

Inside Carabao: Natural Language Processing for XXI Century

By Vadim Berman



Business data does not always come prepackaged in spreadsheets, relational databases or XML. Developers in different industries frequently face challenging tasks involving processing unstructured text. As the expectations on computers being 'smarter' persist, text mining and text analytics have become increasingly important parts of the Business Intelligence.

Overcoming linguistic barriers and structuring unstructured knowledge are the next big technological challenges next to setting up global communication network. As human translation or analysis is not always available or economically viable, advanced natural language processing technologies such as text analytics and machine translation (MT) are the crux of all efforts in this direction.

Multilingual Natural Language Processing: Unsatisfied Demands

Modern MT systems have reached a level where they can produce a fairly readable gist of the source text; many of them are already used in production.

In most cases, however, end users expect human translation quality in all the results. Consequently, they frequently assume that the application is "useless" when in fact, it is quite possible to understand the result. It is interesting that text analytics, on the other hand, are perceived as a technology that has been proved to be reliable. This is easily explained by the fact that the errors are less obvious; a malformed sentence generated by an MT software can be spotted by everyone, while a city which wasn't extracted from a huge text might not be ever discovered.

This pushes MT developers, in particular, to go on a wild goose chase for miraculous breakthroughs in order to achieve human translation quality. While massive attention is focused on this particular cause, little attention is paid to other important aspects:

- Difficult and slow development of new languages
- Often there is no clear distinction between the tasks assigned to the programmers and the linguists in a development team
- As a result, the available translation pairs are limited to a number of languages and the quality of dictionaries is very poor
- Once developed, the end user has little or no control over the translation. While recently more and more off-the-shelf systems offer customization, in many cases what they offer is not enough.
- Typically enormously large footprint, high CPU and memory requirements

All these are also relevant to the text analytics software.

Carabao Language Kit, the system we are presenting here, addresses these issues without conflicting with existing natural language processing methodologies. It is a complete solution for MT, text analytics and cross-lingual information retrieval.

Architecture of Carabao

N-gram Design

Instead of translation pairs in the traditional sense, in **Carabao** bits of lexical data are assigned a universal ID number. A translation pair is created on the fly by going from source lexical data to target lexical data. This is a way not only to save on development on translation pairs, but also to obtain minority translation pairs which are not likely to be created at all. From the text analytics point of view, this approach enables cross-lingual information retrieval.

One of the problems in building a multilingual dictionary is aligning the data; it is not aligned the same way it is in the "paper dictionaries". Building a dictionary from scratch though can be trickier than assembling a paper dictionary. Is "king" the same as "monarch"? Is "hello" the same as "howdy" or "g'day"? How does one translate "howdy" into French? Traditional MT systems don't support fuzzy logic; it's either the same sense or not. With classic N-gram design, it gets even worse. One has to align every entry with all the others and "multiply out" if there is a different sense in any of the supported languages.

However, **Carabao** introduces a slightly different approach which makes dictionary management much easier. The most important difference is that the words are grouped in what we call *families*. *Family* is a group of words or phrases with the same or similar meaning. For now, let's use the concept of "synonyms" (later we'll expand the definition). Words of the same *family* share the same *family ID* code across all the languages in a database. Cases when two or more words in a language bear exactly the same meaning, without differences in nuances, are very rare. Usually the subtle differences are in style, or professional domain, or geography of usage.

A field holds an array of *style tags* whilst a *family ID* + an array of *style tags* provide a unique key for a record. For example, consider the family of the word *hello*. The same family will contain *g'day* tagged with a code *Australian* and *howdy* having *South US* stored somewhere in the *style tags* array. We can also add the word *greetings* to the same family assigning the value *formal* to another member of that array.

