

How to use the Partitur-Editor with written data

This document explains how to use the EXMARaLDA transcription editor when working with written data.

These instructions apply to "ordinary" written texts, i.e. not to transcriptions of spoken language. For transcription data created with a text editor or a word processor, please refer to the information about the "simple EXMARaLDA" format.

Some parts are instructions on how to use certain import options for "customized" text import, i.e. in ways not originally intended. This information is placed in grey boxes; you can skip these parts if you don't need this extra information.

Before you start reading this document, you should read

- Understanding the basics of EXMARaLDA

Contents

A. EXMARaLDA options for text import

The EXMARaLDA Partitur-Editor features three ways of importing text: Apart from import of plain text files, you can import (text) files written by the TreeTagger program and text files in the Simple EXMARaLDA format, which is intended to be used for transcriptions created with a text editor or word processor. The following instructions focus on import of written data and therefore on the first two options.

1. Importing plain text

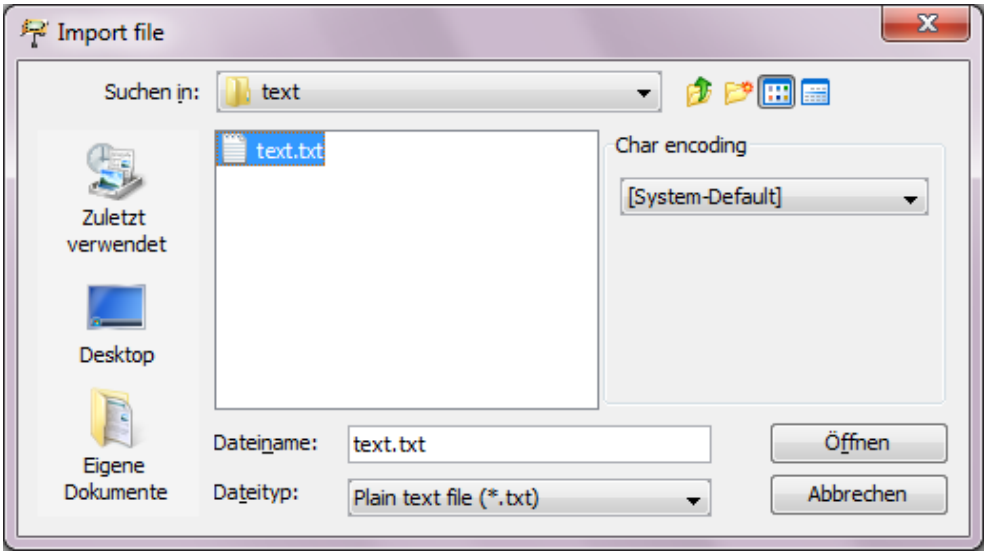
If you only intend to annotate your text manually and to use EXAKT for analysis, this should be the most straight-forward option to get you started.

a. Preparing the file for import

The Partitur-Editor needs plain text (extension `.txt`) as input format, not Word, PDF, etc. Obviously, you will lose all formatting information (e.g. instances of bold or italic text) when you save your document as plain text (`.txt`). If this is not a problem,[1] all you have to do is to save your document as plain text (e.g. in Word by choosing **Save as...** and then **Plain Text (.txt)** in the format drop-down list). You can then open and edit your text file with e.g. Notepad if you're under Windows.

b. Importing the file into the Partitur-Editor

Import the text by choosing **Import** in the **File** menu. First locate the text file you want to import. Then make sure you've chosen the right filter, i.e. for the file type **Plain text file (*.txt)** and the appropriate **Char encoding**, i.e. the same character encoding as in your text file. If you don't know the character encoding, first try the default choice **System-Default**.



If the chosen character encoding doesn't match the one of your file, special characters might not display properly after import has been completed through the next step. Should this happen, try saving your text file with another encoding, e.g. UTF-8. This is done by e.g. choosing **Save as...** in Notepad under Windows and then specifying the encoding. Then try to import

[1] If you have a document where formatting information is crucial, you need to use some work-around based on one of the more complex import options described below or (have someone) convert the document directly into the basic transcription xml-format.

the file again with the chosen character encoding. The next step in the import process is choosing how the text should be split into events during import.

