T-KEIR Documentation

Release 1.0.0

Theresis AI6

Sep 17, 2021

CONTENTS

The searchai tools allows to apply numerous NLP tools, search, index and classify. The tools cover

- advanced tokenizer configuration
- named entities with validations rules
- keywords extraction
- SVO based on syntactic dependencies and rules
- automatic summarization
- sentiment analysis
- unsupervised classification
- relation clustering
- question and answering
- advanced query formulation and expansion based on clustering and text analysis

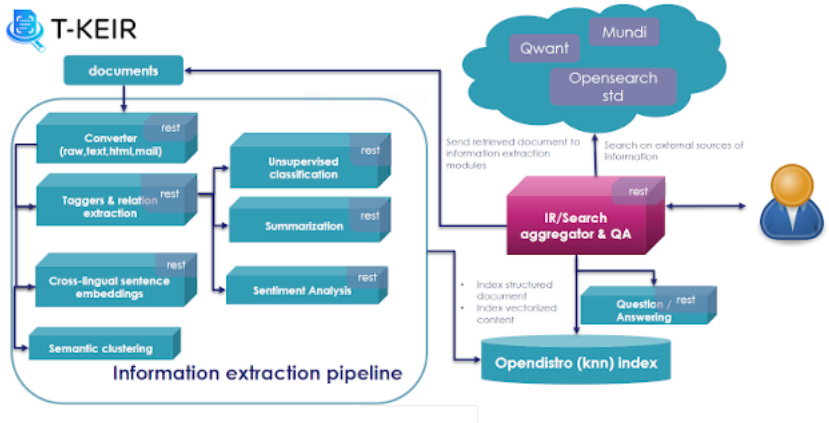
CHAPTER

ONE

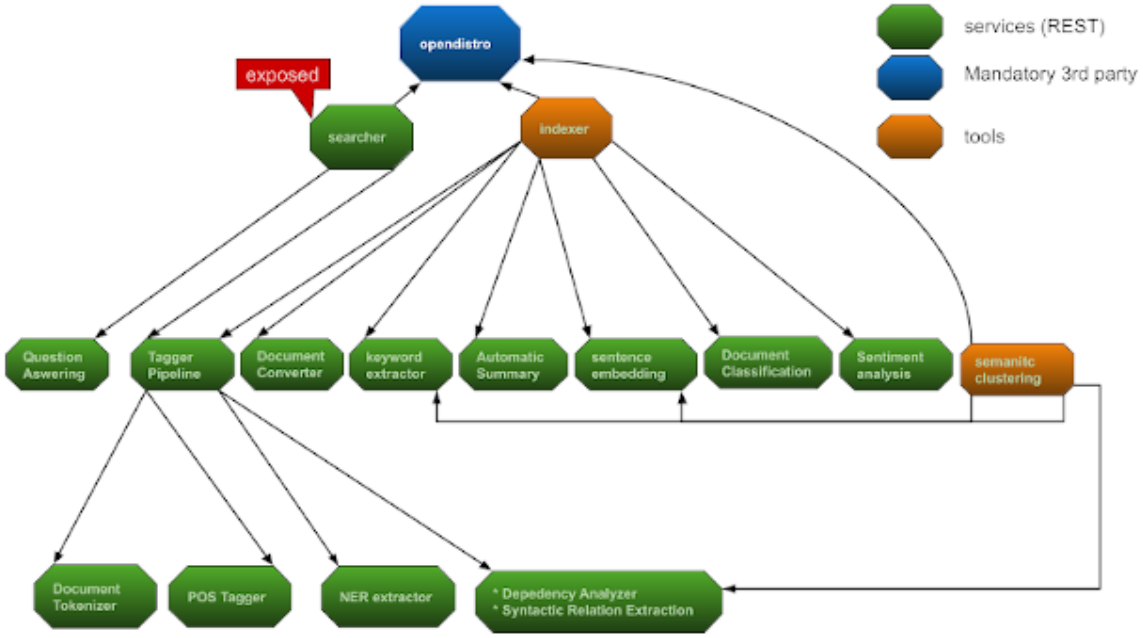
CONTENT

1.1 Overview

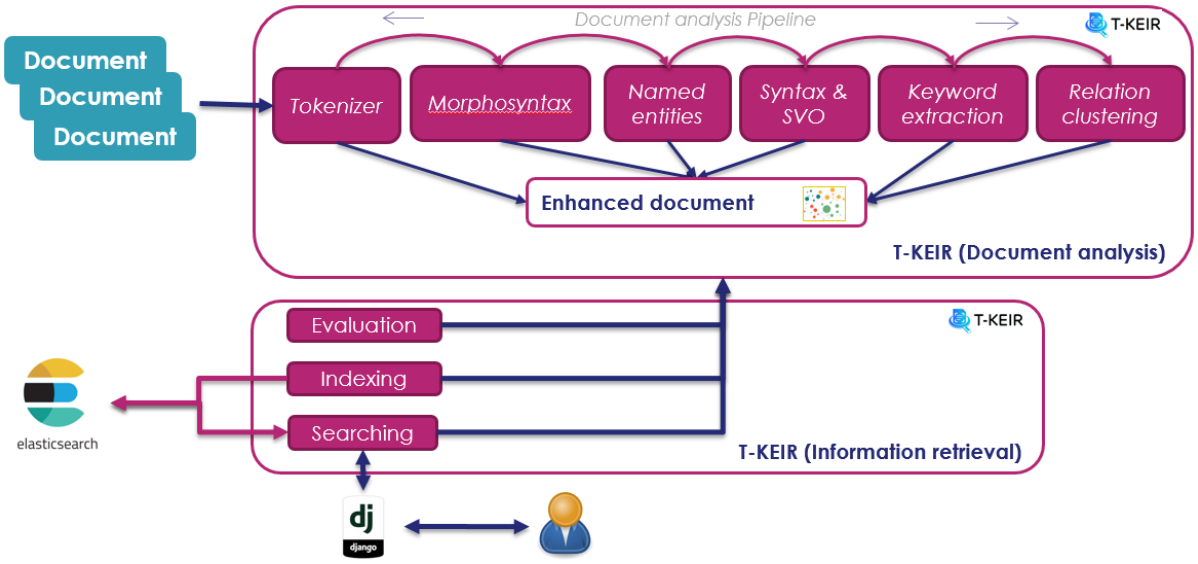
TKEIR (Thales Kowledge Extration To Information Retrieval) is a set of REST services allowing to do knowledge extraction (advanced tokenization, morphosyntax, named entities recognition, dependencies analysis, keyword extraction, unsupervised classification, sentiment analysis, automatic summary, question and answering, indexing and information retrieval)



The services architecture is:



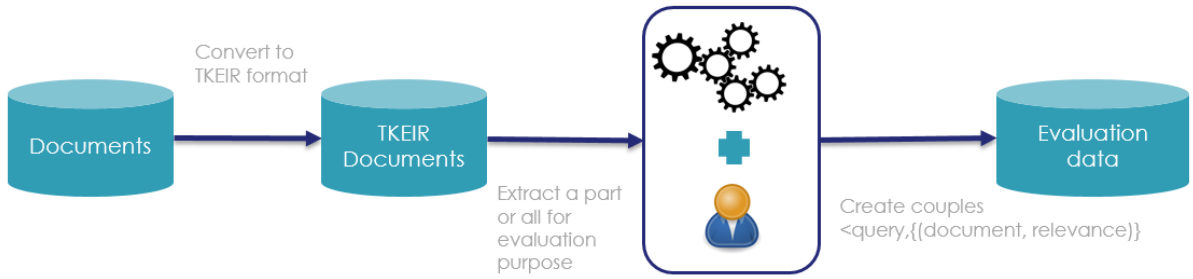
1.2 Conception

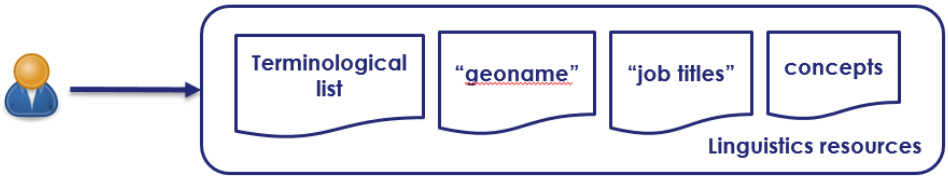


1.2.1 Data preparation

Data preparation consists in

- transform the documents into a format adapted to the tools,
- build terminological and structured resources (such as ontology concepts for example),
- construct the evaluation data.





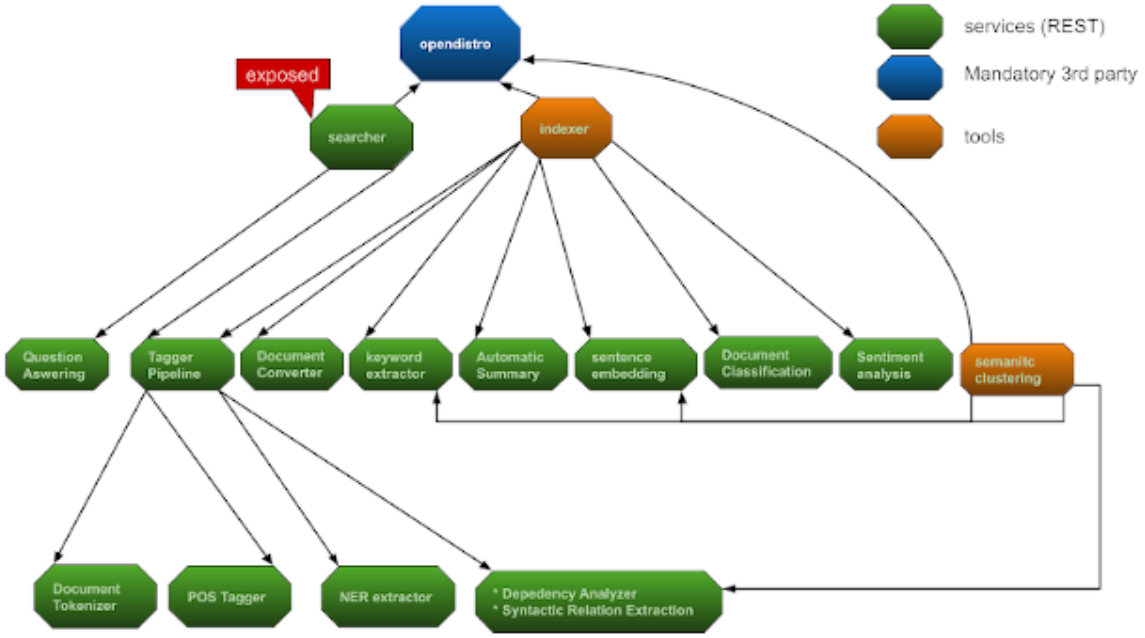
Construction of terminological lists

Linguistic resources are used by document analysis tools to extract data typed in the target domain as well as generic data such as city names (to improve the detection of named entities).

Preparation of the evaluation data

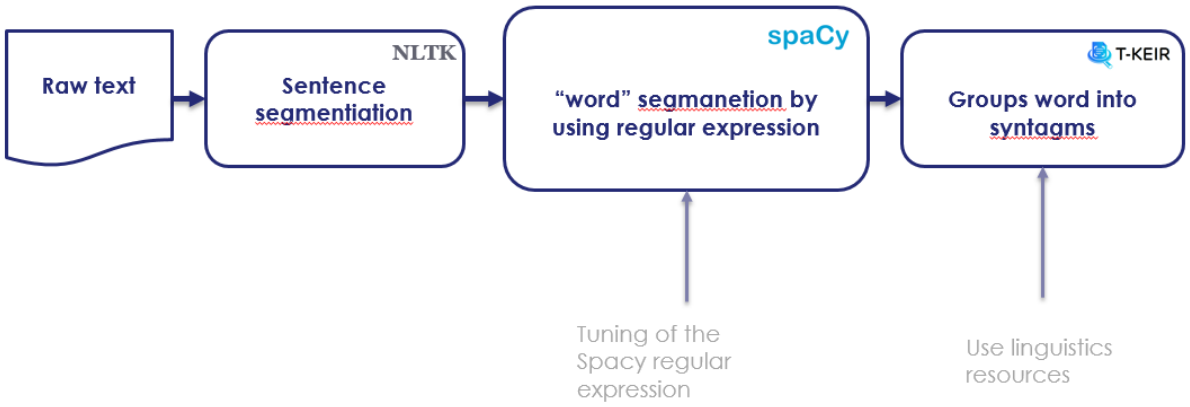
The evaluation data are constructed to know the relevance of the results returned by the search system. We seek to have a set of queries associated with the relevant documents to be returned. The goal is ultimately to assess the capacity of the search engine

1.2.2 Document Analysis



Tokenization

The tokenization phase allows a text to be segmented into linguistic units: sentences, phrases, words.



Principles Segmentation is a delicate phase that requires the use of regular expressions and strategies to group compound word.

Using regular expressions

Regular expressions allow you to define segmentation rules. These rules cover, among other things:

- The fact that the '.' Is not systematically used as the end of a sentence, in the case of a decimal number in English for example where the period is a separator
- The fact that the '-' at the beginning at the end of words is separated
- The fact that the '-' in the middle of a word is not segmented
- ...

Grouping of detached-compound words

Detached compound words often represent semantic units, for example the sequence "hot dog" should be taken as a phrase and should not be segmented into two words ("hot", "dog"). This problem is addressed in T-KEIR tools through the use of phrase list and a Trie type tree data structure.

Resource usage

Linguistic resources provide a list of phrases that can be typed (for example, the list of city names in the geoname database is labeled as a place). They can also define a notion of hierarchy in the case of an ontology of concepts. All of these resources are "compiled" into a Trie structure. This data structure can be configured to remove diacritics (add-ascii-folding option), to add a morphosyntaxic label (pos option) or a named entity label (label option)

```
{
    "format":{
        "type":"csv",
        "header":true,
        "sep":",",
        "columns":[
            {"id":4}
        ]
    },
    "name":"fortune500-industry",
    "path":"fortune500.csv",
    "exceptions": ["stopwords.txt"],
    "pos":"NOUN",
    "add-ascii-folding":true,
    "label":"industry",
    "type":"named-entity",
    "weight":10
},
{
    "format":{
        "type":"csv-zip",
        "header":true,
        "sep":"\t",
        "columns":[
            {"id":0,"concept-type":"instance"},
            {"id":1,"concept-type":"parent-instance"}
        ]

    },
    "name":"chemistry-concepts",
    "path":"chemistry.csv.zip",
    "exceptions": ["stopwords.txt"],
    "pos":"NOUN",
    "label":"chemistry-terminology",
    "type":"concept",
    "add-ascii-folding":true,
    "weight":10
},
```

Example of resource configuration file

```
,"Chevron","San Ramon, CA","Petroleum Refining","CA
"Berkshire Hathaway","Omaha, NE","Insurance: Proper
"Apple","Cupertino, CA","Computers, Office Equipmen
,"General Motors","Detroit, MI","Motor Vehicles and
,"Phillips 66","Houston, TX","Petroleum Refining","
,"General Electric","Fairfield, CT","Diversified Fi
,"Ford Motor","Dearborn, MI","Motor Vehicles and Pa
,"CVS Health","Woonsocket, RI","Food and Drug Store
,"McKesson","San Francisco, CA","Wholesalers: Healt
"AT&T","Dallas, TX","Telecommunications","TX","Dall
,"Valero Energy","San Antonio, TX","Petroleum Refin
,"UnitedHealth Group","Minnetonka, MN","Health Care
```

Example of resource file

Normalization rule

T-KEIR tools provide the ability to normalize words and perform spell checking of the most common mistakes. Here simple transformation rules are set up by means of configuration files.


```
"typos":[
    {"misspelling":"accomodation","correct":"accommodation"},
    {"misspelling":"acommodation","correct":"accommodation"},
    {"misspelling":"acheive","correct":"achieve"},
    {"misspelling":"accross","correct":"across"},
    {"misspelling":"adress","correct":"address"},
    {"misspelling":"agressive","correct":"aggressive"},
    {"misspelling":"alot","correct":"a lot"},
    {"misspelling":"apparantly","correct":"apparently"},
    {"misspelling":"appearence","correct":"appearance"},
    {"misspelling":"arguement","correct":"argument"},
    {"misspelling":"assasination","correct":"assassination"},
    {"misspelling":"basicly","correct":"basically"},
    {"misspelling":"beggining","correct":"beginning"},
    {"misspelling":"beleive","correct":"believe"},
    {"misspelling":"bizzare","correct":"bizarre"},
    {"misspelling":"buisness","correct":"business"},
    {"misspelling":"carribean","correct":"caribbean"},
    {"misspelling":"chauffer","correct":"chauffeur"},
    {"misspelling":"cemetary","correct":"cemetery"},
    {"misspelling":"collegue","correct":"colleague"},
```

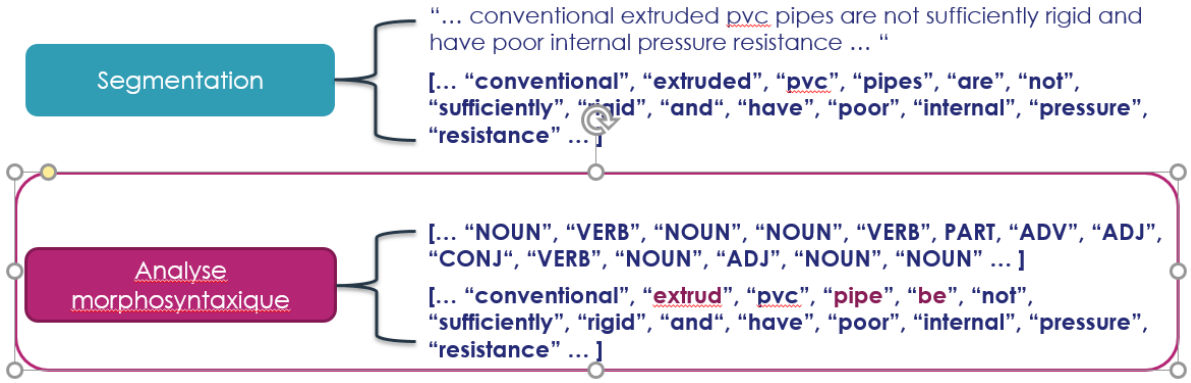

Example of typo configuration file


```
"normalization": {
    "word-mapping" : [
        {"from":"accessorise", "to":"accessorize"},
        {"from":"accessorised", "to":"accessorized"},
        {"from":"accessorises", "to":"accessorizes"},
        {"from":"accessorising", "to":"accessorizing"},
        {"from":"acclimatisation", "to":"acclimatization"},
        {"from":"acclimatise", "to":"acclimatize"},
        {"from":"acclimatised", "to":"acclimatized"},
        {"from":"acclimatises", "to":"acclimatizes"},
        {"from":"acclimatising", "to":"acclimatizing"},
        {"from":"accoutrements", "to":"accouterments"},
        {"from":"aeon", "to":"eon"},
        {"from":"aeons", "to":"eons"},
        {"from":"aerogramme", "to":"aerogram"},
        {"from":"aerogrammes", "to":"aerograms"},
        {"from":"aeroplane", "to":"airplane"},
        {"from":"aeroplanes", "to":"airplanes"},
        {"from":"aesthete", "to":"esthete"},
        {"from":"aesthetes", "to":"esthetes"},
        {"from":"aesthetic", "to":"esthetic"},
        {"from":"aesthetically", "to":"esthetically"},
        {"from":"aesthetics", "to":"esthetics"},
        {"from":"aetiology", "to":"etiology"},
        {"from":"ageing", "to":"aging"},
        {"from":"aggrandisement", "to":"aggrandizement"},
        {"from":"agonise", "to":"agonize"},
```

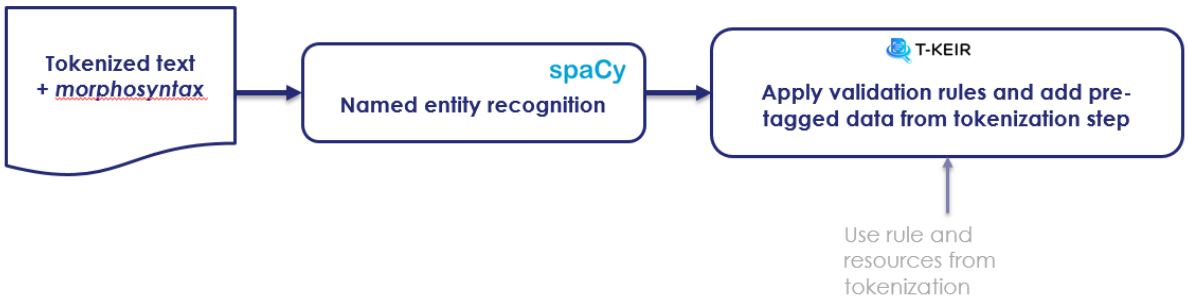

Example of normalization configuration file

Morphosyntax

The morphosyntax module is built on the Spacy library. It provides the possibility of giving each segmented term during the tokenization phase a morphosyntactic label (noun, verb, adjective, ...). This module takes advantage of the pre-tagged information from the segmentation phase by "forcing" the tags of phrases and terminology often unrecognized by the original morphosyntactic tagger. It is also this module which provides the lemmatized form (this is the canonical form of a word, for example the verb "doing" has the lemma "do") of words.



Named entities extraction



Named feature extraction involves labeling textual elements. They can be seen as <text, label> pairs where the label is the type of data, for example "city", "person", "organization", ...

Principles

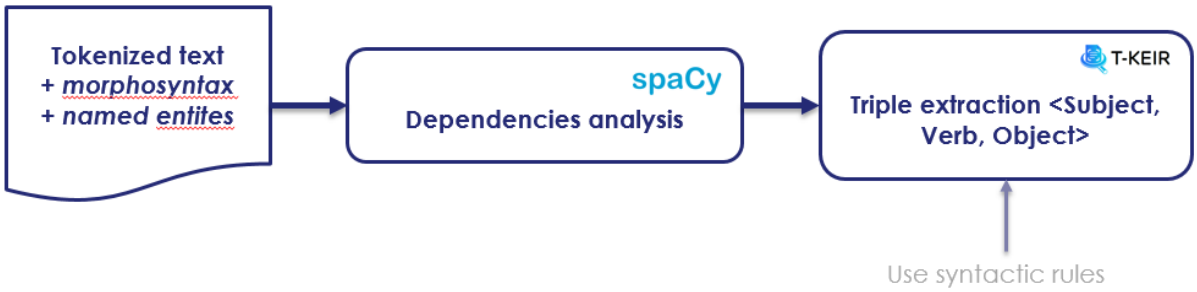
The tagger implemented in the use case uses the Spacy library and uses the elements extracted during the segmentation phase as well as validation rules built with the morphosyntactic elements.

Use validation rules

Validation rules help to avoid basic errors such as associating a city name with a verb.

```
{
    "ner-pos-validation":[
        {"label":"person","possible-pos-in-syntagm":["PROPN","DET","ADJ","NOUN"],"at-least":["PROPN"]},
        {"label":"organization","possible-pos-in-syntagm":["PROPN","DET","ADJ","NOUN"],"at-least":["PROPN"]},
        {"label":"location","possible-pos-in-syntagm":["PROPN","DET","ADJ","NOUN"],"at-least":["PROPN"]},
        {"label":"location.city","possible-pos-in-syntagm":["PROPN","DET","ADJ","NOUN"],"at-least":["PROPN"]},
        {"label":"location.country","possible-pos-in-syntagm":["PROPN","DET","ADJ","NOUN"],"at-least":["PROPN"]},
        {"label":"product","possible-pos-in-syntagm":["PROPN","DET","ADJ","NOUN"],"at-least":["PROPN"]},
        {"label":"facility","possible-pos-in-syntagm":["PROPN","DET","ADJ","NOUN"],"at-least":["NOUN","PROPN"]},
        {"label":"event","possible-pos-in-syntagm":["PROPN","DET","ADJ","NOUN"],"at-least":["NOUN","PROPN"]}]
}
```

Dependencies analysis & triple <Subject, Verb, Object> extraction



Dependency analysis allows the discovery of relationships between the different structuring elements of a sentence. It therefore provides the possibility of creating <Subject, Verb, Object> triples which will form the basis of a knowledge graph automatically constructed by T-KEIR.

Principles

The Dependency Analyzer relies on the Spacy library to extract syntactic dependencies. This analysis is improved by taking advantage of the groupings carried out during the previous phases (segmentation, morphosyntax and extraction of named entities). Thus the structured elements detected by Spacy are extended with the data from the previous phases. Finally, the <Subject, Verb, Object> triples are extracted using syntactic patterns defined in a configuration file.

Syntactic rules

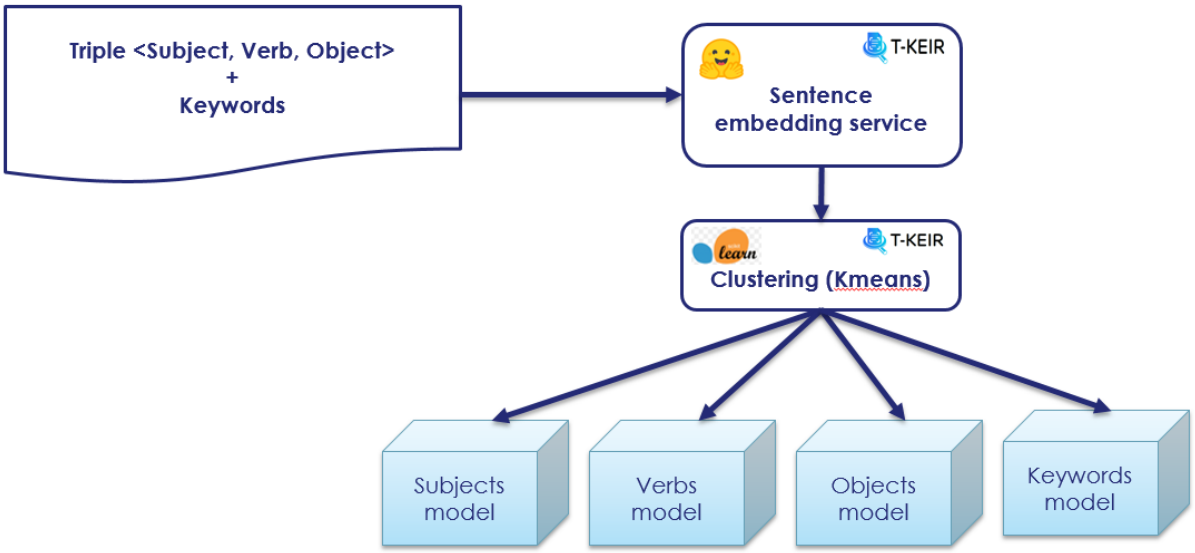
The syntactic rules allow the definition of patterns corresponding to phrases, verbal groups or prepositional groups. The creation of these rules is governed by the syntax defined in the Spacy library.

