

How to use segmentation

This document explains how to use the EXMARaLDA Partitur-Editor's built-in finite state machine segmentation algorithms.

No previous knowledge on algorithms, finite state machines or regular languages is needed to understand the contents of the first two parts of this document as all relevant aspects will be explained. However, before you start reading this document, you should read

- Understanding the basics of EXMARaLDA

Contents

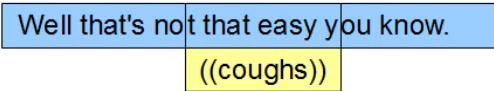
1. About EXMARaLDA and segmentation	2
Basic and segmented transcriptions.....	2
Segments in transcription systems	4
2. The segmentation algorithms.....	5
How it works	5
Implications for transcription	5
3. Segmentation in the Partitur-Editor	6
Segmentation options	6
Segmentation errors	9
Working with error lists	11
Exporting segmented transcriptions.....	11
Performing a Segment count	11
Generating a Word list.....	11
Appendix: GAT2 Transcription Conventions for the minimal transcript	13

1. About EXMARaLDA and segmentation

Basic and segmented transcriptions

Strictly there is not *the* EXMARaLDA transcription format, but two formats – the Basic transcription format and the Segmented transcription format. The transcription created in the Partitur-Editor is an EXMARaLDA Basic transcription (hence the file extension .exb¹). Although the events with their boundaries occur as segments, they are merely visually segments. The timeline items or time points are used to organize the different events – which event occurs before which, which events occur at the same time, which events (partly) overlap or include others etc. There is no linguistic or otherwise transcription-relevant meaning apart from this organizational information to these events in the basic transcription. This means, that even if you create one event for each word, these boundaries will not be recognized as word boundaries. The segmentation process will always recognize the events as such, but as you can tell from the graphic below, you might need to use events in other ways to describe overlapping, and so their meaning would be lost. The segmentation process relies only on space and punctuation as indications for word boundaries.

	0	1	2	3	4	5	6	7	8	9
TOM [v]	Well	that's	no t	that	easy	y	ou	know.		
TOM [mv]				((coughs))						



Furthermore, since words or other units denoted by the transcription conventions' different symbols aren't recognized before segmentation, the correctness of the transcription is not evaluated in any way. During segmentation, correct Basic transcriptions are transformed into the other format, the EXMARaLDA Segmented transcription (with the extension .exs). This segmented format expresses the information about different units in the transcription explicitly; it is partly organized according to these units, e.g. an utterance consisting of words.

Well	that's	not	that	easy	you	know
------	--------	-----	------	------	-----	------

With the information on the various segments in the transcription, it is possible to create many different visualizations from one and the same Segmented transcription. While the Partitur-Editor, which is used for input, is based on the musical score format, the output can be in a completely different format. If the segmentation of your transcription includes utterances (as e.g. HIAT does), you can create a list of those utterances, sorted by time, and even with overlapping speech marked. The utterance list corresponds to a vertical transcription format and there is little resemblance to the musical score format. Please refer to the Download section at www.exmaralda.org for various style sheets for visualization.

Other units of the segmented transcription can also be used for further visualization and presentation formats: By collecting all words, then sorting and counting them, EXAKT creates word lists with frequencies for any corpus with a word segmentation.

¹ The extensions .exb and .exs are only there for the EXMARaLDA tools to recognize, all EXMARaLDA files are XML files.

