

Sequences 101 Guide for Carabao Language Kit



By Vadim Berman

February 19, 2009

How the System Works

The sequences in Carabao have two main objectives:

1. Find combinations of words.
2. Transform these combinations of words. This only works in translation or transformation mode.

Not all positively validated sequences proceed to the transformation stage – in fact, usually only a small fraction do. Depending on considerations such as consistency with other sequences, how comprehensive are the criteria of a sequence (more specific sequences overwrite less specific ones), and others, sequences may be removed prior to the transformation stage.

Once the sequences are validated and are through the refining process, they move on to the transformation stage. On this stage, the words are already reduced to their language-neutral “essence”, namely, family ID, rule units, and style units. The sequences use the same attributes which are used for validation, to modify this “essence”. The members are removed, added, and modified according to the differences between the source and the target sequences. Member identity numbers are used to find out who is linked to who between the source and the target sequences.

Important: Optional and “not” members, as well as “unmapped” members are not mapped as part of the sequence. They are only validated. So if you are building a new sequence based on an existing sequence, and set a member as optional, because in some cases it's not there – it will be **removed**, which is more or less OK, but in the opposite direction, it will be inserted, which is more dangerous: you can get a duplicated word.

The modification of the existing members is simply an assignment of the attributes. For example, if the member is a **verb** whose family ID is **123456**, and the target sequence member is **T=12345\$R1=NOUN**, then it will become a **noun** whose family ID is **12345**.

Matching Source and Target Sequences

The sequences, just like words in Carabao database, are organized into families. Therefore, the relation is also not one to one, but many to many (or “bag to bag”). The matching of the source to the target sequence has a purpose of finding the closest match – by rule units, and style units.

While the procedures matching the sequences are different of those matching words, the logic is largely the same: the winner is the record with maximum matching rule units, and, if there is a “tie” in the number of rule units - style units.

Just like with words, usually **there is no need to provide an exact translation for every record in the family**. In fact, artificial duplication not only results in a performance hit, but also may cause incorrect translations.

What is the purpose of language independent rule units in sequences? More or less the same as with words, simply to coordinate between different sequences under the same family.

In some cases, however, it may be required to have correlation between multiple sequences. In application such as MiaMia, when Carabao is used for text analysis, the application's rules may be linked to the family IDs. In order to capture as many patterns as possible, it is a good idea to group many different sequences under one family ID. For example, a sequence like 905 in the default database (“translate to <language>”) may have different way of expressing the same idea: “translate this to <language>”, “say in <language>”, etc. While these sequences are not conveying the same idea, for the needs of this application they are the same. Still, however, there does not have to be strict correlation between the languages – but all the variants must be covered.

Late Sequences

In some cases, in order to decide how to transform the text, the system needs grammatical information from the target language. For example, if we are translating a superlative adjective from French to English, we need to know, how English marks the superlative grade. Is it “*most complex*” or “*slightest*”? We cannot deduce it from the source language, and we don't know it on the stage when most sequences are transforming the “language-neutral essence”.

So-called *late sequences* solve this problem. One sequence, which is called a *gateway sequence*, is doing the initial transformation. A specially tagged member of the sequence (the checkbox *Use in late sequences* in Sequence Member Sheet) is then examined and compared with the other sequences to choose from. This is how the member is tagged:

Edit Sequence Member

Anything (marks possible gap between neighbors)

Definitions

Not Attached Use Rule Units for Late Sequences Except

-> Family ID:

-> Category ID:

-> Map ID:

-> Syntax Delimiter:

-> Numeric Value:

-> Anywhere First In the middle Last

Case Default Upper Lower Capitalized

Next Element Case Default Upper Lower Capitalized

Regular Expression:

Control Priority:

Optional Identity:

Rule Units

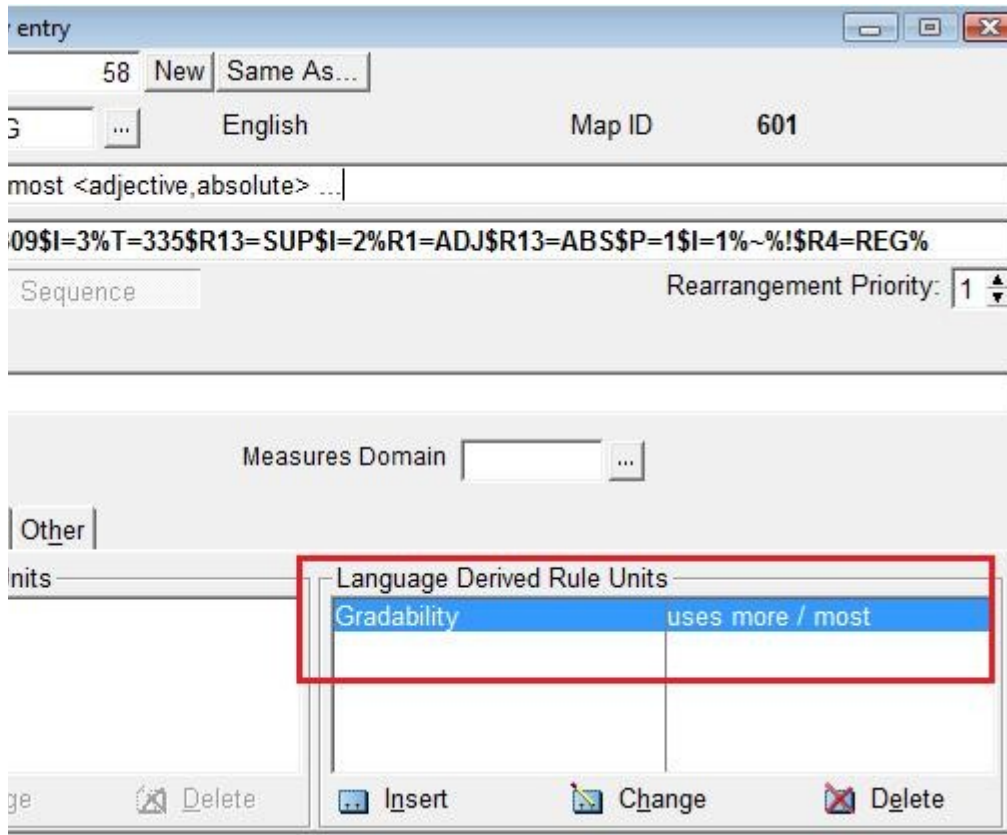
Word class	adjective
Grade	superlative

Style Units

--	--

Insert Change Delete

In the example with the English adjectives, we are looking at the rule unit “Comparability”, which holds the way this adjective inflects. If the value is “regular”, it inflects like “slightest” or “slowest”. We need to pick the sequence - which is also tagged with the “regular comparability” rule unit and only sets the gradability of the adjective to superlative. If the adjective inflects like “most complex” (by using the word *most*), we will pick a sequence which adds (if required) the word *most*. The rule units to compare are assigned to the language **derived** rule units in the sequence, as shown on the figure below:



Debugging the Sequences

Carabao Translation Console has the option to trace all the major calls that the system does when processing. It can be switched on from the *Debug* tab, by checking either of the following options:

- *Morphological analysis* – on this stage, the system tries to build all the possible guesses based on its knowledge of the morphology of the source language.
- *Disambiguation* – on this stage the sequences are validated based on the guesses, then refined along with the guesses. Later, the sequences are applied to the guesses.
- *Synthesis* – on this stage, the late sequences are called, and the words are searched in the target language for the transformed elements.

If you are debugging sequences, in 99% of cases you only need to trace the disambiguation stage. Normally you should be using the tracer window. If, on a very rare occurrence, something is so wrong that it causes the system to GPF, select *Black box mode* – it will write everything to a log file under the working directory.

What you should be looking for if you are not sure whether your sequence is working?

1. Look for the translation stage. Try to find the string “**translat**” in the Tracer. After the “Translating sequences” header, scroll down until you see your sequence. All sequences are starting with the header, saying, *Translating (<number of the sequence as detected> / <number of the sequence in the dictionary>) <sequence code>*. Under the header you can see what the system is doing: inserting, deleting members, or moving and updating. Eventually, it sets up the agreement lists, if applicable. If you don't see your sequence there, or it says, “No sequences to translate”, go to the next stage.
2. Let's see if the sequence made it to the “possible sequences”. Look up for the line: *Adding possible sequence [<the sequence family ID>]*. If you can find your sequence, it means that it passed the initial check (only one element). So now we can look at the way it passed the validation.
3. Look up for the line: *Validating [<the sequence family ID>]*. It has to be there, if the sequence was added to the list of possible sequences. Here, element by element, the sequence is checked, and mapped. Every time a successive member is found, there is a message, *[#<order No. in the sentence>, txtCfgNo#<guess configuration number>]... fits!* But the outcome is reported later. If and only if you have the line, *Adding sequence [<the sequence family ID>]...* - it means that the sequence was validated positively. It is still “on probation” though. Note that the system does not discard the sequence unless it is 100% certain either the conditions are not

satisfied, or the negative members are too obvious (for example, “the” before what is supposedly a verb).

4. If you see that the sequence was positively validated, but did not make it to the translation stage, it means that it was “killed”. It happened either in conflict with another, more powerful sequence, or because the guess that this sequence bets on, was deemed incorrect. When it was the sequence conflict, the reason is either that the winning sequence had more members or more conditions, and therefore assessed as more specific. A word can be a secondary member in only one sequence, and the refining process is inspecting guesses to find the conflicting sequences. Look up for *Delete seq#*<your sequence order (not family ID)>. When you found it (it has to be there) – a few lines above you see how the system compares the sequences where that word is a secondary member. Among the sequences you can find the sequence that won. Pay attention to *srcMembsNo* and *specScore*: these are the values that decide who is the winner. The specification score can be increased by additional criteria. Do not rush to change it though; maybe you need to modify the other sequence.

Troubleshooting

So you designed a sequence, but it doesn't do what it is supposed to do. The table below describes a few common problems, which can be resolved fairly quickly.

Symptom	Usual cause	Solution
1. Once added, all the translations become messier, and new words appear.	The sequence is validated positively in situations when it's not supposed to. It usually happens when translating from more analytical language (that is, with less morphology) to more morphologically rich.	Set up “not” and “optional” members to refine the situation. Make sure you did not remove a member accidentally.
2. The conditions seem correct, but the tracer shows that sequence is not validated positively. It does not even appear as the possible sequence.	The conditions are too strict. It is possible that the sequence was copied from another language without adjusting the conditions. For example, verb tenses were left in Chinese.	Review all conditions, try to disable some and see whether it helps. Make sure not to remove the conditions you really need.
3. The sequence is validated in a situation when it's not supposed to.	Same as 1.	
4. The sequence is validated positively, but the guess is deemed incorrect.	Not enough sequences are validated correctly, or the sequence has negative consistency points.	To raise the sequence's “voting weight”, increase its frequency value. Don't be too liberal though. The idea is that when it's too ambiguous, more sequences weigh in. If the consistency points are negative, one of the “not” members was deemed highly probable. Review the condition and see how it can be refined.

5. The system incorrectly guesses a part of speech of a word.	Same as 4, or a correct sequence does not exist.	Same as 4.