It's important to remember that the events in an EXMARaLDA basic transcription, which is what you get when you import plain text, are entirely time-based, perhaps in the case of written data rather "space-based". Even though the events might have been created to split a text in e.g. words, this doesn't mean that that text has been segmented or tokenized in a way the EXMARaLDA tools can understand. The segmentation of the text into words is a different process, where start and end points for the words are automatically recognized. These might or might not correspond to the start and end points created for the "space-based" events during import. These "space-based" events and time points are better thought of as a kind of grid or guide for aligning stretches of texts, possibly words, with annotations, e.g. their respective POS-tags.

c. Choosing of the text splitter

The Partitur-Editor allows you to choose your preferred text splitting option, in more detail:

Split at paragraphs

"Paragraphs" are recognized as simple line breaks by this function. Additional blank lines between paragraphs, as used in this document, will therefore result in additional empty events. (You can find and replace these e.g. with `^13^13` using the regular expression option in MS Word's **Find and Replace** function.) Please remember to save your file after import.

Split at non-word character

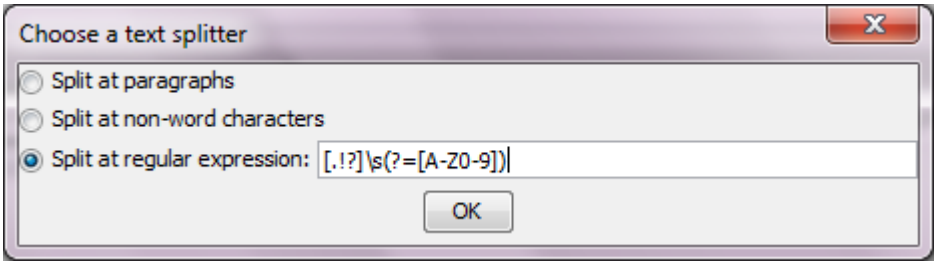
This option is not a tokenizer but a function that will create events roughly corresponding to words. In each event there is one "word" consisting only of alphabetic characters and one string of all non-word characters following the "word". These are some consequences for different non-word characters you might want to consider:

- Basic punctuation signs (. ! ? : ; ,)
  `|existerat. |`
- Opening parentheses
  `|fönsterrutor (|`
- Closing parentheses
  `|kaféerna) |`
- Opening quotes
  `|där "|`
- Closing quotes
  `|gubbarna" |`
- Hyphen
  `|på 1980-|`
- Slash
  `|Torsgatan/|`
- Numbers
  `|pilsner 2,8 |`

Please remember to save your file after import.

Split at regular expression

With this option you can define event boundaries as a regular expression. Obviously, splitting the text with some regular expression won't be as good as a language specific tokenizer when it comes to detecting words. If this is relevant to your work, you should perhaps consider working with a tokenizer. Together with the Partitur-Editor, you could either use the EXMARaLDA importer developed by SFB 632 in Potsdam or the tokenizer that comes with the TreeTagger to tokenize your text. Both options are described below. The regular expression option might however be the better option if you want to annotate some rather uncommon "chunks" that could be described this way, or for languages for which you haven't got a proper tokenizer.



The expression in this example would split the text after each blank space following a period, an exclamation mark or a question mark, as long as the character following the space either one of the letters a-z in upper case or numerical. The syntax for regular expressions is described on this web page.[2] Regardless of the expression used, the text is still split at paragraph boundaries, i.e. line breaks. Please remember to save your file after import.