- ▶ [T38] **T:** a:aa STÖHNT
- ▶ [T39] **M:** mach net aaa un mach weiter
- ▶ [T12] **T:** ja un früher
- ▶ [T13] **M:** ja frü/ wie früher
- ▶ [T14] **T:** aj halb zwölf oder um 18[elf]19
- ▶ [T36] **M:** 18[nee]19 auch net

Word	Frequency
dann	833
ja	810
du	712
nach	641
und	598
äh	568
links	372
ich	368
bis	355

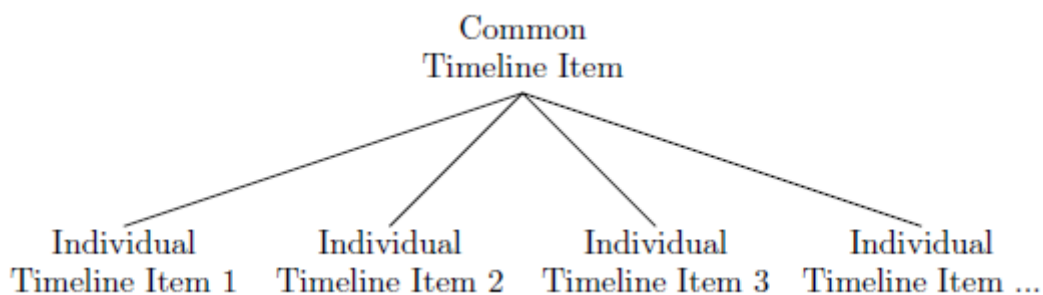
Segmented transcriptions can't be opened and inspected in the Partitur-Editor. Currently, there is no way to inspect or modify the result of the segmentation (apart from working with the plain XML file). However, in the Download section at www.exmaralda.org there is a style sheet for visualization as html ("Segmented transcription as HTML output") that you can apply to your segmented transcription if you want to see the result of the segmentation.

sc (T120/T122)

HIAT:u (T120/T122)									
HIAT:ip	HIAT:ip	HIAT:non-pho (T120/T121)	HIAT:ip	HIAT:ip	HIAT:ip	HIAT:w (T121/T121.TIE1.1)	HIAT:ip	HIAT:w (T121.TIE1.1/T121.TIE1.2)	HL
((0,5s)))	ich)	sehe	HL

One very important feature of the Segmented transcription is the so called *time forks*. Since the Partitur-Editor doesn't require each word to be manually aligned with all other parallel events, e.g. simultaneous speech, there can't be a *common* timeline with time points – timeline items – for each new segment in the Segmented transcription. The EXMARALDA data model allows the transcription tiers to have timeline items that are *not* common. During segmentation, tier specific or individual timeline items “within” or “below” the common timeline level are therefore created as time forks. These are totally independent from individual timeline items in other transcription tiers.

So, when Tom says “Hello Tim!”, and Tim says “Hi Tom, how are you?” while starting and ending at the same time, we don't need to know which words overlap and to what extent. The start and end points are part of the common timeline, and for each tier there is a time fork with individual timeline items for each created word segment. It's important to remember that this is always done by the segmentation algorithm whenever no common timeline items exist.



Segments in transcription systems

Transcription systems have different analytical units and different symbols to mark these. For example, the transcription system HIAT uses periods, question marks and exclamation marks to segment utterances and indicate utterance mode.

[1]

	0	1	2	3
Sw205 [v]	• Har ni, har ni någon som är homosexuell?			
Nw202 [v]	Ja.		• • Nei, ikke så vidt jeg vet.	

[2]

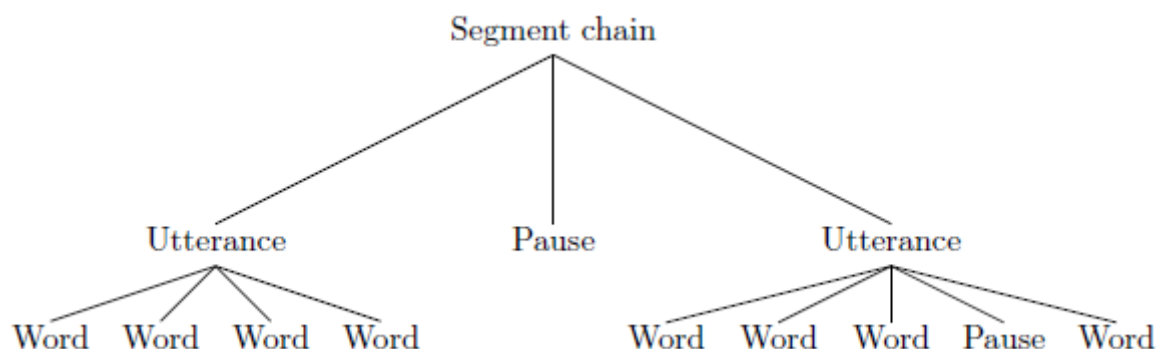
	4	5	6	7	8	9	10	11	
Sw205 [v]	Kära nån! • • Titta, redan här förstår man...						• Ja absolut!		
Nw202 [v]	Nei.			Er det et kriterium?					

