D. Python Natural Language Functions

Def2domain Semantic function

Function definition:

```
json Def2domain (word [, wos=[], source=true|false])
```

Parameters:

```
string word - searched word
int wos [optional] - list of WOS marks in the output
boolean source [optional] - show definition source
```

Return values:

```
Returns JSON object (domain) with words that are related to the input
word based on the word’s definition.
```

Example:

```
=Def2domain('mozak')

# OUTPUT:
[['mozak', 'visok', 'dio', 'središnji', 'u', 'životinja', 'i',
    'središte', 'primati', 'informacija', 'iz', 'obrađivati', 'ih',
    'slati', 'uputa', 'izvršan', 'organ', 'on', 'sjedište',
    'inteligencija', 'nizak', 'živčan', 'sustav', 'čin', 'mreža',
    'međusoban', 'dug', 'funkcija', 'im', 'primanje', 'podražaj',
    'predavanje' ... ]]
```

Function `Def2domain` uses the definition of the word from related SOW tags to create a domain of words related to the source word. Every word from such definition is lemmatized and returned as a JSON object. The function accepts two additional optional parameters (`wos` and `source`). If `wos` parameter is given then the output will be filtered only to these words that are tagged with listed values. It is extremely useful if some word classes (e.g. conjunctions) for which it is usually not important that tehy are a part of resulting domain are excluded from the output. Finally, the `source` parameter defines whether the source of the used definition will be shown or not.

MarkovChain Statistical function

Function definition:

```
string MarkovChain (docs, n [, startword])
```

Parameters:

```
int docs - documents that are used as a source for Markov chain
           algorithm
int n - number of words in resulting sentence
string startword [optional] - starting word of resulting sentence.
                              If left empty, random word is used.
```

Return values:

```
Returns sentence generated using Markov chain algorithm and based on
given input parameters.
```

Example:

```
=MarkovChain([1,2,3,4,5], 5, 'i')

# OUTPUT:
'i vjetar ljulja ogradu polako'
```

Generation of new sentences based on the already loaded corpora in the SSF can be done using the function `MarkovChain`. The function accepts two mandatory and one optional parameter. The first parameter `docs` is a list of ID's of documents from the corpora that will be used as a base for calculation of word occurrence probabilities. The second parameter `n` is a number of words that will appear in the resulting sentence. The third optional parameter `startword` is a string that will be used as a starting point of Markov chain algorithm. If this parameter is left out, the random word from corpora is used.

ChangeTense Morphological function

Function definition:

```
string ChangeTense (sentence, time)
```

Parameters:

```
string sentence - source sentence
string time - verb time to which source sentence will be converted
```

Return values:

```
Returns sentence in verb time defined as second parameter
```

Example:

```
=ChangeTense('vidim plavu kuću', 'aorist')

# OUTPUT:
'vidjeh plavu kuću'
```

Function `ChangeTense` is used when there is a need to change the verb's tense in sentence(s). The first parameter sentence accepts sentence(s) in which verb's tense will be changed. It is not necessary to provide whole sentence; the function will also accept only one word that needs to be changed. The second parameter is target `time`. The function first iterates over words inside a sentence. If the currently observed word is a verb (based on WOS tags the word is associated to) then function steps into a procedure of verb tense change. If the observed word is not a verb it is simply copied to an output. The verb tense change procedure firstly finds a lemma of the word and after that it goes through the lexicon and searches for a word that is tagged as a verb, in the same number and person as the source verb but with targeted verb's tense. When such a word is found it is pushed to the output stack. When the iteration process is ended, the stacked output is returned as a string from the function.

The NLF function `Gets` converts the source sentence to a list of tags under the WOS branch which is passed as a second parameter. Source sentence is tokenized and then iterated. In each iteration step, the algorithm checks which WOS tags from the corresponding branch (from the second parameter) the observed word has assigned to it. If a child tag is found it is pushed to an output stack, and if there is no appropriate tag that is assigned to an observed word - the `ULO` (Unidentified Linguistic Object) mark is returned.

Gets WOS/SOW function

Function definition:

```
string Gets (sentence, tags)
```

Parameters:

```
string sentence - source sentence
string tags - parent WOS/SOW tag
```

Return values:

```
Converts sentence to a tag pattern based on WOS parent parameter
```

Example:

```
=Gets('vidim plavu kuću', 'Vrsta riječi')

# OUTPUT:
Glagol Pridjev Imenica
```

Function `Plural` as an input parameter takes string (a source sentence) which is converted into plural. The function iterates over words in the sentence and for every word that is in singular first it finds its lemma which is used to find a word in lexicon with the same lemma but is tagged with WOS mark of plural. Such word is pushed to an output stack. Once the iteration process is finished the stack output is returned as a string value.

