c_{12}, c_1, p) b(c_2 - c_{12}, N - c_1, p)}{b(c_{12}, c_1, p_1) b(c_2 - c_{12}, N - c_1, p_2)} \\ &= -2(log\ L(c_{12}, c_1, p) + log\ L(c_2 - c_{12}, N - c_1, p) \\ &\quad - log\ L(c_{12}, c_1, p_1) - log\ L(c_2 - c_{12}, N - c_1, p_2)) \end{aligned}$$

where:

$$L(k, n, x) = x^k (1-x)^{n-k}$$

Using this cohesion metric, scores can be assigned to chunks of parser-operations. Chunks with a high score possess high cohesion: they are likely to be chunked. Chunks with a low score are less likely to be chunked. If Hale's theory holds, there should be a correlation when comparing cohesion of text to human reading times of said text. Text with high cohesion scores should have lower reading times than text with low cohesion scores.

[1] In the experiment, 1 is added to the denominator of p_1, and 1 is added to the numerator of p_2, this is to avoid any division by zero and taking the logarithm of zero issues. This has very little effect on cohesion numbers, and a negligible effect on the results of the experiment.

4 Methodology

4.1 Hale's experiment

Using both the left-corner parser, and the cohesion metric, Hale performed an experiment that attempted to compare cohesion and reading times. Hale used two separate corpora for specific purposes. The Penn Treebank was used to build a cohesion database, and the Dundee corpus was used to determine reading time per chunk. Both corpora were parsed as provided using an implementation of the left-corner parser. The chunk size used was three, parser-operations within these chunks note which production they used in the operation, and shift-operations note the POS tag, not the word, belonging to the word shifted. This leads to a great amount of unique chunks. Then, all chunks generated from the Dundee corpus for which the *middle* action is a shift of a certain word are chosen for analysis.

This experiment falls short on several points. Firstly, the Dundee corpus does not include hand-annotated parse trees or POS tags. Hale used the Stanford parser to automatically generate such parse trees. Such parsers make more mistakes than human experts. Since some chunks are wrongly tagged, they will receive cohesion numbers that do not actually belong to the reading time provided. This is troublesome, as the mismatched cohesion/reading time numbers will introduce noise into the data.

Secondly, Hale parsed trees as provided by both corpora. The use of parse-trees as-provided is problematic: it introduces large amounts of ambiguity and backtracking. However, there exists evidence that a transformation called *binarisation* brings these parse trees closer to ones seen in human language parsing. (Roark, 2001; Roark & Johnson, 2000).

Lastly, for reasons unexplained, Hale only decides to use a small subset of all chunks. It is likely that the chunking effect is strongest in the "shift chunks" Hale used, however, if the effect holds even for chunks that are less directly relevant to the word in question, a stronger case can be made for the chunking theory in human language parsing.

In the next section, improvements on these issues will be discussed.

4.2 Improvements on Hale's experiment.

In this thesis, the framework for Hale's experiment will be kept identical. Two corpora will be parsed, one to build a cohesion database, and the second to compare reading times to cohesion of chunks. However, the aforementioned issues will be addressed in this research.

Firstly, the Dundee corpus will be exchanged for another corpus that does include human-annotated parse trees.

Secondly, parse trees included in both the corpus used for the cohesion database, as well as the trees provided in the reading-time-corpus, will be binarised and CNF-transformed.

Lastly, more chunks will be considered, not only ones which include a shift operation.

One design choice that is not necessarily an improvement is the chunk size. In this experiment, the chunk size used will be two. There exist a staggering amount of different chunks, and the databases will not be sufficient to calculate cohesion numbers for all of them: they simply do not exist in the corpus. This effect worsens with chunk size. To avoid this data sparsity problems, the lowest chunk size will be used.

4.3 Data

For this study, two different corpora have been employed. The first of these is the Penn Treebank, used for the construction of the cohesion database. The second corpus is the Natural Stories corpus. Using this corpus, chunks have been tied to reading times.

The Penn Treebank (Marcus, Santorini, Marcinkiewicz, 1993) consists of a wide array of articles extracted from 1989 Wall Street Journal publications. Additionally, it also contains the entirety of the Brown corpus. The PTB provides hand-annotated parses of these texts, including POS tags and parse trees. Usually, an annotated version of the brown corpus is included in this treebank, however, the brown corpus part of the PTB has not been used in this thesis. This restricted version of the Penn Treebank still included around a million words.