What happens if another particularly uninventive language will have only one entry in the family of *hello*? When this language is the target, the translation is trivial - we don't really have a choice but to select this one entry. However, in the reverse situation - we can pick an entry with the best matching *style tags*. If there are no *style tags* in the original entry, then the entry with the least number of filled tag slots will be selected (the most "neutral" one). On a more advanced stage, it is possible to actually force and avoid certain styles by paraphrasing sentences. For example, select formal entries when possible or avoid vulgar language even if the original contains some. This can be done by **Carabao** as well.

This way, creating the database becomes much easier while all the benefits of N-gram design are preserved. When in doubt, simply add the word to a closely related family and mark with a tag!

In traditional lexical databases, both pair based and N-gram based, a noun is coordinated with a noun, an adjective with an adjective, and so on. This makes sense in most cases. But what if one or all of the parts of the speech are not preserved? Language Riau, spoken in some parts of Malaysia, does not have parts of speech at all. So do some artificial languages, like Lojban. In ancient Chinese parts of speech are assigned according to the position of words in a sentence.

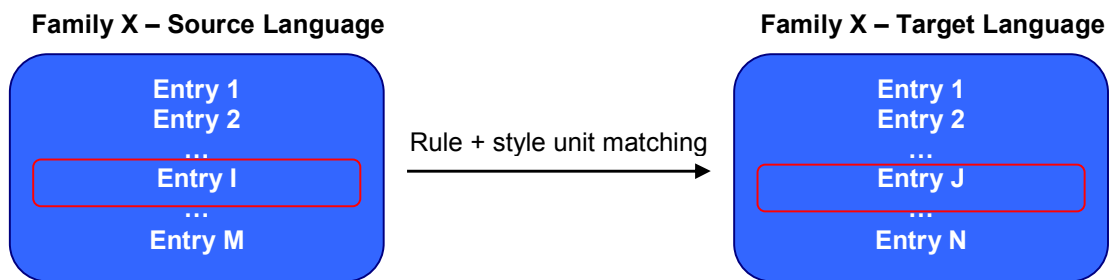
Carabao data model offers a simple and effective solution to overcome the structural differences between the languages. Let's define a *rule unit* - a tag bearing any piece of grammatical data: part of speech, number contrast, gender, conjugation pattern or anything else. Every index in the array, just like with the *style tags*, stands for a type of a *rule unit*. For example, a member number 1 will contain parts of speech, number 2 - number contrast values, etc. Like with styles, fuzzy matching is used to coordinate a source and a target. Now we can group different parts of speech or inflected versions of the same word in the same *family*. Matching any source with any target is a breeze.

However, there are more advantages to this structure. *Rule units* allow effective and easy definition of irregular forms, if the system on top of the database is set to pick the most specific entry from a family. Let us consider the verb *buy*. When generating an entry from grammatical definitions, a system that is aware of the

common conjugation pattern, might erroneously create *buyed*. To avoid this kind of error, a lookup table of exceptions must be implemented. With **Carabao** data model, it is possible to store *bought* in the same family and specified as a past tense form. When a verb in past tense form is required, the system will look for the best matching entry in the family and, in our case, fetch *bought*. Otherwise, when a past tense form is not found, **Carabao** will fetch the next closely matching one (for example, base form of a verb) and will try completing it to the required form by using internally specified rules.

Note that unlike in traditional pair-based design, the dictionary designer does not have to be fluent in **all** of the languages stored in the database. It is enough to know the language being edited and **any** other language which currently has this entry.

The diagram below summarizes the process of translation in **Carabao**:



Implementation of Transfer & Validation Rules

Can translation transfer rules also be arranged in N-gram fashion? Yes, they can.

Carabao uses a special mini-language to specify any kind of sequence of words, syntax delimiters, etc. It can be an idiom, a collocation, a grammatical rule, or mix of any of the above. This entity is called *sequence*. *Sequences* are also stored as N-gram families.

A regular pattern of *sequence* usage is the same as with words / lexical units. *Sequences* are matched by *family ID* and *style tags*. There is no *rule unit* matching, as a *sequence* is a group of words which will on a later stage undergo this rule unit matching process anyway, on a one-on-one basis.