Plural Morphological function

Function definition:

```
string Plural (sentence)
```

Parameters:

```
string sentence - source sentence
```

Return values:

```
Converts sentence in singular to a plural
```

Example:

```
=Plural('vidim plavu kuću')

# OUTPUT:
vidimo plave kuće
```

Functions `wos` and `sow` are used when there is a need to get an ID of a specific WOS/SOW tag for later usage inside other functions. Both return JSON objects and are called without any parameters.

wos **WOS/SOW function**

Function definition:

```
json wos()
```

Parameters:

```
NONE
```

Return values:

```
Returns JSON object with all WOS marks.  Each element has ID of
the WOS mark and its name.
```

Example:

```
=wos()

# OUTPUT:
[(1, 'Vrsta riječi'), (2, 'Imenica'), (3, 'Zamjenica'), (4,
    'Pridjev'), (5, 'Glagol'), (6, 'Broj'), (7, 'Prilog'), ... ]
```

sow **WOS/SOW function**

Function definition:

```
json sow()
```

Parameters:

```
NONE
```

Return values:

```
Returns JSON object with all SOW marks.  Each element has ID of
the SOW mark and its name
```

Example:

```
=sow()

# OUTPUT:
[(1, 'Opće'), (2, 'Živo'), (3, 'Pojam'), (9, 'Ime'), (16, 'Osoba'),
(17, 'Tijelo'), (18, 'Doživljaj'), (19, 'Spoznaja') ... ]
```

Following two functions (`CountWOS` and `CountSOW`) are used for counting words in the lexicon which are tagged with some specific tag. The functions accept ID of the WOS/SOW tag as the input parameter, and returns the number of occurrences as an integer.

CountWOS WOS/SOW function

Function definition:

```
int CountWOS (id)
```

Parameters:

```
int id - ID of the WOS tag
```

Return values:

```
Returns the number of words tagged with target tag
```

Example:

```
=CountWOS(2)


# OUTPUT:
178968
```

CountSOW WOS/SOW function

Function definition:

```
int CountSOW (id)
```

Parameters:

```
int id - ID of the SOW tag
```

Return values:

```
Returns the number of words tagged with target tag
```

Example:

```
=CountSOW(97)


# OUTPUT:
35699
```

Above examples shows the output of function `CountWOS` for WOS tag with the ID 2 (Nouns), and the result shows that in Croatian language (according to the SSF's lexicon), there are 178,968 words which are tagged as nouns. The function `CountSOW` when called with the parameter ID 97 (Croatian WordNet definition), show that there are 35,699 words in the SSF's lexicon which have definition.

The following set of functions are intended for use in the administrative purposes (creating new words, multiwords, lemmas and assigning or removing specific WOS/SOW tags to them). For usage of all assignment and removing functions, the administrative rights are required.

AssignWOS WOS/SOW function

Function definition:

```
    boolean AssignWOS (wordid, wosid, [, wosvalue])
```

Parameters:

```
    int wordid - ID of the target word
    int wosid - ID of the WOS tag
    string wosvalue [optional] - if applicable, the value of the WOS tag
```

Return values:

```
    Returns True upon success, or False upon failure
```

Example:

```
    =AssignWOS(2301, 5)
    # OUTPUT:
    True
```

AssignSOW WOS/SOW function

Function definition:

```
    boolean AssignSOW (wordid, sowid, [, sowvalue])
```

Parameters:

```
    int wordid - ID of the target word
    int sowid - ID of the SOW tag
    string sowvalue [optional] - if applicable, the value of the SOW tag
```

Return values:

```
    Returns True upon success, or False upon failure
```

Example:

```
    =AssignSOW(2301, 12)
    # OUTPUT:
    True
```

Both functions, `AssignWOS` and `AssignSOW` are used for assigning WOS/SOW tags to words. These functions are extremely useful in the process of automated enrichment of lexicon (either by the inclusion of external resources, or in redacting purposes).

Similarly to functions `AssignWOS` and `AssignSOW`, function `AssignWOSLemma` and `AssignSOWLemma` are used for assigning WOS/SOW tags to lemmas.