The Natural Stories corpus (Futrell et al., 2017) consists of ten purposefully-created stories. This corpus provides self-paced reading time data, alongside hand-corrected parses of these stories, including POS tags and parse trees. The reading times of individual words are tracked across ten participants. This corpus is significantly smaller than the Penn Treebank, consisting of only 10.257 words. Furthermore, this corpus provides additional metrics for the included words: word frequency and maximum likelihood estimations for bigrams, trigrams, and quadrigrams. Word frequencies and the maximum likelihood estimates are extracted from the Google Books English corpus. From this corpus, data from the year 1990 to 2017 was used.

The Natural Stories corpus was designed with low-frequency syntactic structures in mind. Many reading time corpora are based on naturalistic text. While this gives an accurate cross-section of reading times of oft-occurring linguistic elements, interesting findings might be found in the low-frequency structures. According to the authors of the Natural Stories corpus (Futrell et al., 2017, p.9), it contains "especially high rates of nonlocal VP conjunction, nonrestrictive SRCs,

idioms, adjective conjunction, noncanonical ORCs, local NP/S ambiguities, and it-clefts" compared to the Dundee corpus.

However, this strength is also a weakness: some low-frequency chunks might not exist in the cohesion-database. Moreover, it might be the case that cohesion is a good predictor for rarer syntactic structures, but not for common syntactic structures. Unfortunately, there is no easy method to determine the rarity of a particular syntactic structure, given that for the rare structures, the database will be sparse. There simply exists insufficient data to determine ground-truth rarity for certain chunks.

4.4 Method

A database of cohesion numbers was extracted from the Penn Treebank using an implementation of the left-corner parser. Using the cohesion formula as described earlier in this thesis, the cohesion for chunks of size two has been calculated. In total, around 220.000 unique bigrams of parser-operations are included in this database.

The cohesion numbers in this database were later used to determine the cohesion scores of chunks in the Natural Stories corpus. This corpus, like the Penn Treebank, has been parsed using the left-corner parser. After parsing, the list of parser-operations were divided into chunks of size two, with overlap. Chunks where the *first* action of the bigram was a shift action, were considered to signal the start of the next word, i.e., the cohesion numbers of such chunks, and the chunks that followed, were considered to be part of the next word. Punctuation was always considered part of the previous word, unless the punctuation was parsed before any words were shifted. In this case, the punctuation was part of the next word. The final cohesion number of a word was the mean of cohesion of all chunks belonging to a certain word. If a chunk could not be found in the database, the word the chunk was part of was discarded. In total, 2.858 out of 10.254 words were disqualified for this reason.[2]

An ordinary least squares linear regression was performed on this data, with cohesion as the central independent variable and reading time in milliseconds as dependent variable. Several combinations of predictors were additionally analysed: cohesion, log cohesion (due to the large difference in values between cohesion numbers), log word frequency, bigram surprisal, word length, as well as the interaction between length and word frequency. These other metrics are well-researched predictors for reading times (Hale, 2011; Levy, 2008; Smith & Levy, 2013). Furthermore, one additional dependent variables was analysed: the logarithm of the geometric mean reading time.

[2]For the analysis of bigram surprisal included later, another 100 words were excluded, since the MLE of these bigrams could not be determined.

5 Results

In Table 1, some examples of cohesion numbers for chunks of size two are listed. Note the high cohesion for chunks which involve common POS tags. A determiner shift followed by a noun phrase projection intuitively seems a common chunk. Meanwhile, chunks which feature highly nested (which are denoted by a high amount of carets/hats, introduced by binarisation) syntactic structures, or chunks which feature rare POS tags (for example, *NX*) have low cohesion.