Without support for HIAT in the Partitur-Editor, you would perhaps encode the information about the extent and mode of utterances somewhat like this:

7	1	[03:28.1]	2	[03:28.4]	3	[03:29]	4	[03:29.8*]	5	[03:31.3]	6	[03:32]	7	[03:32.6]	8	[03:32.8]	9	[03:33]	10	[03:33.3]
	har ni,	har ni någon som är homosexuell							kära nån	• •	titta,	redan här förstår	man							
e	question								exclamation	pause	aborted utterance									
	ja		• •	nei, ikke så vidt jeg vet							nei			er	det					
	assertion		pause	assertion							assertion				question					

With built-in support for the transcription conventions, EXMARaLDA lets you use the familiar symbols of the respective transcription. For all the functions listed below [SEGMENTATION](#) in the [Transcription](#) menu, the Partitur-Editor then uses the symbols of the transcription system to segment the transcription.

The segmentation in general can be thought of as a transformation of each transcription tier into several very simple (shallow) trees (as you know them from formal/generative grammars), whose structures depend on the structures described in the transcription system: The largest unit is *in all cases* the EXMARaLDA inherent segment chain – the uninterrupted sequence of events belonging to one speaker – and within (but never across!) segment chains there can be e.g. different kinds of utterances, in their turn containing words, pauses, non-phonological phenomena etc.



2. The segmentation algorithms

The use of finite state machines is not specific to EXMARaLDA and as long as you just want to use segmentation for supported transcription systems, you don't really need to know much about it and you can skip the first section below. Nevertheless, understanding how the segmentation works could help understanding and thus avoiding segmentation errors.

How it works

The main idea is to have this “machine” read the transcription tier; symbol for symbol, to decide if what has been input complies with the conventions. And to create a corresponding Segmented transcription from the input. For a transcription to be correct, all symbols in the transcription tier must belong to the set of symbols allowed in this sort of transcription and they must be used in the correct way (cf. lexicon and syntax). Obviously the machine must be able to recognize all imaginable transcriptions complying with the transcription conventions – a situation quite familiar from natural (human) languages.

Fortunately, transcriptions can be described by less complex grammars than natural languages and therefore recognized and produced by these “machines” called *finite state machines*. When you check for segmentation errors, the machine starts with the first segment chain in the transcription tier, reading the symbols one at a time until it reaches the end of the transcription tier or an illegal sequence of characters. Since the machine can't remember each symbol it reads, it only recognizes errors by knowing in which *state* it is, e.g. “inside a word inside an utterance inside a segment chain” or “inside some non-phonological phenomenon inside an utterance inside a segment chain”, and what characters are allowed as input in this state. Depending on the source state and the input the machine moves to another state (e.g. it reads a space and moves from the state of being in a word to the state of being directly in an utterance, outside of any words) or remains in the same state when reading each character.