AssignWOSLemma WOS/SOW function

Function definition:

```
    boolean AssignWOSLemma (lemmaid, wosid, [, wosvalue])
```

Parameters:

```
    int lemmaid - ID of the target lemma
    int wosid - ID of the WOS tag
    string wosvalue [optional] - if applicable, the value of the WOS tag
```

Return values:

```
    Returns True upon success, or False upon failure
```

Example:

```
    =AssignWOSLemma(1200, 5)

    # OUTPUT:
    True
```

AssignSOWLemma WOS/SOW function

Function definition:

```
    boolean AssignSOWLemma (lemmaid, sowid, [, sowvalue])
```

Parameters:

```
    int lemmaid - ID of the target lemma
    int sowid - ID of the SOW tag
    string sowvalue [optional] - if applicable, the value of the SOW tag
```

Return values:

```
    Returns True upon success, or False upon failure
```

Example:

```
    =AssignSOWLemma(1200, 12)

    # OUTPUT:
    True
```

Finally, the last lexicon entities, multiword expressions (MWEs), can also be tagged with special functions `AssignWOSMWE` and `AssignSOWMWE`.

AssignWOSMWE WOS/SOW function

Function definition:

```
boolean AssignWOSMWE (mweid, wosid, [, wosvalue])
```

Parameters:

```
int mweid - ID of the target MWE
int wosid - ID of the WOS tag
string wosvalue [optional] - if applicable, the value of the WOS tag
```

Return values:

```
Returns True upon success, or False upon failure
```

Example:

```
=AssignWOSMWE(3103, 5)

# OUTPUT:
True
```

AssignSOWMWE WOS/SOW function

Function definition:

```
boolean AssignSOWMWE (mweid, sowid, [, sowvalue])
```

Parameters:

```
int mweid - ID of the target MWE
int sowid - ID of the SOW tag
string sowvalue [optional] - if applicable, the value of the SOW tag
```

Return values:

```
Returns True upon success, or False upon failure
```

Example:

```
=AssignSOWMWE(3103, 12)

# OUTPUT:
True
```

Removing tags from words, lemmas and MWE's can be done with functions: `RemoveWOS` and `RemoveSOW` for word's lexicon, `RemoveWOSLemma` and `RemoveSOWLemma` for lexicon of lemmas and `RemoveWOSMWE` and `RemoveSOWMWE` for the MWE lexicon. The remove functions have the same parameters layout as assignment functions.

RemoveWOS WOS/SOW function

Function definition:

```
boolean RemoveWOS (wordid, wosid, [, wosvalue])
```

Parameters:

```
int wordid - ID of the target word
int wosid - ID of the WOS tag
string wosvalue [optional] - if applicable, the value of the WOS tag
```

Return values:

```
Returns True upon success, or False upon failure
```

Example:

```
=RemoveWOS(2301, 5)


# OUTPUT:
True
```

RemoveSOW WOS/SOW function

Function definition:

```
boolean RemoveSOW (wordid, sowid, [, sowvalue])
```

Parameters:

```
int wordid - ID of the target word
int sowid - ID of the SOW tag
string sowvalue [optional] - if applicable, the value of the SOW tag
```

Return values:

```
Returns True upon success, or False upon failure
```

Example:

```
=RemoveSOW(2301, 12)


# OUTPUT:
True
```

RemoveWOSLemma WOS/SOW function

Function definition:

```
    boolean RemoveWOSLemma (lemmaid, wosid, [, wosvalue])
```

Parameters:

```
    int lemmaid - ID of the target lemma
    int wosid - ID of the WOS tag
    string wosvalue [optional] - if applicable, the value of the WOS tag
```

Return values:

```
    Returns True upon success, or False upon failure
```

Example:

```
    =RemoveWOSLemma(1200, 5)


    # OUTPUT:
    True
```

RemoveSOWLemma WOS/SOW function

Function definition:

```
    boolean RemoveSOWLemma (lemmaid, sowid, [, sowvalue])
```

Parameters:

```
    int lemmaid - ID of the target lemma
    int sowid - ID of the SOW tag
    string sowvalue [optional] - if applicable, the value of the SOW tag
```

Return values:

```
    Returns True upon success, or False upon failure
```

Example:

```
    =RemoveSOWLemma(1200, 12)


    # OUTPUT:
    True
```

RemoveWOSMWE   WOS/SOW function

Function definition:

```
    boolean RemoveWOSMWE (mweid, wosid, [, wosvalue])
```