To handle situations when a rearrangement or deletion / insertion of sequence members is required, each lexical unit (member) of a sequence is given a locally unique *identity number*. These numbers will be preserved across languages. When translating, **Carabao** implicitly generates transfer rules like in Q-systems.

Let us consider a translation between English and French using the sequence “there” + “is, are, was, were” + (any element(s)) + noun. Its French equivalent is “il” + “y” + “a, await” + (any element(s)) + noun. As we can see, while the noun and the verb “to be” are preserved, “there” replaces “il” (or vice versa, depending on the direction), and “y” is inserted or deleted, depending on the direction.

The identity numbers will hint **Carabao** Sequence Manager on how to handle lexical holes and what are the sources and the targets of each transformation. In our example, the identity numbers will be:

English: “there”_{identity=1} + “is, are, was, were”_{identity=2} + (any element(s)) + noun_{identity=3}
 French: “il”_{identity=1} + “y”_{identity=4} + “a, await”_{identity=2} + (any element(s)) + noun_{identity=3}

As “y” has *identity number* 4 which wasn’t found in the source sequence, it will be created when translating from English to French. When going in the opposite direction, it will be deleted, because it is not found in the target sequence. Note that the identity numbers do not have to be sequential or ordered in a particular

fashion. Therefore, as with the lexical units, a linguist fluent in the language being edited and in **any** other language in the database, is able to edit the sequence for **all** possible translation directions.

The sequence mechanism is also responsible for enforcing grammatical agreement between the relevant members of the sequence. Unlike in most systems which only allow for one head word, **Carabao** is capable of forcing (or validating) multilevel agreement. That is, a head word governs another sequence member, and this other member governs yet another sequence member.

Even though sequences appear “flat” on the surface, Sequence Manager concatenates them into a lexical tree when parsing a sentence. A sequence does not have to be continuous. This feature allows for greater flexibility and reusable rule design. For example, if we add another sequence:

English: **adjective**_{identity=2} + **noun**_{identity=1}
 French: **noun**_{identity=1} + **adjective**_{identity=2}

...it will be detected along with the “there...” structure. Sequence Manager will concatenate it to the noun and determine the order of performing the transformations such as the rearrangement of the adjective, or adding “y”.

Sequences can express virtually every aspect – grammatical data, style, family, specific word, category of words, order in a sentence, etc.

However, sometimes selection of the target sequence depends on grammatical properties of the target word. For example, some English adjectives use “more” / “most” for comparative or superlative forms, while other attach “-er” / “-est”. As *sequences* are validated on a stage before the target words are fetched, the grammatical properties of the target word are still unknown. To solve this catch-22, **Carabao** uses what we call “*late sequences*”. To do this, a regular sequence performs its basic transformations then a database designer must specify that it might be used as a basis for *late sequences*. A sequence member (or members) marked for use in *late sequences* will be used to compare with other sequences in the same family.

Let us examine the example of comparative adjectives in English. We need to define 3 sequences. One will serve as a “gateway”. The other two will be “more” + adjective” and “adjective in comparative form”. The “gateway” sequence will not bear any rule unit tags. The other two, however, will.

“more”_{identity=2} + **adjective**_{use rule units for late sequences, identity=1}

The rule units of member with identity 1 (the adjective) will be exported to the sequence header rule units and used for comparison with the other two sequences:

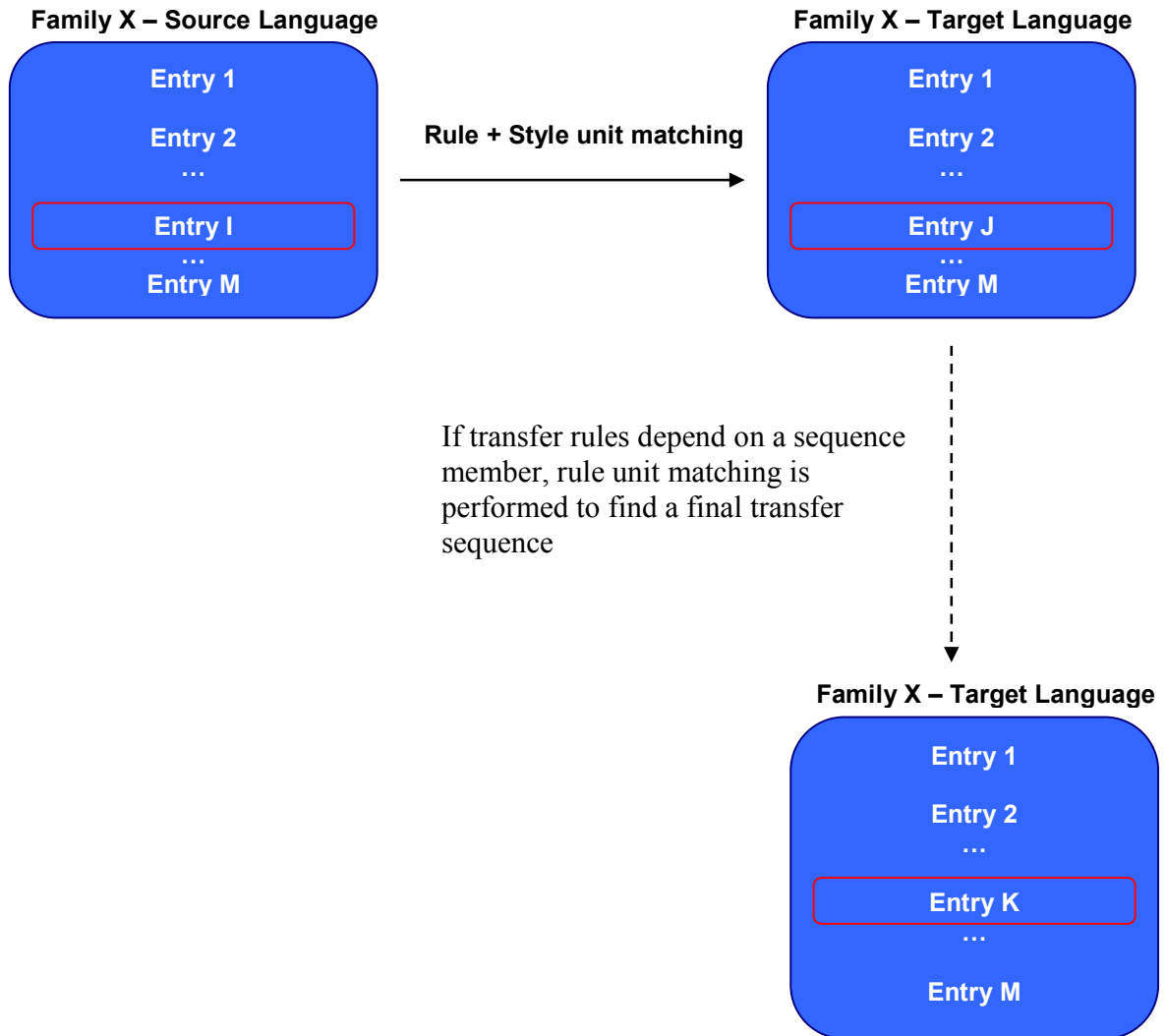
“more”_{identity=2} + **adjective**_{identity=1 (rule units: comparative form= “more”)}
adjective in comparative form_{identity=1 (rule units: comparative form= “-er”)}

The rule unit that matters to us is the one holding the way comparative form is built. It will be compared to rule units in the target sequences. After having selected the sequence, a normal transformation process will be initiated – with the difference that the source will be the “gateway” sequence.