Implications for transcription

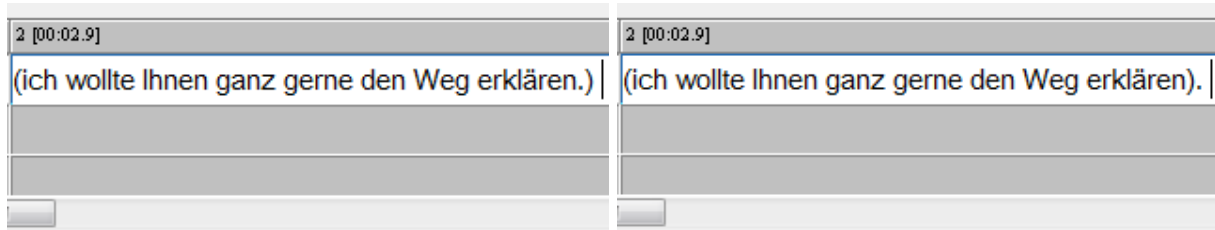
Since each symbol or symbol combination that is part of the transcription conventions is read without regarding much context, the meaning of the symbols is not as flexible as human interpretation would allow. For example, in HIAT, a full stop is used to mark the end of a declarative utterance, and it therefore can't be used in abbreviations for any HIAT-like transcription conventions. Since the Partiture-Editor's segmentation algorithms can't keep track of the textual context and therefore doesn't know if it's currently reading a (language specific) abbreviation, the German sentence “Das sind z. B. Hühner.” would be segmented into the following three utterances:

1. Das sind z.
2. B.
3. Hühner.

The difference between three single full stops in a row (...) and the ellipsis sign (...) to mark an interrupted utterance is therefore also very important to recognize. Repeating the full stop would mean trying to create several empty declarative utterances. Which meaning was intended is obvious to the human reader, but not to the segmentation algorithm.

Because of the functionality of the segmentation algorithms, some other details that might seem unimportant aren't. For example, if you're using HIAT and transcribing an utterance with two uncertain words at the end – which you bracket using single parentheses – you must put the closing bracket before the utterance end symbol, i.e. first end the uncertain part *within*

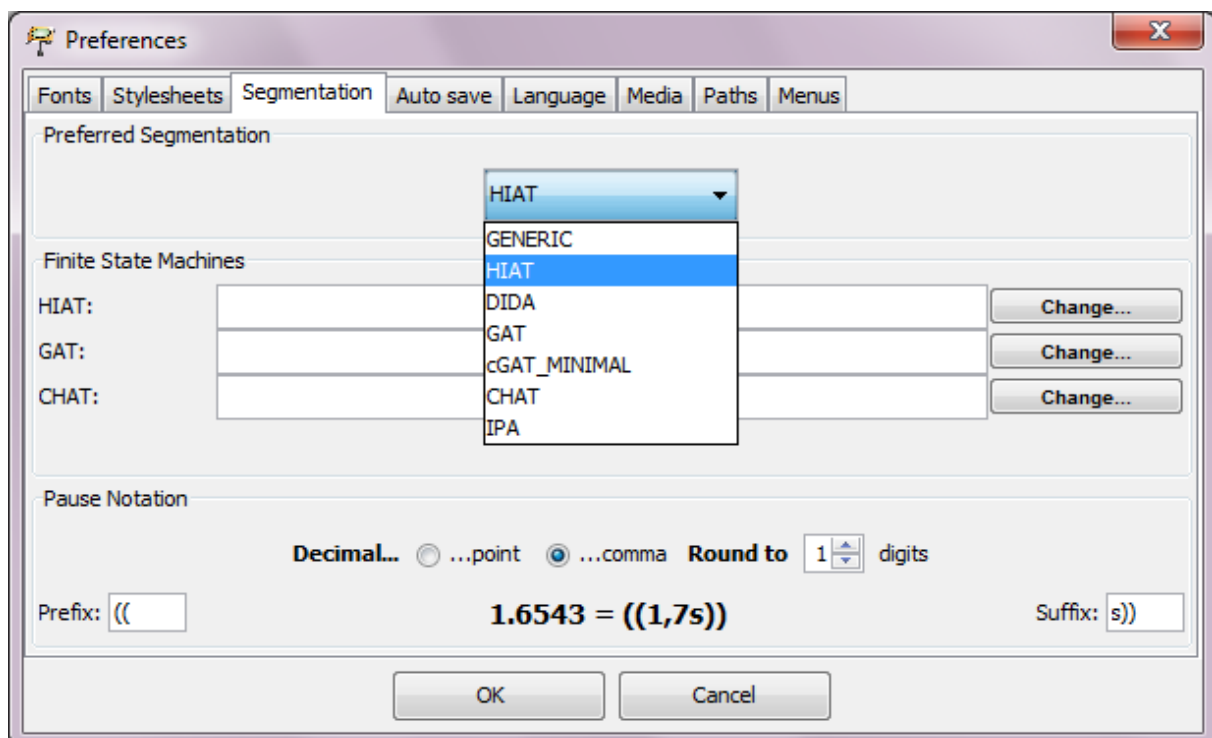
the utterance before you end the utterance itself. This is even true where the whole utterance is uncertain, since the utterance is considered to start before the opening bracket starts the uncertain part. Thus, the leftmost option will cause a segmentation error, the rightmost won't.