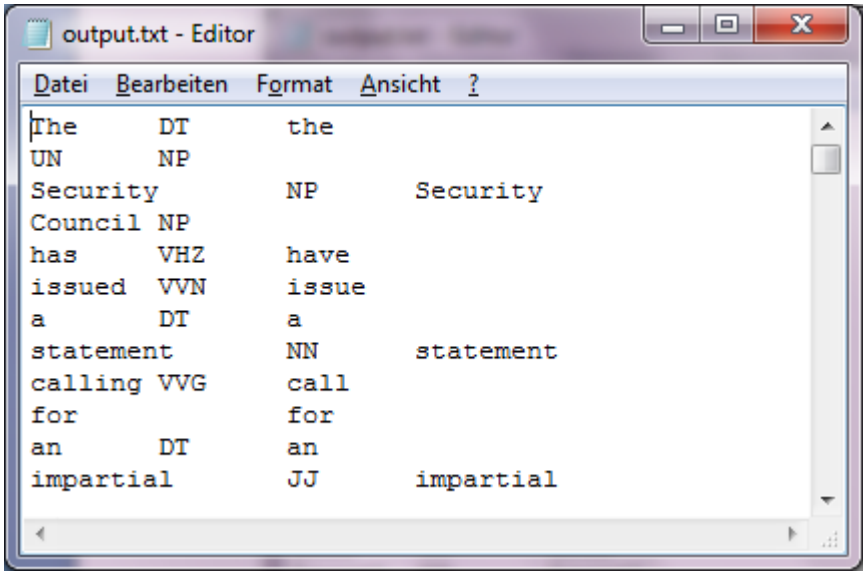
2. Importing TreeTagger Output

The output of the widely used TreeTagger (Schmidt 1994, web page)[3] can be imported into the Partitur-Editor. In the EXMARaLDA file, a transcription tier with one token per event is created for the text itself, with one or two annotation tiers holding the respective POS-tag and, if this TreeTagger option was used, lemma information for each token. This option might be convenient if you want to POS-tag and/or lemmatize your text automatically before proceeding with manual annotation in EXMARaLDA. TreeTagger parameter files exist for several languages and tag sets, but the tagger can also be trained. Thus the TreeTagger can learn to tag data using any tag set for which there is manually annotated data for the training.

a. Preparing the file for import

Instructions for using the TreeTagger can be found on the web page.[4] For those who prefer graphical user interfaces to working with the command prompt, there's also a separate Windows interface[5] for the TreeTagger. The installation of the interface and the TreeTagger is described in detail on the respective web pages. Depending on the language and tag set, your output file should look something like this:

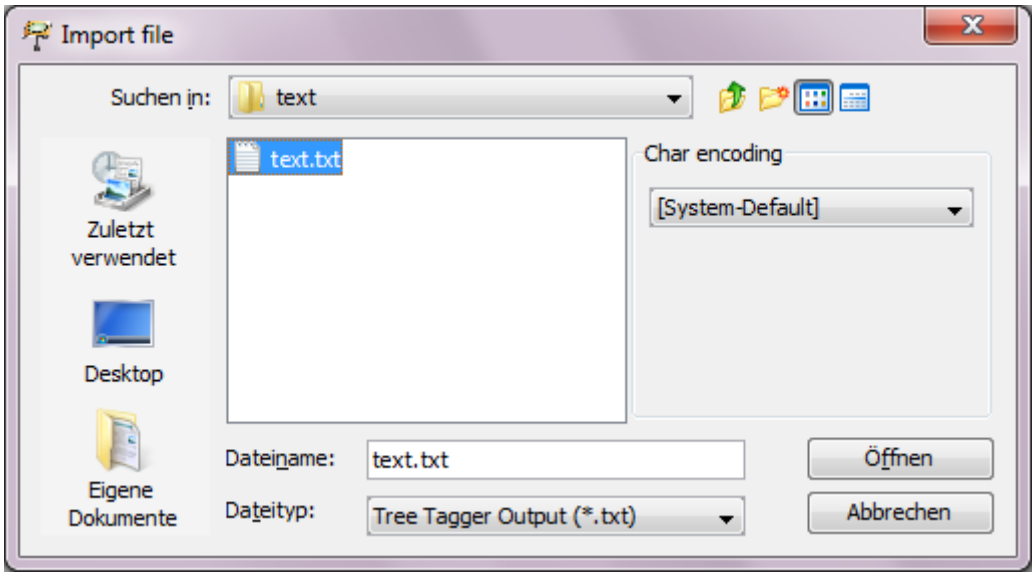
[2] http://java.sun.com/javase/6/docs/api/java/util/regex/Pattern.html#sum
[3] http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/
[4] http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/
[5] http://www.smo.uhi.ac.uk/~oduibhin/oideasra/interfaces/winttinterface.htm



The TreeTagger (and Simple EXMARaLDA) import options aren't merely interesting if you're using the TreeTagger (or have created a transcription in Word). You could also use these to "customize" the text import options. Basically, the TreeTagger import creates a transcription from a tab separated text file. An event is created for each new line and annotation tiers (which properties you can easily edit) are created for the second and third columns. This means you could use this import option for any text file in TreeTagger input (one token per line) or output format (one column for text, two or three columns for annotations, one event per line). Since the TreeTagger tokenizer strips blank space from the original text file, during import an additional blank space is reinserted at the end of each token/event in the text/transcription tier.