Parameters:

```
    int mweid - ID of the target MWE
    int wosid - ID of the WOS tag
    string wosvalue [optional] - if applicable, the value of the WOS tag
```

Return values:

```
    Returns True upon success, or False upon failure
```

Example:

```
    =RemoveWOSMWE(3103, 5)


    # OUTPUT:
    True
```

RemoveSOWMWE   WOS/SOW function

Function definition:

```
    boolean RemoveSOWMWE (mweid, sowid, [, sowvalue])
```

Parameters:

```
    int mweid - ID of the target MWE
    int sowid - ID of the SOW tag
    string sowvalue [optional] - if applicable, the value of the SOW tag
```

Return values:

```
    Returns True upon success, or False upon failure
```

Example:

```
    =RemoveSOWMWE(3103, 12)


    # OUTPUT:
    True
```

The creation of new words in the SSF's lexicon is usually done with the integrated morphology generator (which can be accessed from the administrative module of the GUI). In cases when the process of words creation is done from external applications (over the API), or as a part of the NLF script, the usage of functions `InsertWord`, `InsertMWE` and `InsertLemma` is necessary.

InsertWord Syntactic function

Function definition:

```
    int InsertWord (word)
```

Parameters:

```
    string word - the word that is inserted
```

Return values:

```
    Returns the inserted word ID
```

Example:

```
    =InsertWord("informatika")


    # OUTPUT:
    288588
```

InsertMWE Syntactic function

Function definition:

```
    int InsertMWE (mwe)
```

Parameters:

```
    string mwe - the multiword expression that is inserted
```

Return values:

```
    Returns the inserted MWE ID
```

Example:

```
    =InsertMWE("morski pas")


    # OUTPUT:
    43317
```

Each multiword expression is first tokenized, and then for every token (word), checked over the words lexicon. If the word exists in the lexicon, the ID of the word is used in the MWE building procedure. When the word does not exist within the word's lexicon, it is automatically inserted, and assigned to the multiword expression as its part.

InsertLemma Syntactic function

Function definition:

```
int InsertLemma (lemma)
```

Parameters:

```
string lemma - the lemma that is inserted
```

Return values:

```
Returns the inserted lemma ID
```

Example:

```
=InsertLemma("raditi")

# OUTPUT:
3252
```

The inserted lemma is automatically visible in the lemmas lexicon, but to assign it to the specific word in the word's lexicon, the function `AssignLemma2Word` is used. The function accepts two parameters, the ID of the word, and the ID of the lemma and return True if the operation is successfully completed, or False if the error in the process of assignment occurred (e.g. invalid ID of word/lemma was used).

AssignLemma2Word Syntactic function

Function definition:

```
boolean AssignLemma2Word (wordid, lemmaid)
```

Parameters:

```
int wordid - ID of the word
int lemmaid - ID of the lemma
```

Return values:

```
Returns True upon success, or False upon failure
```

Example:

```
=AssignLemma2Word(3223, 2452)

# OUTPUT:
True
```

Function **Antonym** as a first parameter takes a word and generates its antonym based on a antonymity criteria which is given as the second parameter. This function relies on semantic domains which are described in Section 6.3. For the given example in the **Antonym** function the word hrv. "mladić" (eng. "*young man*") is given as the first parameter and the second parameter (criteria of antonymity) is the given "gender". The output is hrv. "djevojka" (eng. *young girl*). If for instance, instead of "gender" an "age" is set as criteria of antonymity the word hrv. "starac" (eng. "*old man*") would be returned as an output. In order that this function might work, the semantic domain with properly ordered words must exist. For the given example at least two such domains are used. First, the domain `gender` which is made of two elements (hrv. muško (eng. *male*) and hrv. žensko (eng. *female*), and the domain `age` which is made of elements hrv. rođenje (eng. *birth*), hrv. mladost (eng. *youth*), hrv. zrelost (eng. *maturity*), hrv. starost (eng. *senility*) and hrv. smrt (eng. *death*). Since all words are tagged with proper SOW features (e.g. the word hrv. mladić (eng. *young man*) is tagged with *male* and *youth* tag and the word hrv. djevojka (eng. *young girl*) is tagged with *youth* and *female* tag) it is easy to use semantic domains and find corresponding word with opposite meaning.