3. Segmentation in the Partitur-Editor

Segmentation options

The EXMARaLDA Partitur-Editor comes with built-in segmentation for the transcription systems most commonly used (HIAT, DIDA, GAT, CHAT) and IPA. Furthermore, there's a generic segmentation that can be used for transcriptions without inline symbols (perhaps using annotations instead) or for written data. You can choose your preferred segmentation using the drop-down menu at [Edit > Preferences > Segmentation](#). All segmentations use the notion of segment chains² as a top level, i.e. if a segment chain is erroneously interrupted, the segmentation will too be erroneous because the segment will be split.



Generic

You can always use the Generic Segmentation. It will recognize words – as delimited by space or other non-word characters (punctuation etc.), which means *no linguistic information*

²Segment chains are the uninterrupted series of events belonging to one speaker.

is used – and segment chains. With a transcription segmented with the generic segmentation algorithm you can create Word Lists or Segment Chain Lists.

HIAT

The HIAT Segmentation recognizes utterances, words, pause symbols and non-phonological segments according to the HIAT conventions. Other meaningful punctuation (e.g. quotes for quotations, slash for repairs, hyphens for word fragments, commas for repetition) is also recognized but only transferred to the segmented transcription as punctuation (IP), but does not get evaluated like words or utterances, i.e. there are no “quotation” or “aborted word” segments in the segmented transcription. Below is the segmentation of a segment chain containing two utterances, one with an internal pause and one with an internal non-phonological event, where the speaker coughs:

	0	1
Jim [v]	Hi • Joe! How are ((coughs)) you?	
Joe [v]		Oh hellc

Segment Chain													
Utterance						Utterance							
W ³	IP	Non-Pho	IP	W	IP	W	IP	W	IP	Non-Pho	IP	W	IP
Hi		•		Joe	!	How		are	((coughs))	you	?

DIDA

The DIDA Segmentation segments words, pauses and non-morphemic utterances (NMÄ). It also keeps track of other symbols used in the transcription. However, in DIDA there are no utterances or equivalent “chunk”. Again, the segmentation of a segment chain, this time with some additional lengthening and stress to the first word, is shown below:

	0	1
Jim [v]	hi:” * joe how are COUGHS you	
Joe [v]		oh hellc

Segment Chain												
W	IP	PAUSE	IP	W	IP	W	IP	W	IP	NMÄ	IP	W
Hi:”		*		joe		how		are		COUGHS		you

GAT and cGAT minimal

For the traditional GAT transcription conventions, there is only segmentation into Intonation Units (PhrasierungsEinheiten) according to the corresponding end signs. Jim’s greeting (roughly the same version as above) would therefore only consist of two rather long segments within the segment chain:

	0	1
Jim [v]	Hi: (.) joe? how are ((coughs)) you.	
Joe [v]		oh hellc

³ = Word

Segment Chain	
PE	PE
Hi: (.) joe?	how are ((coughs)) you.