Keywords extraction

The keywords are the most relevant words or sequences of words in a document. When they are weighted, they allow, for example, the creation of word clouds. Extracting them is a good way to naively summarize a document by pointing to the most relevant elements.

To judge the relevance of the different terms we used the Rake algorithm. It is built on the observation that keywords are found between empty words and punctuation marks. The algorithm extracts and weights these word sequences using a method described in “Rose, Stuart & Engel, Dave & Cramer, Nick & Cowley, Wendy. (2010). Automatic Keyword Extraction from Individual Documents. 10.1002 / 9780470689646.ch1 (Automatic Keyword Extraction from Individual Documents (researchgate.net))”. T-KEIR uses a modified version of Rake taking into account lemmatized forms and their morphosyntaxic tags. Thus empty words will be associated with the labels of determinants and other conjunctions while the delimiters will be associated with the punctuation tags.

Relation clustering



The relations clustering allows to associate a semantic class with the elements of <Subject, Verb, Object> triples (and keywords) in an unsupervised manner (without human intervention).

The construction of the classes is carried out in two stages:

- The elements of the <Subject, Verb, Object> triples and the keywords are vectorized using a transformer neural network (here we use the LaBSE transformer from the Huggingface library) pre-trained on a large amount of data covering a number varied fields. Each of these vectors can be seen as an embedding of sequences of words (associated with a Subject, a Verb, an Object or a keyword) in a semantic space created by the Transformer.
- These four sets of vectors (associated respectively with the Subject, Verb, Object and Keywords) are clustered by a clusterin algorithm. From then on, each vector is assigned a class number and the algorithm creates a model to predict the class of a new vector.

Using a clustering algorithm is a good way to do semantic quantization: we don't store the vectors, only their semantic classes predicted by the clustering model.

1.2.3 Indexing

Index design

The index built by the T-KEIR library is the subject of a particular design responding to the various functionalities which are:

- Standard information search (by key words / phrase).
- Structured research by using a knowledge graph with access to Subject, Verb / Predicate / Property, Object type triples. It is essential to note here, that beyond the standard search, the index has been constructed in such a way that the business concepts extracted from the ontology can be used as search criteria. They are therefore considered as <Subject, Predicate, Object> triples of a knowledge graph.

The index is built of several facets:

- title: document title
- content: content of the document
- lemma_title: lemmatized version of the title where the tool words and punctuation have been removed
- lemma_content: lemmatized version of the content (without tool words)
- data_source / index_document: pointer to document
- text_suggester: list of keywords allowing the completion of a query (in the case of a "keywords" type query).
- kg: knowledge graph, a focus is made thereafter.

The kg field contains the basic structure for building a knowledge graph: the <Subject, Predicate / Property, Object> triples. In the T-KEIR index each element of this triplet includes the following fields:

- class: the class resulting from the cluster of relations or keywords when it is available. It is a cluster identifier linked to the model calculated during the relationship clustering phase
- content / lemma_content: the textual content of the element
- label: the label associated with the content, in the case of a named entity this will for example be "place" or "person"
- positions: the positions of the element when they are available

From documentary analysis to index

The index is fill with results of the linguistic analysis. Each document is analyzed: it follows the diagram **Documentary analysis**. The indexing tool uses the content of the result of this analysis to generate indexing ElasticSearch queries following the scheme defined in the Design section of the index.

1.2.4 Searching

The document searching is the step of querying indexes. The rich structure of the index offers many possibilities for "querying". Developing a query involves analyzing the user query, constructing an Elasticsearch query and manipulating the results

Analysis of the request

Query analysis follows the same process as document analysis. In this case study, the query corresponds to all or part of a document and not simply to a sentence or a juxtaposition of keywords. Using an identical documentary analysis ensures that we have the same documentary enrichment as that carried out for the indexed documents. The construction of the query to Elasticsearch will therefore be easier.

Building Elasticsearch queries

Elasticsearch provides a very advanced query language (DSL: Domain Specific Language). This language makes it possible to carry out multifaceted interrogation by integrating notions of "boosting" of query elements (to give more weight to these elements), advanced combinations of Boolean clauses (OR and AND), notions of "slop" to manage the alignments between two sequences . . .

Standard queries

Standard querying simply uses a bag of words constructed from the content of the document (the tokenization phase). This bag of words is sorted by how often the words appear in the document.

```
"basic-querying":{
    "uniq-word-query":true,
    "boosted-uniq-word-query":false,
    "cut-query":4096
},
```

It is possible to configure the request with 2 criteria: * we use a bag of words sorted by frequency of occurrence without weighting (uniq-word-query)
- Either we "boost" each word according to their frequency.

In both cases, a maximum number of words must be defined in the query (cut-query option).

"Advanced" queries allow you to create complex queries based on all fields of the index.

```
"advanced-querying":{
    "use-lemma":false,
    "use-keywords":false,
    "use-knowledge-graph":false,
    "use-semantic-keywords":false,
    "use-semantic-knowledge-graph":true,
    "use-concepts":true,
    "use-sentences":false,
    "querying":{
        "match-phrase-slop":1,
        "match-phrase-boosting":0.5,
        "match-sentence":{
            "number-and-symbol-filtering":true,
            "max-number-of-words":30
        },
        "match-keyword":{
            "number-and-symbol-filtering":true,
            "semantic-skip-highest-ranked-classes":3,
            "semantic-max-boosting":5
        },
        "match-svo":{
            "semantic-use-class-triple":false,
            "semantic-use-lemma-property-object":false,
            "semantic-use-subject-lemma-object":false,
            "semantic-use-subject-property-lemma":false,
            "semantic-use-lemma-lemma-object":true,
            "semantic-use-lemma-property-lemma":true,
            "semantic-use-subject-lemma-lemma":true,
            "semanic-max-boosting":5
        },
        "match-concept":{
            "concept-boosting":0.2,
            "concept-pruning":10
        }
    }
},
```

The created query can take into account * Fields prefixed by "**lemma_**" (use-lemma option) * Keywords by comparing the keywords from the query with

those extracted during the indexing phase (use-keyword options)

- The knowledge graph by building disjunctive queries or the sub-queries will be conjunctions of the elements of the triples <Subject, Property / Verb / Predicat, Object / Value> (use-knowledge-graph option)
- The semantics of the keywords with the classes of the clusters: in the same way as the relations, the keywords are the object of a clustering and have at this tritre of classes which one can interrogate (option use-semantic-keyword)
- The semantics of <Subject, Verb, Object> triples: each element of the triplet has semantic classes resulting from clustering (use-semantic-knowledge-graph option)
- The concepts of ontology (those extracted by Linguamatics)
- Sentences in documents.

Then the types of queries defined above can be refined by configuring them (querying field). Thus it is possible to define the slop (maximum distance between the words of each sequence), the boosting value, etc.

The use of <Subject, Verb, Subject> triples and sentences leads to the construction of potentially very large queries. They are all the greater in that to increase the relevance of the results we combine three forms of query: OR, AND and ExactMatch.

To limit the size of the requests, we apply a clustering algorithm (HDBSCAN) to a TF.IDF type vectorization of the sequences (document sentences, elements of triples). The clusters thus created, we only use the most relevant sequences (those close to the centers of the clusters).

Query Expansion

In the study case, a query expansion option by document was implemented. Here we are looking to extend the query using other patents. The strategy implemented is to use the term vectors provided by ElasticsSearch. These vectors contain for each document the list of terms of the document. Each term contains statistics related to the index: how often the term appears in the document, throughout the index, and the number of documents in which the term appears. The idea is to combine all the vectors so as to build a query taking into account all the documents: query and document to be extended with their associated statistics.

Combine & Re-Ranking of results

```
"scoring":{
    "normalize-score":true,
    "document-query-intersection-penalty":"by-query-size",
    "run-clause-separately":true,
    "expand-results":50
},
```

Combination strategies

When creating queries we saw that it was possible to take into account several options: use-keyword, use-concepts, use-knowledge-graph, use-semantic-keywords, use-semanic-knowledgre graph; these different options can be combined within the same query in which case ElasticSearch will sum the values of the similarities of each option or else be executed in several sub-queries and in this case the merge is done a posteriori. In this last strategy the lists of results of each subquery are combined by sum of the scores. To ensure that there are as many common documents as possible between the search results of the different sub-queries, it is possible to extend the number of documents returned by ElasticSearch (expand-results option). The configuration of the choice between merging within a single query or after the execution of several queries is given by the run-clause-separately field.

Scoring of results

The result scoring is generally presented to the user, unfortunately the score provided by ElasticSearch is not bounded and is difficult to interpret as it is. So we have implemented several re-scoring strategies * By score normalization: the score is normalized by the score of the first ordered document

(the highest score returned by ElasticSearch). The interpretation of the score is simple, corresponding to the degree of relevance to the most relevant document. But This implies that the most relevant document will always have a score of 1, even if the actual relevance is low.

- Using metrics on the intersection between the terms of the term vectors of the request and those of the document: * **by-query-size**: we calculate the size of the intersection between a document and the query which we divide

  by the size of the query. The score should reflect the presence of all the words of the query in the document. This is the default configuration.

  - **by-document-size**: we calculate the size of the intersection between a document and the query which we divide by the size of the document. The score should reflect a maximum match between the query and the document.
  - **by-union-size: we calculate the size of the intersection between a document and the query which we divide** by the size of the union between the query and the document (Jaccard similarity). The score must reflect a maximum correspondence between the document and the query and penalize the elements of the query (or document) that do not intersect.
  - **no-normalization**: no normalization. The score will be the one provided by Elasticsearch.

The two types of normalization are multiplied to obtain the final score

1.3 Installation

These tools work on *nix and docker environment.

1.3.1 Directory structure

- **app/bin** : scripts and tools for server execution
- **doc** : buildable documentation
- **runtimes/docker** : docker environment
- **runtimes/k8s** : minikube environment
- **resources** : contain testing resources & automatic index creation data
- **thot** : searchai source code

The installation is hightly dockerized

1.3.2 Installation Prerequists

The only prerequists is docker, docker compose and sphinx doc to build the documentation. To build the documentation goes in doc directory and type

```
#> builddoc.sh
```

1.3.3 Use docker image

Build searchai docker image

You should build the docker base image. This image contains os and python dependencies and code of search ai with 6 entry points:

- init : initialize directories and models
- searcher : run search engine
- indexer : run index engine
- web : run web demonstrator
- shell : run sleep loop. It allows interactive use of tools (useful for develop, debugging and testing)
- tests : run services tests

Go in docker directory and run the following command:

- for cuda build:

```
#> ./builddocker.sh
```

- for no-cuda build:

```
#> ./builddocker-nocuda.sh
```

Configure docker compose and directory paths

You need to configure some path in **".env"** file in directory docker

- OPENDISTRO_VERSION : version of opendistrop (1.12.0)
- OPENDISTRO_HOST : opendistro host (0.0.0.0)
- OPENDISTRO_PORT : opendistro port (9200)
- SEARCHAI_PATH : searchai soruce path (/data/searchai/dev/)
- SEARCHAI_DATA_PATH : searchai data path (/data/searchai/)
- SEARCHAI_MODEL_PATH : search ai model model. Containing hugging face models (/data/searchai/resources/modeling). Take care to make **transformers** sub directory.
- CONVERTER_PORT : converter service port
- TOKENIZER_PORT : tokenizer service port
- MSTAGGER_PORT : morphosyntactic service port
- NERTAGGER_PORT : named entities service port
- SYNTAXTAGGER_PORT : syntax and relation service port
- SENT_EMBEDDING_PORT : sentence embedding service port
- TAGGER_PIPELINE_PORT
- KEYWORD_PORT=10007
- AUTOMATIC_SUMMARY_PORT=10008
- SENTIMENT_ANALYSIS_PORT=10009
- CLASSIFICATION_PORT=10010
- QA_PORT=10011
- GRAPH_SEARCH_PORT=8001
- SEARCH_PORT=8000
- WEB_PORT=8080
- CONVERTER_HOST=0.0.0.0
- TOKENIZER_HOST=0.0.0.0
- MSTAGGER_HOST=0.0.0.0
- NERTAGGER_HOST=0.0.0.0
- SYNTAXTAGGER_HOST=0.0.0.0
- SENT_EMBEDDING_HOST=0.0.0.0
- TAGGER_PIPELINE_HOST=0.0.0.0
- KEYWORD_HOST=0.0.0.0
- AUTOMATIC_SUMMARY_HOST=0.0.0.0
- SENTIMENT_ANALYSIS_HOST=0.0.0.0
- CLASSIFICATION_HOST=0.0.0.0
- QA_HOST=0.0.0.0
- GRAPH_SEARCH_HOST=0.0.0.0
- SEARCH_HOST=0.0.0.0
- WEB_HOST=0.0.0.0

- SERVICE_CONFIG=configs/tests/services.json

Ideally just copy .env (the one pull from git) to a file **env.local** before editing the environment variables.

1.3.4 Copy or create data

Searchai comes with default configuration file. Nevertheless you can modify or add file. Most of them are configuration (see configuration section). The data are stores in the directories defined into the file "**.env**" . The folder associated with **RESOURCES_DIRECTORY** contains resources data (index mapping and resources files) and models

Index mappings

Index mapping is store in **RESOURCES_DIRECTORY/indices/indices_mapping**. if you create new mapping it MUST contains the same fields. You can freely change the analyzers.

Resources

The resources are stored in **RESOURCES_DIRECTORY/modeling/tokenizer/[en|fr...]**. This directory contains file with list or csv tables. The descriptions of these file are in **CONFIGS/annotation-resources.json**

1.3.5 Initialize/Load the models

When resources, configuration file and indices are well configurated. In directory **docker**, you can run the command:

```
#> docker-compose -f docker-compose-init.yml up
```

Take care of proxies. Please set correclty $HOME/.docker/config.json like that:

```
{
  "proxies":
  {
    "default":
    {
      "httpProxy": "your_http_proxy",
      "httpsProxy": "your_https_proxy",
      "noProxy": "your_no_proxy,searchai_opendistro"
    }
  }
}
```

Don't forget to add **searchai_opendistro** in no_proxy

1.4 Tools

1.4.1 Content

Tools Overview

Tkeir tools

TKeir tools is mostly a set of REST service, except Converter each service use generally as 'Searchai' document which store the extracted information (i.e. tokens, morphosyntax, named entities, syntax and SPO triples, semantic classes ...)

How run the services

The services can be run a a same way:

```
python3 thot/<service_name>_svc.py -c service_config.json
```

How use the services

There is two way to consume the service:

1. by developing your own access to the REST api
2. by using the python client

```
python3 thot/<service_name>_client.py -c service_config.json [client␣
↪specific option]
```

Quick start / Example

Here we use all pre-configured service: the configuration environement is in directory **configs/defaults***. The dataset we use is the enronmail dataset provided freely on [https://www.cs.cmu.edu/~enron/]

Start with document conversion

This task aims to convert raw document (here e-mails) to document compliant with the T-KEIR tools.

- First run converter REST service:

```
python3 thot/converter_svc.py --config=/home/searchai_svc/searchai/configs/default/
↪configs/converter.json
```

- On another console run the client to build document: the folder containing. The folder containing the document is **/home/searchai_svc/searchai/thot/tests/data/test-raw/mail** the target directory is **/home/searchai_svc/searchai/thot/tests/data/test-inputs/**

```
python3 thot/converter_client.py -c /home/searchai_svc/searchai/configs/default/
↪configs/converter.json -t email -i /home/searchai_svc/searchai/thot/tests/data/
↪test-raw/mail -o /home/searchai_svc/searchai/thot/tests/data/test-inputs/
```

Document Tokenization

This task allows to cut text into small units (generally words). The first step to perform before tokenization to create resource model containing for example a list of city or a list of specific term (like financial terminology).

To create these resources simply run

```
python3 thot/tasks/tokenizer/createAnnotationResource.py --entries=/home/searchai_
↪svc/searchai/configs/default/configs/annotation-resources.json --output=/home/
↪searchai_svc/searchai/configs/default/resources/modeling/tokenizer/en/searchai_
↪mwe.pkl
```

To run the command type simply from searchai directory:

```
python3 thot/tokenizer_svc.py --config=/home/searchai_svc/searchai/configs/default/
↪configs/tokenizer.json
```

Run client

```
python3 thot/tokenizer_client.py --config=/home/searchai_svc/searchai/configs/
↪default/configs/converter.json --input=/home/searchai_svc/searchai/thot/tests/
↪data/test-inputs/ --output=/home/searchai_svc/searchai/thot/tests/data/test-
↪tokenizer/
```

Converter

Converter is a tool allowing to convert different kind of document format to one compliant with **tkeir** tools. This tools is a rest service.

Converter API

OPTIONS /api/converter/run

run the converter

Run the converter according to input. It select the good converter and return searchai representation

Request JSON Object

- **data** (*string*) – base64 encoded document
- **datatype** (*string*) – type of data return (available list could be returned by list-type
- **source** (*string*) – source of the document on host file system

Status Codes

- 200 OK – return the converted document
- 422 Unprocessable Entity – Something wrong on the request parameters
- 500 Internal Server Error – Internal Error, and exception occured

Response JSON Object

- **config** (*string*) – service configuration file
- **date** (*string*) – release date of the service
- **info** (*string*) – service info id
- **result.content** (*string*) – nested array of text representing the content of the document
- **result.content_morphosyntax[].lemma** (*string*) – lemma of token - the form
- **result.content_morphosyntax[].pos** (*string*) – Part Of Speech
- **result.content_ner[].end** (*integer*) – end offset of the token of the named entity
- **result.content_ner[].label** (*string*) – label of the named entity
- **result.content_ner[].start** (*integer*) – start offset of the token of the named entity
- **result.content_ner[].text** (*string*) – text of the named entity
- **result.content_tokens[]** (*string*) – a token, possibly MWE
- **result.data_source** (*string*) – source of the document (could be host file system, web, . . .
- **result.embeddings[].content** (*string*) – text of the sentence
- **result.embeddings[].embeding[]** (*integer*) – embedding feature

- **result.embeddings[].field** (*string*) – Field from which the embedding is computed (can be empty)
- **result.embeddings[].position.length** (*integer*) – length of the sentence
- **result.embeddings[].position.start** (*integer*) – start offset of the sentence
- **result.kg[].property.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].property.content** (*string*) – Content of triple item
- **result.kg[].property.label** (*string*) – label of the content
- **result.kg[].property.lemma_content** (*string*) – Lemmatized content
- **result.kg[].property.pos[]** (*string*) – POS tag
- **result.kg[].property.positions[]** (*integer*) – Position in document
- **result.kg[].subject.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].subject.content** (*string*) – Content of triple item
- **result.kg[].subject.label** (*string*) – label of the content
- **result.kg[].subject.lemma_content** (*string*) – Lemmatized content
- **result.kg[].subject.pos[]** (*string*) – POS tag
- **result.kg[].subject.positions[]** (*integer*) – Position in document
- **result.kg[].value.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].value.content** (*string*) – Content of triple item
- **result.kg[].value.label** (*string*) – label of the content
- **result.kg[].value.lemma_content** (*string*) – Lemmatized content
- **result.kg[].value.pos[]** (*string*) – POS tag
- **result.kg[].value.positions[]** (*integer*) – Position in document
- **result.source_doc_id** (*string*) – uniq id of the document
- **result.title** (*string*) – title of document
- **result.title_morphosyntax[].lemma** (*string*) – lemma of token - the form
- **result.title_morphosyntax[].pos** (*string*) – Part Of Speech
- **result.title_ner[].end** (*integer*) – end offset of the token of the named entity
- **result.title_ner[].label** (*string*) – label of the named entity
- **result.title_ner[].start** (*integer*) – start offset of the token of the named entity
- **result.title_ner[].text** (*string*) – text of the named entity
- **result.title_tokens[]** (*string*) – a token, possibly MWE
- **version** (*string*) – version of the service

POST /api/converter/run
run the converter

Run the converter according to input. It select the good converter and return searchai representation

Request JSON Object

- **data** (*string*) – base64 encoded document
- **datatype** (*string*) – type of data return (available list could be returned by list-type
- **source** (*string*) – source of the document on host file system

Status Codes

- 200 OK – return the converted document
- 422 Unprocessable Entity – Something wrong on the request parameters
- 500 Internal Server Error – Internal Error, and exception occured

Response JSON Object

- **config** (*string*) – service configuration file
- **date** (*string*) – release date of the service
- **info** (*string*) – service info id
- **result.content** (*string*) – nested array of text representing the content of the document
- **result.content_morphosyntax[].lemma** (*string*) – lemma of token - the form
- **result.content_morphosyntax[].pos** (*string*) – Part Of Speech
- **result.content_ner[].end** (*integer*) – end offset of the token of the named entity
- **result.content_ner[].label** (*string*) – label of the named entity
- **result.content_ner[].start** (*integer*) – start offset of the token of the named entity
- **result.content_ner[].text** (*string*) – text of the named entity
- **result.content_tokens[]** (*string*) – a token, possibly MWE
- **result.data_source** (*string*) – source of the document (could be host file system, web, ...
- **result.embeddings[].content** (*string*) – text of the sentence
- **result.embeddings[].embeding[]** (*integer*) – embedding feature
- **result.embeddings[].field** (*string*) – Field from which the embedding is computed (can be empty)
- **result.embeddings[].position.length** (*integer*) – length of the sentence
- **result.embeddings[].position.start** (*integer*) – start offset of the sentence
- **result.kg[].property.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].property.content** (*string*) – Content of triple item
- **result.kg[].property.label** (*string*) – label of the content
- **result.kg[].property.lemma_content** (*string*) – Lemmatized content

- **result.kg[].property.pos[]** (*string*) – POS tag
- **result.kg[].property.positions[]** (*integer*) – Position in document
- **result.kg[].subject.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].subject.content** (*string*) – Content of triple item
- **result.kg[].subject.label** (*string*) – label of the content
- **result.kg[].subject.lemma_content** (*string*) – Lemmatized content
- **result.kg[].subject.pos[]** (*string*) – POS tag
- **result.kg[].subject.positions[]** (*integer*) – Position in document
- **result.kg[].value.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].value.content** (*string*) – Content of triple item
- **result.kg[].value.label** (*string*) – label of the content
- **result.kg[].value.lemma_content** (*string*) – Lemmatized content
- **result.kg[].value.pos[]** (*string*) – POS tag
- **result.kg[].value.positions[]** (*integer*) – Position in document
- **result.source_doc_id** (*string*) – uniq id of the document
- **result.title** (*string*) – title of document
- **result.title_morphosyntax[].lemma** (*string*) – lemma of token - the form
- **result.title_morphosyntax[].pos** (*string*) – Part Of Speech
- **result.title_ner[].end** (*integer*) – end offset of the token of the named entity
- **result.title_ner[].label** (*string*) – label of the named entity
- **result.title_ner[].start** (*integer*) – start offset of the token of the named entity
- **result.title_ner[].text** (*string*) – text of the named entity
- **result.title_tokens[]** (*string*) – a token, possibly MWE
- **version** (*string*) – version of the service

OPTIONS /api/converter/list-types

Get list of managed type

Retrieve th elist of managed type, that must be a parameter of run function

Status Codes

- 200 OK – return the list of available converter formats
- 500 Internal Server Error – Internal Error, and exception occured

Response JSON Object

- **config** (*string*) – configuration file path
- **date** (*string*) – service development date
- **info** (*string*) – service uniq id
- **results** (*string*) – list of possible input document format
- **version** (*string*) – version of the service

GET /api/converter/list-types
Get list of managed type

Retrieve th elist of managed type, that must be a parameter of run function

Status Codes

- 200 OK – return the list of available converter formats
- 500 Internal Server Error – Internal Error, and exception occured

Response JSON Object

- **config** (*string*) – configuration file path
- **date** (*string*) – service development date
- **info** (*string*) – service uniq id
- **results** (*string*) – list of possible input document format
- **version** (*string*) – version of the service

OPTIONS /api/converter/health
Ask the health of the service

This function allows to know if the service is up and healthy. It is use by client to see if they can use it.

Status Codes

- 200 OK – Service successfuly loaded
- 500 Internal Server Error – Service is not loaded

Response JSON Object

- **config** (*string*) – configuration file path
- **date** (*string*) – service development date
- **health** (*string*) – ok if service is healthy
- **info** (*string*) – service uniq id
- **version** (*string*) – version of the service

POST /api/converter/health
Ask the health of the service

This function allows to know if the service is up and healthy. It is use by client to see if they can use it.

Status Codes

- 200 OK – Service successfuly loaded
- 500 Internal Server Error – Service is not loaded

Response JSON Object

- **config** (*string*) – configuration file path
- **date** (*string*) – service development date
- **health** (*string*) – ok if service is healthy
- **info** (*string*) – service uniq id
- **version** (*string*) – version of the service

GET /api/converter/health
Ask the health of the service

This function allows to know if the service is up and healthy. It is use by client to see if they can use it.

