**Extended notes on Search (and introductory notes on feature vectors)**

Revision: Z::1.5j

Tokens, represented by individual letters in this document, are ordinarily words in the lexicon to be searched. However, they can instead be a variety of feature of the lexical word:

* Lemmas
* Person/Number (1st, 2nd, or 3rd person / singular or plural)
* Part of Speech
* etc…

These feature items can be utilized instead of the word itself and they can be ANDed ( & ) and ORed ( | ). Consider a token as follows:

couldst

Such a search could alternatively be expressed using lemma (#-prefix) and part-of-speech:

#could&/vmd2/

Or a more relaxed search using a suffix could be executed as:

could#modern

**Bitwise Part-of-Speech and person number is also possible using these tables:**

WordClass (12 bits)

Person/Number (4 bits)

|  |  |  |
| --- | --- | --- |
| **Description** |  | **Left-Most Nibble** |
| Person bits |  | 0x3--- (0b--11) |
| Number bits |  | 0xC--- (0b11--) |
| Indefinite |  | 0x0--- (0b--00) |
| 1st Person |  | 0x1--- (0b--01) |
| 2nd Person |  | 0x2--- (0b--10) |
| 3rd Person |  | 0x3--- (0b--11) |
| Singular |  | 0x4--- (0b01--) |
| Plural |  | 0x8--- (0b10--) |
| WH\* |  | 0xC--- (0b00--) |

|  |  |
| --- | --- |
| **Description** | **Right 3 Nibbles** |
| NounOrPronoun | 0x-03- |
| Noun | 0x-01- |
| Noun: unknown gender | 0x-010 |
| Proper Noun | 0x-03- |
| Pronoun | 0x-02- |
| Pronoun: Neuter | 0x-021 |
| Pronoun: Masculine | 0x-022 |
| Pronoun: Non-feminine**\*** | 0x-023 |
| Pronoun: Feminine | 0x-024 |
| Pronoun/Noun: Genitive | 0x-0-8 |
| Pronoun: Nominative | 0x-06- |
| Pronoun: Objective | 0x-0A- |
| Pronoun: Reflexive | 0x-0E- |
| Pronoun: no case/gender | 0x-020 |
| Verb | 0x-1-- |
| to | 0x-200 |
| Preposition | 0x-400 |
| Interjection | 0x-800 |
| Adjective | 0x-A00 |
| Numeric | 0x-B00 |
| Conjunction | 0x-C0- |
| Determiner | 0x-D0- |
| Particle | 0x-E00 |
| Adverb | 0x-F00 |

Therefore, while terse, the previous search could be precisely executed as:

could#modern&/61--/

**Consider:** “p q … r s … [t u] ... v (w x) [y z]” Since we search ordered tokens sequentially, we don’t need to track position (only adjacency).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Token** | **Token Sequence** | **Anchor/Adjacency**  **(-1/1)** | **Subgroup**  **(unordered)** | **Function Call** | **Results:** |
| p | Anchor/T1 | -1 | 0 | P = find(&cursor, p, 1, &span) | ++cursor, --span |
| q | T2 | 1 | 0 | Q = P & find(&cursor, q, 1, &span) | ++cursor, --span |
| r | T3 | 0 | 0 | R = Q & find(&cursor, r, span, &span) | cursor+=pos, span-=pos |
| s | T4 | 1 | 0 | S = R & find(&cursor, s, 1, &span) | ++cursor, --span |
| t | T5 | 0 | (1) | TU = S & find(&cursor, [t u], span, &span) | cursor+=pos, span-=pos |
| u | T6 | 0 | (1) |  | cursor+=pos, span-=pos |
| v | T7 | 0 | 0 | V = TU & find(&cursor, v, span, &span) |  |
| (w x) | T8 | 1 | 0 | WX = V & find(&cursor, w|x, span, &span) |  |
| y | T9 | 0 | (2) | YZ = WX & find(&cursor, [y z], span, &span) |  |
| z | Terminal/T10 | 0 | (2) |  |  |

FOUND=Z

Consider the same list unquoted: p q r s t u v (w x) y z

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Token** | **Token Sequence** | **No Adjacency** | **Unordered** | **Function Call** | **Results:** |
| p | T1 | 0 | (1) | P = find(&cursor, p, span, &span) | n/a |
| q | T2 | 0 | (1) | Q = find(&cursor, q, span, &span) | n/a |
| r | T3 | 0 | (1) | R = find(&cursor, r, span, &span) | n/a |
| s | T4 | 0 | (1) | S = find(&cursor, S, span, &span) | n/a |
| t | T5 | 0 | (1) | T = find(&cursor, t, span, &span) | n/a |
| u | T6 | 0 | (1) | U = find(&cursor, u, span, &span) | n/a |
| v | T7 | 0 | (1) | V = find(&cursor, p, span, &span) | n/a |
| (w x) | T8 | 0 | (1) | WX = find(&cursor, w|x, span, &span) | n/a |
| y | T9 | 0 | (1) | Y = find(&cursor, y span, &span) | n/a |
| z | T10 | 0 | (1) | Z = find(&cursor, z, span, &span) | n/a |

FOUND = P & Q & R & S & T U & V & WX & Y & Z

**NOTES:**

* … [t u] is equivalent to [t u] without the ellipses prefix. But the ellipses postfix [t u] ... is significant.
* Quelle spec version 1.0.1.5i deprecated the (w x) syntax, replacing it at the feature level as w|x. Supporting a single [unified] or-syntax greatly reduces the complexity of Quelle search algebra. Currently, (w x) is be silently mapped to w|x by the standard Quelle driver.

Outer methods will be used to walk through the text:

* HashSet<UInt16> bcv = QuotedSearch(int span, bool exact, string domain, SearchSpec spec);)
* HashSet<UInt16> bcv = UnquotedSearch(int span, bool exact, string domain, SearchSpec spec);