To allow more detailed segmentation, a new set of conventions, the [cGAT minimal](#)⁴ conventions, was developed. The cGAT conventions are described in detail – in German, the English translation is to be published shortly – in the [transcription manual](#)⁵ for the tool Folker. A short “unofficial” description in English has been appended to this document. Using these, it is possible to segment our segment chain more accurately. On the other hand, the cGAT minimal conventions don’t feature end signs for Intonation Units, so currently there is no way to benefit from both segmentation options. As you can see, neither lengthening nor stress is a part of the cGAT minimal conventions:

	0	1
Jim [v]	hi (.) joe how are ((coughs)) you	
Joe [v]		oh hellc

Segment Chain													
W	S ⁶	Pause	S	W	S	W	S	W	S	Non-Pho	S	W	S
hi		(.)		joe		how		are		((coughs))		you	

CHAT

Transcriptions in the CHAT format, which is described in the [CHAT manual](#)⁷, are segmented into utterances according to the utterance terminators used. Looking at roughly the same segment chain as before, we receive a segmentation into two utterances:

	0	1
Jim [v]	hi # joe. how are &=coughs you?	
Joe [v]		oh hellc

Segment Chain	
U ⁸	U
Hi # joe.	how are &=coughs you?

IPA

The International Phonetic Alphabet can also be used for transcriptions in the Partitur-Editor. The IPA conventions allowing a segmentation into words and syllables were described in:

- Thoma, Dieter & Tracy, Rosemarie (2005): L1 and Early L2: What's the difference? Talk, DGfS-Jahrestagung in Cologne.

There are no published conventions, but the only segmentation relevant conventions are the space to mark word boundaries and the full stop to mark syllable boundaries. Since Jim only uses monosyllabic words, this segmentation algorithm is illustrated by another example:

⁴ <http://agd.ids-mannheim.de/html/FOLKER-Transkriptionshandbuch.pdf>

⁵ <http://agd.ids-mannheim.de/html/FOLKER-Transkriptionshandbuch.pdf>

⁶ = Space

⁷ <http://childes.psy.cmu.edu/manuals/chat.pdf>

⁸ = Utterance

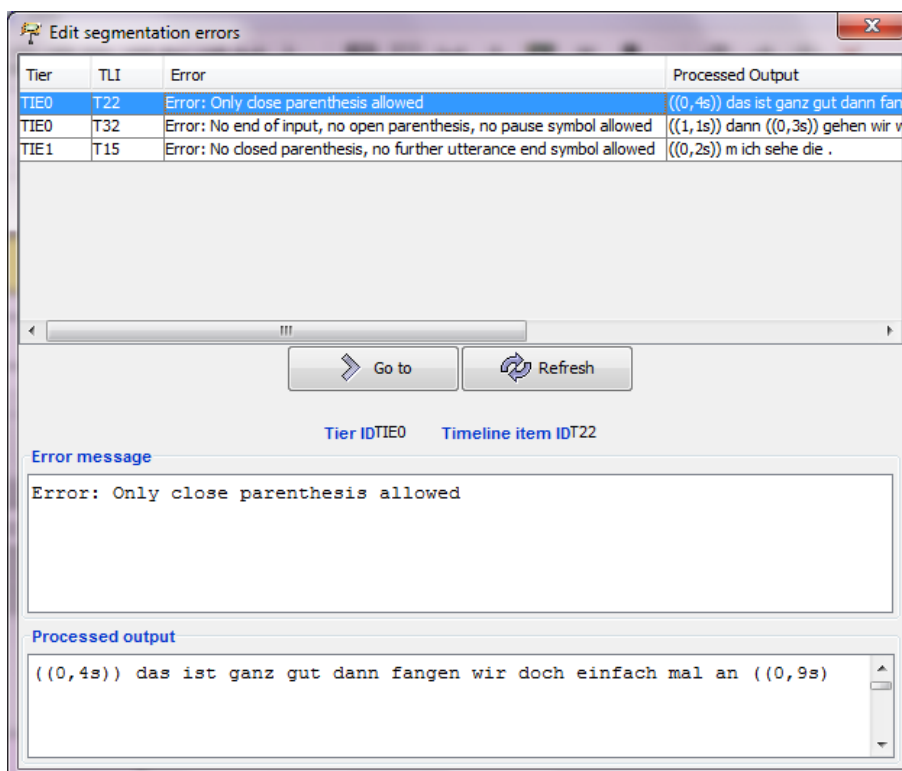
S [v-pho] næ:.ra fʏ:.tər ɪŋ:..ən ha:..rɛ