Status Codes

- 200 OK – Service successfuly loaded
- 500 Internal Server Error – Service is not loaded

Response JSON Object

- **config** (*string*) – configuration file path
- **date** (*string*) – service development date
- **health** (*string*) – ok if service is healthy
- **info** (*string*) – service uniq id
- **version** (*string*) – version of the service

This API is also available via the service itself on http://<service host>:<service port>/swagger

Converter configuration

Example of Configuration:

```
{
    "logger": {
        "logging-file": "test.log",
        "logging-path": "/tmp",
        "logging-level": {"file": "info", "screen": "debug"}
    },
    "converter": {
        "network": {
            "host":"localhost",
            "port":10000,
            "associate-environment": {
                "host":"CONVERTER_HOST",
                "port":"CONVERTER_PORT"
            }
        },
        "runtime":{
            "request-max-size":100000000,
            "request-buffer-queue-size":100,
            "keep-alive":true,
            "keep-alive-timeout":5,
            "graceful-shutown-timeout":15.0,
            "request-timeout":60,
            "response-timeout":60,
            "workers":1
        },
        "serialize":{
            "input":{
                "path":"/tmp",
                "keep-service-info":true
            },
            "output":{
                "path":"/tmp",
                "keep-service-info":true
            }
        }

    }
}
```

Converter is an aggreation of network configuration, serialize configuration, runtime configuration (in field converter), logger (at top level).

Configure converter logger

Logger is configuration at top level of json in *logger* field.

Example of Configuration:

```
{
    "logger": {
        "logging-file": "test.log",
        "logging-path": "/tmp",
        "logging-level": {"file": "info", "screen": "debug"}
    }
}
```

The logger fields are:

- **logging-file** is the filename of the log file (notice that "-<logname>" will be added to this name=
- **logging-path** is the path to the logfile (if it does not exist it will be created)
- **logging-level** contains two fields:
  - **file** for the logging level of the file
  - **screen** for the logging level on screen output

  Both can be set to the following values:
  - **debug** for the debug level and developper information
  - **info** for the level of information
  - **warning** to display only warning and errors
  - **error** to display only error

Configure converter Network

Example of Configuration:

```
{
    "network": {
        "host":"0.0.0.0",
        "port":8080,
        "associate-environment": {
            "host":"HOST_ENVNAME",
            "port":"PORT_ENVNAME"
        }
    }
}
```

The network fields:

- **host** : hostname
- **port** : port of the service
- **associated-environement** : is the "host" and "port" associated environment variables that allows to replace the default one. This field is not mandatory.
  - "host" : associated "host" environment variable
  - "port" : associated "port" environment variable

Configure converter Serialize

Example of Configuration:

```
{
    "serialize":{
        "input":{
            "path":"/tmp",
            "keep-service-info":true
        },
        "output":{
            "path":"/tmp",
            "keep-service-info":true
        }
    }
}
```

The serialize fields:

- **input** : serialize input of service
  - **path** : path of serialized fields
  - **keep-service-info** : True if serialize info is kept
  - **associated-environement** : is the "path" and "keep-service-info" associated environment variables that allows to replace the default one. This field is not mandatory.
    - **path** : associated "path" environment variable
    - **keep-service-info** : associated "keep-service-info" environment variable
- output : serialize output of service
  - **path** : path of serialized fields
  - **keep-service-info** : True if serialize info is kept
  - **associated-environement** : if the "path" and "keep-service-info" associated environment variables that allows to replace the default one. This field is not mandatory.
    - **path** : associated "path" environment variable
    - **keep-service-info** : associated "keep-service-info" environment variable

Configure converter runtime

Example of Configuration:

```
{
    "runtime":{
        "request-max-size":100000000,
        "request-buffer-queue-size":100,
        "keep-alive":true,
        "keep-alive-timeout":5,
        "graceful-shutown-timeout":15.0,
        "request-timeout":60,
        "response-timeout":60,
        "workers":1
    }
}
```

The Runtime fields:

- **request-max-size** : how big a request may be (bytes)

- **request-buffer-queue-size**: request streaming buffer queue size
- **request-timeout** : how long a request can take to arrive (sec)
- **response-timeout** : how long a response can take to process (sec)
- **keep-alive**: keep-alive
- **keep-alive-timeout**: how long to hold a TCP connection open (sec)
- **graceful-shutdown_timeout** : how long to wait to force close non-idle connection (sec)
- **workers** : number of workers for the service on a node
- **associated-environement** : if one of previous field is on the associated environment variables that allows to replace the default one. This field is not mandatory.
  - **request-max-size** : overwrite with environement variable
  - **request-buffer-queue-size**: overwrite with environement variable
  - **request-timeout** : overwrite with environement variable
  - **response-timeout** : overwrite with environement variable
  - **keep-alive**: overwrite with environement variable
  - **keep-alive-timeout**: overwrite with environement variable
  - **graceful-shutdown_timeout** : overwrite with environement variable
  - **workers** : overwrite with environement variable

Converter service

The available input format are:

- **raw** : a raw text format
- **email** : a mail format

To run the command type simply from tkeir directory:

```
python3 thot/converter_svc.py --config=/home/searchai_svc/searchai/configs/default/
↪configs/converter.json
```

A light client can be run through the command

```
python3 thot/converter_client.py -c /home/searchai_svc/searchai/configs/default/
↪configs/converter.json -t email -i /home/searchai_svc/searchai/thot/tests/data/
↪test-raw/mail -o /home/searchai_svc/searchai/thot/tests/data/test-inputs/
```

Converter Tests

The converter service come with unit and functional testing.

Converter Unit tests

Unittest allows to test Converters classes only.

```
python3 -m unittest thot/tests/unittests/TestConverterConfiguration.py
python3 -m unittest thot/tests/unittests/TestConverter.py
python3 -m unittest thot/tests/unittests/TestEmailConverter.py
python3 -m unittest thot/tests/unittests/TestRawConverter.py
```

Converter Functional tests

```
python3 -m unittest thot/tests/functional_tests/TestConverterSvc.py
```

Tokenizer

The tokenizer is a tool allowing to tokenize "title" and "content" field of searchai document. This tools is a rest service. Tokenization depends on annotation model created by the tool stored in **searchai/thot/tasks/tokenizer/createAnnotationResouces.py** This tools allows to create typed compound word list.

Tokenizer API

OPTIONS /api/tokenizer/run

run the tokenizer

Run the tokenizer according to input. It return searchai representation

Request JSON Object

- **config** (*string*) – service configuration file
- **date** (*string*) – release date of the service
- **info** (*string*) – service info id
- **result.content** (*string*) – nested array of text representing the content of the document
- **result.content_morphosyntax[].lemma** (*string*) – lemma of token - the form
- **result.content_morphosyntax[].pos** (*string*) – Part Of Speech
- **result.content_ner[].end** (*integer*) – end offset of the token of the named entity
- **result.content_ner[].label** (*string*) – label of the named entity
- **result.content_ner[].start** (*integer*) – start offset of the token of the named entity
- **result.content_ner[].text** (*string*) – text of the named entity
- **result.content_tokens[]** (*string*) – a token, possibly MWE
- **result.data_source** (*string*) – source of the document (could be host file system, web, ...
- **result.embeddings[].content** (*string*) – text of the sentence
- **result.embeddings[].embeding[]** (*integer*) – embedding feature

- **result.embeddings[].field** (*string*) – Field from which the embedding is computed (can be empty)
- **result.embeddings[].position.length** (*integer*) – length of the sentence
- **result.embeddings[].position.start** (*integer*) – start offset of the sentence
- **result.kg[].property.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].property.content** (*string*) – Content of triple item
- **result.kg[].property.label** (*string*) – label of the content
- **result.kg[].property.lemma_content** (*string*) – Lemmatized content
- **result.kg[].property.pos[]** (*string*) – POS tag
- **result.kg[].property.positions[]** (*integer*) – Position in document
- **result.kg[].subject.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].subject.content** (*string*) – Content of triple item
- **result.kg[].subject.label** (*string*) – label of the content
- **result.kg[].subject.lemma_content** (*string*) – Lemmatized content
- **result.kg[].subject.pos[]** (*string*) – POS tag
- **result.kg[].subject.positions[]** (*integer*) – Position in document
- **result.kg[].value.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].value.content** (*string*) – Content of triple item
- **result.kg[].value.label** (*string*) – label of the content
- **result.kg[].value.lemma_content** (*string*) – Lemmatized content
- **result.kg[].value.pos[]** (*string*) – POS tag
- **result.kg[].value.positions[]** (*integer*) – Position in document
- **result.source_doc_id** (*string*) – uniq id of the document
- **result.title** (*string*) – title of document
- **result.title_morphosyntax[].lemma** (*string*) – lemma of token - the form
- **result.title_morphosyntax[].pos** (*string*) – Part Of Speech
- **result.title_ner[].end** (*integer*) – end offset of the token of the named entity
- **result.title_ner[].label** (*string*) – label of the named entity
- **result.title_ner[].start** (*integer*) – start offset of the token of the named entity
- **result.title_ner[].text** (*string*) – text of the named entity
- **result.title_tokens[]** (*string*) – a token, possibly MWE
- **version** (*string*) – version of the service

Status Codes

- 200 OK – return the searchai doc with tokens

- 422 Unprocessable Entity – Something wrong on the request parameters
- 500 Internal Server Error – Internal Error, and exception occured

Response JSON Object

- **config** (*string*) – service configuration file
- **date** (*string*) – release date of the service
- **info** (*string*) – service info id
- **result.content** (*string*) – nested array of text representing the content of the document
- **result.content_morphosyntax[].lemma** (*string*) – lemma of token - the form
- **result.content_morphosyntax[].pos** (*string*) – Part Of Speech
- **result.content_ner[].end** (*integer*) – end offset of the token of the named entity
- **result.content_ner[].label** (*string*) – label of the named entity
- **result.content_ner[].start** (*integer*) – start offset of the token of the named entity
- **result.content_ner[].text** (*string*) – text of the named entity
- **result.content_tokens[]** (*string*) – a token, possibly MWE
- **result.data_source** (*string*) – source of the document (could be host file system, web, ...
- **result.embeddings[].content** (*string*) – text of the sentence
- **result.embeddings[].embeding[]** (*integer*) – embedding feature
- **result.embeddings[].field** (*string*) – Field from which the embedding is computed (can be empty)
- **result.embeddings[].position.length** (*integer*) – length of the sentence
- **result.embeddings[].position.start** (*integer*) – start offset of the sentence
- **result.kg[].property.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].property.content** (*string*) – Content of triple item
- **result.kg[].property.label** (*string*) – label of the content
- **result.kg[].property.lemma_content** (*string*) – Lemmatized content
- **result.kg[].property.pos[]** (*string*) – POS tag
- **result.kg[].property.positions[]** (*integer*) – Position in document
- **result.kg[].subject.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].subject.content** (*string*) – Content of triple item
- **result.kg[].subject.label** (*string*) – label of the content
- **result.kg[].subject.lemma_content** (*string*) – Lemmatized content
- **result.kg[].subject.pos[]** (*string*) – POS tag
- **result.kg[].subject.positions[]** (*integer*) – Position in document

- **result.kg[].value.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].value.content** (*string*) – Content of triple item
- **result.kg[].value.label** (*string*) – label of the content
- **result.kg[].value.lemma_content** (*string*) – Lemmatized content
- **result.kg[].value.pos[]** (*string*) – POS tag
- **result.kg[].value.positions[]** (*integer*) – Position in document
- **result.source_doc_id** (*string*) – uniq id of the document
- **result.title** (*string*) – title of document
- **result.title_morphosyntax[].lemma** (*string*) – lemma of token - the form
- **result.title_morphosyntax[].pos** (*string*) – Part Of Speech
- **result.title_ner[].end** (*integer*) – end offset of the token of the named entity
- **result.title_ner[].label** (*string*) – label of the named entity
- **result.title_ner[].start** (*integer*) – start offset of the token of the named entity
- **result.title_ner[].text** (*string*) – text of the named entity
- **result.title_tokens[]** (*string*) – a token, possibly MWE
- **version** (*string*) – version of the service

POST /api/tokenizer/run
run the tokenizer

Run the tokenizer according to input. It return searchai representation

Request JSON Object

- **config** (*string*) – service configuration file
- **date** (*string*) – release date of the service
- **info** (*string*) – service info id
- **result.content** (*string*) – nested array of text representing the content of the document
- **result.content_morphosyntax[].lemma** (*string*) – lemma of token - the form
- **result.content_morphosyntax[].pos** (*string*) – Part Of Speech
- **result.content_ner[].end** (*integer*) – end offset of the token of the named entity
- **result.content_ner[].label** (*string*) – label of the named entity
- **result.content_ner[].start** (*integer*) – start offset of the token of the named entity
- **result.content_ner[].text** (*string*) – text of the named entity
- **result.content_tokens[]** (*string*) – a token, possibly MWE
- **result.data_source** (*string*) – source of the document (could be host file system, web, ...
- **result.embeddings[].content** (*string*) – text of the sentence

- **result.embeddings[].embeding[]** (*integer*) – embedding feature
- **result.embeddings[].field** (*string*) – Field from which the embedding is computed (can be empty)
- **result.embeddings[].position.length** (*integer*) – length of the sentence
- **result.embeddings[].position.start** (*integer*) – start offset of the sentence
- **result.kg[].property.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].property.content** (*string*) – Content of triple item
- **result.kg[].property.label** (*string*) – label of the content
- **result.kg[].property.lemma_content** (*string*) – Lemmatized content
- **result.kg[].property.pos[]** (*string*) – POS tag
- **result.kg[].property.positions[]** (*integer*) – Position in document
- **result.kg[].subject.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].subject.content** (*string*) – Content of triple item
- **result.kg[].subject.label** (*string*) – label of the content
- **result.kg[].subject.lemma_content** (*string*) – Lemmatized content
- **result.kg[].subject.pos[]** (*string*) – POS tag
- **result.kg[].subject.positions[]** (*integer*) – Position in document
- **result.kg[].value.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].value.content** (*string*) – Content of triple item
- **result.kg[].value.label** (*string*) – label of the content
- **result.kg[].value.lemma_content** (*string*) – Lemmatized content
- **result.kg[].value.pos[]** (*string*) – POS tag
- **result.kg[].value.positions[]** (*integer*) – Position in document
- **result.source_doc_id** (*string*) – uniq id of the document
- **result.title** (*string*) – title of document
- **result.title_morphosyntax[].lemma** (*string*) – lemma of token - the form
- **result.title_morphosyntax[].pos** (*string*) – Part Of Speech
- **result.title_ner[].end** (*integer*) – end offset of the token of the named entity
- **result.title_ner[].label** (*string*) – label of the named entity
- **result.title_ner[].start** (*integer*) – start offset of the token of the named entity
- **result.title_ner[].text** (*string*) – text of the named entity
- **result.title_tokens[]** (*string*) – a token, possibly MWE
- **version** (*string*) – version of the service

Status Codes

- 200 OK – return the searchai doc with tokens
- 422 Unprocessable Entity – Something wrong on the request parameters
- 500 Internal Server Error – Internal Error, and exception occured

Response JSON Object

- **config** (*string*) – service configuration file
- **date** (*string*) – release date of the service
- **info** (*string*) – service info id
- **result.content** (*string*) – nested array of text representing the content of the document
- **result.content_morphosyntax[].lemma** (*string*) – lemma of token - the form
- **result.content_morphosyntax[].pos** (*string*) – Part Of Speech
- **result.content_ner[].end** (*integer*) – end offset of the token of the named entity
- **result.content_ner[].label** (*string*) – label of the named entity
- **result.content_ner[].start** (*integer*) – start offset of the token of the named entity
- **result.content_ner[].text** (*string*) – text of the named entity
- **result.content_tokens[]** (*string*) – a token, possibly MWE
- **result.data_source** (*string*) – source of the document (could be host file system, web, ...
- **result.embeddings[].content** (*string*) – text of the sentence
- **result.embeddings[].embeding[]** (*integer*) – embedding feature
- **result.embeddings[].field** (*string*) – Field from which the embedding is computed (can be empty)
- **result.embeddings[].position.length** (*integer*) – length of the sentence
- **result.embeddings[].position.start** (*integer*) – start offset of the sentence
- **result.kg[].property.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].property.content** (*string*) – Content of triple item
- **result.kg[].property.label** (*string*) – label of the content
- **result.kg[].property.lemma_content** (*string*) – Lemmatized content
- **result.kg[].property.pos[]** (*string*) – POS tag
- **result.kg[].property.positions[]** (*integer*) – Position in document
- **result.kg[].subject.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].subject.content** (*string*) – Content of triple item
- **result.kg[].subject.label** (*string*) – label of the content
- **result.kg[].subject.lemma_content** (*string*) – Lemmatized content
- **result.kg[].subject.pos[]** (*string*) – POS tag

- **result.kg[].subject.positions[]** (*integer*) – Position in document
- **result.kg[].value.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].value.content** (*string*) – Content of triple item
- **result.kg[].value.label** (*string*) – label of the content
- **result.kg[].value.lemma_content** (*string*) – Lemmatized content
- **result.kg[].value.pos[]** (*string*) – POS tag
- **result.kg[].value.positions[]** (*integer*) – Position in document
- **result.source_doc_id** (*string*) – uniq id of the document
- **result.title** (*string*) – title of document
- **result.title_morphosyntax[].lemma** (*string*) – lemma of token - the form
- **result.title_morphosyntax[].pos** (*string*) – Part Of Speech
- **result.title_ner[].end** (*integer*) – end offset of the token of the named entity
- **result.title_ner[].label** (*string*) – label of the named entity
- **result.title_ner[].start** (*integer*) – start offset of the token of the named entity
- **result.title_ner[].text** (*string*) – text of the named entity
- **result.title_tokens[]** (*string*) – a token, possibly MWE
- **version** (*string*) – version of the service

OPTIONS /api/tokenizer/health

Ask the health of the service

This function allows to know if the service is up and healthy. It is use by client to see if they can use it.

Status Codes

- 200 OK – Service successfuly loaded
- 500 Internal Server Error – Service is not loaded

Response JSON Object

- **config** (*string*) – configuration file path
- **date** (*string*) – service development date
- **health** (*string*) – ok if service is healthy
- **info** (*string*) – service uniq id
- **version** (*string*) – version of the service

GET /api/tokenizer/health

Ask the health of the service

This function allows to know if the service is up and healthy. It is use by client to see if they can use it.

Status Codes

- 200 OK – Service successfuly loaded
- 500 Internal Server Error – Service is not loaded

Response JSON Object

- **config** (*string*) – configuration file path

- **date** (*string*) – service development date
- **health** (*string*) – ok if service is healthy
- **info** (*string*) – service uniq id
- **version** (*string*) – version of the service

POST /api/tokenizer/health

Ask the health of the service

This function allows to know if the service is up and healthy. It is use by client to see if they can use it.

Status Codes

- 200 OK – Service successfuly loaded
- 500 Internal Server Error – Service is not loaded

Response JSON Object

- **config** (*string*) – configuration file path
- **date** (*string*) – service development date
- **health** (*string*) – ok if service is healthy
- **info** (*string*) – service uniq id
- **version** (*string*) – version of the service

This API is also available via the service itself on http://<service host>:<service port>/swagger

Tokenizer configuration

Example of Configuration:

```
{
    "logger": {
        "logging-file": "tokenizer.log",
        "logging-path": "/tmp",
        "logging-level": {"file": "info", "screen": "debug"}
    },
    "tokenizers": {
        "segmenters":[{
            "language":"en",
            "resources-base-path":"/home/searchai_svc/searchai/configs/default/
↪resources/modeling/tokenizer/en",
            "mwe": "searchai_mwe.pkl",
            "normalization-rules":"tokenizer-rules.json"
        }],
        "network": {
            "host":"0.0.0.0",
            "port":10001,
            "associate-environment": {
                "host":"TOKENIZER_HOST",
                "port":"TOKENIZER_PORT"
            }
        },
        "runtime":{
            "request-max-size":100000000,
            "request-buffer-queue-size":100,
            "keep-alive":true,
            "keep-alive-timeout":500,
            "graceful-shutown-timeout":15.0,
            "request-timeout":600,
```

(continues on next page)

(continued from previous page)

```
            "response-timeout":600,
            "workers":1
        },
        "serialize":{
            "input":{
                "path":"/tmp",
                "keep-service-info":true
            },
            "output":{
                "path":"/tmp",
                "keep-service-info":true
            }
        }

    }
}
```

Tokenizer is an aggreation of network configuration, serialize configuration, runtime configuration (in field converter), logger (at top level). The segmenter configuration is a table containing path to Multiple Word Expression entries (MWE):

- **language** :the language of tokenizer
- **resources-base-path**: the path to the resources (containing file created by tools *createAnnotationResources.py*
- **mwe** : the file containing MWE entries
- *normalization-rules** : the file containing normalization rules

Tokenizer accepts a rule file to select parser (not yet implemented), common typos fixing and word mapping (for example map english words to us words). The normalization rule is a simple json file with the following fields:

- **parsers** (NOT YET IMPLEMENTED) : the available parser (for exemple pyvalem to parse chemestry formulas)
- **normalization/word-mapping**: mapping words
- **normalization/typos** : typos fixing

```
{
    "parsers": {
        "on-document":["texsoup"],
        "on-tokens":[
            {"parsers":"chemparse","max-tokens-merge":50}
        ]
    },

    "normalization": {
        "word-mapping" : [
            {"from":"accessorise", "to":"accessorize"},
            {"from":"accessorised", "to":"accessorized"},
            {"from":"accessorises", "to":"accessorizes"},
            {"from":"accessorising", "to":"accessorizing"},
            {"from":"acclimatisation", "to":"acclimatization"},
            {"from":"acclimatise", "to":"acclimatize"},
            {"from":"acclimatised", "to":"acclimatized"},
            {"from":"acclimatises", "to":"acclimatizes"},
            {"from":"acclimatising", "to":"acclimatizing"},
            {"from":"accoutrements", "to":"accouterments"},
            {"from":"aeon", "to":"eon"},
            {"from":"aeons", "to":"eons"},
            {"from":"aerogramme", "to":"aerogram"},
            {"from":"aerogrammes", "to":"aerograms"},
```

(continues on next page)

(continued from previous page)


```
            {"from":"aeroplane", "to":"airplane"},
            {"from":"aeroplanes", "to":"airplanes"},
            {"from":"aesthete", "to":"esthete"},
            {"from":"aesthetes", "to":"esthetes"},
            {"from":"aesthetic", "to":"esthetic"},
            {"from":"aesthetically", "to":"esthetically"},
            {"from":"aesthetics", "to":"esthetics"},
            {"from":"aetiology", "to":"etiology"},
            {"from":"ageing", "to":"aging"},
            {"from":"aggrandisement", "to":"aggrandizement"},
            {"from":"agonise", "to":"agonize"},
            {"from":"agonised", "to":"agonized"},
            {"from":"agonises", "to":"agonizes"},
            {"from":"agonising", "to":"agonizing"},
            {"from":"agonisingly", "to":"agonizingly"},
            {"from":"almanack", "to":"almanac"},
            {"from":"almanacks", "to":"almanacs"},
            {"from":"aluminium", "to":"aluminum"},
            {"from":"amortisable", "to":"amortizable"},
            {"from":"amortisation", "to":"amortization"},
            {"from":"amortisations", "to":"amortizations"},
            {"from":"amortise", "to":"amortize"},
            {"from":"amortised", "to":"amortized"},
            {"from":"amortises", "to":"amortizes"},
            {"from":"amortising", "to":"amortizing"},
            {"from":"amphitheatre", "to":"amphitheater"},
            {"from":"amphitheatres", "to":"amphitheaters"},
            {"from":"anaemia", "to":"anemia"},
            {"from":"anaemic", "to":"anemic"},
            {"from":"anaesthesia", "to":"anesthesia"},
            {"from":"anaesthetic", "to":"anesthetic"},
            {"from":"anaesthetics", "to":"anesthetics"},
            {"from":"anaesthetise", "to":"anesthetize"},
            {"from":"anaesthetised", "to":"anesthetized"},
            {"from":"anaesthetises", "to":"anesthetizes"},
            {"from":"anaesthetising", "to":"anesthetizing"},
            {"from":"anaesthetist", "to":"anesthetist"},
            {"from":"anaesthetists", "to":"anesthetists"},
            {"from":"anaesthetize", "to":"anesthetize"},
            {"from":"anaesthetized", "to":"anesthetized"},
            {"from":"anaesthetizes", "to":"anesthetizes"},
            {"from":"anaesthetizing", "to":"anesthetizing"},
            {"from":"analogue", "to":"analog"},
            {"from":"analogues", "to":"analogs"},
            {"from":"analyse", "to":"analyze"},
            {"from":"analysed", "to":"analyzed"},
            {"from":"analyses", "to":"analyzes"},
            {"from":"analysing", "to":"analyzing"},
            {"from":"anglicise", "to":"anglicize"},
            {"from":"anglicised", "to":"anglicized"},
            {"from":"anglicises", "to":"anglicizes"},
            {"from":"anglicising", "to":"anglicizing"},
            {"from":"annualised", "to":"annualized"},
            {"from":"antagonise", "to":"antagonize"},
            {"from":"antagonised", "to":"antagonized"},
            {"from":"antagonises", "to":"antagonizes"},
            {"from":"antagonising", "to":"antagonizing"},
            {"from":"apologise", "to":"apologize"},
            {"from":"apologised", "to":"apologized"},
            {"from":"apologises", "to":"apologizes"},
            {"from":"apologising", "to":"apologizing"},
```


(continues on next page)