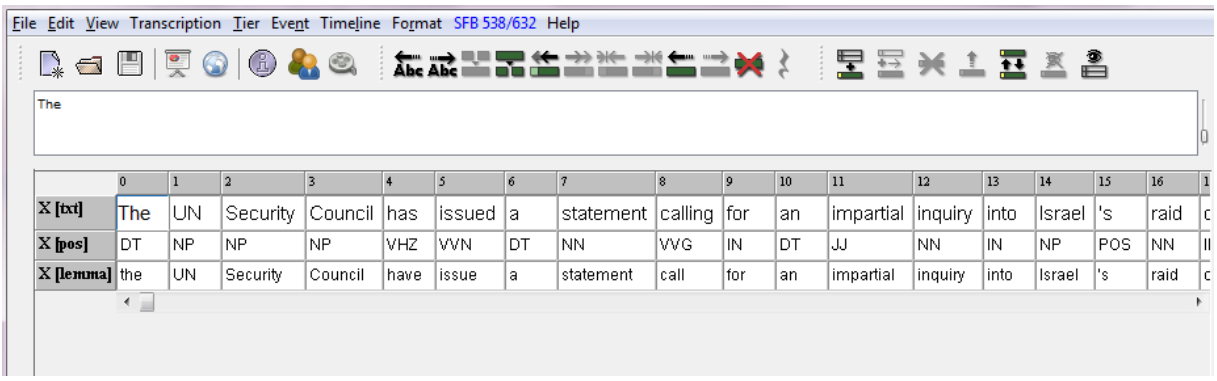
b. Importing the file into the Partitur-Editor

Import the TreeTagger file by choosing **File > Import** and selecting the file type **Tree Tagger Output (*.txt)** to receive a transcription file. Then save your file.



If you want to use existing EXMARaLDA segmentation and visualizations for the text, you have to remove the inserted blank space after words followed by punctuation marks during the manual control of the tagger result. If you would want to e.g. have compound nouns not

recognized as such tagged with one tag – in the text below “UN Security Council” – use the merge function (**Merge** in the **Event** menu).



3. Importing the Simple EXMARaLDA format

This option is intended to be used for transcriptions in .txt format created in a regular text editor or word processor. The Simple EXMARaLDA format allows you to create additional annotation tiers for formatting, mark-up and/or annotations during import, and to define events boundaries according to information in the original file. The format and the intended use are described in the document “How to import text transcriptions”. If you want to use this format to “customize” text import, you should start by reading that document.

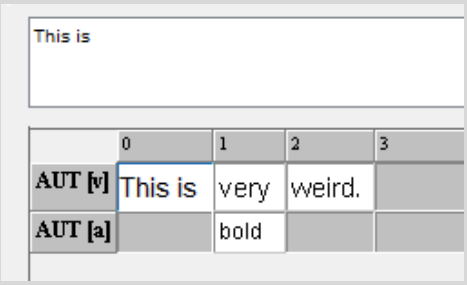
a. Preparing the file for import

You can use line breaks and curly (and square) brackets to create annotations (description events). The annotated text should be in a separate line, starting with the speaker abbreviation, and the annotation text should be in curly brackets at the end of the line. Microsoft Word and OpenOffice Writer both have a regular expression option in the Find and Replace function that will let you search for and replace formatting, and use the found expression as part of the replacing expression. To keep formatting information, the sentence “This is **very** weird.” could then be transformed into:

```
AUT: This is
AUT: very {bold}
AUT: weird.
```

b. Importing the file into the Partitur-Editor

Import is done via **File > Import**, choose **Simple EXMARaLDA text file (*.txt)** as file type. You might want to change the tier categories and display names. (Click on the speaker abbreviation of the tier to highlight the tier and then from the **Tier** menu choose **Tier Properties**.)