Antonym Semantic function

Function definition:

```
json Antonym (word, criteria)
```

Parameters:

```
string word - source word
string criteria - criteria of antonimity
```

Return values:

```
Returns an antonym of a given word based on specific criteria
```

Example #1:

```
=Antonym('mladić','spol')

# OUTPUT:
['djevojka']
```

Example #2:

```
=Antonym('mladić','dob')

# OUTPUT:
['starac']
```

Synonym Semantic function

Function definition:

```
json Synonym (word)
```

Parameters:

```
string word - source word
```

Return values:

```
Returns a list of synonyms for a given word
```

Example:

```
=Synonym("raditi")

# OUTPUT:
['izrađivati', 'uraditi', 'proizvoditi', 'stvoriti', 'proizvesti',
    'tvoriti', 'praviti', 'činiti', 'učiniti', 'stvarati', 'izraditi',
    'načiniti', 'napraviti', 'izgraditi', 'izgrađivati',
    'konstruirati', 'graditi', 'sagraditi', 'specifičan', 'prosječan',
    'obraditi', 'obrađivati', 'fabricirati', 'upravljati', 'dvoriti',
    'služiti', 'poslužiti', 'posluživati', 'umjeren', 'funkcionirati',
    'slab', 'vršiti', 'obavljati', 'izvršavati', 'provoditi',
    'producirati', 'robijati', 'dirinčiti', 'nespecifičan', 'crnčiti',
    'kulučiti', 'rintati']
```

As described in Section 3.3, the function `Synonym` uses the information from WOS or SOW tags to create a list of synonyms for a given word and return it in the form of JSON object. These lexical functions can also be chained, for example, if the function `Anyonym` is called with parameters hrv. "mladić" (eng. *young man*) as word, and hrv. "dob" (eng. *age*) as the criteria of anonymity, the output is word hrv. "djevojka" (eng. *young girl*). This output can be directly the input for the function `Synonym`. The function call `=Synonym(Antonym("mladić", "spol")[0])` will result with the following JSON object: `['dekla', 'gospođica', 'gđica', 'cura', 'mlada dama']`. In the same way any other function output can be input in another function. The function `Antonym("mladić", "spol")` is executed first, resulting with the list of antonyms for the word hrv. "mladić" (eng. *young man*). The first element of the list (i.e. word hrv. "djevojka") is then passed as the first argument in the function `Synonym`.

Collocation Semantic function

Function definition:

```
json Collocation (word)
```

Parameters:

```
string word - source word
```

Return values:

```
Returns JSON object with all collocations for a given word.
```

Example:

```
=Collocation('labav')

# OUTPUT:
['labava carinska unija', 'labava federacija', 'labava granica',
    'labava kompozicija', 'labava konfederacija', 'labava unija',
    'labave cijene', 'labave uzde', 'labave veze', 'labav režim',
    'labava politička suradnja', 'labava veza među koalicijskim
    strankama']
```

Collocations are multiword expressions (MWE) which are related to a specific word. The function `Collocation` outputs all these MWEs as a JSON object.

Freq Statistical function

Function definition:

```
int Freq (docs, tags)
```

Parameters:

```
int docs - list of document ID's
string tags - tags that are counted
```

Return values:

```
Returns number of occurances of a words which are tagged with
WOS/SOW tags from the second parameter.
```

Example:

```
=Freq([1,2], ['Imenica'])

# OUTPUT:
7283
```

MatchPattern Syntactic function

Function definition:

```
string MatchPattern (sentence, patternname)
```

Parameters:

```
string sentence - sentence which is tested against a pattern
string patternname - pattern name
```

Return values:

```
Returns part of sentence which mathes certain pattern.
```

Example:

```
=MatchPattern('Sutra ću više raditi', 'futur')

# OUTPUT:
ću raditi
```

Function `MatchPattern` tests source sentence against an already stored pattern from database. The process of pattern testing is similar to one described in Section 5.6. In the first step the sentence is enriched with relevant WOS/SOW marks and then tested against pre-prepared regular expressions which are stored in the database. Some of possible values (pattern names) are: `s` (subject), `p` (predicate), `o` (object), `aorist`, `apozicija`, `futur`, `imperfekt`, `infinitiv`, `perfekt`, `prezent`, etc.). Example pattern `MatchPattern` function (future tense), which is used to test if a sentence has auxiliary verb will (WOS ID 153) followed by a main verb (WOS ID 150) in infinitive (WOS ID 183), may look like:

```
.*?\s*(\S+)\[w:153\]\s*.*?\s*(\S+)\[w:150,183\]\s*.*?\s*
```

and when tested against enriched version of the sentence:

```
Sutra␣ću[w:153]␣više␣raditi[w:150,183]
```

matched result is:

```
ću␣raditi
```

This is a very simple example involving only WOS tags, but in some complex scenarios it is possible to combine it also with SOW tags, or even define specific word forms independently to WOS/SOW marks, just by using standard regular expression rules.