Strictly speaking, the “gateway” can be a default final destination as well. In our example, if the comparative form is formed by adding *more*, nothing will change. Therefore, it is unnecessary to create 3 sequences – one can be dropped. If the rule units don’t match, there will be no additional transformation.

Note that, again, it does not matter what is the source language. A dictionary designer working on this rule just needs to coordinate this *family* of sequences with **any** other equivalent sequence in the database, and essentially it will enable transformation of all the other sequences in all the other languages in the database.

The diagram below summarizes how sequences work.



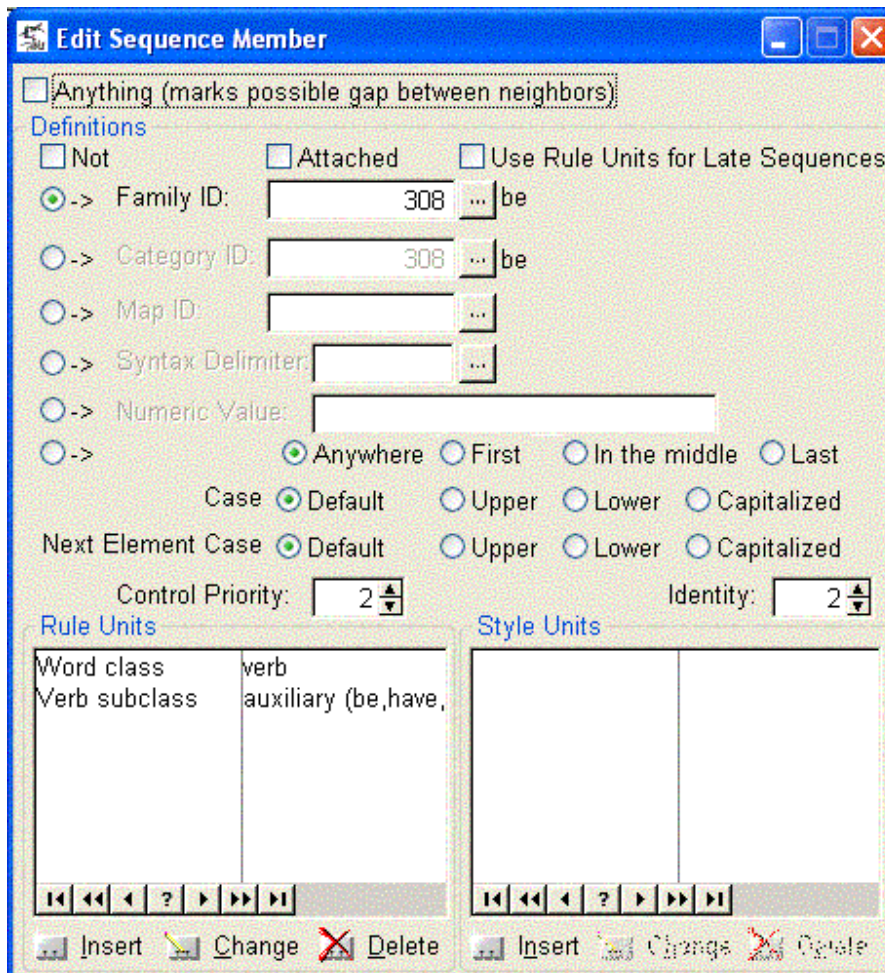
When the system is working in text analytics mode, the sequences are used to find out the exact meaning of every word, and are no less important than in the translation mode. However, there is no transformation involved.