(continued from previous page)


```
            {"from":"appal", "to":"appall"},
            {"from":"appals", "to":"appalls"},
            {"from":"appetiser", "to":"appetizer"},
            {"from":"appetisers", "to":"appetizers"},
            {"from":"appetising", "to":"appetizing"},
            {"from":"appetisingly", "to":"appetizingly"},
            {"from":"arbour", "to":"arbor"},
            {"from":"arbours", "to":"arbors"},
            {"from":"archaeological", "to":"archeological"},
            {"from":"archaeologically", "to":"archeologically"},
            {"from":"archaeologist", "to":"archeologist"},
            {"from":"archaeologists", "to":"archeologists"},
            {"from":"archaeology", "to":"archeology"},
            {"from":"ardour", "to":"ardor"},
            {"from":"armour", "to":"armor"},
            {"from":"armoured", "to":"armored"},
            {"from":"armourer", "to":"armorer"},
            {"from":"armourers", "to":"armorers"},
            {"from":"armouries", "to":"armories"},
            {"from":"armoury", "to":"armory"},
            {"from":"artefact", "to":"artifact"},
            {"from":"artefacts", "to":"artifacts"},
            {"from":"authorise", "to":"authorize"},
            {"from":"authorised", "to":"authorized"},
            {"from":"authorises", "to":"authorizes"},
            {"from":"authorising", "to":"authorizing"},
            {"from":"axe", "to":"ax"},
            {"from":"backpedalled", "to":"backpedaled"},
            {"from":"backpedalling", "to":"backpedaling"},
            {"from":"bannister", "to":"banister"},
            {"from":"bannisters", "to":"banisters"},
            {"from":"baptise", "to":"baptize"},
            {"from":"baptised", "to":"baptized"},
            {"from":"baptises", "to":"baptizes"},
            {"from":"baptising", "to":"baptizing"},
            {"from":"bastardise", "to":"bastardize"},
            {"from":"bastardised", "to":"bastardized"},
            {"from":"bastardises", "to":"bastardizes"},
            {"from":"bastardising", "to":"bastardizing"},
            {"from":"battleaxe", "to":"battleax"},
            {"from":"baulk", "to":"balk"},
            {"from":"baulked", "to":"balked"},
            {"from":"baulking", "to":"balking"},
            {"from":"baulks", "to":"balks"},
            {"from":"bedevilled", "to":"bedeviled"},
            {"from":"bedevilling", "to":"bedeviling"},
            {"from":"behaviour", "to":"behavior"},
            {"from":"behavioural", "to":"behavioral"},
            {"from":"behaviourism", "to":"behaviorism"},
            {"from":"behaviourist", "to":"behaviorist"},
            {"from":"behaviourists", "to":"behaviorists"},
            {"from":"behaviours", "to":"behaviors"},
            {"from":"behove", "to":"behoove"},
            {"from":"behoved", "to":"behooved"},
            {"from":"behoves", "to":"behooves"},
            {"from":"bejewelled", "to":"bejeweled"},
            {"from":"belabour", "to":"belabor"},
            {"from":"belaboured", "to":"belabored"},
            {"from":"belabouring", "to":"belaboring"},
            {"from":"belabours", "to":"belabors"},
            {"from":"bevelled", "to":"beveled"},
```


(continues on next page)

(continued from previous page)


```
            {"from":"bevvies", "to":"bevies"},
            {"from":"bevvy", "to":"bevy"},
            {"from":"biassed", "to":"biased"},
            {"from":"biassing", "to":"biasing"},
            {"from":"bingeing", "to":"binging"},
            {"from":"bougainvillaea", "to":"bougainvillea"},
            {"from":"bougainvillaeas", "to":"bougainvilleas"},
            {"from":"bowdlerise", "to":"bowdlerize"},
            {"from":"bowdlerised", "to":"bowdlerized"},
            {"from":"bowdlerises", "to":"bowdlerizes"},
            {"from":"bowdlerising", "to":"bowdlerizing"},
            {"from":"breathalyse", "to":"breathalyze"},
            {"from":"breathalysed", "to":"breathalyzed"},
            {"from":"breathalyser", "to":"breathalyzer"},
            {"from":"breathalysers", "to":"breathalyzers"},
            {"from":"breathalyses", "to":"breathalyzes"},
            {"from":"breathalysing", "to":"breathalyzing"},
            {"from":"brutalise", "to":"brutalize"},
            {"from":"brutalised", "to":"brutalized"},
            {"from":"brutalises", "to":"brutalizes"},
            {"from":"brutalising", "to":"brutalizing"},
            {"from":"buses", "to":"busses"},
            {"from":"busing", "to":"bussing"},
            {"from":"caesarean", "to":"cesarean"},
            {"from":"caesareans", "to":"cesareans"},
            {"from":"calibre", "to":"caliber"},
            {"from":"calibres", "to":"calibers"},
            {"from":"calliper", "to":"caliper"},
            {"from":"callipers", "to":"calipers"},
            {"from":"callisthenics", "to":"calisthenics"},
            {"from":"canalise", "to":"canalize"},
            {"from":"canalised", "to":"canalized"},
            {"from":"canalises", "to":"canalizes"},
            {"from":"canalising", "to":"canalizing"},
            {"from":"cancellation", "to":"cancelation"},
            {"from":"cancellations", "to":"cancelations"},
            {"from":"cancelled", "to":"canceled"},
            {"from":"cancelling", "to":"canceling"},
            {"from":"candour", "to":"candor"},
            {"from":"cannibalise", "to":"cannibalize"},
            {"from":"cannibalised", "to":"cannibalized"},
            {"from":"cannibalises", "to":"cannibalizes"},
            {"from":"cannibalising", "to":"cannibalizing"},
            {"from":"canonise", "to":"canonize"},
            {"from":"canonised", "to":"canonized"},
            {"from":"canonises", "to":"canonizes"},
            {"from":"canonising", "to":"canonizing"},
            {"from":"capitalise", "to":"capitalize"},
            {"from":"capitalised", "to":"capitalized"},
            {"from":"capitalises", "to":"capitalizes"},
            {"from":"capitalising", "to":"capitalizing"},
            {"from":"caramelise", "to":"caramelize"},
            {"from":"caramelised", "to":"caramelized"},
            {"from":"caramelises", "to":"caramelizes"},
            {"from":"caramelising", "to":"caramelizing"},
            {"from":"carbonise", "to":"carbonize"},
            {"from":"carbonised", "to":"carbonized"},
            {"from":"carbonises", "to":"carbonizes"},
            {"from":"carbonising", "to":"carbonizing"},
            {"from":"carolled", "to":"caroled"},
            {"from":"carolling", "to":"caroling"},
```


(continues on next page)

(continued from previous page)


```
            {"from":"catalogue", "to":"catalog"},
            {"from":"catalogued", "to":"cataloged"},
            {"from":"catalogues", "to":"catalogs"},
            {"from":"cataloguing", "to":"cataloging"},
            {"from":"catalyse", "to":"catalyze"},
            {"from":"catalysed", "to":"catalyzed"},
            {"from":"catalyses", "to":"catalyzes"},
            {"from":"catalysing", "to":"catalyzing"},
            {"from":"categorise", "to":"categorize"},
            {"from":"categorised", "to":"categorized"},
            {"from":"categorises", "to":"categorizes"},
            {"from":"categorising", "to":"categorizing"},
            {"from":"cauterise", "to":"cauterize"},
            {"from":"cauterised", "to":"cauterized"},
            {"from":"cauterises", "to":"cauterizes"},
            {"from":"cauterising", "to":"cauterizing"},
            {"from":"cavilled", "to":"caviled"},
            {"from":"cavilling", "to":"caviling"},
            {"from":"centigramme", "to":"centigram"},
            {"from":"centigrammes", "to":"centigrams"},
            {"from":"centilitre", "to":"centiliter"},
            {"from":"centilitres", "to":"centiliters"},
            {"from":"centimetre", "to":"centimeter"},
            {"from":"centimetres", "to":"centimeters"},
            {"from":"centralise", "to":"centralize"},
            {"from":"centralised", "to":"centralized"},
            {"from":"centralises", "to":"centralizes"},
            {"from":"centralising", "to":"centralizing"},
            {"from":"centre", "to":"center"},
            {"from":"centred", "to":"centered"},
            {"from":"centrefold", "to":"centerfold"},
            {"from":"centrefolds", "to":"centerfolds"},
            {"from":"centrepiece", "to":"centerpiece"},
            {"from":"centrepieces", "to":"centerpieces"},
            {"from":"centres", "to":"centers"},
            {"from":"channelled", "to":"channeled"},
            {"from":"channelling", "to":"channeling"},
            {"from":"characterise", "to":"characterize"},
            {"from":"characterised", "to":"characterized"},
            {"from":"characterises", "to":"characterizes"},
            {"from":"characterising", "to":"characterizing"},
            {"from":"cheque", "to":"check"},
            {"from":"chequebook", "to":"checkbook"},
            {"from":"chequebooks", "to":"checkbooks"},
            {"from":"chequered", "to":"checkered"},
            {"from":"cheques", "to":"checks"},
            {"from":"chilli", "to":"chili"},
            {"from":"chimaera", "to":"chimera"},
            {"from":"chimaeras", "to":"chimeras"},
            {"from":"chiselled", "to":"chiseled"},
            {"from":"chiselling", "to":"chiseling"},
            {"from":"circularise", "to":"circularize"},
            {"from":"circularised", "to":"circularized"},
            {"from":"circularises", "to":"circularizes"},
            {"from":"circularising", "to":"circularizing"},
            {"from":"civilise", "to":"civilize"},
            {"from":"civilised", "to":"civilized"},
            {"from":"civilises", "to":"civilizes"},
            {"from":"civilising", "to":"civilizing"},
            {"from":"clamour", "to":"clamor"},
            {"from":"clamoured", "to":"clamored"},
```


(continues on next page)

(continued from previous page)


```
{"from":"clamouring", "to":"clamoring"},
{"from":"clamours", "to":"clamors"},
{"from":"clangour", "to":"clangor"},
{"from":"clarinettist", "to":"clarinetist"},
{"from":"clarinettists", "to":"clarinetists"},
{"from":"collectivise", "to":"collectivize"},
{"from":"collectivised", "to":"collectivized"},
{"from":"collectivises", "to":"collectivizes"},
{"from":"collectivising", "to":"collectivizing"},
{"from":"colonisation", "to":"colonization"},
{"from":"colonise", "to":"colonize"},
{"from":"colonised", "to":"colonized"},
{"from":"coloniser", "to":"colonizer"},
{"from":"colonisers", "to":"colonizers"},
{"from":"colonises", "to":"colonizes"},
{"from":"colonising", "to":"colonizing"},
{"from":"colour", "to":"color"},
{"from":"colourant", "to":"colorant"},
{"from":"colourants", "to":"colorants"},
{"from":"coloured", "to":"colored"},
{"from":"coloureds", "to":"coloreds"},
{"from":"colourful", "to":"colorful"},
{"from":"colourfully", "to":"colorfully"},
{"from":"colouring", "to":"coloring"},
{"from":"colourize", "to":"colorize"},
{"from":"colourized", "to":"colorized"},
{"from":"colourizes", "to":"colorizes"},
{"from":"colourizing", "to":"colorizing"},
{"from":"colourless", "to":"colorless"},
{"from":"colours", "to":"colors"},
{"from":"commercialise", "to":"commercialize"},
{"from":"commercialised", "to":"commercialized"},
{"from":"commercialises", "to":"commercializes"},
{"from":"commercialising", "to":"commercializing"},
{"from":"compartmentalise", "to":"compartmentalize"},
{"from":"compartmentalised", "to":"compartmentalized"},
{"from":"compartmentalises", "to":"compartmentalizes"},
{"from":"compartmentalising", "to":"compartmentalizing"},
{"from":"computerise", "to":"computerize"},
{"from":"computerised", "to":"computerized"},
{"from":"computerises", "to":"computerizes"},
{"from":"computerising", "to":"computerizing"},
{"from":"conceptualise", "to":"conceptualize"},
{"from":"conceptualised", "to":"conceptualized"},
{"from":"conceptualises", "to":"conceptualizes"},
{"from":"conceptualising", "to":"conceptualizing"},
{"from":"connexion", "to":"connection"},
{"from":"connexions", "to":"connections"},
{"from":"contextualise", "to":"contextualize"},
{"from":"contextualised", "to":"contextualized"},
{"from":"contextualises", "to":"contextualizes"},
{"from":"contextualising", "to":"contextualizing"},
{"from":"cosier", "to":"cozier"},
{"from":"cosies", "to":"cozies"},
{"from":"cosiest", "to":"coziest"},
{"from":"cosily", "to":"cozily"},
{"from":"cosiness", "to":"coziness"},
{"from":"cosy", "to":"cozy"},
{"from":"councillor", "to":"councilor"},
{"from":"councillors", "to":"councilors"},
{"from":"counselled", "to":"counseled"},
```


(continues on next page)

(continued from previous page)


```
            {"from":"counselling", "to":"counseling"},
            {"from":"counsellor", "to":"counselor"},
            {"from":"counsellors", "to":"counselors"},
            {"from":"crenellated", "to":"crenelated"},
            {"from":"criminalise", "to":"criminalize"},
            {"from":"criminalised", "to":"criminalized"},
            {"from":"criminalises", "to":"criminalizes"},
            {"from":"criminalising", "to":"criminalizing"},
            {"from":"criticise", "to":"criticize"},
            {"from":"criticised", "to":"criticized"},
            {"from":"criticises", "to":"criticizes"},
            {"from":"criticising", "to":"criticizing"},
            {"from":"crueller", "to":"crueler"},
            {"from":"cruellest", "to":"cruelest"},
            {"from":"crystallisation", "to":"crystallization"},
            {"from":"crystallise", "to":"crystallize"},
            {"from":"crystallised", "to":"crystallized"},
            {"from":"crystallises", "to":"crystallizes"},
            {"from":"crystallising", "to":"crystallizing"},
            {"from":"cudgelled", "to":"cudgeled"},
            {"from":"cudgelling", "to":"cudgeling"},
            {"from":"customise", "to":"customize"},
            {"from":"customised", "to":"customized"},
            {"from":"customises", "to":"customizes"},
            {"from":"customising", "to":"customizing"},
            {"from":"cypher", "to":"cipher"},
            {"from":"cyphers", "to":"ciphers"},
            {"from":"decentralisation", "to":"decentralization"},
            {"from":"decentralise", "to":"decentralize"},
            {"from":"decentralised", "to":"decentralized"},
            {"from":"decentralises", "to":"decentralizes"},
            {"from":"decentralising", "to":"decentralizing"},
            {"from":"decriminalisation", "to":"decriminalization"},
            {"from":"decriminalise", "to":"decriminalize"},
            {"from":"decriminalised", "to":"decriminalized"},
            {"from":"decriminalises", "to":"decriminalizes"},
            {"from":"decriminalising", "to":"decriminalizing"},
            {"from":"defence", "to":"defense"},
            {"from":"defenceless", "to":"defenseless"},
            {"from":"defences", "to":"defenses"},
            {"from":"dehumanisation", "to":"dehumanization"},
            {"from":"dehumanise", "to":"dehumanize"},
            {"from":"dehumanised", "to":"dehumanized"},
            {"from":"dehumanises", "to":"dehumanizes"},
            {"from":"dehumanising", "to":"dehumanizing"},
            {"from":"demeanour", "to":"demeanor"},
            {"from":"demilitarisation", "to":"demilitarization"},
            {"from":"demilitarise", "to":"demilitarize"},
            {"from":"demilitarised", "to":"demilitarized"},
            {"from":"demilitarises", "to":"demilitarizes"},
            {"from":"demilitarising", "to":"demilitarizing"},
            {"from":"demobilisation", "to":"demobilization"},
            {"from":"demobilise", "to":"demobilize"},
            {"from":"demobilised", "to":"demobilized"},
            {"from":"demobilises", "to":"demobilizes"},
            {"from":"demobilising", "to":"demobilizing"},
            {"from":"democratisation", "to":"democratization"},
            {"from":"democratise", "to":"democratize"},
            {"from":"democratised", "to":"democratized"},
            {"from":"democratises", "to":"democratizes"},
            {"from":"democratising", "to":"democratizing"},
```


(continues on next page)

(continued from previous page)


```
            {"from":"demonise", "to":"demonize"},
            {"from":"demonised", "to":"demonized"},
            {"from":"demonises", "to":"demonizes"},
            {"from":"demonising", "to":"demonizing"},
            {"from":"demoralisation", "to":"demoralization"},
            {"from":"demoralise", "to":"demoralize"},
            {"from":"demoralised", "to":"demoralized"},
            {"from":"demoralises", "to":"demoralizes"},
            {"from":"demoralising", "to":"demoralizing"},
            {"from":"denationalisation", "to":"denationalization"},
            {"from":"denationalise", "to":"denationalize"},
            {"from":"denationalised", "to":"denationalized"},
            {"from":"denationalises", "to":"denationalizes"},
            {"from":"denationalising", "to":"denationalizing"},
            {"from":"deodorise", "to":"deodorize"},
            {"from":"deodorised", "to":"deodorized"},
            {"from":"deodorises", "to":"deodorizes"},
            {"from":"deodorising", "to":"deodorizing"},
            {"from":"depersonalise", "to":"depersonalize"},
            {"from":"depersonalised", "to":"depersonalized"},
            {"from":"depersonalises", "to":"depersonalizes"},
            {"from":"depersonalising", "to":"depersonalizing"},
            {"from":"deputise", "to":"deputize"},
            {"from":"deputised", "to":"deputized"},
            {"from":"deputises", "to":"deputizes"},
            {"from":"deputising", "to":"deputizing"},
            {"from":"desensitisation", "to":"desensitization"},
            {"from":"desensitise", "to":"desensitize"},
            {"from":"desensitised", "to":"desensitized"},
            {"from":"desensitises", "to":"desensitizes"},
            {"from":"desensitising", "to":"desensitizing"},
            {"from":"destabilisation", "to":"destabilization"},
            {"from":"destabilise", "to":"destabilize"},
            {"from":"destabilised", "to":"destabilized"},
            {"from":"destabilises", "to":"destabilizes"},
            {"from":"destabilising", "to":"destabilizing"},
            {"from":"dialled", "to":"dialed"},
            {"from":"dialling", "to":"dialing"},
            {"from":"dialogue", "to":"dialog"},
            {"from":"dialogues", "to":"dialogs"},
            {"from":"diarrhoea", "to":"diarrhea"},
            {"from":"digitise", "to":"digitize"},
            {"from":"digitised", "to":"digitized"},
            {"from":"digitises", "to":"digitizes"},
            {"from":"digitising", "to":"digitizing"},
            {"from":"disc", "to":"disk"},
            {"from":"discolour", "to":"discolor"},
            {"from":"discoloured", "to":"discolored"},
            {"from":"discolouring", "to":"discoloring"},
            {"from":"discolours", "to":"discolors"},
            {"from":"discs", "to":"disks"},
            {"from":"disembowelled", "to":"disemboweled"},
            {"from":"disembowelling", "to":"disemboweling"},
            {"from":"disfavour", "to":"disfavor"},
            {"from":"dishevelled", "to":"disheveled"},
            {"from":"dishonour", "to":"dishonor"},
            {"from":"dishonourable", "to":"dishonorable"},
            {"from":"dishonourably", "to":"dishonorably"},
            {"from":"dishonoured", "to":"dishonored"},
            {"from":"dishonouring", "to":"dishonoring"},
            {"from":"dishonours", "to":"dishonors"},
```


(continues on next page)

(continued from previous page)


```
            {"from":"disorganisation", "to":"disorganization"},
            {"from":"disorganised", "to":"disorganized"},
            {"from":"distil", "to":"distill"},
            {"from":"distils", "to":"distills"},
            {"from":"dramatisation", "to":"dramatization"},
            {"from":"dramatisations", "to":"dramatizations"},
            {"from":"dramatise", "to":"dramatize"},
            {"from":"dramatised", "to":"dramatized"},
            {"from":"dramatises", "to":"dramatizes"},
            {"from":"dramatising", "to":"dramatizing"},
            {"from":"draught", "to":"draft"},
            {"from":"draughtboard", "to":"draftboard"},
            {"from":"draughtboards", "to":"draftboards"},
            {"from":"draughtier", "to":"draftier"},
            {"from":"draughtiest", "to":"draftiest"},
            {"from":"draughts", "to":"drafts"},
            {"from":"draughtsman", "to":"draftsman"},
            {"from":"draughtsmanship", "to":"draftsmanship"},
            {"from":"draughtsmen", "to":"draftsmen"},
            {"from":"draughtswoman", "to":"draftswoman"},
            {"from":"draughtswomen", "to":"draftswomen"},
            {"from":"draughty", "to":"drafty"},
            {"from":"drivelled", "to":"driveled"},
            {"from":"drivelling", "to":"driveling"},
            {"from":"duelled", "to":"dueled"},
            {"from":"duelling", "to":"dueling"},
            {"from":"economise", "to":"economize"},
            {"from":"economised", "to":"economized"},
            {"from":"economises", "to":"economizes"},
            {"from":"economising", "to":"economizing"},
            {"from":"edoema", "to":"edema"},
            {"from":"editorialise", "to":"editorialize"},
            {"from":"editorialised", "to":"editorialized"},
            {"from":"editorialises", "to":"editorializes"},
            {"from":"editorialising", "to":"editorializing"},
            {"from":"empathise", "to":"empathize"},
            {"from":"empathised", "to":"empathized"},
            {"from":"empathises", "to":"empathizes"},
            {"from":"empathising", "to":"empathizing"},
            {"from":"emphasise", "to":"emphasize"},
            {"from":"emphasised", "to":"emphasized"},
            {"from":"emphasises", "to":"emphasizes"},
            {"from":"emphasising", "to":"emphasizing"},
            {"from":"enamelled", "to":"enameled"},
            {"from":"enamelling", "to":"enameling"},
            {"from":"enamoured", "to":"enamored"},
            {"from":"encyclopaedia", "to":"encyclopedia"},
            {"from":"encyclopaedias", "to":"encyclopedias"},
            {"from":"encyclopaedic", "to":"encyclopedic"},
            {"from":"endeavour", "to":"endeavor"},
            {"from":"endeavoured", "to":"endeavored"},
            {"from":"endeavouring", "to":"endeavoring"},
            {"from":"endeavours", "to":"endeavors"},
            {"from":"energise", "to":"energize"},
            {"from":"energised", "to":"energized"},
            {"from":"energises", "to":"energizes"},
            {"from":"energising", "to":"energizing"},
            {"from":"enrol", "to":"enroll"},
            {"from":"enrols", "to":"enrolls"},
            {"from":"enthral", "to":"enthrall"},
            {"from":"enthrals", "to":"enthralls"},
```


(continues on next page)

(continued from previous page)


```
            {"from":"epaulette", "to":"epaulet"},
            {"from":"epaulettes", "to":"epaulets"},
            {"from":"epicentre", "to":"epicenter"},
            {"from":"epicentres", "to":"epicenters"},
            {"from":"epilogue", "to":"epilog"},
            {"from":"epilogues", "to":"epilogs"},
            {"from":"epitomise", "to":"epitomize"},
            {"from":"epitomised", "to":"epitomized"},
            {"from":"epitomises", "to":"epitomizes"},
            {"from":"epitomising", "to":"epitomizing"},
            {"from":"equalisation", "to":"equalization"},
            {"from":"equalise", "to":"equalize"},
            {"from":"equalised", "to":"equalized"},
            {"from":"equaliser", "to":"equalizer"},
            {"from":"equalisers", "to":"equalizers"},
            {"from":"equalises", "to":"equalizes"},
            {"from":"equalising", "to":"equalizing"},
            {"from":"eulogise", "to":"eulogize"},
            {"from":"eulogised", "to":"eulogized"},
            {"from":"eulogises", "to":"eulogizes"},
            {"from":"eulogising", "to":"eulogizing"},
            {"from":"evangelise", "to":"evangelize"},
            {"from":"evangelised", "to":"evangelized"},
            {"from":"evangelises", "to":"evangelizes"},
            {"from":"evangelising", "to":"evangelizing"},
            {"from":"exorcise", "to":"exorcize"},
            {"from":"exorcised", "to":"exorcized"},
            {"from":"exorcises", "to":"exorcizes"},
            {"from":"exorcising", "to":"exorcizing"},
            {"from":"extemporisation", "to":"extemporization"},
            {"from":"extemporise", "to":"extemporize"},
            {"from":"extemporised", "to":"extemporized"},
            {"from":"extemporises", "to":"extemporizes"},
            {"from":"extemporising", "to":"extemporizing"},
            {"from":"externalisation", "to":"externalization"},
            {"from":"externalisations", "to":"externalizations"},
            {"from":"externalise", "to":"externalize"},
            {"from":"externalised", "to":"externalized"},
            {"from":"externalises", "to":"externalizes"},
            {"from":"externalising", "to":"externalizing"},
            {"from":"factorise", "to":"factorize"},
            {"from":"factorised", "to":"factorized"},
            {"from":"factorises", "to":"factorizes"},
            {"from":"factorising", "to":"factorizing"},
            {"from":"faecal", "to":"fecal"},
            {"from":"faeces", "to":"feces"},
            {"from":"familiarisation", "to":"familiarization"},
            {"from":"familiarise", "to":"familiarize"},
            {"from":"familiarised", "to":"familiarized"},
            {"from":"familiarises", "to":"familiarizes"},
            {"from":"familiarising", "to":"familiarizing"},
            {"from":"fantasise", "to":"fantasize"},
            {"from":"fantasised", "to":"fantasized"},
            {"from":"fantasises", "to":"fantasizes"},
            {"from":"fantasising", "to":"fantasizing"},
            {"from":"favour", "to":"favor"},
            {"from":"favourable", "to":"favorable"},
            {"from":"favourably", "to":"favorably"},
            {"from":"favoured", "to":"favored"},
            {"from":"favouring", "to":"favoring"},
            {"from":"favourite", "to":"favorite"},
```