DetectPattern Syntactic function

Function definition:

```
string DetectPattern (sentence, patternname)
```

Parameters:

```
string sentence - sentence which is tested against a pattern
boolean extend - show matched part of the sentence
```

Return values:

```
Detects pattern within the sentence.
```

Example:

```
=DetectPattern('Prestao sam pisati', extend=True)

# OUTPUT:
prezent (Prestao sam)
```

Like the `MatchPattern`, the function `DetectPattern` tests the sentence in the same way but against all stored patterns in the database and outputs those that have a match.

FreqDist Statistical function

Function definition:

```
int FreqDist (docs, wos, pattern, n, len)
```

Parameters:

```
int docs - list of documents
int wos - list of WOS marks in the output
string pattern - pattern for filtering words
int n - number of letters, morphs or syllables
string len - sylab, morph or char
```

Return values:

```
Returns a number of occurances of words that have n letters, morphs
or syllables and are tagged with specific WOS tags.
```

Example:

```
=FreqDist([1,2], wos=['Imenica'], pattern='%', n=4, len='sylab')

# OUTPUT:
258
```

Function `FreqDist` returns the number of occurrences of a specific word defined by parameters within selected documents. The example above shows that there are 258 nouns that have 4 syllables in documents 1 and 2.

Ngrams | Syntactic function

Function definition:

```
string[] Ngrams (sentence, n, d)
```

Parameters:

```
string sentence - source sentence
int n - size of ngram
int d - number of neighbouring words to combine
```

Return values:

```
Returns a list of n-grams from the source sentence based on
the size of sliding window
```

Example:

```
=Ngrams("Čitala je njegove pjesme kao da prvi put otkriva snagu
    pjesničke riječi", 2, 2)

# OUTPUT:
[['Čitala', 'je'], ['Čitala', 'njegove'], ['je', 'njegove'], ['je',
    'pjesme'], ['njegove', 'pjesme'], ['njegove', 'kao'], ['pjesme',
    'kao'], ['pjesme', 'da'], ['kao', 'da'], ['kao', 'prvi'], ['da',
    'prvi'], ['da', 'put'], ['prvi', 'put'], ['prvi', 'otkriva'],
    ['put', 'otkriva'], ['put', 'snagu'], ['otkriva', 'snagu'],
    ['otkriva', 'pjesničke'], ['snagu', 'pjesničke'], ['snagu',
    'riječi'], ['pjesničke', 'riječi']]
```

Function `Ngrams`, as name says, extracts n-grams from sentences. It accepts two parametes, the first parameter `n` defines the size of n-gram, and parameter `d` defines how many neighbouring words next to the observed word will be in the output.

GwMSY WOS/SOW function

Function definition:

```
json GwMSY (ms, like, orderno)
```

Parameters:

```
string ms - morphs or syllables
string like - pattern to match syllable or morph
int orderno - position of morph or syllable within the word
                (-1 means last)
```

Return values:

```
Returns JSON object with words split into morphs or syllables
matching specific pattern.
```

Example:

```
=GwMSY('syllable','ja',-1)

# OUTPUT:
{'ja': ['bo-ja', 'im-ple-men-ta-ci-ja', 'i-de-ja',
    'ko-mu-ni-ka-ci-ja', 'ge-sti-ku-la-ci-ja', 'op-ci-ja',
    'se-lek-ci-ja', 'kog-ni-ci-ja', 'ap-strak-ci-ja', 'po-zi-ci-ja',
    'fun-kci-ja', 're-gi-ja', 'kre-a-ci-ja', 'or-ga-ni-za-ci-ja',
    'si-tu-a-ci-ja', 'for-ma-ci-ja', 'spo-zna-ja', 'pro-fe-si-ja',
    'ka-te-go-ri-ja', 'in-for-ma-ci-ja', 'e-du-ka-ci-ja',
    're-la-ci-ja', 'pu-bli-ka-ci-ja', 'lo-ka-ci-ja', 'e-mo-ci-ja',
    'kon-struk-ci-ja', 'e-sen-ci-ja', 'di-stri-bu-ci-ja', ... ]}
```