Separation of Linguistics and System Logic

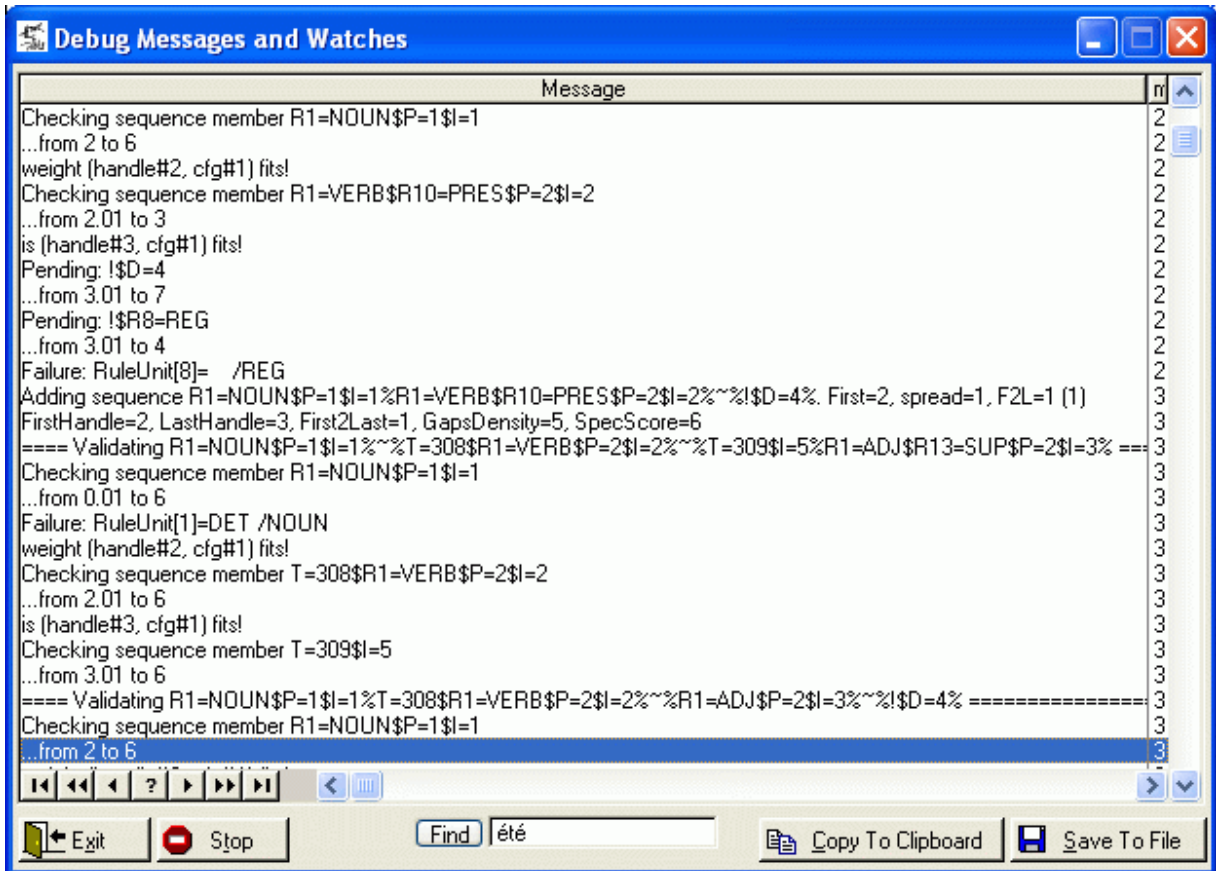
As mentioned earlier, one of the flaws in the current systems is the lack of separation of tasks for the linguist and the programmer. Sometimes linguists must learn a special language to specify the rules; in other cases, the programmers have to hard-code them. **Carabao** was designed to empower the linguists to design the database themselves without any kind of special IT knowledge or need to collaborate with the programmers or even recompile the application. Another goal was to enable end users to make their own changes and to customize the results of translation on any level. Perhaps the biggest advantage of this approach is the ability to limit the lexical database to the rules and lexicon of the handled domain, no more, no less, yielding on-demand specialized system.

The core logic of **Carabao** is abstract. It does not contain any linguistic concepts. Even the basics – part of speech, conjugation patterns, gender, and agreement in collocations – are stored in the database and can be changed to accommodate the user’s needs.

