Regular Expressions

Sometimes, you want to query for different variants of a word or for different words at the same time. E.g. you might want to find different forms of *to eat* in French, like *manger, mangent, mangé* etc. For such queries you need Regular Expressions (short: RegEx).

In order to formulate RegEx expressions in ANNIS, you put your query in between slashes. In the example above, the query might look like *tok=/mang.*/*

A (very) short introduction to RegEx

"In computing, regular expressions, also referred to as RegEx or RegExp, provide a concise and flexible means for matching strings of text, such as particular characters, words, or patterns of characters." (http://en.wikipedia.org/wiki/Regular_expression][Wikipedia).

As Wikipedia tells us, RegEx takes a pattern of characters you enter into the search field and looks for matches of these characters in the database. Let us assume that the database to be queried is a string of characters like "the man manually attached the tube in Manchester" without any separation of tokens and words. Let us now query the three letters *man*. In this case, RegEx looks for the letter <m> followed by an <a> and then an <n> in the database, regardless of what the pattern is preceded or followed by. As a result, you will get *man* and *manual*, but you will not get *Manchester*, because the RegEx search is case sensitive, see below.

However, RegEx also allows you to search for such things as alternatives (*man* or *men*), for word boundaries or for the beginning or end of a line (here: beginning or end of a message). In fact, RegEx offers more than you can dream of. It is a syntax widely spread in programming languages, but of course we cannot cover all the details here. Instead, we try to offer an easy overview over the functions you might use most often in this corpus. For more information, we refer you to http://www.regular-expressions.info/regular-expressions.info.

Case sensitivity

Your search is case sensitive, i.e. the system does not differentiate between upper and lower case. A query for *MAN* or for *man* or for *mAn* are completely different. If you want to query for all three variants, you have to work with alternatives (see below), thus, eg. */[mM][aA][nN]/* or *(man|Man|MAN|MAn|MaN|maN)*.

Characters, letters and digits

In RegEx, a character is not the same as a letter. Basically, everything you enter with one key on your keyboard is one character (Sometimes you use two keys to type in one character, e.g. in the case of a circumflex such as <ê>. This is still considered to be one character). So a digit, a full stop (.), a TAB or a carriage return (ENTER, RETURN etc.) is also a character you can search for.

Letters

Simple

You can search for every letter or combination thereof in the corpus by just typing it into the search field. Your sequence of letters, however, have to make up the whole token

Example: <man> will search for a lowercase <m> followed by a lowercase <a> and a lowercase <n>.

Alternatives

If you want to have alternative letters in one specific spot you can put the alternatives into square brackets.

Example: m[aei]n will look for occurrences of either

- man
- men
- min

Variable letters

If you are looking for any letter, you can use <\w> (Remember as: word character.), i.e. a backslash followed by a <w>.

Example: m\wn will look for (among others):

- mAn
- mBn
- mCn

- man
- mbn
- mcn

Something similar can be achieved with [a-z] and [A-Z] respectively. Here you look for occurrences of any letter as well, but this time case sensitive.

E.g <m[A-Z]n>

This search string can also be reduced to e.g. <[m-q]>to find any letter between a-q, however useful this may be.

N.B.: <\w> covers all letters from A to z, i.e. uppercase and lowercase. In our corpus, it also includes special letters like äöüàéèß. However, it does not include special characters such as punctuation, spaces, <&> etc.

Any character

If you want to search for any character, use a fullstop.

Example: m.n will look for (among others): mAn mBn man mbn m&n m n m_n m?n

Diacritica

This corpus is set up so as to recognize umlauts and letters with accents as individuals (Keep in mind that this is not the case in many other uses of RegEx. Especially in programs that were developed in the US, a <ü> is not considered as a letter but rather as a boundary). Searching for *mange* will therefore not find any occurrences of *mangé*.

Digits

Just like <\w> above, you can use <\d> (Remember as: digit) to stand in for any digit.

Example: n\d will look for (among others): n0 n1 n9

Separators

Individual separating characters

Many different characters can occur in between your letters and digits: comas, full stops, spaces etc. Most of these can just be used as they are, i.e:

- space
- coma
- dash (-)
- semicolon (;)
- curly brackets ({})
- colon (:)
- ampersand (&)
- percent (%)
- exclamation mark (!)