Function `GwMSY` returns JSON object with words split into syllables or morphs based on criteria provided in parameters. The first is `ms` parameter which decides whether the morphs or syllables are displayed. The `like` parameter is pattern or exact string of syllable/morph which is queried, and `orderno` parameter defines the position of searched syllable/morph within the word. If `orderno` value is set to -1 it means that only syllables/morphs which are at the end of the word are returned. The example above queries lexicon for all words with the last syllable "*ja*".

SplitSentences Syntactic function

Function definition:

```
string[] SplitSentences (sentence [, allmatches])
```

Parameters:

```
string sentence - source sentence
boolean allmatches [optional] - Show all possible matches
```

Return values:

```
Returns a list of sentences parts along with corresponding types.
```

Example:

```
=SplitSentences("Koji ne može sebi zapovijedati, ne može ni drugom.")

# OUTPUT:
[['subjektna', 0, '(^\S+\[w:136\].+\[w:5\].+),(.+\[w:5\].+)',
    'Koji[w:136] ne može[w:5] sebi zapovijedati[w:5] , ne može[w:5] ni
    drugom . ', ['Koji ne može sebi zapovijedati', 'ne može ni drugom
    .']]]
```

The `SplitSentences` function uses predefined set of syntactical patterns (known as O-structures) to split the enriched version of the complex sentence in its parts (see function `EnrichSentence`) and detect the type of the sentence. As an output it returns a list of elements. The first element is type of sentence the algorithm has found. The second element indicates if the matched sentence is in its regular or inverted version (value `0` means the sentence is in regular form, whereas `1` means that it is inverted sentence). The third element of the output list is the regular expression which was tested against the enriched version of the sentence. The fourth elements is the enriched sentence which was tested with the regular expression (the third element), and finally, the last element of the output list is another list which contains all of the sentence elements. The SSF currently contains 14 different sentence types (apozicijska (eng. appositional), atributna (eng. attribute), dopusna (eng. permissible), mjesna (eng. place), namjerna (eng. intentional), načinska (eng. modal), objektna (eng. object), poredbena (eng. compareable), posljedična (eng. subsequent), predikatna (eng. predicate), subjektna (eng. subject), uvjetna (eng. conditiona), uzročna (eng. causative), vremenska (eng. time)) which in all their variants make over 40 different syntactic patterns used by the splitting algorithm. These patterns can be edited in the O-structure tab as described in Section 7.4, with the possibility of creating a completely new patterns.

The following three functions: `DetectS`, `DetectP` and `DetectO` are used to detect subject, predicate and object within the sentences. These functions basically implement `MatchPattern` function.

DetectS Syntactic function

Function definition:

```
string DetectS (sentence)
```

Parameters:

```
string sentence - source sentence
```

Return values:

```
Returns subject from the sentence.
```

Example:

```
=DetectS('Željka kuha ručak')


# OUTPUT:
Željka
```

DetectP Syntactic function

Function definition:

```
string DetectP (sentence)
```

Parameters:

```
string sentence - source sentence
```

Return values:

```
Returns predicate from the sentence.
```

Example:

```
=DetectS('Željka kuha ručak')


# OUTPUT:
kuha ručak
```

DetectO Syntactic function

Function definition:

```
string DetectO (sentence)
```

Parameters:

```
string sentence - source sentence
```

Return values:

```
Returns object from the sentence.
```

Example:

```
=DetectO('Željka kuha ručak')

# OUTPUT:
ručak
```

For easier usage all three function are aggregated into one function `DetectSPO`

DetectSPO Syntactic function

Function definition:

```
json DetectSPO (sentence)
```

Parameters:

```
string sentence - source sentence
```

Return values:

```
Returns subject, predicate and object from the sentence as a
JSON object.
```

Example:

```
=DetectSPO('Željka kuha ručak')

# OUTPUT:
{'subject': 'Željka', 'predicate': 'kuha ručak', 'object': 'ručak'}
```

Detection of metonymies and metaphors is described in Section 7.3 and the function `DetectMetonymy` implements these principles to extract metonymy from the sentence. In the first pass sentence elements (words) are lemmatized, and checked if the word is tagged as a symbol. If such word is found, in the next step the algorithm looks for core verbs that are tagged to a symbol word. If the verb in the sentence is not in the list of the core verbs it is recognized as a metonymy.