(continues on next page)

(continued from previous page)


```
            {"from":"favourites", "to":"favorites"},
            {"from":"favouritism", "to":"favoritism"},
            {"from":"favours", "to":"favors"},
            {"from":"feminise", "to":"feminize"},
            {"from":"feminised", "to":"feminized"},
            {"from":"feminises", "to":"feminizes"},
            {"from":"feminising", "to":"feminizing"},
            {"from":"fertilisation", "to":"fertilization"},
            {"from":"fertilise", "to":"fertilize"},
            {"from":"fertilised", "to":"fertilized"},
            {"from":"fertiliser", "to":"fertilizer"},
            {"from":"fertilisers", "to":"fertilizers"},
            {"from":"fertilises", "to":"fertilizes"},
            {"from":"fertilising", "to":"fertilizing"},
            {"from":"fervour", "to":"fervor"},
            {"from":"fibre", "to":"fiber"},
            {"from":"fibreglass", "to":"fiberglass"},
            {"from":"fibres", "to":"fibers"},
            {"from":"fictionalisation", "to":"fictionalization"},
            {"from":"fictionalisations", "to":"fictionalizations"},
            {"from":"fictionalise", "to":"fictionalize"},
            {"from":"fictionalised", "to":"fictionalized"},
            {"from":"fictionalises", "to":"fictionalizes"},
            {"from":"fictionalising", "to":"fictionalizing"},
            {"from":"fillet", "to":"filet"},
            {"from":"filleted", "to":"fileted"},
            {"from":"filleting", "to":"fileting"},
            {"from":"fillets", "to":"filets"},
            {"from":"finalisation", "to":"finalization"},
            {"from":"finalise", "to":"finalize"},
            {"from":"finalised", "to":"finalized"},
            {"from":"finalises", "to":"finalizes"},
            {"from":"finalising", "to":"finalizing"},
            {"from":"flautist", "to":"flutist"},
            {"from":"flautists", "to":"flutists"},
            {"from":"flavour", "to":"flavor"},
            {"from":"flavoured", "to":"flavored"},
            {"from":"flavouring", "to":"flavoring"},
            {"from":"flavourings", "to":"flavorings"},
            {"from":"flavourless", "to":"flavorless"},
            {"from":"flavours", "to":"flavors"},
            {"from":"flavoursome", "to":"flavorsome"},
            {"from":"flyer / flier", "to":"flier / flyer"},
            {"from":"foetal", "to":"fetal"},
            {"from":"foetid", "to":"fetid"},
            {"from":"foetus", "to":"fetus"},
            {"from":"foetuses", "to":"fetuses"},
            {"from":"formalisation", "to":"formalization"},
            {"from":"formalise", "to":"formalize"},
            {"from":"formalised", "to":"formalized"},
            {"from":"formalises", "to":"formalizes"},
            {"from":"formalising", "to":"formalizing"},
            {"from":"fossilisation", "to":"fossilization"},
            {"from":"fossilise", "to":"fossilize"},
            {"from":"fossilised", "to":"fossilized"},
            {"from":"fossilises", "to":"fossilizes"},
            {"from":"fossilising", "to":"fossilizing"},
            {"from":"fraternisation", "to":"fraternization"},
            {"from":"fraternise", "to":"fraternize"},
            {"from":"fraternised", "to":"fraternized"},
            {"from":"fraternises", "to":"fraternizes"},
```


(continues on next page)

(continued from previous page)


```
            {"from":"fraternising", "to":"fraternizing"},
            {"from":"fulfil", "to":"fulfill"},
            {"from":"fulfilment", "to":"fulfillment"},
            {"from":"fulfils", "to":"fulfills"},
            {"from":"funnelled", "to":"funneled"},
            {"from":"funnelling", "to":"funneling"},
            {"from":"galvanise", "to":"galvanize"},
            {"from":"galvanised", "to":"galvanized"},
            {"from":"galvanises", "to":"galvanizes"},
            {"from":"galvanising", "to":"galvanizing"},
            {"from":"gambolled", "to":"gamboled"},
            {"from":"gambolling", "to":"gamboling"},
            {"from":"gaol", "to":"jail"},
            {"from":"gaolbird", "to":"jailbird"},
            {"from":"gaolbirds", "to":"jailbirds"},
            {"from":"gaolbreak", "to":"jailbreak"},
            {"from":"gaolbreaks", "to":"jailbreaks"},
            {"from":"gaoled", "to":"jailed"},
            {"from":"gaoler", "to":"jailer"},
            {"from":"gaolers", "to":"jailers"},
            {"from":"gaoling", "to":"jailing"},
            {"from":"gaols", "to":"jails"},
            {"from":"gases", "to":"gasses"},
            {"from":"gauge", "to":"gage"},
            {"from":"gauged", "to":"gaged"},
            {"from":"gauges", "to":"gages"},
            {"from":"gauging", "to":"gaging"},
            {"from":"generalisation", "to":"generalization"},
            {"from":"generalisations", "to":"generalizations"},
            {"from":"generalise", "to":"generalize"},
            {"from":"generalised", "to":"generalized"},
            {"from":"generalises", "to":"generalizes"},
            {"from":"generalising", "to":"generalizing"},
            {"from":"ghettoise", "to":"ghettoize"},
            {"from":"ghettoised", "to":"ghettoized"},
            {"from":"ghettoises", "to":"ghettoizes"},
            {"from":"ghettoising", "to":"ghettoizing"},
            {"from":"gipsies", "to":"gypsies"},
            {"from":"glamorise", "to":"glamorize"},
            {"from":"glamorised", "to":"glamorized"},
            {"from":"glamorises", "to":"glamorizes"},
            {"from":"glamorising", "to":"glamorizing"},
            {"from":"glamour", "to":"glamor"},
            {"from":"globalisation", "to":"globalization"},
            {"from":"globalise", "to":"globalize"},
            {"from":"globalised", "to":"globalized"},
            {"from":"globalises", "to":"globalizes"},
            {"from":"globalising", "to":"globalizing"},
            {"from":"glueing", "to":"gluing"},
            {"from":"goitre", "to":"goiter"},
            {"from":"goitres", "to":"goiters"},
            {"from":"gonorrhoea", "to":"gonorrhea"},
            {"from":"gramme", "to":"gram"},
            {"from":"grammes", "to":"grams"},
            {"from":"gravelled", "to":"graveled"},
            {"from":"grey", "to":"gray"},
            {"from":"greyed", "to":"grayed"},
            {"from":"greying", "to":"graying"},
            {"from":"greyish", "to":"grayish"},
            {"from":"greyness", "to":"grayness"},
            {"from":"greys", "to":"grays"},
```


(continues on next page)

(continued from previous page)


```
            {"from":"grovelled", "to":"groveled"},
            {"from":"grovelling", "to":"groveling"},
            {"from":"groyne", "to":"groin"},
            {"from":"groynes", "to":"groins"},
            {"from":"gruelling", "to":"grueling"},
            {"from":"gruellingly", "to":"gruelingly"},
            {"from":"gryphon", "to":"griffin"},
            {"from":"gryphons", "to":"griffins"},
            {"from":"gynaecological", "to":"gynecological"},
            {"from":"gynaecologist", "to":"gynecologist"},
            {"from":"gynaecologists", "to":"gynecologists"},
            {"from":"gynaecology", "to":"gynecology"},
            {"from":"haematological", "to":"hematological"},
            {"from":"haematologist", "to":"hematologist"},
            {"from":"haematologists", "to":"hematologists"},
            {"from":"haematology", "to":"hematology"},
            {"from":"haemoglobin", "to":"hemoglobin"},
            {"from":"haemophilia", "to":"hemophilia"},
            {"from":"haemophiliac", "to":"hemophiliac"},
            {"from":"haemophiliacs", "to":"hemophiliacs"},
            {"from":"haemorrhage", "to":"hemorrhage"},
            {"from":"haemorrhaged", "to":"hemorrhaged"},
            {"from":"haemorrhages", "to":"hemorrhages"},
            {"from":"haemorrhaging", "to":"hemorrhaging"},
            {"from":"haemorrhoids", "to":"hemorrhoids"},
            {"from":"harbour", "to":"harbor"},
            {"from":"harboured", "to":"harbored"},
            {"from":"harbouring", "to":"harboring"},
            {"from":"harbours", "to":"harbors"},
            {"from":"harmonisation", "to":"harmonization"},
            {"from":"harmonise", "to":"harmonize"},
            {"from":"harmonised", "to":"harmonized"},
            {"from":"harmonises", "to":"harmonizes"},
            {"from":"harmonising", "to":"harmonizing"},
            {"from":"homoeopath", "to":"homeopath"},
            {"from":"homoeopathic", "to":"homeopathic"},
            {"from":"homoeopaths", "to":"homeopaths"},
            {"from":"homoeopathy", "to":"homeopathy"},
            {"from":"homogenise", "to":"homogenize"},
            {"from":"homogenised", "to":"homogenized"},
            {"from":"homogenises", "to":"homogenizes"},
            {"from":"homogenising", "to":"homogenizing"},
            {"from":"honour", "to":"honor"},
            {"from":"honourable", "to":"honorable"},
            {"from":"honourably", "to":"honorably"},
            {"from":"honoured", "to":"honored"},
            {"from":"honouring", "to":"honoring"},
            {"from":"honours", "to":"honors"},
            {"from":"hospitalisation", "to":"hospitalization"},
            {"from":"hospitalise", "to":"hospitalize"},
            {"from":"hospitalised", "to":"hospitalized"},
            {"from":"hospitalises", "to":"hospitalizes"},
            {"from":"hospitalising", "to":"hospitalizing"},
            {"from":"humanise", "to":"humanize"},
            {"from":"humanised", "to":"humanized"},
            {"from":"humanises", "to":"humanizes"},
            {"from":"humanising", "to":"humanizing"},
            {"from":"humour", "to":"humor"},
            {"from":"humoured", "to":"humored"},
            {"from":"humouring", "to":"humoring"},
            {"from":"humourless", "to":"humorless"},
```


(continues on next page)

(continued from previous page)


```
            {"from":"humours", "to":"humors"},
            {"from":"hybridise", "to":"hybridize"},
            {"from":"hybridised", "to":"hybridized"},
            {"from":"hybridises", "to":"hybridizes"},
            {"from":"hybridising", "to":"hybridizing"},
            {"from":"hypnotise", "to":"hypnotize"},
            {"from":"hypnotised", "to":"hypnotized"},
            {"from":"hypnotises", "to":"hypnotizes"},
            {"from":"hypnotising", "to":"hypnotizing"},
            {"from":"hypothesise", "to":"hypothesize"},
            {"from":"hypothesised", "to":"hypothesized"},
            {"from":"hypothesises", "to":"hypothesizes"},
            {"from":"hypothesising", "to":"hypothesizing"},
            {"from":"idealisation", "to":"idealization"},
            {"from":"idealise", "to":"idealize"},
            {"from":"idealised", "to":"idealized"},
            {"from":"idealises", "to":"idealizes"},
            {"from":"idealising", "to":"idealizing"},
            {"from":"idolise", "to":"idolize"},
            {"from":"idolised", "to":"idolized"},
            {"from":"idolises", "to":"idolizes"},
            {"from":"idolising", "to":"idolizing"},
            {"from":"immobilisation", "to":"immobilization"},
            {"from":"immobilise", "to":"immobilize"},
            {"from":"immobilised", "to":"immobilized"},
            {"from":"immobiliser", "to":"immobilizer"},
            {"from":"immobilisers", "to":"immobilizers"},
            {"from":"immobilises", "to":"immobilizes"},
            {"from":"immobilising", "to":"immobilizing"},
            {"from":"immortalise", "to":"immortalize"},
            {"from":"immortalised", "to":"immortalized"},
            {"from":"immortalises", "to":"immortalizes"},
            {"from":"immortalising", "to":"immortalizing"},
            {"from":"immunisation", "to":"immunization"},
            {"from":"immunise", "to":"immunize"},
            {"from":"immunised", "to":"immunized"},
            {"from":"immunises", "to":"immunizes"},
            {"from":"immunising", "to":"immunizing"},
            {"from":"impanelled", "to":"impaneled"},
            {"from":"impanelling", "to":"impaneling"},
            {"from":"imperilled", "to":"imperiled"},
            {"from":"imperilling", "to":"imperiling"},
            {"from":"individualise", "to":"individualize"},
            {"from":"individualised", "to":"individualized"},
            {"from":"individualises", "to":"individualizes"},
            {"from":"individualising", "to":"individualizing"},
            {"from":"industrialise", "to":"industrialize"},
            {"from":"industrialised", "to":"industrialized"},
            {"from":"industrialises", "to":"industrializes"},
            {"from":"industrialising", "to":"industrializing"},
            {"from":"inflexion", "to":"inflection"},
            {"from":"inflexions", "to":"inflections"},
            {"from":"initialise", "to":"initialize"},
            {"from":"initialised", "to":"initialized"},
            {"from":"initialises", "to":"initializes"},
            {"from":"initialising", "to":"initializing"},
            {"from":"initialled", "to":"initialed"},
            {"from":"initialling", "to":"initialing"},
            {"from":"instal", "to":"install"},
            {"from":"instalment", "to":"installment"},
            {"from":"instalments", "to":"installments"},
```


(continues on next page)

(continued from previous page)


```
            {"from":"instals", "to":"installs"},
            {"from":"instil", "to":"instill"},
            {"from":"instils", "to":"instills"},
            {"from":"institutionalisation", "to":"institutionalization"},
            {"from":"institutionalise", "to":"institutionalize"},
            {"from":"institutionalised", "to":"institutionalized"},
            {"from":"institutionalises", "to":"institutionalizes"},
            {"from":"institutionalising", "to":"institutionalizing"},
            {"from":"intellectualise", "to":"intellectualize"},
            {"from":"intellectualised", "to":"intellectualized"},
            {"from":"intellectualises", "to":"intellectualizes"},
            {"from":"intellectualising", "to":"intellectualizing"},
            {"from":"internalisation", "to":"internalization"},
            {"from":"internalise", "to":"internalize"},
            {"from":"internalised", "to":"internalized"},
            {"from":"internalises", "to":"internalizes"},
            {"from":"internalising", "to":"internalizing"},
            {"from":"internationalisation", "to":"internationalization"},
            {"from":"internationalise", "to":"internationalize"},
            {"from":"internationalised", "to":"internationalized"},
            {"from":"internationalises", "to":"internationalizes"},
            {"from":"internationalising", "to":"internationalizing"},
            {"from":"ionisation", "to":"ionization"},
            {"from":"ionise", "to":"ionize"},
            {"from":"ionised", "to":"ionized"},
            {"from":"ioniser", "to":"ionizer"},
            {"from":"ionisers", "to":"ionizers"},
            {"from":"ionises", "to":"ionizes"},
            {"from":"ionising", "to":"ionizing"},
            {"from":"italicise", "to":"italicize"},
            {"from":"italicised", "to":"italicized"},
            {"from":"italicises", "to":"italicizes"},
            {"from":"italicising", "to":"italicizing"},
            {"from":"itemise", "to":"itemize"},
            {"from":"itemised", "to":"itemized"},
            {"from":"itemises", "to":"itemizes"},
            {"from":"itemising", "to":"itemizing"},
            {"from":"jeopardise", "to":"jeopardize"},
            {"from":"jeopardised", "to":"jeopardized"},
            {"from":"jeopardises", "to":"jeopardizes"},
            {"from":"jeopardising", "to":"jeopardizing"},
            {"from":"jewelled", "to":"jeweled"},
            {"from":"jeweller", "to":"jeweler"},
            {"from":"jewellers", "to":"jewelers"},
            {"from":"jewellery", "to":"jewelry"},
            {"from":"judgement", "to":"judgment"},
            {"from":"kilogramme", "to":"kilogram"},
            {"from":"kilogrammes", "to":"kilograms"},
            {"from":"kilometre", "to":"kilometer"},
            {"from":"kilometres", "to":"kilometers"},
            {"from":"labelled", "to":"labeled"},
            {"from":"labelling", "to":"labeling"},
            {"from":"labour", "to":"labor"},
            {"from":"laboured", "to":"labored"},
            {"from":"labourer", "to":"laborer"},
            {"from":"labourers", "to":"laborers"},
            {"from":"labouring", "to":"laboring"},
            {"from":"labours", "to":"labors"},
            {"from":"lacklustre", "to":"lackluster"},
            {"from":"legalisation", "to":"legalization"},
            {"from":"legalise", "to":"legalize"},
```


(continues on next page)

(continued from previous page)


```
            {"from":"legalised", "to":"legalized"},
            {"from":"legalises", "to":"legalizes"},
            {"from":"legalising", "to":"legalizing"},
            {"from":"legitimise", "to":"legitimize"},
            {"from":"legitimised", "to":"legitimized"},
            {"from":"legitimises", "to":"legitimizes"},
            {"from":"legitimising", "to":"legitimizing"},
            {"from":"leukaemia", "to":"leukemia"},
            {"from":"levelled", "to":"leveled"},
            {"from":"leveller", "to":"leveler"},
            {"from":"levellers", "to":"levelers"},
            {"from":"levelling", "to":"leveling"},
            {"from":"libelled", "to":"libeled"},
            {"from":"libelling", "to":"libeling"},
            {"from":"libellous", "to":"libelous"},
            {"from":"liberalisation", "to":"liberalization"},
            {"from":"liberalise", "to":"liberalize"},
            {"from":"liberalised", "to":"liberalized"},
            {"from":"liberalises", "to":"liberalizes"},
            {"from":"liberalising", "to":"liberalizing"},
            {"from":"licence", "to":"license"},
            {"from":"licenced", "to":"licensed"},
            {"from":"licences", "to":"licenses"},
            {"from":"licencing", "to":"licensing"},
            {"from":"likeable", "to":"likable"},
            {"from":"lionisation", "to":"lionization"},
            {"from":"lionise", "to":"lionize"},
            {"from":"lionised", "to":"lionized"},
            {"from":"lionises", "to":"lionizes"},
            {"from":"lionising", "to":"lionizing"},
            {"from":"liquidise", "to":"liquidize"},
            {"from":"liquidised", "to":"liquidized"},
            {"from":"liquidiser", "to":"liquidizer"},
            {"from":"liquidisers", "to":"liquidizers"},
            {"from":"liquidises", "to":"liquidizes"},
            {"from":"liquidising", "to":"liquidizing"},
            {"from":"litre", "to":"liter"},
            {"from":"litres", "to":"liters"},
            {"from":"localise", "to":"localize"},
            {"from":"localised", "to":"localized"},
            {"from":"localises", "to":"localizes"},
            {"from":"localising", "to":"localizing"},
            {"from":"louvre", "to":"louver"},
            {"from":"louvred", "to":"louvered"},
            {"from":"louvres", "to":"louvers"},
            {"from":"lustre", "to":"luster"},
            {"from":"magnetise", "to":"magnetize"},
            {"from":"magnetised", "to":"magnetized"},
            {"from":"magnetises", "to":"magnetizes"},
            {"from":"magnetising", "to":"magnetizing"},
            {"from":"manoeuvrability", "to":"maneuverability"},
            {"from":"manoeuvrable", "to":"maneuverable"},
            {"from":"manoeuvre", "to":"maneuver"},
            {"from":"manoeuvred", "to":"maneuvered"},
            {"from":"manoeuvres", "to":"maneuvers"},
            {"from":"manoeuvring", "to":"maneuvering"},
            {"from":"manoeuvrings", "to":"maneuverings"},
            {"from":"marginalisation", "to":"marginalization"},
            {"from":"marginalise", "to":"marginalize"},
            {"from":"marginalised", "to":"marginalized"},
            {"from":"marginalises", "to":"marginalizes"},
```


(continues on next page)

(continued from previous page)


```
            {"from":"marginalising", "to":"marginalizing"},
            {"from":"marshalled", "to":"marshaled"},
            {"from":"marshalling", "to":"marshaling"},
            {"from":"marvelled", "to":"marveled"},
            {"from":"marvelling", "to":"marveling"},
            {"from":"marvellous", "to":"marvelous"},
            {"from":"marvellously", "to":"marvelously"},
            {"from":"materialisation", "to":"materialization"},
            {"from":"materialise", "to":"materialize"},
            {"from":"materialised", "to":"materialized"},
            {"from":"materialises", "to":"materializes"},
            {"from":"materialising", "to":"materializing"},
            {"from":"maximisation", "to":"maximization"},
            {"from":"maximise", "to":"maximize"},
            {"from":"maximised", "to":"maximized"},
            {"from":"maximises", "to":"maximizes"},
            {"from":"maximising", "to":"maximizing"},
            {"from":"meagre", "to":"meager"},
            {"from":"mechanisation", "to":"mechanization"},
            {"from":"mechanise", "to":"mechanize"},
            {"from":"mechanised", "to":"mechanized"},
            {"from":"mechanises", "to":"mechanizes"},
            {"from":"mechanising", "to":"mechanizing"},
            {"from":"mediaeval", "to":"medieval"},
            {"from":"memorialise", "to":"memorialize"},
            {"from":"memorialised", "to":"memorialized"},
            {"from":"memorialises", "to":"memorializes"},
            {"from":"memorialising", "to":"memorializing"},
            {"from":"memorise", "to":"memorize"},
            {"from":"memorised", "to":"memorized"},
            {"from":"memorises", "to":"memorizes"},
            {"from":"memorising", "to":"memorizing"},
            {"from":"mesmerise", "to":"mesmerize"},
            {"from":"mesmerised", "to":"mesmerized"},
            {"from":"mesmerises", "to":"mesmerizes"},
            {"from":"mesmerising", "to":"mesmerizing"},
            {"from":"metabolise", "to":"metabolize"},
            {"from":"metabolised", "to":"metabolized"},
            {"from":"metabolises", "to":"metabolizes"},
            {"from":"metabolising", "to":"metabolizing"},
            {"from":"metre", "to":"meter"},
            {"from":"metres", "to":"meters"},
            {"from":"micrometre", "to":"micrometer"},
            {"from":"micrometres", "to":"micrometers"},
            {"from":"militarise", "to":"militarize"},
            {"from":"militarised", "to":"militarized"},
            {"from":"militarises", "to":"militarizes"},
            {"from":"militarising", "to":"militarizing"},
            {"from":"milligramme", "to":"milligram"},
            {"from":"milligrammes", "to":"milligrams"},
            {"from":"millilitre", "to":"milliliter"},
            {"from":"millilitres", "to":"milliliters"},
            {"from":"millimetre", "to":"millimeter"},
            {"from":"millimetres", "to":"millimeters"},
            {"from":"miniaturisation", "to":"miniaturization"},
            {"from":"miniaturise", "to":"miniaturize"},
            {"from":"miniaturised", "to":"miniaturized"},
            {"from":"miniaturises", "to":"miniaturizes"},
            {"from":"miniaturising", "to":"miniaturizing"},
            {"from":"minibuses", "to":"minibusses"},
            {"from":"minimise", "to":"minimize"},
```


(continues on next page)

(continued from previous page)


```
            {"from":"minimised", "to":"minimized"},
            {"from":"minimises", "to":"minimizes"},
            {"from":"minimising", "to":"minimizing"},
            {"from":"misbehaviour", "to":"misbehavior"},
            {"from":"misdemeanour", "to":"misdemeanor"},
            {"from":"misdemeanours", "to":"misdemeanors"},
            {"from":"misspelt", "to":"misspelled"},
            {"from":"mitre", "to":"miter"},
            {"from":"mitres", "to":"miters"},
            {"from":"mobilisation", "to":"mobilization"},
            {"from":"mobilise", "to":"mobilize"},
            {"from":"mobilised", "to":"mobilized"},
            {"from":"mobilises", "to":"mobilizes"},
            {"from":"mobilising", "to":"mobilizing"},
            {"from":"modelled", "to":"modeled"},
            {"from":"modeller", "to":"modeler"},
            {"from":"modellers", "to":"modelers"},
            {"from":"modelling", "to":"modeling"},
            {"from":"modernise", "to":"modernize"},
            {"from":"modernised", "to":"modernized"},
            {"from":"modernises", "to":"modernizes"},
            {"from":"modernising", "to":"modernizing"},
            {"from":"moisturise", "to":"moisturize"},
            {"from":"moisturised", "to":"moisturized"},
            {"from":"moisturiser", "to":"moisturizer"},
            {"from":"moisturisers", "to":"moisturizers"},
            {"from":"moisturises", "to":"moisturizes"},
            {"from":"moisturising", "to":"moisturizing"},
            {"from":"monologue", "to":"monolog"},
            {"from":"monologues", "to":"monologs"},
            {"from":"monopolisation", "to":"monopolization"},
            {"from":"monopolise", "to":"monopolize"},
            {"from":"monopolised", "to":"monopolized"},
            {"from":"monopolises", "to":"monopolizes"},
            {"from":"monopolising", "to":"monopolizing"},
            {"from":"moralise", "to":"moralize"},
            {"from":"moralised", "to":"moralized"},
            {"from":"moralises", "to":"moralizes"},
            {"from":"moralising", "to":"moralizing"},
            {"from":"motorised", "to":"motorized"},
            {"from":"mould", "to":"mold"},
            {"from":"moulded", "to":"molded"},
            {"from":"moulder", "to":"molder"},
            {"from":"mouldered", "to":"moldered"},
            {"from":"mouldering", "to":"moldering"},
            {"from":"moulders", "to":"molders"},
            {"from":"mouldier", "to":"moldier"},
            {"from":"mouldiest", "to":"moldiest"},
            {"from":"moulding", "to":"molding"},
            {"from":"mouldings", "to":"moldings"},
            {"from":"moulds", "to":"molds"},
            {"from":"mouldy", "to":"moldy"},
            {"from":"moult", "to":"molt"},
            {"from":"moulted", "to":"molted"},
            {"from":"moulting", "to":"molting"},
            {"from":"moults", "to":"molts"},
            {"from":"moustache", "to":"mustache"},
            {"from":"moustached", "to":"mustached"},
            {"from":"moustaches", "to":"mustaches"},
            {"from":"moustachioed", "to":"mustachioed"},
            {"from":"multicoloured", "to":"multicolored"},
```