B. The SFB 632 EXMARaLDA importer

The SFB 632[6] has created importers[7] for the automatic generation of EXMARaLDA files (and files in other formats used by other annotation tools) from plain text files. The importer creates an EXMARaLDA file with the text from the plain text file in a transcription tier with the category "word" and event boundaries at each of these words. In two annotation tiers there are annotations corresponding to sentence (category "sent", annotation "S") and paragraph boundaries (category "para", annotation "P"). You upload a text file and receive a link to the corresponding file in the EXMARaLDA basic transcription format, which you then download. By default, the process includes tokenizing to recognize word, sentence and paragraph boundaries. If the result of the automatic tokenization is unsatisfactory, the text can be manually corrected (tokens separated by space, one sentence per line, paragraphs separated by an empty line) and imported with the option for importing tokenized text.

Since the Potsdam "dialect" doesn't share the interpretation of basic EXMARaLDA concepts like events and segmentation with the rest of the EXMARaLDA tools, you won't be able to use EXAKT for concordances and analysis unless you convert this kind of basic transcription into an EXMARaLDA segmented transcription. The "native" text import options preserve blank spaces used as word separators in the original text (the TreeTagger import even adds some), whereas in the Potsdam dialect, spaces are stripped during the conversion. The Partitur-Editor needs the spaces for the automatic segmentation, i.e. to create a segmented transcription, and EXAKT only works with segmented transcriptions.

C. Annotating text

1. The AUT Speaker

Since EXMARaLDA was originally developed to create transcriptions of speech featuring several speakers, it is based on the assumption that there is one speaker for each transcription tier. Even in this case, where the "speaker" of your written data might be unknown or simply irrelevant, there must be one. Annotation tiers, and thus annotations, are linked to the transcription tier, and thus to the annotated text, via the speaker abbreviation, therefore all added tiers must be assigned to this one speaker.

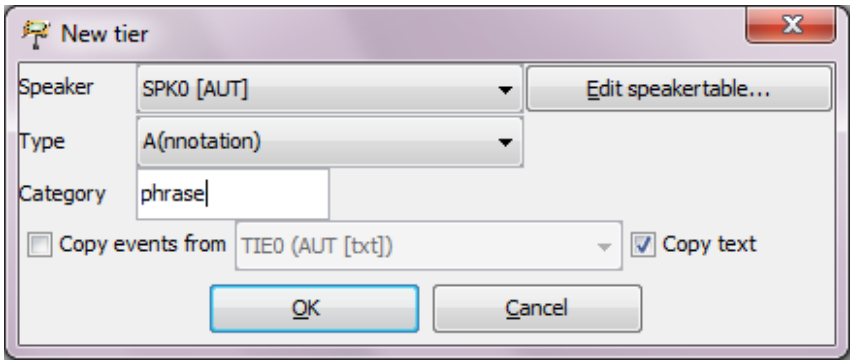
As of version 1.4.5., the Partitur-Editor automatically creates a dummy speaker called AUT along with the tier for the imported text. If your version is an earlier version of EXMARaLDA, either update to the latest version (preferable) or create a speaker manually for your imported text (**Transcription > Speakertable… > Add speaker**) and assign the speaker to the tier (click on the speaker abbreviation of the tier to highlight the tier and then from the **Tier** menu choose **Tier Properties** and select the speaker).

2. Annotation tiers

In order to annotate your text, you need to create additional tiers, of type "A", for "Annotation". The number of annotation tiers and their categories depend on your annotation scheme. Since the Partitur-Editor was developed for transcription of spoken language, it handles parallel, independent events or annotations very well. However, the EXMARaLDA data model has no built in way to express the dependence needed to define hierarchies or other relations between annotations or annotation tiers.

[6] http://www.sfb632.uni-potsdam.de/
[7] https://141.89.100.100/homes/d1/services/paula_webservice/for_KorpTA/index_en.php