DetectMetonymy Semantic function

Function definition:

```
json DetectMetonymy (sentence)
```

Parameters:

```
string sentence - source sentence
```

Return values:

```
Returns a metonymy if detected.
```

Example:

```
=DetectMetonymy('Kruna je rekla da su se odnosi poboljšali')

# OUTPUT:
Metonimija: Kruna rekla
```

Similarly to the `DetectMetonymy()` function, the function `DetectMetaphor()` uses SOW symbol tag and word's domains to detect metaphors from sentences. In the first step function extracts SPO roles, and for any SPO role which is tagged as a symbol, checks core domains. If the word is not in core domains, the metaphor is detected.

DetectMetaphor Semantic function

Function definition:

```
json DetectMetaphor (sentence)
```

Parameters:

```
string sentence - source sentence
```

Return values:

```
Returns a metaphor if detected.
```

Example:

```
=DetectMetaphor("Njihov predsjednik je hrabri lav.")

# OUTPUT:
Metafora: predsjednik je lav
```

The function `DetectColoc` extracts collocations from the sentence. Collocations are by definition - sequences of words or terms that co-occur more often than would be expected by chance. The function has only one parameter (sentence) which is parsed and matched against the MWE dictionary.

DetectColoc Semantic function

Function definition:

```
json DetectColoc (sentence)
```

Parameters:

```
string sentence - source sentence
```

Return values:

```
Returns a collocations if detected.
```

Example:

```
=DetectColoc('Bila je to labava carinska unija')


# OUTPUT:
labava carinska unija
```

The function `DetectFigure` parses a sentence and tries to find a stylistic figure using O-structures patterns (described in Section 7.4)

DetectFigure Semantic function

Function definition:

```
json DetectFigure (sentence)
```

Parameters:

```
string sentence - source sentence
```

Return values:

```
Returns a stylistic figure if detected.
```

Example:

```
=DetectFigure('Bilo je dugo bablje ljeto')


# OUTPUT:
Antonomasia => bablje ljeto
```

The function `EnrichSentence` is commonly used in situations where sentence needs to be expanded with WOS/SOW marks in order to be easily matched against O-structure patterns.

EnrichSentence Syntactic function

Function definition:

```
json EnrichSentence (sentence, wos, sow)
```

Parameters:

```
string sentence - source sentence
int wos - list of WOS tags which are relevant for enrichment
int sow - list of SOW tags which are relevant for enrichment
```

Return values:

```
Returns enriched version of the sentence.
```

Example:

```
=EnrichSentence('Mario voli čitati knjige','1,2,3','1,2,3')

# OUTPUT:
Mario[w:2] voli čitati knjige[w:2]
```

The function `Imperative()` takes as an argument word and returns its imperative version. This function is part of Mel’čuks MTT.

Imperative Syntactic function

Function definition:

```
string Imperative (word)
```

Parameters:

```
string word - source word
```

Return values:

```
Imperative of the source word.
```

Example:

```
=Imperative("stajati")

# OUTPUT:
stoj
```

Following two functions `Word2MULTEXT` and `Wid2MULTEXT` use the WOS/SOW tags of words for creating the MULTEXT-East tags. Both functions work on the same principle, the only difference is that the function `Word2MULTEXT` as a parameter expects string representation of a word, whereas function `Wid2MULTEXT` expects integer value of word's ID. For the given word funciton chech WOS/SOW tags which are assigned to the words and generates the MULTEXT-East string using the rules from Appendix F.

Word2MULTEXT WOS/SOW function

Function definition:

```
string[] Word2MULTEXT (word)
```

Parameters:

```
string word - target word
```

Return values:

```
Returns list of MULTEXT-East tags for a given word.
```

Example:

```
=Word2MULTEXT("stol")

# OUTPUT:
['Nmsa-', 'Nmsn-']
```

Wid2MULTEXT WOS/SOW function

Function definition:

```
string Wid2MULTEXT (wid)
```

Parameters:

```
int wid - target word's ID
```

Return values:

```
Returns MULTEXT-East tags for a given word.
```

Example:

```
=Wid2MULTEXT(2301)

# OUTPUT:
Nfpg-
```