With the – optional – segmentation into syllables included, the segmentation looks as follows:

Segment Chain														
W			WB ⁹	W			WB	W			WB	W		
SL ¹⁰	SB ¹¹	SL		SL	SB	SL		SL	SB	SL		SL	SB	SL
næ:	.	ra		fʏ:	.	tər		ɪŋ:	.	ən		ha:	.	rɛ

Segmentation errors

If the basic transcription doesn't follow the conventions of the transcription system, it can't be transformed into a proper segmented transcription. Therefore all errors and mistakes must be corrected before the segmented transcription can be exported. To apply the segmentation algorithm specified at [Edit > Preferences > Segmentation](#) and list all segmentation errors, choose [Segmentation errors...](#) from the [Transcription](#) menu. Remember to make sure the preferred segmentation is active. If there are no segmentation errors, you can apply the segmentation algorithm to count different segments or export a segmented transcription.



If there are segmentation errors, you'll receive a list with the following information for each segmentation error:

- Tier: the tier in which the error was encountered
- TLI: the time-line item where the error was encountered
- Error: a description of the error, i.e. why the input couldn't be processed
- Processed output: the last part that could be processed and turned into output.

⁹ = Word boundary

¹⁰ = Syllable

¹¹ = Syllable boundary

You can view this information in detail by selecting the line. To correct an error in the transcription, simply click the [Go to](#)-button and then edit the transcription. If you need to, please refer to the transcription conventions and the information about segmentation in this document. To check whether the error was indeed corrected, click refresh. Remember you need to either press enter, “step out” of the event or save the transcription for the correction to be recognized.

Generic

The Generic Segmentation never causes segmentation errors.

HIAT

For HIAT, the most common segmentation errors seem to be caused by:

- Using three full stops instead of the ellipsis sign. With [Replace in events](#) this is easily taken care of.
- Placing the closing bracket of an uncertain part after the utterance end symbol. It has to be before the utterance end symbol.
- Forgetting one bracket in a pair of opening or closing brackets.
- Forgetting the closing quotation sign before starting a new quote or using inappropriate quotation signs.

Errors that aren't detected include:

- Utterances erroneously split into several parts since the top level segment chain was interrupted, i.e. there was an empty (“grey”) event in the middle of the utterance.
- Utterances ending (inevitably) with the end of the segment chain they're part of but lacking an utterance end sign.
- Words merged at event boundaries since the final space was left out. Remember it is possible to create event boundaries in the middle of words without splitting them!

DIDA

When working with the DIDA conventions it's important not to use capitals for other purposes than non-morphemic utterances.

GAT and cGAT minimal

For traditional *GAT*, since there is not much segmentation to the segmentation algorithm, there are really only errors due to incorrect usage of the end signs for intonation units. These should not occur as parts of words etc.

With *cGAT*, one has to be more careful and make sure all conventions have been applied correctly. In particular there are some differences between *GAT* and *cGAT*, like the use of capitalization and the notation of lengthening and stress.