(continues on next page)

(continued from previous page)


```
            {"from":"nationalisation", "to":"nationalization"},
            {"from":"nationalisations", "to":"nationalizations"},
            {"from":"nationalise", "to":"nationalize"},
            {"from":"nationalised", "to":"nationalized"},
            {"from":"nationalises", "to":"nationalizes"},
            {"from":"nationalising", "to":"nationalizing"},
            {"from":"naturalisation", "to":"naturalization"},
            {"from":"naturalise", "to":"naturalize"},
            {"from":"naturalised", "to":"naturalized"},
            {"from":"naturalises", "to":"naturalizes"},
            {"from":"naturalising", "to":"naturalizing"},
            {"from":"neighbour", "to":"neighbor"},
            {"from":"neighbourhood", "to":"neighborhood"},
            {"from":"neighbourhoods", "to":"neighborhoods"},
            {"from":"neighbouring", "to":"neighboring"},
            {"from":"neighbourliness", "to":"neighborliness"},
            {"from":"neighbourly", "to":"neighborly"},
            {"from":"neighbours", "to":"neighbors"},
            {"from":"neutralisation", "to":"neutralization"},
            {"from":"neutralise", "to":"neutralize"},
            {"from":"neutralised", "to":"neutralized"},
            {"from":"neutralises", "to":"neutralizes"},
            {"from":"neutralising", "to":"neutralizing"},
            {"from":"normalisation", "to":"normalization"},
            {"from":"normalise", "to":"normalize"},
            {"from":"normalised", "to":"normalized"},
            {"from":"normalises", "to":"normalizes"},
            {"from":"normalising", "to":"normalizing"},
            {"from":"odour", "to":"odor"},
            {"from":"odourless", "to":"odorless"},
            {"from":"odours", "to":"odors"},
            {"from":"oesophagus", "to":"esophagus"},
            {"from":"oesophaguses", "to":"esophaguses"},
            {"from":"oestrogen", "to":"estrogen"},
            {"from":"offence", "to":"offense"},
            {"from":"offences", "to":"offenses"},
            {"from":"omelette", "to":"omelet"},
            {"from":"omelettes", "to":"omelets"},
            {"from":"optimise", "to":"optimize"},
            {"from":"optimised", "to":"optimized"},
            {"from":"optimises", "to":"optimizes"},
            {"from":"optimising", "to":"optimizing"},
            {"from":"organisation", "to":"organization"},
            {"from":"organisational", "to":"organizational"},
            {"from":"organisations", "to":"organizations"},
            {"from":"organise", "to":"organize"},
            {"from":"organised", "to":"organized"},
            {"from":"organiser", "to":"organizer"},
            {"from":"organisers", "to":"organizers"},
            {"from":"organises", "to":"organizes"},
            {"from":"organising", "to":"organizing"},
            {"from":"orthopaedic", "to":"orthopedic"},
            {"from":"orthopaedics", "to":"orthopedics"},
            {"from":"ostracise", "to":"ostracize"},
            {"from":"ostracised", "to":"ostracized"},
            {"from":"ostracises", "to":"ostracizes"},
            {"from":"ostracising", "to":"ostracizing"},
            {"from":"outmanoeuvre", "to":"outmaneuver"},
            {"from":"outmanoeuvred", "to":"outmaneuvered"},
            {"from":"outmanoeuvres", "to":"outmaneuvers"},
            {"from":"outmanoeuvring", "to":"outmaneuvering"},
```


(continues on next page)

(continued from previous page)


```
            {"from":"overemphasise", "to":"overemphasize"},
            {"from":"overemphasised", "to":"overemphasized"},
            {"from":"overemphasises", "to":"overemphasizes"},
            {"from":"overemphasising", "to":"overemphasizing"},
            {"from":"oxidisation", "to":"oxidization"},
            {"from":"oxidise", "to":"oxidize"},
            {"from":"oxidised", "to":"oxidized"},
            {"from":"oxidises", "to":"oxidizes"},
            {"from":"oxidising", "to":"oxidizing"},
            {"from":"paederast", "to":"pederast"},
            {"from":"paederasts", "to":"pederasts"},
            {"from":"paediatric", "to":"pediatric"},
            {"from":"paediatrician", "to":"pediatrician"},
            {"from":"paediatricians", "to":"pediatricians"},
            {"from":"paediatrics", "to":"pediatrics"},
            {"from":"paedophile", "to":"pedophile"},
            {"from":"paedophiles", "to":"pedophiles"},
            {"from":"paedophilia", "to":"pedophilia"},
            {"from":"palaeolithic", "to":"paleolithic"},
            {"from":"palaeontologist", "to":"paleontologist"},
            {"from":"palaeontologists", "to":"paleontologists"},
            {"from":"palaeontology", "to":"paleontology"},
            {"from":"panelled", "to":"paneled"},
            {"from":"panelling", "to":"paneling"},
            {"from":"panellist", "to":"panelist"},
            {"from":"panellists", "to":"panelists"},
            {"from":"paralyse", "to":"paralyze"},
            {"from":"paralysed", "to":"paralyzed"},
            {"from":"paralyses", "to":"paralyzes"},
            {"from":"paralysing", "to":"paralyzing"},
            {"from":"parcelled", "to":"parceled"},
            {"from":"parcelling", "to":"parceling"},
            {"from":"parlour", "to":"parlor"},
            {"from":"parlours", "to":"parlors"},
            {"from":"particularise", "to":"particularize"},
            {"from":"particularised", "to":"particularized"},
            {"from":"particularises", "to":"particularizes"},
            {"from":"particularising", "to":"particularizing"},
            {"from":"passivisation", "to":"passivization"},
            {"from":"passivise", "to":"passivize"},
            {"from":"passivised", "to":"passivized"},
            {"from":"passivises", "to":"passivizes"},
            {"from":"passivising", "to":"passivizing"},
            {"from":"pasteurisation", "to":"pasteurization"},
            {"from":"pasteurise", "to":"pasteurize"},
            {"from":"pasteurised", "to":"pasteurized"},
            {"from":"pasteurises", "to":"pasteurizes"},
            {"from":"pasteurising", "to":"pasteurizing"},
            {"from":"patronise", "to":"patronize"},
            {"from":"patronised", "to":"patronized"},
            {"from":"patronises", "to":"patronizes"},
            {"from":"patronising", "to":"patronizing"},
            {"from":"patronisingly", "to":"patronizingly"},
            {"from":"pedalled", "to":"pedaled"},
            {"from":"pedalling", "to":"pedaling"},
            {"from":"pedestrianisation", "to":"pedestrianization"},
            {"from":"pedestrianise", "to":"pedestrianize"},
            {"from":"pedestrianised", "to":"pedestrianized"},
            {"from":"pedestrianises", "to":"pedestrianizes"},
            {"from":"pedestrianising", "to":"pedestrianizing"},
            {"from":"penalise", "to":"penalize"},
```


(continues on next page)

(continued from previous page)


```
            {"from":"penalised", "to":"penalized"},
            {"from":"penalises", "to":"penalizes"},
            {"from":"penalising", "to":"penalizing"},
            {"from":"pencilled", "to":"penciled"},
            {"from":"pencilling", "to":"penciling"},
            {"from":"personalise", "to":"personalize"},
            {"from":"personalised", "to":"personalized"},
            {"from":"personalises", "to":"personalizes"},
            {"from":"personalising", "to":"personalizing"},
            {"from":"pharmacopoeia", "to":"pharmacopeia"},
            {"from":"pharmacopoeias", "to":"pharmacopeias"},
            {"from":"philosophise", "to":"philosophize"},
            {"from":"philosophised", "to":"philosophized"},
            {"from":"philosophises", "to":"philosophizes"},
            {"from":"philosophising", "to":"philosophizing"},
            {"from":"philtre", "to":"filter"},
            {"from":"philtres", "to":"filters"},
            {"from":"phoney", "to":"phony"},
            {"from":"plagiarise", "to":"plagiarize"},
            {"from":"plagiarised", "to":"plagiarized"},
            {"from":"plagiarises", "to":"plagiarizes"},
            {"from":"plagiarising", "to":"plagiarizing"},
            {"from":"plough", "to":"plow"},
            {"from":"ploughed", "to":"plowed"},
            {"from":"ploughing", "to":"plowing"},
            {"from":"ploughman", "to":"plowman"},
            {"from":"ploughmen", "to":"plowmen"},
            {"from":"ploughs", "to":"plows"},
            {"from":"ploughshare", "to":"plowshare"},
            {"from":"ploughshares", "to":"plowshares"},
            {"from":"polarisation", "to":"polarization"},
            {"from":"polarise", "to":"polarize"},
            {"from":"polarised", "to":"polarized"},
            {"from":"polarises", "to":"polarizes"},
            {"from":"polarising", "to":"polarizing"},
            {"from":"politicisation", "to":"politicization"},
            {"from":"politicise", "to":"politicize"},
            {"from":"politicised", "to":"politicized"},
            {"from":"politicises", "to":"politicizes"},
            {"from":"politicising", "to":"politicizing"},
            {"from":"popularisation", "to":"popularization"},
            {"from":"popularise", "to":"popularize"},
            {"from":"popularised", "to":"popularized"},
            {"from":"popularises", "to":"popularizes"},
            {"from":"popularising", "to":"popularizing"},
            {"from":"pouffe", "to":"pouf"},
            {"from":"pouffes", "to":"poufs"},
            {"from":"practise", "to":"practice"},
            {"from":"practised", "to":"practiced"},
            {"from":"practises", "to":"practices"},
            {"from":"practising", "to":"practicing"},
            {"from":"praesidium", "to":"presidium"},
            {"from":"praesidiums", "to":"presidiums"},
            {"from":"pressurisation", "to":"pressurization"},
            {"from":"pressurise", "to":"pressurize"},
            {"from":"pressurised", "to":"pressurized"},
            {"from":"pressurises", "to":"pressurizes"},
            {"from":"pressurising", "to":"pressurizing"},
            {"from":"pretence", "to":"pretense"},
            {"from":"pretences", "to":"pretenses"},
            {"from":"primaeval", "to":"primeval"},
```


(continues on next page)

(continued from previous page)


```
            {"from":"prioritisation", "to":"prioritization"},
            {"from":"prioritise", "to":"prioritize"},
            {"from":"prioritised", "to":"prioritized"},
            {"from":"prioritises", "to":"prioritizes"},
            {"from":"prioritising", "to":"prioritizing"},
            {"from":"privatisation", "to":"privatization"},
            {"from":"privatisations", "to":"privatizations"},
            {"from":"privatise", "to":"privatize"},
            {"from":"privatised", "to":"privatized"},
            {"from":"privatises", "to":"privatizes"},
            {"from":"privatising", "to":"privatizing"},
            {"from":"professionalisation", "to":"professionalization"},
            {"from":"professionalise", "to":"professionalize"},
            {"from":"professionalised", "to":"professionalized"},
            {"from":"professionalises", "to":"professionalizes"},
            {"from":"professionalising", "to":"professionalizing"},
            {"from":"programme", "to":"program"},
            {"from":"programmes", "to":"programs"},
            {"from":"prologue", "to":"prolog"},
            {"from":"prologues", "to":"prologs"},
            {"from":"propagandise", "to":"propagandize"},
            {"from":"propagandised", "to":"propagandized"},
            {"from":"propagandises", "to":"propagandizes"},
            {"from":"propagandising", "to":"propagandizing"},
            {"from":"proselytise", "to":"proselytize"},
            {"from":"proselytised", "to":"proselytized"},
            {"from":"proselytiser", "to":"proselytizer"},
            {"from":"proselytisers", "to":"proselytizers"},
            {"from":"proselytises", "to":"proselytizes"},
            {"from":"proselytising", "to":"proselytizing"},
            {"from":"psychoanalyse", "to":"psychoanalyze"},
            {"from":"psychoanalysed", "to":"psychoanalyzed"},
            {"from":"psychoanalyses", "to":"psychoanalyzes"},
            {"from":"psychoanalysing", "to":"psychoanalyzing"},
            {"from":"publicise", "to":"publicize"},
            {"from":"publicised", "to":"publicized"},
            {"from":"publicises", "to":"publicizes"},
            {"from":"publicising", "to":"publicizing"},
            {"from":"pulverisation", "to":"pulverization"},
            {"from":"pulverise", "to":"pulverize"},
            {"from":"pulverised", "to":"pulverized"},
            {"from":"pulverises", "to":"pulverizes"},
            {"from":"pulverising", "to":"pulverizing"},
            {"from":"pummelled", "to":"pummel"},
            {"from":"pummelling", "to":"pummeled"},
            {"from":"pyjama", "to":"pajama"},
            {"from":"pyjamas", "to":"pajamas"},
            {"from":"pzazz", "to":"pizzazz"},
            {"from":"quarrelled", "to":"quarreled"},
            {"from":"quarrelling", "to":"quarreling"},
            {"from":"radicalise", "to":"radicalize"},
            {"from":"radicalised", "to":"radicalized"},
            {"from":"radicalises", "to":"radicalizes"},
            {"from":"radicalising", "to":"radicalizing"},
            {"from":"rancour", "to":"rancor"},
            {"from":"randomise", "to":"randomize"},
            {"from":"randomised", "to":"randomized"},
            {"from":"randomises", "to":"randomizes"},
            {"from":"randomising", "to":"randomizing"},
            {"from":"rationalisation", "to":"rationalization"},
            {"from":"rationalisations", "to":"rationalizations"},
```


(continues on next page)

(continued from previous page)


```
            {"from":"rationalise", "to":"rationalize"},
            {"from":"rationalised", "to":"rationalized"},
            {"from":"rationalises", "to":"rationalizes"},
            {"from":"rationalising", "to":"rationalizing"},
            {"from":"ravelled", "to":"raveled"},
            {"from":"ravelling", "to":"raveling"},
            {"from":"realisable", "to":"realizable"},
            {"from":"realisation", "to":"realization"},
            {"from":"realisations", "to":"realizations"},
            {"from":"realise", "to":"realize"},
            {"from":"realised", "to":"realized"},
            {"from":"realises", "to":"realizes"},
            {"from":"realising", "to":"realizing"},
            {"from":"recognisable", "to":"recognizable"},
            {"from":"recognisably", "to":"recognizably"},
            {"from":"recognisance", "to":"recognizance"},
            {"from":"recognise", "to":"recognize"},
            {"from":"recognised", "to":"recognized"},
            {"from":"recognises", "to":"recognizes"},
            {"from":"recognising", "to":"recognizing"},
            {"from":"reconnoitre", "to":"reconnoiter"},
            {"from":"reconnoitred", "to":"reconnoitered"},
            {"from":"reconnoitres", "to":"reconnoiters"},
            {"from":"reconnoitring", "to":"reconnoitering"},
            {"from":"refuelled", "to":"refueled"},
            {"from":"refuelling", "to":"refueling"},
            {"from":"regularisation", "to":"regularization"},
            {"from":"regularise", "to":"regularize"},
            {"from":"regularised", "to":"regularized"},
            {"from":"regularises", "to":"regularizes"},
            {"from":"regularising", "to":"regularizing"},
            {"from":"remodelled", "to":"remodeled"},
            {"from":"remodelling", "to":"remodeling"},
            {"from":"remould", "to":"remold"},
            {"from":"remoulded", "to":"remolded"},
            {"from":"remoulding", "to":"remolding"},
            {"from":"remoulds", "to":"remolds"},
            {"from":"reorganisation", "to":"reorganization"},
            {"from":"reorganisations", "to":"reorganizations"},
            {"from":"reorganise", "to":"reorganize"},
            {"from":"reorganised", "to":"reorganized"},
            {"from":"reorganises", "to":"reorganizes"},
            {"from":"reorganising", "to":"reorganizing"},
            {"from":"revelled", "to":"reveled"},
            {"from":"reveller", "to":"reveler"},
            {"from":"revellers", "to":"revelers"},
            {"from":"revelling", "to":"reveling"},
            {"from":"revitalise", "to":"revitalize"},
            {"from":"revitalised", "to":"revitalized"},
            {"from":"revitalises", "to":"revitalizes"},
            {"from":"revitalising", "to":"revitalizing"},
            {"from":"revolutionise", "to":"revolutionize"},
            {"from":"revolutionised", "to":"revolutionized"},
            {"from":"revolutionises", "to":"revolutionizes"},
            {"from":"revolutionising", "to":"revolutionizing"},
            {"from":"rhapsodise", "to":"rhapsodize"},
            {"from":"rhapsodised", "to":"rhapsodized"},
            {"from":"rhapsodises", "to":"rhapsodizes"},
            {"from":"rhapsodising", "to":"rhapsodizing"},
            {"from":"rigour", "to":"rigor"},
            {"from":"rigours", "to":"rigors"},
```


(continues on next page)

(continued from previous page)


```
            {"from":"ritualised", "to":"ritualized"},
            {"from":"rivalled", "to":"rivaled"},
            {"from":"rivalling", "to":"rivaling"},
            {"from":"romanticise", "to":"romanticize"},
            {"from":"romanticised", "to":"romanticized"},
            {"from":"romanticises", "to":"romanticizes"},
            {"from":"romanticising", "to":"romanticizing"},
            {"from":"rumour", "to":"rumor"},
            {"from":"rumoured", "to":"rumored"},
            {"from":"rumours", "to":"rumors"},
            {"from":"sabre", "to":"saber"},
            {"from":"sabres", "to":"sabers"},
            {"from":"saltpetre", "to":"saltpeter"},
            {"from":"sanitise", "to":"sanitize"},
            {"from":"sanitised", "to":"sanitized"},
            {"from":"sanitises", "to":"sanitizes"},
            {"from":"sanitising", "to":"sanitizing"},
            {"from":"satirise", "to":"satirize"},
            {"from":"satirised", "to":"satirized"},
            {"from":"satirises", "to":"satirizes"},
            {"from":"satirising", "to":"satirizing"},
            {"from":"saviour", "to":"savior"},
            {"from":"saviours", "to":"saviors"},
            {"from":"savour", "to":"savor"},
            {"from":"savoured", "to":"savored"},
            {"from":"savouries", "to":"savories"},
            {"from":"savouring", "to":"savoring"},
            {"from":"savours", "to":"savors"},
            {"from":"savoury", "to":"savory"},
            {"from":"scandalise", "to":"scandalize"},
            {"from":"scandalised", "to":"scandalized"},
            {"from":"scandalises", "to":"scandalizes"},
            {"from":"scandalising", "to":"scandalizing"},
            {"from":"sceptic", "to":"skeptic"},
            {"from":"sceptical", "to":"skeptical"},
            {"from":"sceptically", "to":"skeptically"},
            {"from":"scepticism", "to":"skepticism"},
            {"from":"sceptics", "to":"skeptics"},
            {"from":"sceptre", "to":"scepter"},
            {"from":"sceptres", "to":"scepters"},
            {"from":"scrutinise", "to":"scrutinize"},
            {"from":"scrutinised", "to":"scrutinized"},
            {"from":"scrutinises", "to":"scrutinizes"},
            {"from":"scrutinising", "to":"scrutinizing"},
            {"from":"secularisation", "to":"secularization"},
            {"from":"secularise", "to":"secularize"},
            {"from":"secularised", "to":"secularized"},
            {"from":"secularises", "to":"secularizes"},
            {"from":"secularising", "to":"secularizing"},
            {"from":"sensationalise", "to":"sensationalize"},
            {"from":"sensationalised", "to":"sensationalized"},
            {"from":"sensationalises", "to":"sensationalizes"},
            {"from":"sensationalising", "to":"sensationalizing"},
            {"from":"sensitise", "to":"sensitize"},
            {"from":"sensitised", "to":"sensitized"},
            {"from":"sensitises", "to":"sensitizes"},
            {"from":"sensitising", "to":"sensitizing"},
            {"from":"sentimentalise", "to":"sentimentalize"},
            {"from":"sentimentalised", "to":"sentimentalized"},
            {"from":"sentimentalises", "to":"sentimentalizes"},
            {"from":"sentimentalising", "to":"sentimentalizing"},
```


(continues on next page)

(continued from previous page)


```
            {"from":"sepulchre", "to":"sepulcher"},
            {"from":"sepulchres", "to":"sepulchers"},
            {"from":"serialisation", "to":"serialization"},
            {"from":"serialisations", "to":"serializations"},
            {"from":"serialise", "to":"serialize"},
            {"from":"serialised", "to":"serialized"},
            {"from":"serialises", "to":"serializes"},
            {"from":"serialising", "to":"serializing"},
            {"from":"sermonise", "to":"sermonize"},
            {"from":"sermonised", "to":"sermonized"},
            {"from":"sermonises", "to":"sermonizes"},
            {"from":"sermonising", "to":"sermonizing"},
            {"from":"sheikh", "to":"sheik"},
            {"from":"shovelled", "to":"shoveled"},
            {"from":"shovelling", "to":"shoveling"},
            {"from":"shrivelled", "to":"shriveled"},
            {"from":"shrivelling", "to":"shriveling"},
            {"from":"signalise", "to":"signalize"},
            {"from":"signalised", "to":"signalized"},
            {"from":"signalises", "to":"signalizes"},
            {"from":"signalising", "to":"signalizing"},
            {"from":"signalled", "to":"signaled"},
            {"from":"signalling", "to":"signaling"},
            {"from":"smoulder", "to":"smolder"},
            {"from":"smouldered", "to":"smoldered"},
            {"from":"smouldering", "to":"smoldering"},
            {"from":"smoulders", "to":"smolders"},
            {"from":"snivelled", "to":"sniveled"},
            {"from":"snivelling", "to":"sniveling"},
            {"from":"snorkelled", "to":"snorkeled"},
            {"from":"snorkelling", "to":"snorkeling"},
            {"from":"snowplough", "to":"snowplow"},
            {"from":"snowploughs", "to":"snowplow"},
            {"from":"socialisation", "to":"socialization"},
            {"from":"socialise", "to":"socialize"},
            {"from":"socialised", "to":"socialized"},
            {"from":"socialises", "to":"socializes"},
            {"from":"socialising", "to":"socializing"},
            {"from":"sodomise", "to":"sodomize"},
            {"from":"sodomised", "to":"sodomized"},
            {"from":"sodomises", "to":"sodomizes"},
            {"from":"sodomising", "to":"sodomizing"},
            {"from":"solemnise", "to":"solemnize"},
            {"from":"solemnised", "to":"solemnized"},
            {"from":"solemnises", "to":"solemnizes"},
            {"from":"solemnising", "to":"solemnizing"},
            {"from":"sombre", "to":"somber"},
            {"from":"specialisation", "to":"specialization"},
            {"from":"specialisations", "to":"specializations"},
            {"from":"specialise", "to":"specialize"},
            {"from":"specialised", "to":"specialized"},
            {"from":"specialises", "to":"specializes"},
            {"from":"specialising", "to":"specializing"},
            {"from":"spectre", "to":"specter"},
            {"from":"spectres", "to":"specters"},
            {"from":"spiralled", "to":"spiraled"},
            {"from":"spiralling", "to":"spiraling"},
            {"from":"splendour", "to":"splendor"},
            {"from":"splendours", "to":"splendors"},
            {"from":"squirrelled", "to":"squirreled"},
            {"from":"squirrelling", "to":"squirreling"},
```


(continues on next page)

(continued from previous page)


```
            {"from":"stabilisation", "to":"stabilization"},
            {"from":"stabilise", "to":"stabilize"},
            {"from":"stabilised", "to":"stabilized"},
            {"from":"stabiliser", "to":"stabilizer"},
            {"from":"stabilisers", "to":"stabilizers"},
            {"from":"stabilises", "to":"stabilizes"},
            {"from":"stabilising", "to":"stabilizing"},
            {"from":"standardisation", "to":"standardization"},
            {"from":"standardise", "to":"standardize"},
            {"from":"standardised", "to":"standardized"},
            {"from":"standardises", "to":"standardizes"},
            {"from":"standardising", "to":"standardizing"},
            {"from":"stencilled", "to":"stenciled"},
            {"from":"stencilling", "to":"stenciling"},
            {"from":"sterilisation", "to":"sterilization"},
            {"from":"sterilisations", "to":"sterilizations"},
            {"from":"sterilise", "to":"sterilize"},
            {"from":"sterilised", "to":"sterilized"},
            {"from":"steriliser", "to":"sterilizer"},
            {"from":"sterilisers", "to":"sterilizers"},
            {"from":"sterilises", "to":"sterilizes"},
            {"from":"sterilising", "to":"sterilizing"},
            {"from":"stigmatisation", "to":"stigmatization"},
            {"from":"stigmatise", "to":"stigmatize"},
            {"from":"stigmatised", "to":"stigmatized"},
            {"from":"stigmatises", "to":"stigmatizes"},
            {"from":"stigmatising", "to":"stigmatizing"},
            {"from":"storey", "to":"story"},
            {"from":"storeys", "to":"stories"},
            {"from":"subsidisation", "to":"subsidization"},
            {"from":"subsidise", "to":"subsidize"},
            {"from":"subsidised", "to":"subsidized"},
            {"from":"subsidiser", "to":"subsidizer"},
            {"from":"subsidisers", "to":"subsidizers"},
            {"from":"subsidises", "to":"subsidizes"},
            {"from":"subsidising", "to":"subsidizing"},
            {"from":"succour", "to":"succor"},
            {"from":"succoured", "to":"succored"},
            {"from":"succouring", "to":"succoring"},
            {"from":"succours", "to":"succors"},
            {"from":"sulphate", "to":"sulfate"},
            {"from":"sulphates", "to":"sulfates"},
            {"from":"sulphide", "to":"sulfide"},
            {"from":"sulphides", "to":"sulfides"},
            {"from":"sulphur", "to":"sulfur"},
            {"from":"sulphurous", "to":"sulfurous"},
            {"from":"summarise", "to":"summarize"},
            {"from":"summarised", "to":"summarized"},
            {"from":"summarises", "to":"summarizes"},
            {"from":"summarising", "to":"summarizing"},
            {"from":"swivelled", "to":"swiveled"},
            {"from":"swivelling", "to":"swiveling"},
            {"from":"symbolise", "to":"symbolize"},
            {"from":"symbolised", "to":"symbolized"},
            {"from":"symbolises", "to":"symbolizes"},
            {"from":"symbolising", "to":"symbolizing"},
            {"from":"sympathise", "to":"sympathize"},
            {"from":"sympathised", "to":"sympathized"},
            {"from":"sympathiser", "to":"sympathizer"},
            {"from":"sympathisers", "to":"sympathizers"},
            {"from":"sympathises", "to":"sympathizes"},
```