Chunk	Cohesion
('shift', DT), ('project', NP ->DT NPˆDT)	517204.2
('shift', IN), ('project', PP ->IN PPˆIN)	363123.1
('shift', VBD), ('project', VP ->VBD VPˆVBD)	346959.5
('shift', VB), ('project', VP ->VB VPˆVB)	301342.2
('shift', VBZ), ('project', VP ->VBZ VPˆVBZ)	264303.7
(('complete', PP-MNRˆIN ->S-NOM), ('complete', VPˆADVPˆVBNˆNP ->PP-MNR))	14.1
(('project', NX ->-NONE-), ('complete', NPˆDTˆJJ ->NX))	11.9
(('project', SBARQ ->WHADVP SBARQˆWHADVP), ('shift', VBP))	7.8
(('shift', NNS), ('complete', NP-LGSˆJJSˆNNPˆNNP ->NNS))	5.7
(('complete', Sˆ“ˆSˆ, ->CC Sˆ“ˆSˆ,ˆCC), ('shift', MD))	1.3

Table 1: Example chunks from the PTB cohesion database

Both cohesion and log cohesion were found to be significant predictors for reading times. The individual analyses are noted below. A scatter plot of cohesion and mean reading times can be seen in figure 1.

Predictor	Coefficient	Std. error	p value	
Intercept	340.9442	0.827		$R^2 = 0.002$
Cohesion	-2.175e-05	5.39e-06	<0.001	

Table 2: Cohesion as sole predictor for reading time

Predictor	Coefficient	Std. error	p value	
Intercept	354.4908	3.261		$R^2 = 0.003$
Log cohesion	-1.4888	0.298	<0.001	

Table 3: Log cohesion as sole predictor for reading time

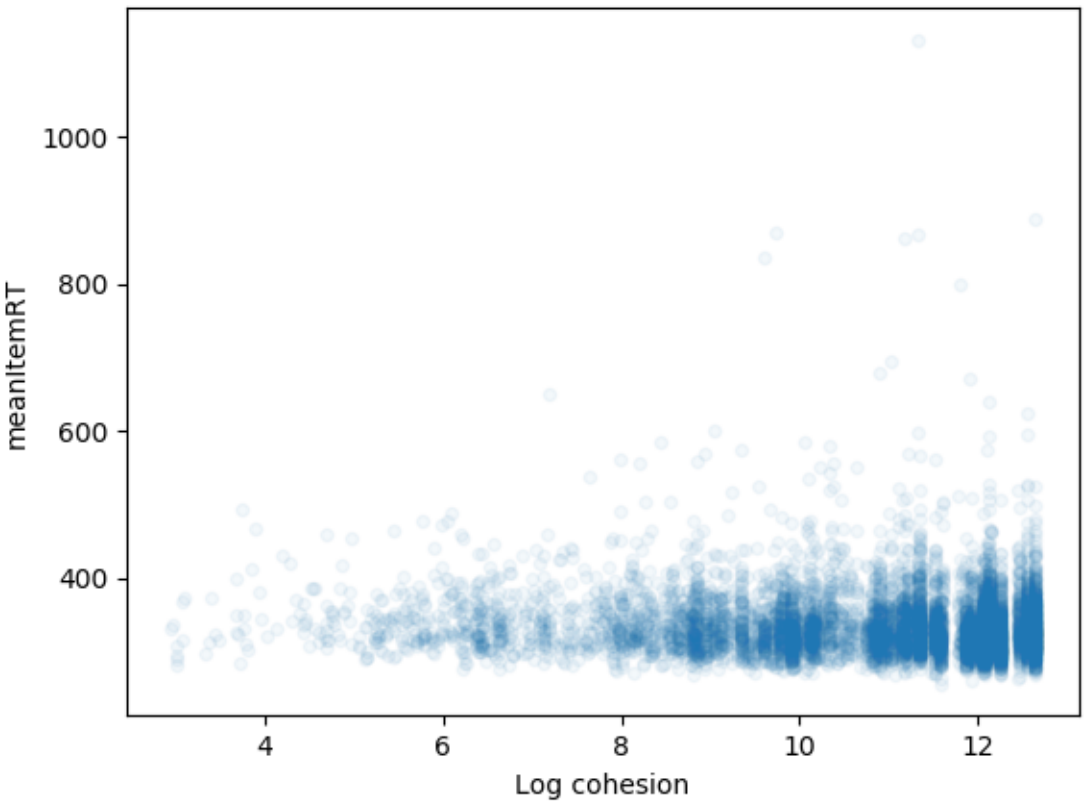

Figure 1: A scatter plot with log cohesion on the x axis, mean reading time on the y axis

Even after accounting for of the co-predictors, cohesion still remained a significant negative predictor of reading time, as can be seen in table 4. The combination of predictors listed below had the best R^2 value of all predictor combinations. For this analysis, the dependent variable has been changed to the logarithm of the geometric mean reading time. To determine the contribution of cohesion to these predictors, the results of a separate analysis without cohesion has been included in table 5. Additionally, the results of the model including all predictors has been included in table 7, note that, due to the inclusion of frequency, bigram surprisal becomes an insignificant predictor.[3]

[3]Bigram surprisal, without including word frequency as a co-predictor, remains a significant predictor, the result for this analysis hasn't been included since it is not relevant to the research at hand.