The entire lexical database, including dictionaries, grammatical rules, collocations, and even shallow parsing, is managed by **Carabao** Data Manager Utility. To simplify data entry and eliminate the need to learn the mini-language of **Carabao**’s sequences, the sequences are automatically created by a GUI. See snapshot below:



As building transfer rules is never an easy task, a special tracer logs the entire translation process:



The tracer acts as a debugger for the rules and sequences.

Small Footprint

Using relational database allows for loading rules and lexical data on demand – what we call AdaptiveLoading™. Except for a small amount of fixed data loaded during the initial stage, a word or a sequence is not loaded until there is a possibility that it could be used in the currently processed content.

Once a record is read, it is cached into memory to be used in the next reads. To prevent redundant attempts to read from the database, if a word encountered in the text is not found in the database, this failed attempt is also stored in the cache. A lexicon of an average human rarely exceeds 2,000-3,000 words. Most NLP systems today load entire dictionary, which is no less than 10,000-20,000 and is usually closer to 100,000 entries. Our tests have shown that as the word usage grows, the caching efficiency approaches $O(\log N)$, with an initial peak of fixed amount of data loaded. We also provide tools to indicate the number of loaded rules and lexical entries.

In addition to the efficient algorithm, we selected a backend suitable for our needs, without any bells and whistles, a transactional database working in local mode. It means that it is well suited for usage in mobile devices with limited resources.

What about Accuracy?

Having said all this, we are not going to dismiss the most daunting problem of machine translation and natural language processing – accuracy. We mimic the human translation process in which weighed response is given based on syntactic, contextual and semantic analysis.

In addition to the traditional means of disambiguation, such as morphologic, contextual and semantic analysis, **Carabao's** architecture allows for more means giving it higher accuracy.

The editable dictionary makes it easy to experiment with and add grammatical information, and the morphological analysis is enhanced with “pseudo-grammatical” information. For example, prepositions pertaining to events or time units are only used in conjunction with time units or events. A word *speech* can mean a means of communication or an act of delivering a speech. If we say, *during the speech*, the meaning has to be an act.

Contextual analysis is implemented via sequences. In general, meanings that result in a tree of sequences with the most branches are assumed to be the most likely to be correct. The results of the contextual analysis will be one of the inputs for the selection routine. Other inputs are more interesting, however.

As a general rule, a text's subject is a continuous function. This assumption is widely used when restricting dictionary in use to a certain domain – technical, financial, etc. **Carabao** has taken this a bit further. Not only is the domain of discourse is determined automatically on a sentence level, but also multiple domains within one text are supported. In addition, domains are derived from the lexical units, thus unlimited hierarchy of domains is possible. **Carabao** keeps track of the most dominant domain text-wide (global), and sentence-wide (local). This helps resolving ambiguities, for example, when *nutty professor* and *nutty flavor* are found in the same text.

As we all know, selection of styles is usually even more consistent and continuous than domains of discourse. Dominant styles text-wide, sentence-wide and sequence-wide are counted and analyzed. Carabao is even able to make sense of colloquial language – such as “*He shot someone and got a chair*”. The domain is identified as “crime” and “chair” is interpreted as “electric chair”.

The semantic analysis can be dependent on information accumulated during the process. For example, *I saw him mixing an orange juice and vodka. What are you doing, I asked. Screwdriver, he replied.* Carabao is able to identify ‘screwdriver’ as a cocktail.

Summary

There is much more to **Carabao** than was presented above. We have only covered the essential aspects. We do not claim to have achieved a breakthrough; **Carabao** is a clever engineering solution rather than an invention. It was meant to address the problems of the modern natural language processing systems utilizing their achievements. Most of its enhancements and unique features rely on or derive from the way the data is organized.

Contact Details



To find out more, please email: info@digitalsonata.com