CHAT

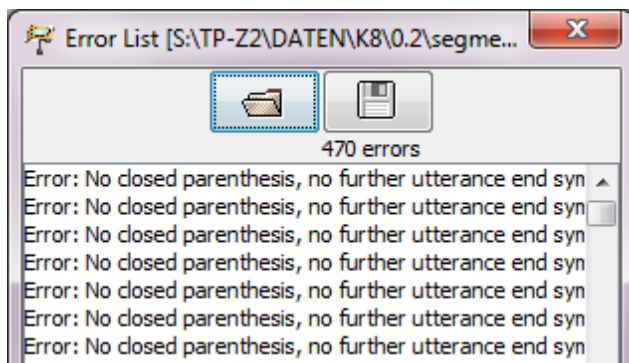
As all other *CHAT* conventions are irrelevant to the segmentation, segmentation errors are only caused by incorrect usage of the utterance end signs. All basic and special utterance terminators must be used according to the conventions.

IPA

Remember to insert spaces everywhere you want to separate words. Full stops shouldn't be used for other purposes than to mark syllable boundaries.

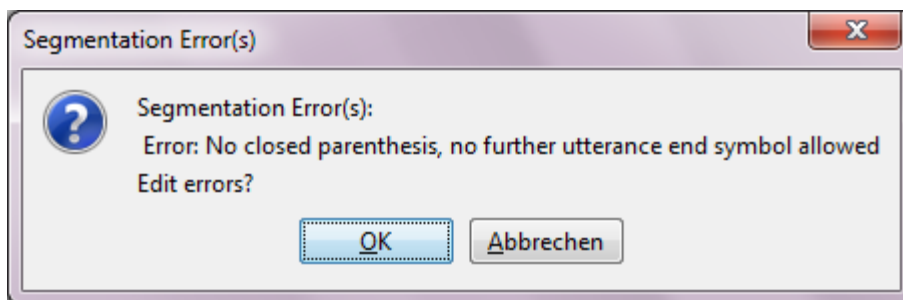
Working with error lists

The error list generated for individual transcriptions in the Partitur-Editor can also be generated for every transcription in a corpus in Coma. In fact, separate lists for segmentation and structure errors are generated. An error list is an XML file with information about each error – as shown in the above section – and in which transcription file it was encountered. The error list file is loaded into the Partitur-Editor by clicking the folder in the dialog window you receive via [Transcription > Error list...](#) After opening an error list file, you can see all errors in the current corpus. A double-click on one of the errors opens the corresponding transcription and moves to the location of the error for you to correct it, then it is removed from the list. If there are too many errors for you to correct all at once, you could save the list of remaining errors at any time by clicking the floppy disk icon.



Exporting segmented transcriptions

To create a segmented transcription with the segmentation algorithm specified at [Edit > Preferences > Segmentation](#), choose [Export Segmented Transcription...](#) from the [Transcription](#) menu. If there are no segmentation errors, you can save the segmented transcription, preferably with a name related to the original basic transcription, e.g. by using the suffix `_s`. If the transcription is not correct, you will receive an error message like this:



Click OK to arrive at the list of segmentation errors described in the section above.

Performing a Segment count

To generate a list of frequencies for the different kind of segments in the transcription with the chosen segmentation algorithm, choose [Count Segments...](#) from the [Transcription](#) menu. Should you receive a message about segmentation errors, edit these as described in the section above.

Generating a Word list

To generate a Word list, choose [Word list...](#) in the [Transcription](#) menu. The Word list, which will be displayed in a new window, contains all items segmented as words in the transcription. By clicking on the header rows [Word](#) and [Speaker](#) you can sort the list alphabetically according to these criteria. You can save the Word list as an html-file – to do so, click [Save as...](#) – with the following options: [Simple word list \(HTML\)](#) for a list with the words

sorted alphabetically or [Word list by speaker \(HTML\)](#) for a list with the words sorted by speaker and then alphabetically. Should you receive a message about segmentation errors, edit these as described in the section above.