Predictor	Coefficient	Std. error	p value	
Intercept	5.7333	0.827		
Log cohesion	-0.0044	0.001	<0.001	$R^2 = 0.169$
Length	0.0284	0.002	<0.001	
Length:log frequency	-0.0011	<0.001	<0.001	

Table 4: Cohesion along with co-predictors as predictors for the logarithm of the geometric mean reading time.

Predictor	Coefficient	Std. error	p value	
Intercept	5.6941	0.003		
Length	0.0284	0.001	<0.001	$R^2 = 0.162$
Length:log frequency	-0.0011	<0.001	<0.001	

Table 5: Length, and the interaction between length and the log of word frequency as predictors for the logarithm of the geometric mean reading time

Predictor	Coefficient	Std. error	p value	
Intercept	5.6941	0.003		
Log cohesion	-0.0041	0.001	<0.001	
Length	0.0100	0.001	<0.001	$R^2 = 0.140$
Log frequency	-0.0037	0.001	<0.001	
Bigram surprisal	0.0003	<0.001	0.532	

Table 6: Log cohesion, length, log frequency, bigram surprisal as predictors for the logarithm of the geometric reading time

6 Discussison

6.1 Results

In the results of this experiment, a significant effect was found between cohesion and reading times. Both cohesion itself, and log-cohesion were found to be significant negative predictors of reading times. Moreover, the found coefficients for cohesion were in the same order of magnitude as Hale's results, both before, and after accounting for co-predictors.

The effect is fairly small, with an R^2 value below 0.01, meaning that in the current analysis, only a small part of the variance can be explained through cohesion alone. Indeed, the difference in R^2 values between the analyses using cohesion and co-predictors both (table 4), and the analysis only using the co-predictors (table 5), is only slight. This means that cohesion only made a small contribution to the predictive power of the model. However, this low R^2 value can in part be explained by the great variability of human reading time itself. Such reading time data is inherently noisy due to the great number of factors determining reading times. Some of these factors are present in the methodology of the creation of such corpora. For example, participants must not only read, but also read the text carefully to answer questions that follow the text. Participants must also press buttons to load the next word. Other factors have a more human-general cause:

humans may find themselves distracted or unfocused during such reading tasks, which are not easily modelled using linguistic/information-theoretical metrics alone.

Despite the small R^2 value, the small p-value signals the existence of an effect of cohesion on reading times. The fact that this metric remains significant even when other predictors are accounted for suggests that cohesion is not explained by well-known effects. Furthermore, the results suggest that the cohesion of parser-operator chunks, a metric measuring how well said chunk should be embedded in memory, plays a role in human language processing.

6.2 Future work and improvements

While this thesis confirms the findings of Hale in a generalised analysis, more research has to be done on this topic to reinforce Hale's chunking theory. A significant link has been found between cohesion and reading times, however, the R^2 value is low. Future work can be improved in several ways, these will be discussed below.

A great amount of words in the Natural Stories corpus had to be removed from the analysis, due to the presence of chunks which do not exist in the cohesion-database. This can be remedied by using, or creating, a larger database from which the cohesion is calculated. For example, multiple corpora could be combined to create one large database. This has the added effect of including more different styles of text within the database. The Penn Treebank used in this thesis only contains text from the Wall Street Journal. This, however, means that the database mostly centres around news articles. This may bias the cohesion database to chunks that are seen often in such articles, while leaving beside other writing styles.

While the Natural Stories corpus attempts to improve on previous self-paced-reading-task corpora, it is far from perfect. The study includes only a small number of participants, ten in total. The variance between individual participants still plays a large role in the mean reading times.