(continues on next page)

(continued from previous page)


```
            {"from":"sympathising", "to":"sympathizing"},
            {"from":"synchronisation", "to":"synchronization"},
            {"from":"synchronise", "to":"synchronize"},
            {"from":"synchronised", "to":"synchronized"},
            {"from":"synchronises", "to":"synchronizes"},
            {"from":"synchronising", "to":"synchronizing"},
            {"from":"synthesise", "to":"synthesize"},
            {"from":"synthesised", "to":"synthesized"},
            {"from":"synthesiser", "to":"synthesizer"},
            {"from":"synthesisers", "to":"synthesizers"},
            {"from":"synthesises", "to":"synthesizes"},
            {"from":"synthesising", "to":"synthesizing"},
            {"from":"syphon", "to":"siphon"},
            {"from":"syphoned", "to":"siphoned"},
            {"from":"syphoning", "to":"siphoning"},
            {"from":"syphons", "to":"siphons"},
            {"from":"systematisation", "to":"systematization"},
            {"from":"systematise", "to":"systematize"},
            {"from":"systematised", "to":"systematized"},
            {"from":"systematises", "to":"systematizes"},
            {"from":"systematising", "to":"systematizing"},
            {"from":"tantalise", "to":"tantalize"},
            {"from":"tantalised", "to":"tantalized"},
            {"from":"tantalises", "to":"tantalizes"},
            {"from":"tantalising", "to":"tantalizing"},
            {"from":"tantalisingly", "to":"tantalizingly"},
            {"from":"tasselled", "to":"tasseled"},
            {"from":"technicolour", "to":"technicolor"},
            {"from":"temporise", "to":"temporize"},
            {"from":"temporised", "to":"temporized"},
            {"from":"temporises", "to":"temporizes"},
            {"from":"temporising", "to":"temporizing"},
            {"from":"tenderise", "to":"tenderize"},
            {"from":"tenderised", "to":"tenderized"},
            {"from":"tenderises", "to":"tenderizes"},
            {"from":"tenderising", "to":"tenderizing"},
            {"from":"terrorise", "to":"terrorize"},
            {"from":"terrorised", "to":"terrorized"},
            {"from":"terrorises", "to":"terrorizes"},
            {"from":"terrorising", "to":"terrorizing"},
            {"from":"theatre", "to":"theater"},
            {"from":"theatregoer", "to":"theatergoer"},
            {"from":"theatregoers", "to":"theatergoers"},
            {"from":"theatres", "to":"theaters"},
            {"from":"theorise", "to":"theorize"},
            {"from":"theorised", "to":"theorized"},
            {"from":"theorises", "to":"theorizes"},
            {"from":"theorising", "to":"theorizing"},
            {"from":"tonne", "to":"ton"},
            {"from":"tonnes", "to":"tons"},
            {"from":"towelled", "to":"toweled"},
            {"from":"towelling", "to":"toweling"},
            {"from":"toxaemia", "to":"toxemia"},
            {"from":"tranquillise", "to":"tranquilize"},
            {"from":"tranquillised", "to":"tranquilized"},
            {"from":"tranquilliser", "to":"tranquilizer"},
            {"from":"tranquillisers", "to":"tranquilizers"},
            {"from":"tranquillises", "to":"tranquilizes"},
            {"from":"tranquillising", "to":"tranquilizing"},
            {"from":"tranquillity", "to":"tranquility"},
            {"from":"tranquillize", "to":"tranquilize"},
```


(continues on next page)

(continued from previous page)


```
            {"from":"tranquillized", "to":"tranquilized"},
            {"from":"tranquillizer", "to":"tranquilizer"},
            {"from":"tranquillizers", "to":"tranquilizers"},
            {"from":"tranquillizes", "to":"tranquilizes"},
            {"from":"tranquillizing", "to":"tranquilizing"},
            {"from":"tranquilly", "to":"tranquility"},
            {"from":"transistorised", "to":"transistorized"},
            {"from":"traumatise", "to":"traumatize"},
            {"from":"traumatised", "to":"traumatized"},
            {"from":"traumatises", "to":"traumatizes"},
            {"from":"traumatising", "to":"traumatizing"},
            {"from":"travelled", "to":"traveled"},
            {"from":"traveller", "to":"traveler"},
            {"from":"travellers", "to":"travelers"},
            {"from":"travelling", "to":"traveling"},
            {"from":"travelogue", "to":"travelog"},
            {"from":"travelogues", "to":"travelogs"},
            {"from":"trialled", "to":"trialed"},
            {"from":"trialling", "to":"trialing"},
            {"from":"tricolour", "to":"tricolor"},
            {"from":"tricolours", "to":"tricolors"},
            {"from":"trivialise", "to":"trivialize"},
            {"from":"trivialised", "to":"trivialized"},
            {"from":"trivialises", "to":"trivializes"},
            {"from":"trivialising", "to":"trivializing"},
            {"from":"tumour", "to":"tumor"},
            {"from":"tumours", "to":"tumors"},
            {"from":"tunnelled", "to":"tunneled"},
            {"from":"tunnelling", "to":"tunneling"},
            {"from":"tyrannise", "to":"tyrannize"},
            {"from":"tyrannised", "to":"tyrannized"},
            {"from":"tyrannises", "to":"tyrannizes"},
            {"from":"tyrannising", "to":"tyrannizing"},
            {"from":"tyre", "to":"tire"},
            {"from":"tyres", "to":"tires"},
            {"from":"unauthorised", "to":"unauthorized"},
            {"from":"uncivilised", "to":"uncivilized"},
            {"from":"underutilised", "to":"underutilized"},
            {"from":"unequalled", "to":"unequaled"},
            {"from":"unfavourable", "to":"unfavorable"},
            {"from":"unfavourably", "to":"unfavorably"},
            {"from":"unionisation", "to":"unionization"},
            {"from":"unionise", "to":"unionize"},
            {"from":"unionised", "to":"unionized"},
            {"from":"unionises", "to":"unionizes"},
            {"from":"unionising", "to":"unionizing"},
            {"from":"unorganised", "to":"unorganized"},
            {"from":"unravelled", "to":"unraveled"},
            {"from":"unravelling", "to":"unraveling"},
            {"from":"unrecognisable", "to":"unrecognizable"},
            {"from":"unrecognised", "to":"unrecognized"},
            {"from":"unrivalled", "to":"unrivaled"},
            {"from":"unsavoury", "to":"unsavory"},
            {"from":"untrammelled", "to":"untrammeled"},
            {"from":"urbanisation", "to":"urbanization"},
            {"from":"urbanise", "to":"urbanize"},
            {"from":"urbanised", "to":"urbanized"},
            {"from":"urbanises", "to":"urbanizes"},
            {"from":"urbanising", "to":"urbanizing"},
            {"from":"utilisable", "to":"utilizable"},
            {"from":"utilisation", "to":"utilization"},
```


(continues on next page)

(continued from previous page)


```
            {"from":"utilise", "to":"utilize"},
            {"from":"utilised", "to":"utilized"},
            {"from":"utilises", "to":"utilizes"},
            {"from":"utilising", "to":"utilizing"},
            {"from":"valour", "to":"valor"},
            {"from":"vandalise", "to":"vandalize"},
            {"from":"vandalised", "to":"vandalized"},
            {"from":"vandalises", "to":"vandalizes"},
            {"from":"vandalising", "to":"vandalizing"},
            {"from":"vaporisation", "to":"vaporization"},
            {"from":"vaporise", "to":"vaporize"},
            {"from":"vaporised", "to":"vaporized"},
            {"from":"vaporises", "to":"vaporizes"},
            {"from":"vaporising", "to":"vaporizing"},
            {"from":"vapour", "to":"vapor"},
            {"from":"vapours", "to":"vapors"},
            {"from":"verbalise", "to":"verbalize"},
            {"from":"verbalised", "to":"verbalized"},
            {"from":"verbalises", "to":"verbalizes"},
            {"from":"verbalising", "to":"verbalizing"},
            {"from":"victimisation", "to":"victimization"},
            {"from":"victimise", "to":"victimize"},
            {"from":"victimised", "to":"victimized"},
            {"from":"victimises", "to":"victimizes"},
            {"from":"victimising", "to":"victimizing"},
            {"from":"videodisc", "to":"videodisk"},
            {"from":"videodiscs", "to":"videodisks"},
            {"from":"vigour", "to":"vigor"},
            {"from":"visualisation", "to":"visualization"},
            {"from":"visualisations", "to":"visualizations"},
            {"from":"visualise", "to":"visualize"},
            {"from":"visualised", "to":"visualized"},
            {"from":"visualises", "to":"visualizes"},
            {"from":"visualising", "to":"visualizing"},
            {"from":"vocalisation", "to":"vocalization"},
            {"from":"vocalisations", "to":"vocalizations"},
            {"from":"vocalise", "to":"vocalize"},
            {"from":"vocalised", "to":"vocalized"},
            {"from":"vocalises", "to":"vocalizes"},
            {"from":"vocalising", "to":"vocalizing"},
            {"from":"vulcanised", "to":"vulcanized"},
            {"from":"vulgarisation", "to":"vulgarization"},
            {"from":"vulgarise", "to":"vulgarize"},
            {"from":"vulgarised", "to":"vulgarized"},
            {"from":"vulgarises", "to":"vulgarizes"},
            {"from":"vulgarising", "to":"vulgarizing"},
            {"from":"waggon", "to":"wagon"},
            {"from":"waggons", "to":"wagons"},
            {"from":"watercolour", "to":"watercolor"},
            {"from":"watercolours", "to":"watercolors"},
            {"from":"weaselled", "to":"weaseled"},
            {"from":"weaselling", "to":"weaseling"},
            {"from":"westernisation", "to":"westernization"},
            {"from":"westernise", "to":"westernize"},
            {"from":"westernised", "to":"westernized"},
            {"from":"westernises", "to":"westernizes"},
            {"from":"westernising", "to":"westernizing"},
            {"from":"womanise", "to":"womanize"},
            {"from":"womanised", "to":"womanized"},
            {"from":"womaniser", "to":"womanizer"},
            {"from":"womanisers", "to":"womanizers"},
```


(continues on next page)

(continued from previous page)


```
            {"from":"womanises", "to":"womanizes"},
            {"from":"womanising", "to":"womanizing"},
            {"from":"woollen", "to":"woolen"},
            {"from":"woollens", "to":"woolens"},
            {"from":"woollies", "to":"woolies"},
            {"from":"woolly", "to":"wooly"},
            {"from":"worshipped", "to":"worshiped"},
            {"from":"worshipping", "to":"worshiping"},
            {"from":"worshipper", "to":"worshiper"},
            {"from":"yodelled", "to":"yodeled"},
            {"from":"yodelling", "to":"yodeling"},
            {"from":"yoghourt", "to":"yogurt"},
            {"from":"yoghourts", "to":"yogurts"},
            {"from":"yoghurt", "to":"yogurt"},
            {"from":"yoghurts", "to":"yogurts"}
        ],
        "typos":[
            {"misspelling":"accomodation","correct":"accommodation"},
            {"misspelling":"acommodation","correct":"accommodation"},
            {"misspelling":"acheive","correct":"achieve"},
            {"misspelling":"accross","correct":"across"},
            {"misspelling":"adress","correct":"address"},
            {"misspelling":"agressive","correct":"aggressive"},
            {"misspelling":"alot","correct":"a lot"},
            {"misspelling":"apparantly","correct":"apparently"},
            {"misspelling":"appearence","correct":"appearance"},
            {"misspelling":"arguement","correct":"argument"},
            {"misspelling":"assasination","correct":"assassination"},
            {"misspelling":"basicly","correct":"basically"},
            {"misspelling":"beggining","correct":"beginning"},
            {"misspelling":"beleive","correct":"believe"},
            {"misspelling":"bizzare","correct":"bizarre"},
            {"misspelling":"buisness","correct":"business"},
            {"misspelling":"carribean","correct":"caribbean"},
            {"misspelling":"chauffer","correct":"chauffeur"},
            {"misspelling":"cemetary","correct":"cemetery"},
            {"misspelling":"collegue","correct":"colleague"},
            {"misspelling":"commitee","correct":"committee"},
            {"misspelling":"committment","correct":"commitment"},
            {"misspelling":"completly","correct":"completely"},
            {"misspelling":"concious","correct":"conscious"},
            {"misspelling":"copywrite","correct":"copyright"},
            {"misspelling":"curiousity","correct":"curiosity"},
            {"misspelling":"decaffinated","correct":"decaffeinated"},
            {"misspelling":"definately","correct":"definitely"},
            {"misspelling":"dependance","correct":"dependence"},
            {"misspelling":"desireable","correct":"desirable"},
            {"misspelling":"diarhea","correct":"diarrhoea"},
            {"misspelling":"dissapoint","correct":"disappoint"},
            {"misspelling":"dissapear","correct":"disappear"},
            {"misspelling":"dispell","correct":"dispel"},
            {"misspelling":"ecstacy","correct":"ecstasy"},
            {"misspelling":"embarass","correct":"embarrass"},
            {"misspelling":"enviroment","correct":"environment"},
            {"misspelling":"Farenheit","correct":"Fahrenheit"},
            {"misspelling":"febuary","correct":"february"},
            {"misspelling":"finaly","correct":"finally"},
            {"misspelling":"fluoroscent","correct":"fluorescent"},
            {"misspelling":"flouride","correct":"fluoride"},
            {"misspelling":"foriegn","correct":"foreign"},
            {"misspelling":"forteen","correct":"fourteen"},
```


(continues on next page)

(continued from previous page)


```
        {"misspelling":"fourty","correct":"forty"},
        {"misspelling":"freind","correct":"friend"},
        {"misspelling":"geneology","correct":"genealogy"},
        {"misspelling":"glamourous","correct":"glamorous"},
        {"misspelling":"goverment","correct":"government"},
        {"misspelling":"grammer","correct":"grammar"},
        {"misspelling":"happend","correct":"happened"},
        {"misspelling":"hemorage","correct":"haemorrhage"},
        {"misspelling":"heros","correct":"heroes"},
        {"misspelling":"hight","correct":"height"},
        {"misspelling":"humourous","correct":"humorous"},
        {"misspelling":"hygeine","correct":"hygiene"},
        {"misspelling":"idiosyncracy","correct":"idiosyncrasy"},
        {"misspelling":"independance","correct":"independence"},
        {"misspelling":"interupt","correct":"interrupt"},
        {"misspelling":"intresting","correct":"interesting"},
        {"misspelling":"juge","correct":"judge"},
        {"misspelling":"knowlege","correct":"knowledge"},
        {"misspelling":"lazer","correct":"laser"},
        {"misspelling":"liason","correct":"liaison"},
        {"misspelling":"libary","correct":"library"},
        {"misspelling":"lightening","correct":"lightning"},
        {"misspelling":"lollypop","correct":"lollipop"},
        {"misspelling":"millenium","correct":"millennium"},
        {"misspelling":"mischievious","correct":"mischievous"},
        {"misspelling":"mispell","correct":"misspell"},
        {"misspelling":"monkies","correct":"monkeys"},
        {"misspelling":"morgage","correct":"mortgage"},
        {"misspelling":"neccessary","correct":"necessary"},
        {"misspelling":"neice","correct":"niece"},
        {"misspelling":"noone","correct":"no one"},
        {"misspelling":"noticable","correct":"noticeable"},
        {"misspelling":"occassion","correct":"occasion"},
        {"misspelling":"occured","correct":"occurred"},
        {"misspelling":"oppurtunity","correct":"opportunity"},
        {"misspelling":"paralell","correct":"parallel"},
        {"misspelling":"pasttime","correct":"pastime"},
        {"misspelling":"peice","correct":"piece"},
        {"misspelling":"persistant","correct":"persistent"},
        {"misspelling":"persue","correct":"pursue"},
        {"misspelling":"pharoah","correct":"pharaoh"},
        {"misspelling":"portugese","correct":"portuguese"},
        {"misspelling":"posession","correct":"possession"},
        {"misspelling":"potatoe","correct":"potato"},
        {"misspelling":"preceeding","correct":"preceding"},
        {"misspelling":"prefered","correct":"preferred"},
        {"misspelling":"pronounciation","correct":"pronunciation"},
        {"misspelling":"propoganda","correct":"propaganda"},
        {"misspelling":"privelige","correct":"privilege"},
        {"misspelling":"publically","correct":"publicly"},
        {"misspelling":"rasberry","correct":"raspberry"},
        {"misspelling":"recieve","correct":"receive"},
        {"misspelling":"reccomend","correct":"recommend"},
        {"misspelling":"rythm","correct":"rhythm"},
        {"misspelling":"shedule","correct":"schedule"},
        {"misspelling":"seige","correct":"siege"},
        {"misspelling":"sentance","correct":"sentence"},
        {"misspelling":"seperate","correct":"separate"},
        {"misspelling":"sieze","correct":"seize"},
        {"misspelling":"sincerly","correct":"sincerely"},
        {"misspelling":"supercede","correct":"supersede"},
```


(continues on next page)

(continued from previous page)

```
            {"misspelling":"suprise","correct":"surprise"},
            {"misspelling":"tatoo","correct":"tattoo"},
            {"misspelling":"tendancy","correct":"tendency"},
            {"misspelling":"thier","correct":"their"},
            {"misspelling":"threshhold","correct":"threshold"},
            {"misspelling":"tommorrow","correct":"tomorrow"},
            {"misspelling":"truely","correct":"truly"},
            {"misspelling":"untill","correct":"until"},
            {"misspelling":"vaccuum","correct":"vacuum"},
            {"misspelling":"vegeterian","correct":"vegetarian"},
            {"misspelling":"wendesday","correct":"wednesday"},
            {"misspelling":"whereever","correct":"wherever"},
            {"misspelling":"wierd","correct":"weird"},
            {"misspelling":"writen","correct":"written"}
        ]
    }
}
```

Configure tokenizer logger

Logger is configuration at top level of json in *logger* field.

Example of Configuration:

```
{
    "logger": {
        "logging-file": "test.log",
        "logging-path": "/tmp",
        "logging-level": {"file": "info", "screen": "debug"}
    }
}
```

The logger fields are:

- **logging-file** is the filename of the log file (notice that "-<logname>" will be added to this name=
- **logging-path** is the path to the logfile (if it does not exist it will be created)
- **logging-level** contains two fields:
  - **file** for the logging level of the file
  - **screen** for the logging level on screen output

  Both can be set to the following values:
  - **debug** for the debug level and developper information
  - **info** for the level of information
  - **warning** to display only warning and errors
  - **error** to display only error

Configure tokenizer Network

Example of Configuration:

```
{
    "network": {
        "host":"0.0.0.0",
        "port":8080,
        "associate-environment": {
            "host":"HOST_ENVNAME",
            "port":"PORT_ENVNAME"
        }
    }
}
```

The network fields:

- **host** : hostname
- **port** : port of the service
- **associated-environement** : is the "host" and "port" associated environment variables that allows to replace the default one. This field is not mandatory.
  - "host" : associated "host" environment variable
  - "port" : associated "port" environment variable

Configure tokenizer Serialize

Example of Configuration:

```
{
    "serialize":{
        "input":{
            "path":"/tmp",
            "keep-service-info":true
        },
        "output":{
            "path":"/tmp",
            "keep-service-info":true
        }
    }
}
```

The serialize fields:

- **input** : serialize input of service
  - **path** : path of serialized fields
  - **keep-service-info** : True if serialize info is kept
  - **associated-environement** : is the "path" and "keep-service-info" associated environment variables that allows to replace the default one. This field is not mandatory.
    - **path** : associated "path" environment variable
    - **keep-service-info** : associated "keep-service-info" environment variable
- output : serialize output of service
  - **path** : path of serialized fields
  - **keep-service-info** : True if serialize info is kept

  - **associated-environement** : if the "path" and "keep-service-info" associated environment variables that allows to replace the default one. This field is not mandatory.
    * **path** : associated "path" environment variable
    * **keep-service-info** : associated "keep-service-info" environment variable

Configure tokenizer runtime

Example of Configuration:

```
{
    "runtime":{
        "request-max-size":100000000,
        "request-buffer-queue-size":100,
        "keep-alive":true,
        "keep-alive-timeout":5,
        "graceful-shutown-timeout":15.0,
        "request-timeout":60,
        "response-timeout":60,
        "workers":1
    }
}
```

The Runtime fields:

- **request-max-size** : how big a request may be (bytes)
- **request-buffer-queue-size**: request streaming buffer queue size
- **request-timeout** : how long a request can take to arrive (sec)
- **response-timeout** : how long a response can take to process (sec)
- **keep-alive**: keep-alive
- **keep-alive-timeout**: how long to hold a TCP connection open (sec)
- **graceful-shutdown_timeout** : how long to wait to force close non-idle connection (sec)
- **workers** : number of workers for the service on a node
- **associated-environement** : if one of previous field is on the associated environment variables that allows to replace the default one. This field is not mandatory.
  - **request-max-size** : overwrite with environement variable
  - **request-buffer-queue-size**: overwrite with environement variable
  - **request-timeout** : overwrite with environement variable
  - **response-timeout** : overwrite with environement variable
  - **keep-alive**: overwrite with environement variable
  - **keep-alive-timeout**: overwrite with environement variable
  - **graceful-shutdown_timeout** : overwrite with environement variable
  - **workers** : overwrite with environement variable

Tokenizer service

To create these resources simply run

```
python3 thot/tasks/tokenizer/createAnnotationResource.py --entries=/home/searchai_
↪svc/searchai/configs/default/configs/annotation-resources.json --output=/home/
↪searchai_svc/searchai/configs/default/resources/modeling/tokenizer/en/searchai_
↪mwe.pkl
```

To run the command type simply from searchai directory:

```
python3 thot/tokenizer_svc.py --config=<path to tokenizer configuration file>
```

A light client can be run through the command

```
python3 thot/tokenizer_client.py --config=<path to tokenizer configuration file> --
↪input=<input directory> --output=<output directory>
```

Tokenizer Tests

The converter service come with unit and functional testing.

Tokenizer Unit tests

Unittest allows to test Tokenizer classes only.

```
python3 -m unittest thot/tests/unittests/TestTokenizerConfiguration.py
python3 -m unittest thot/tests/unittests/TestTokenizer.py
```

Notes:

- if there is error due to the file **searchai_mwe.mkl** it is normal. You can avoid this error by creating the

the resources model

- the model data directory is mapped into docker-compose file, please check if all the configuration files are inside this directory

Tokenizer Functional tests

```
python3 -m unittest thot/tests/functional_tests/TestTokenizerSvc.py
```

Morphosyntactic tagger

The Morphosyntactic Tagger is a tool allowing to tag the fields “title_tokens” and “content_tokens” of the searchai document. This tools is a rest service. Morphosyntactic tagger is based on spacy toolbox.