NB: most of these characters do have a special function as well if they appear in a specific position. As you will see below, { } is one of the possible way to search for repeating characters. Thus, the character { can be recognized as a character in its own right or as a syntactic function depending on its position. The same goes for most of these characters.

Other separators are reserved by the RegEx syntax. To use them by their ordinary value, you have to escape them, i.e. you have to place a backslash in front of them. Thus, you type in <m*n> to look for *m*n*. These characters are:

- asterisk (*)
- full stop (.)
- all other brackets ([()])
- slash, pipe and backslash (/|\)
- question mark (?)
- plus (+)
- dollar ($)
- caret (^)

Did we forget anything? Well possible. Just type in the character you are wondering about on its own. If you get an error, you have to escape it.

Word boundaries

In ANNIS you can query on different layers. You can, e.g. search for a string of characters in every token or you can search for the same string over whole messages (please keep in mind: this approach is very slow and can result in time-outs!). Depending on which approach you choose for, you have to consider the surrounding environment to your search string.

Let us look again at the phrase "the man manually attached the tube in Manchester". On the **token level**, this phrase consists of eight individual tokens:

the	man	manually	attached	the	tube	in	manchester

On the **message level**, on the other hand, this is a string with characters and spaces:

the man manually attached the tube in Manchester

Accordingly, the system for querying is different. if you query for *man* on the token level, you will find exactly one occurrence, namely the token *man*, because all other tokens contain more than those three characters, e.g. *manually* contains five more characters.

If you query for *man* on the message level, you will find nothing, because ANNIS will search for a whole message that contains only these three characters. In order to actually find the word you are looking for, you have to query for "any characters followed by the string *man* followed by any characters" (the function "any characters will be introduces later on). Such a string will look like:

msg=/.*?man.*/

and will find *man* but also *manually*.

If you want to find only *man*, you have to query for the three letters surrounded by boundaries (ie. spaces, tabs, fullstops, commas, new-lines etc.). The string for a boundary is *\b*. The query for *man* and only *man* within a message would thus look as follows:

msg=/.*?\bman\b.*/

Quantifiers

Sometimes you might be looking for an expression which can be written with or without repeating letters. E.g. you might want to look for *hallo, haaallo, haloooo* etc. in the corpus. Since you do not know how often the individual vowels were repeated, you have to use quantifiers to tell the system that the preceding character can be repeated a certain number of times. Your options are as follows:

- **?** a question mark means a repetition of 1 or 0 times
- * an asterisk means a repetition of 0 or more times
- **+** an plus sign means a repetition of 1 or more times
- **{n,m}** two boundaries in curly brackets mean a repetition of at least n but not more than m times

Example: /h+a+l+o+/ will find all variants of hallo

Using quantifiers is much more capable and demanding than this. The examples given here are called *greedy*, there are also *non greedy quantifiers* and *possessive ones*, which basically look for the last instead of the first occurrence of a token. Please refer to a more explicit manual for these functions.

Hint: it you find these options too complicated, consider using <{n,m}> only. With this function, you can fulfill nearly all requirements and it is the easiest function to remember.

Alternatives

Above, you have seen that you can query for different letters in one spot, e.g. you can search for *man* and *men* with the expression <m[ae]n>. But what if you want to look for either *n8* or *night* or *nacht* or *nuit*? Here you have to set a <8> equal to the letters <eight>, <acht> and <uit>. To achieve this,

you set the whole expression in parentheses and separate the individual variants by a pipe (|).

Example: n(8|acht|ight|uit) will look for: n8 nacht night nuit

A final word

What you have read here, is only a selection of the possibilities RegEx offers. To keep things more or less simple for you, we tried to document all the features you are likely to use while omitting everything you probably will not care about. Also, there are different implementations of RegEx in different programs and they support different features or not. Thus, if you want to use RegEx more intensively or in other places, please read the according manual. If you need more functions, check http://www.regular-expressions.info/regular-expressions.info .