This study only considers chunks of size two, the found effect might differ for chunks of greater size. However, a larger database must be built to avoid data sparsity problems.

Both Hale's work and this thesis only focuses on one type of parser: the left-corner parser. Many other types of parsers exist. However, using log-likelihood on parser-operations is still a fairly novel idea. Despite the theory that left-corner parsers model human parsing well, other parser types might perhaps see better results for this particular metric.

Current work has focused on English. Hale has also carried out the experiment on

a French corpus. Results for other languages do not exist yet, and testing the theory on other languages will provide a clearer picture of the effect on human languages in general.

7 Conclusion

In this thesis, Hale's experiment on cohesion and reading times has been replicated. Several shortcomings have been addressed: the amount of chunks used has been greatly increased, a hand-annotated reading-time corpus has been used, parse trees are first binarised and CNF-transformed, furthermore, more low-frequency syntactic structures are considered. Even with these changes, a result similar to Hale's finding has been found. While the found effect is smaller than in Hale's work, it still resides firmly within the same order of magnitude. This does not necessarily prove Hale's theory, however, it does provide evidence that the chunking theory of learning is relevant in human parsing. The cohesion metric, when applied to parser-operation chunks, has been found to be a metric with significance. This metric could be used in future studies comparing reading times and cohesion for other parsers, other corpora, or other languages.

8 Implementation

For this research, Python was used to implement the parser and to perform the analysis. The Python library "Natural Language Toolkit" was used to aid the writing of the parser. This library provides classes and methods for formal grammars and parse trees. For the binarisation and CNF-transformation of the trees, code was provided by the supervisor for this thesis, Jakub Dotlačil. The cohesion formula was implemented without further library usage. "Pandas" Dataframes were used to prepare data for the regression, for which the library "Statsmodels" was used. This library provided regression coefficients, p-values, and R^2 values. Matplotlib was used for the graph included in this thesis.

9 Citations

Hale, J. T. (2014). Automaton theories of human sentence comprehension. Center for the Study of Language and Information.

Smith, N. J., & Levy, R. (2013). The effect of word predictability on reading time is logarithmic. Cognition, 128(3), 302-319.

Levy, R. (2008). Expectation-based syntactic comprehension. Cognition, 106(3), 1126-1177.

Chomsky, N. (1956). Three models for the description of language. IRE Transactions on information theory, 2(3), 113-124.

Huybregts, R. (1984). The weak inadequacy of context-free phrase structure grammars. Van periferie naar kern, 81-99.

Shieber, S. M. (1985). Evidence against the context-freeness of natural language. In Philosophy, Language, and Artificial Intelligence (pp. 79-89). Springer, Dordrecht.

Culy, C. (1985). The complexity of the vocabulary of Bambara. Linguistics and philosophy, 8(3), 345-351.

Resnik, P. (1992, August). Left-corner parsing and psychological plausibility. In Proceedings of the 14th conference on Computational linguistics-Volume 1 (pp. 191-197). Association for Computational Linguistics.

Johnson-Laird, P. N. (1983). Mental models: Towards a cognitive science of language, inference, and consciousness (No. 6). Harvard University Press.

Miller, G. A., & Chomsky, N. (1963). Finitary models of language users.

Crocker, M. W. (1999). Mechanisms for sentence processing. Language processing, 191-232.

Rosenbloom, P., & Newell, A. (1987). Learning by chunking: A production system model of practice. Production system models of learning and development, 221-286.

Manning, C. D., Manning, C. D., &, H. (1999). Foundations of statistical natural language processing. MIT press.

Roark, B. (2001). Probabilistic top-down parsing and language modeling. Computational linguistics, 27(2), 249-276.

Roark, B., & Johnson, M. (2000). Efficient probabilistic top-down and left-corner parsing. arXiv preprint cs/0008017.

Hale, J. (2001, June). A probabilistic Earley parser as a psycholinguistic model. In Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies (pp. 1-8). Association for Computational Linguistics.

Marcus, M., Santorini, B., & Marcinkiewicz, M. A. (1993). Building a large annotated corpus of English: The Penn Treebank.

Futrell, R., Gibson, E., Tily, H., Blank, I., Vishnevetsky, A., Piantadosi, S. T., & Fedorenko, E. (2017). The natural stories corpus. arXiv preprint arXiv:1708.05763.