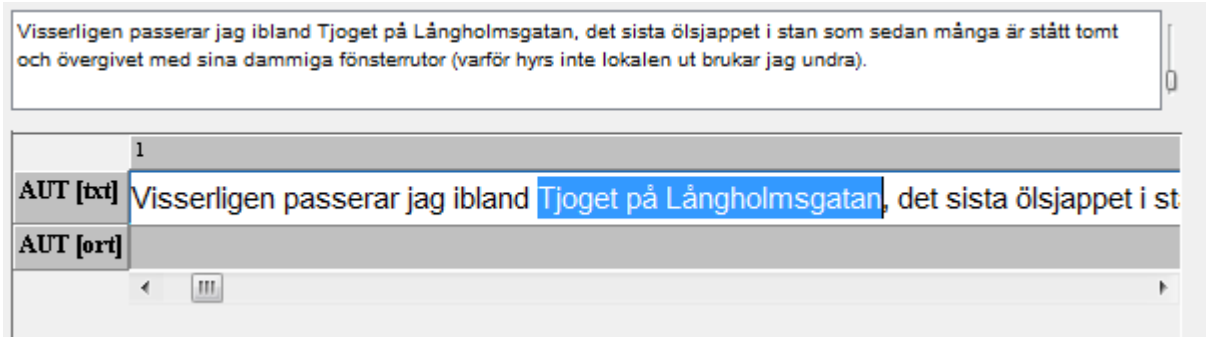
To add an annotation tier, click on the icon **Add tier...** (left) or **Insert tier...** (right). Or choose the corresponding items in the **Tier** menu. With **Insert tier...** you can decide where the new tier is inserted, **Add tier...** will simply add it as the last tier. Select the type "A", for "Annotation", then select the speaker and define the category for the new tier.



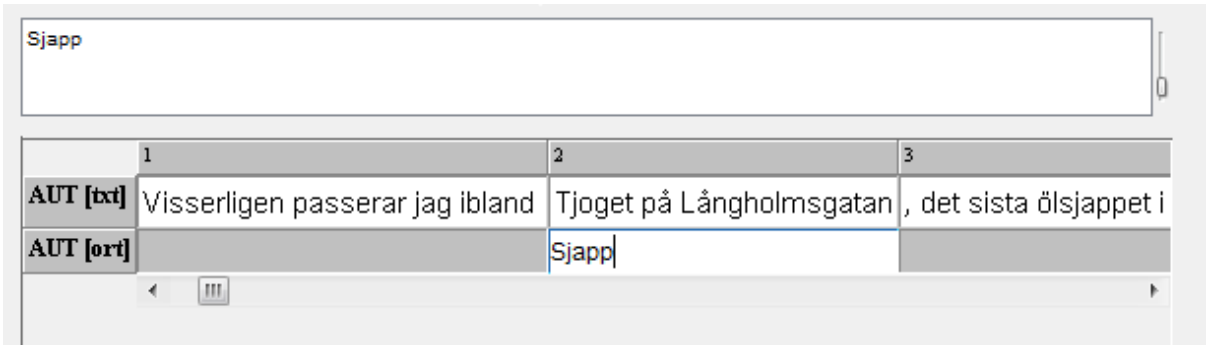
Add / Insert

3. Annotating in the Partitur-Editor

If your annotations are simple, e.g. some kind of commenting, you can simply go ahead annotating. To create events for the annotations, define additional event boundaries: For a single boundary, place the cursor in the text at the intended boundary and then either press **Ctrl + 2**, click on the **Split** icon or choose **Split** from the **Event** menu. To create an event for a stretch of text, select the text you want to annotate and press **Ctrl + 3** or choose **Double split** from the **Event** menu.



The above picture shows the highlighted text, below is the result of the **Double split**.



The annotation panel (**View > Annotation Panel**) will help you annotate consistently and facilitate the annotation process, e.g. by letting you add annotations to more than one event that will then be merged automatically. Your annotation scheme with annotation guide lines will be visible to the person annotating. The panel can suggest annotations, but doesn't im-

pose any restrictions. The use of the panel is described in detail in the document "How to use the annotation panel".

D. Segmentation

The other EXMARaLDA tools – CoMa and EXAKT – require segmented transcriptions. The segmentation in EXMARaLDA is usually based on some transcription conventions where each symbol or pair of symbols carries a unique meaning.

Since this is not the case with standard written language[8], the text can only be segmented into segment chains – not very interesting, since most texts consist of only one segment chain[9] – and words, which is again not the result of tokenization but only depends on the use of blank space as delimiter. For more information on segmentation, and how to use other segmentation algorithms or customize them, please refer to the document "How to use segmentation".

[8] E.g. a period is used both to mark the end of a sentence and as a part of an abbreviation

[9] Though not intended this way, you could theoretically use the segment chain with a linguistic meaning by manually creating empty events (the events with grey background, unless you change the settings) between your linguistically relevant segments encoded as segment chains, i.e. continuous "white" events.