Morphosyntactic tagger API

POST /api/mstagger/run
run the morphosyntactic tagger

Run the morphosyntactic tagger according to input. It return searchai representation

Request JSON Object

- **config** (*string*) – service configuration file
- **date** (*string*) – release date of the service
- **info** (*string*) – service info id
- **result.content** (*string*) – nested array of text representing the content of the document
- **result.content_morphosyntax[].lemma** (*string*) – lemma of token - the form
- **result.content_morphosyntax[].pos** (*string*) – Part Of Speech
- **result.content_ner[].end** (*integer*) – end offset of the token of the named entity
- **result.content_ner[].label** (*string*) – label of the named entity
- **result.content_ner[].start** (*integer*) – start offset of the token of the named entity
- **result.content_ner[].text** (*string*) – text of the named entity
- **result.content_tokens[]** (*string*) – a token, possibly MWE
- **result.data_source** (*string*) – source of the document (could be host file system, web, ...
- **result.embeddings[].content** (*string*) – text of the sentence
- **result.embeddings[].embeding[]** (*integer*) – embedding feature
- **result.embeddings[].field** (*string*) – Field from which the embedding is computed (can be empty)
- **result.embeddings[].position.length** (*integer*) – length of the sentence
- **result.embeddings[].position.start** (*integer*) – start offset of the sentence
- **result.kg[].property.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].property.content** (*string*) – Content of triple item
- **result.kg[].property.label** (*string*) – label of the content
- **result.kg[].property.lemma_content** (*string*) – Lemmatized content
- **result.kg[].property.pos[]** (*string*) – POS tag
- **result.kg[].property.positions[]** (*integer*) – Position in document
- **result.kg[].subject.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].subject.content** (*string*) – Content of triple item
- **result.kg[].subject.label** (*string*) – label of the content
- **result.kg[].subject.lemma_content** (*string*) – Lemmatized content

- **result.kg[].subject.pos[]** (*string*) – POS tag
- **result.kg[].subject.positions[]** (*integer*) – Position in document
- **result.kg[].value.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].value.content** (*string*) – Content of triple item
- **result.kg[].value.label** (*string*) – label of the content
- **result.kg[].value.lemma_content** (*string*) – Lemmatized content
- **result.kg[].value.pos[]** (*string*) – POS tag
- **result.kg[].value.positions[]** (*integer*) – Position in document
- **result.source_doc_id** (*string*) – uniq id of the document
- **result.title** (*string*) – title of document
- **result.title_morphosyntax[].lemma** (*string*) – lemma of token - the form
- **result.title_morphosyntax[].pos** (*string*) – Part Of Speech
- **result.title_ner[].end** (*integer*) – end offset of the token of the named entity
- **result.title_ner[].label** (*string*) – label of the named entity
- **result.title_ner[].start** (*integer*) – start offset of the token of the named entity
- **result.title_ner[].text** (*string*) – text of the named entity
- **result.title_tokens[]** (*string*) – a token, possibly MWE
- **version** (*string*) – version of the service

Status Codes

- 200 OK – return the searchai doc with tokens
- 422 Unprocessable Entity – Something wrong on the request parameters
- 500 Internal Server Error – Internal Error, and exception occured

Response JSON Object

- **config** (*string*) – service configuration file
- **date** (*string*) – release date of the service
- **info** (*string*) – service info id
- **result.content** (*string*) – nested array of text representing the content of the document
- **result.content_morphosyntax[].lemma** (*string*) – lemma of token - the form
- **result.content_morphosyntax[].pos** (*string*) – Part Of Speech
- **result.content_ner[].end** (*integer*) – end offset of the token of the named entity
- **result.content_ner[].label** (*string*) – label of the named entity
- **result.content_ner[].start** (*integer*) – start offset of the token of the named entity
- **result.content_ner[].text** (*string*) – text of the named entity
- **result.content_tokens[]** (*string*) – a token, possibly MWE

- **result.data_source** (*string*) – source of the document (could be host file system, web, ...
- **result.embeddings[].content** (*string*) – text of the sentence
- **result.embeddings[].embeding[]** (*integer*) – embedding feature
- **result.embeddings[].field** (*string*) – Field from which the embedding is computed (can be empty)
- **result.embeddings[].position.length** (*integer*) – length of the sentence
- **result.embeddings[].position.start** (*integer*) – start offset of the sentence
- **result.kg[].property.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].property.content** (*string*) – Content of triple item
- **result.kg[].property.label** (*string*) – label of the content
- **result.kg[].property.lemma_content** (*string*) – Lemmatized content
- **result.kg[].property.pos[]** (*string*) – POS tag
- **result.kg[].property.positions[]** (*integer*) – Position in document
- **result.kg[].subject.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].subject.content** (*string*) – Content of triple item
- **result.kg[].subject.label** (*string*) – label of the content
- **result.kg[].subject.lemma_content** (*string*) – Lemmatized content
- **result.kg[].subject.pos[]** (*string*) – POS tag
- **result.kg[].subject.positions[]** (*integer*) – Position in document
- **result.kg[].value.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].value.content** (*string*) – Content of triple item
- **result.kg[].value.label** (*string*) – label of the content
- **result.kg[].value.lemma_content** (*string*) – Lemmatized content
- **result.kg[].value.pos[]** (*string*) – POS tag
- **result.kg[].value.positions[]** (*integer*) – Position in document
- **result.source_doc_id** (*string*) – uniq id of the document
- **result.title** (*string*) – title of document
- **result.title_morphosyntax[].lemma** (*string*) – lemma of token - the form
- **result.title_morphosyntax[].pos** (*string*) – Part Of Speech
- **result.title_ner[].end** (*integer*) – end offset of the token of the named entity
- **result.title_ner[].label** (*string*) – label of the named entity
- **result.title_ner[].start** (*integer*) – start offset of the token of the named entity
- **result.title_ner[].text** (*string*) – text of the named entity

- **result.title_tokens[]** (*string*) – a token, possibly MWE
- **version** (*string*) – version of the service

OPTIONS /api/mstagger/run
run the morphosyntactic tagger

Run the morphosyntactic tagger according to input. It return searchai representation

Request JSON Object

- **config** (*string*) – service configuration file
- **date** (*string*) – release date of the service
- **info** (*string*) – service info id
- **result.content** (*string*) – nested array of text representing the content of the document
- **result.content_morphosyntax[].lemma** (*string*) – lemma of token - the form
- **result.content_morphosyntax[].pos** (*string*) – Part Of Speech
- **result.content_ner[].end** (*integer*) – end offset of the token of the named entity
- **result.content_ner[].label** (*string*) – label of the named entity
- **result.content_ner[].start** (*integer*) – start offset of the token of the named entity
- **result.content_ner[].text** (*string*) – text of the named entity
- **result.content_tokens[]** (*string*) – a token, possibly MWE
- **result.data_source** (*string*) – source of the document (could be host file system, web, ...
- **result.embeddings[].content** (*string*) – text of the sentence
- **result.embeddings[].embeding[]** (*integer*) – embedding feature
- **result.embeddings[].field** (*string*) – Field from which the embedding is computed (can be empty)
- **result.embeddings[].position.length** (*integer*) – length of the sentence
- **result.embeddings[].position.start** (*integer*) – start offset of the sentence
- **result.kg[].property.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].property.content** (*string*) – Content of triple item
- **result.kg[].property.label** (*string*) – label of the content
- **result.kg[].property.lemma_content** (*string*) – Lemmatized content
- **result.kg[].property.pos[]** (*string*) – POS tag
- **result.kg[].property.positions[]** (*integer*) – Position in document
- **result.kg[].subject.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].subject.content** (*string*) – Content of triple item
- **result.kg[].subject.label** (*string*) – label of the content
- **result.kg[].subject.lemma_content** (*string*) – Lemmatized content

- **result.kg[].subject.pos[]** (*string*) – POS tag
- **result.kg[].subject.positions[]** (*integer*) – Position in document
- **result.kg[].value.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].value.content** (*string*) – Content of triple item
- **result.kg[].value.label** (*string*) – label of the content
- **result.kg[].value.lemma_content** (*string*) – Lemmatized content
- **result.kg[].value.pos[]** (*string*) – POS tag
- **result.kg[].value.positions[]** (*integer*) – Position in document
- **result.source_doc_id** (*string*) – uniq id of the document
- **result.title** (*string*) – title of document
- **result.title_morphosyntax[].lemma** (*string*) – lemma of token - the form
- **result.title_morphosyntax[].pos** (*string*) – Part Of Speech
- **result.title_ner[].end** (*integer*) – end offset of the token of the named entity
- **result.title_ner[].label** (*string*) – label of the named entity
- **result.title_ner[].start** (*integer*) – start offset of the token of the named entity
- **result.title_ner[].text** (*string*) – text of the named entity
- **result.title_tokens[]** (*string*) – a token, possibly MWE
- **version** (*string*) – version of the service

Status Codes

- 200 OK – return the searchai doc with tokens
- 422 Unprocessable Entity – Something wrong on the request parameters
- 500 Internal Server Error – Internal Error, and exception occured

Response JSON Object

- **config** (*string*) – service configuration file
- **date** (*string*) – release date of the service
- **info** (*string*) – service info id
- **result.content** (*string*) – nested array of text representing the content of the document
- **result.content_morphosyntax[].lemma** (*string*) – lemma of token - the form
- **result.content_morphosyntax[].pos** (*string*) – Part Of Speech
- **result.content_ner[].end** (*integer*) – end offset of the token of the named entity
- **result.content_ner[].label** (*string*) – label of the named entity
- **result.content_ner[].start** (*integer*) – start offset of the token of the named entity
- **result.content_ner[].text** (*string*) – text of the named entity
- **result.content_tokens[]** (*string*) – a token, possibly MWE

- **result.data_source** (*string*) – source of the document (could be host file system, web, ...
- **result.embeddings[].content** (*string*) – text of the sentence
- **result.embeddings[].embeding[]** (*integer*) – embedding feature
- **result.embeddings[].field** (*string*) – Field from which the embedding is computed (can be empty)
- **result.embeddings[].position.length** (*integer*) – length of the sentence
- **result.embeddings[].position.start** (*integer*) – start offset of the sentence
- **result.kg[].property.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].property.content** (*string*) – Content of triple item
- **result.kg[].property.label** (*string*) – label of the content
- **result.kg[].property.lemma_content** (*string*) – Lemmatized content
- **result.kg[].property.pos[]** (*string*) – POS tag
- **result.kg[].property.positions[]** (*integer*) – Position in document
- **result.kg[].subject.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].subject.content** (*string*) – Content of triple item
- **result.kg[].subject.label** (*string*) – label of the content
- **result.kg[].subject.lemma_content** (*string*) – Lemmatized content
- **result.kg[].subject.pos[]** (*string*) – POS tag
- **result.kg[].subject.positions[]** (*integer*) – Position in document
- **result.kg[].value.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].value.content** (*string*) – Content of triple item
- **result.kg[].value.label** (*string*) – label of the content
- **result.kg[].value.lemma_content** (*string*) – Lemmatized content
- **result.kg[].value.pos[]** (*string*) – POS tag
- **result.kg[].value.positions[]** (*integer*) – Position in document
- **result.source_doc_id** (*string*) – uniq id of the document
- **result.title** (*string*) – title of document
- **result.title_morphosyntax[].lemma** (*string*) – lemma of token - the form
- **result.title_morphosyntax[].pos** (*string*) – Part Of Speech
- **result.title_ner[].end** (*integer*) – end offset of the token of the named entity
- **result.title_ner[].label** (*string*) – label of the named entity
- **result.title_ner[].start** (*integer*) – start offset of the token of the named entity
- **result.title_ner[].text** (*string*) – text of the named entity

- **result.title_tokens[]** (*string*) – a token, possibly MWE
- **version** (*string*) – version of the service

GET /api/mstagger/health
Ask the health of the service

This function allows to know if the service is up and healthy. It is use by client to see if they can use it.

Status Codes

- 200 OK – Service successfuly loaded
- 500 Internal Server Error – Service is not loaded

Response JSON Object

- **config** (*string*) – configuration file path
- **date** (*string*) – service development date
- **health** (*string*) – ok if service is healthy
- **info** (*string*) – service uniq id
- **version** (*string*) – version of the service

POST /api/mstagger/health
Ask the health of the service

This function allows to know if the service is up and healthy. It is use by client to see if they can use it.

Status Codes

- 200 OK – Service successfuly loaded
- 500 Internal Server Error – Service is not loaded

Response JSON Object

- **config** (*string*) – configuration file path
- **date** (*string*) – service development date
- **health** (*string*) – ok if service is healthy
- **info** (*string*) – service uniq id
- **version** (*string*) – version of the service

OPTIONS /api/mstagger/health
Ask the health of the service

This function allows to know if the service is up and healthy. It is use by client to see if they can use it.

Status Codes

- 200 OK – Service successfuly loaded
- 500 Internal Server Error – Service is not loaded

Response JSON Object

- **config** (*string*) – configuration file path
- **date** (*string*) – service development date
- **health** (*string*) – ok if service is healthy
- **info** (*string*) – service uniq id
- **version** (*string*) – version of the service

This API is also available via the service itself on http://<service host>:<service port>/swagger

Morphosyntactic tagger configuration

Example of Configuration:

```
{
    "logger": {
        "logging-file": "test.log",
        "logging-path": "/tmp",
        "logging-level": {"file": "info", "screen": "debug"}
    },
    "morphosyntax": {
        "taggers":[{
            "language":"en",
            "resources-base-path":"/home/searchai_svc/searchai/configs/default/
↪resources/modeling/tokenizer/en",
            "mwe": "searchai_mwe.pkl",
            "pre-sentencizer": true,
            "pre-tagging-with-concept":true,
            "add-concept-in-knowledge-graph":true
        }],
        "network": {
            "host":"0.0.0.0",
            "port":10002,
            "associate-environment": {
                "host":"MSTAGGER_HOST",
                "port":"MSTAGGER_PORT"
            }
        },
        "runtime":{
            "request-max-size":100000000,
            "request-buffer-queue-size":100,
            "keep-alive":true,
            "keep-alive-timeout":500,
            "graceful-shutown-timeout":15.0,
            "request-timeout":600,
            "response-timeout":600,
            "workers":1
        },
        "serialize":{
            "input":{
                "path":"/tmp",
                "keep-service-info":true
            },
            "output":{
                "path":"/tmp",
                "keep-service-info":true
            }
        }

    }
}
```

Morphosyntactic tagger is an aggreation of network configuration, serialize configuration, runtime configuration (in field converter), logger (at top level). The segmenter configuration is a table containing path to Multiple Word Expression entries (MWE):

- **language** :the language of tokenizer
- **resources-base-path**: the path to the resources (containing file created by tools *createAnnotationResources.py*
- **mwe** : the file containing MWE entries

Configure Morphosyntactic tagger logger

Logger is configuration at top level of json in *logger* field.

Example of Configuration:

```
{
    "logger": {
        "logging-file": "test.log",
        "logging-path": "/tmp",
        "logging-level": {"file": "info", "screen": "debug"}
    }
}
```

The logger fields are:

- **logging-file** is the filename of the log file (notice that "-<logname>" will be added to this name=
- **logging-path** is the path to the logfile (if it does not exist it will be created)
- **logging-level** contains two fields:
  - **file** for the logging level of the file
  - **screen** for the logging level on screen output

  Both can be set to the following values:
  - **debug** for the debug level and developper information
  - **info** for the level of information
  - **warning** to display only warning and errors
  - **error** to display only error

Configure Morphosyntactic tagger Network

Example of Configuration:

```
{
    "network": {
        "host":"0.0.0.0",
        "port":8080,
        "associate-environment": {
            "host":"HOST_ENVNAME",
            "port":"PORT_ENVNAME"
        }
    }
}
```

The network fields:

- **host** : hostname
- **port** : port of the service
- **associated-environement** : is the "host" and "port" associated environment variables that allows to replace the default one. This field is not mandatory.
  - "host" : associated "host" environment variable
  - "port" : associated "port" environment variable

Configure Morphosyntactic tagger Serialize

Example of Configuration:

```
{
    "serialize":{
        "input":{
            "path":"/tmp",
            "keep-service-info":true
        },
        "output":{
            "path":"/tmp",
            "keep-service-info":true
        }
    }
}
```

The serialize fields:

- **input** : serialize input of service
  - **path** : path of serialized fields
  - **keep-service-info** : True if serialize info is kept
  - **associated-environement** : is the "path" and "keep-service-info" associated environment variables that allows to replace the default one. This field is not mandatory.
    - **path** : associated "path" environment variable
    - **keep-service-info** : associated "keep-service-info" environment variable
- output : serialize output of service
  - **path** : path of serialized fields
  - **keep-service-info** : True if serialize info is kept
  - **associated-environement** : if the "path" and "keep-service-info" associated environment variables that allows to replace the default one. This field is not mandatory.
    - **path** : associated "path" environment variable
    - **keep-service-info** : associated "keep-service-info" environment variable

Configure Morphosyntactic tagger runtime

Example of Configuration:

```
{
    "runtime":{
        "request-max-size":100000000,
        "request-buffer-queue-size":100,
        "keep-alive":true,
        "keep-alive-timeout":5,
        "graceful-shutown-timeout":15.0,
        "request-timeout":60,
        "response-timeout":60,
        "workers":1
    }
}
```

The Runtime fields:

- **request-max-size** : how big a request may be (bytes)

- **request-buffer-queue-size**: request streaming buffer queue size
- **request-timeout** : how long a request can take to arrive (sec)
- **response-timeout** : how long a response can take to process (sec)
- **keep-alive**: keep-alive
- **keep-alive-timeout**: how long to hold a TCP connection open (sec)
- **graceful-shutdown_timeout** : how long to wait to force close non-idle connection (sec)
- **workers** : number of workers for the service on a node
- **associated-environement** : if one of previous field is on the associated environment variables that allows to replace the default one. This field is not mandatory.
  - **request-max-size** : overwrite with environement variable
  - **request-buffer-queue-size**: overwrite with environement variable
  - **request-timeout** : overwrite with environement variable
  - **response-timeout** : overwrite with environement variable
  - **keep-alive**: overwrite with environement variable
  - **keep-alive-timeout**: overwrite with environement variable
  - **graceful-shutdown_timeout** : overwrite with environement variable
  - **workers** : overwrite with environement variable

Morphosyntactic tagger service

To run the command type simply from searchai directory:

```
python3 thot/mstagger_svc.py --config=<path to Morphosyntactic Tagger␣
↪configuration file>
```

A light client can be run through the command

```
python3 thot/mstagger_client.py --config=<path to Morphosyntactic Tagger␣
↪configuration file> --input=<input directory> --output=<output directory>
```

Morphosyntactic Tagger Tests

The Morphosyntactic tagger service come with unit and functional testing.

Morphosyntactic Tagger Unit tests

Unittest allows to test Tokenizer classes only.

```
python3 -m unittest thot/tests/unittests/TestMorphoSyntacticTaggerConfiguration.py
python3 -m unittest thot/tests/unittests/TestMorphoSyntacticTagger.py
```

Morphosyntactic Tagger Functional tests

```
python3 -m unittest thot/tests/functional_tests/TestMorphoSyntacticTaggerSvc.py
```

Named entity tagger

The Named entity tagger is a tool allowing to extract Named Entities from "title_tokens" and "content_tokens" field of searchai document. This tools is a rest service where the API is described in **API section** and the configuration file is described in **Configuration section**.

NER Tagger API

OPTIONS /api/nertagger/run

run the ner tagger

Run the ner tagger according to input. It return searchai representation

Request JSON Object

- **config** (*string*) – service configuration file
- **date** (*string*) – release date of the service
- **info** (*string*) – service info id
- **result.content** (*string*) – nested array of text representing the content of the document
- **result.content_morphosyntax[].lemma** (*string*) – lemma of token - the form
- **result.content_morphosyntax[].pos** (*string*) – Part Of Speech
- **result.content_ner[].end** (*integer*) – end offset of the token of the named entity
- **result.content_ner[].label** (*string*) – label of the named entity
- **result.content_ner[].start** (*integer*) – start offset of the token of the named entity
- **result.content_ner[].text** (*string*) – text of the named entity
- **result.content_tokens[]** (*string*) – a token, possibly MWE
- **result.data_source** (*string*) – source of the document (could be host file system, web, ...
- **result.embeddings[].content** (*string*) – text of the sentence
- **result.embeddings[].embeding[]** (*integer*) – embedding feature
- **result.embeddings[].field** (*string*) – Field from which the embedding is computed (can be empty)
- **result.embeddings[].position.length** (*integer*) – length of the sentence
- **result.embeddings[].position.start** (*integer*) – start offset of the sentence
- **result.kg[].property.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].property.content** (*string*) – Content of triple item

- **result.kg[].property.label** (*string*) – label of the content
- **result.kg[].property.lemma_content** (*string*) – Lemmatized content
- **result.kg[].property.pos[]** (*string*) – POS tag
- **result.kg[].property.positions[]** (*integer*) – Position in document
- **result.kg[].subject.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].subject.content** (*string*) – Content of triple item
- **result.kg[].subject.label** (*string*) – label of the content
- **result.kg[].subject.lemma_content** (*string*) – Lemmatized content
- **result.kg[].subject.pos[]** (*string*) – POS tag
- **result.kg[].subject.positions[]** (*integer*) – Position in document
- **result.kg[].value.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].value.content** (*string*) – Content of triple item
- **result.kg[].value.label** (*string*) – label of the content
- **result.kg[].value.lemma_content** (*string*) – Lemmatized content
- **result.kg[].value.pos[]** (*string*) – POS tag
- **result.kg[].value.positions[]** (*integer*) – Position in document
- **result.source_doc_id** (*string*) – uniq id of the document
- **result.title** (*string*) – title of document
- **result.title_morphosyntax[].lemma** (*string*) – lemma of token - the form
- **result.title_morphosyntax[].pos** (*string*) – Part Of Speech
- **result.title_ner[].end** (*integer*) – end offset of the token of the named entity
- **result.title_ner[].label** (*string*) – label of the named entity
- **result.title_ner[].start** (*integer*) – start offset of the token of the named entity
- **result.title_ner[].text** (*string*) – text of the named entity
- **result.title_tokens[]** (*string*) – a token, possibly MWE
- **version** (*string*) – version of the service

Status Codes

- 200 OK – return the searchai doc with tokens
- 422 Unprocessable Entity – Something wrong on the request parameters
- 500 Internal Server Error – Internal Error, and exception occured

Response JSON Object

- **config** (*string*) – service configuration file
- **date** (*string*) – release date of the service
- **info** (*string*) – service info id
- **result.content** (*string*) – nested array of text representing the content of the document

- **result.content_morphosyntax[].lemma** (*string*) – lemma of token - the form
- **result.content_morphosyntax[].pos** (*string*) – Part Of Speech
- **result.content_ner[].end** (*integer*) – end offset of the token of the named entity
- **result.content_ner[].label** (*string*) – label of the named entity
- **result.content_ner[].start** (*integer*) – start offset of the token of the named entity
- **result.content_ner[].text** (*string*) – text of the named entity
- **result.content_tokens[]** (*string*) – a token, possibly MWE
- **result.data_source** (*string*) – source of the document (could be host file system, web, ...
- **result.embeddings[].content** (*string*) – text of the sentence
- **result.embeddings[].embeding[]** (*integer*) – embedding feature
- **result.embeddings[].field** (*string*) – Field from which the embedding is computed (can be empty)
- **result.embeddings[].position.length** (*integer*) – length of the sentence
- **result.embeddings[].position.start** (*integer*) – start offset of the sentence
- **result.kg[].property.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].property.content** (*string*) – Content of triple item
- **result.kg[].property.label** (*string*) – label of the content
- **result.kg[].property.lemma_content** (*string*) – Lemmatized content
- **result.kg[].property.pos[]** (*string*) – POS tag
- **result.kg[].property.positions[]** (*integer*) – Position in document
- **result.kg[].subject.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].subject.content** (*string*) – Content of triple item
- **result.kg[].subject.label** (*string*) – label of the content
- **result.kg[].subject.lemma_content** (*string*) – Lemmatized content
- **result.kg[].subject.pos[]** (*string*) – POS tag
- **result.kg[].subject.positions[]** (*integer*) – Position in document
- **result.kg[].value.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].value.content** (*string*) – Content of triple item
- **result.kg[].value.label** (*string*) – label of the content
- **result.kg[].value.lemma_content** (*string*) – Lemmatized content
- **result.kg[].value.pos[]** (*string*) – POS tag
- **result.kg[].value.positions[]** (*integer*) – Position in document
- **result.source_doc_id** (*string*) – uniq id of the document

- **result.title** (*string*) – title of document
- **result.title_morphosyntax[].lemma** (*string*) – lemma of token - the form
- **result.title_morphosyntax[].pos** (*string*) – Part Of Speech
- **result.title_ner[].end** (*integer*) – end offset of the token of the named entity
- **result.title_ner[].label** (*string*) – label of the named entity
- **result.title_ner[].start** (*integer*) – start offset of the token of the named entity
- **result.title_ner[].text** (*string*) – text of the named entity
- **result.title_tokens[]** (*string*) – a token, possibly MWE
- **version** (*string*) – version of the service

POST /api/nertagger/run

run the ner tagger

Run the ner tagger according to input. It return searchai representation

Request JSON Object

- **config** (*string*) – service configuration file
- **date** (*string*) – release date of the service
- **info** (*string*) – service info id
- **result.content** (*string*) – nested array of text representing the content of the document
- **result.content_morphosyntax[].lemma** (*string*) – lemma of token - the form
- **result.content_morphosyntax[].pos** (*string*) – Part Of Speech
- **result.content_ner[].end** (*integer*) – end offset of the token of the named entity
- **result.content_ner[].label** (*string*) – label of the named entity
- **result.content_ner[].start** (*integer*) – start offset of the token of the named entity
- **result.content_ner[].text** (*string*) – text of the named entity
- **result.content_tokens[]** (*string*) – a token, possibly MWE
- **result.data_source** (*string*) – source of the document (could be host file system, web, ...
- **result.embeddings[].content** (*string*) – text of the sentence
- **result.embeddings[].embeding[]** (*integer*) – embedding feature
- **result.embeddings[].field** (*string*) – Field from which the embedding is computed (can be empty)
- **result.embeddings[].position.length** (*integer*) – length of the sentence
- **result.embeddings[].position.start** (*integer*) – start offset of the sentence
- **result.kg[].property.class_** (*integer*) – Semantic class, comming from clustering prediction

- **result.kg[].property.content** (*string*) – Content of triple item
- **result.kg[].property.label** (*string*) – label of the content
- **result.kg[].property.lemma_content** (*string*) – Lemmatized content
- **result.kg[].property.pos[]** (*string*) – POS tag
- **result.kg[].property.positions[]** (*integer*) – Position in document
- **result.kg[].subject.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].subject.content** (*string*) – Content of triple item
- **result.kg[].subject.label** (*string*) – label of the content
- **result.kg[].subject.lemma_content** (*string*) – Lemmatized content
- **result.kg[].subject.pos[]** (*string*) – POS tag
- **result.kg[].subject.positions[]** (*integer*) – Position in document
- **result.kg[].value.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].value.content** (*string*) – Content of triple item
- **result.kg[].value.label** (*string*) – label of the content
- **result.kg[].value.lemma_content** (*string*) – Lemmatized content
- **result.kg[].value.pos[]** (*string*) – POS tag
- **result.kg[].value.positions[]** (*integer*) – Position in document
- **result.source_doc_id** (*string*) – uniq id of the document
- **result.title** (*string*) – title of document
- **result.title_morphosyntax[].lemma** (*string*) – lemma of token - the form
- **result.title_morphosyntax[].pos** (*string*) – Part Of Speech
- **result.title_ner[].end** (*integer*) – end offset of the token of the named entity
- **result.title_ner[].label** (*string*) – label of the named entity
- **result.title_ner[].start** (*integer*) – start offset of the token of the named entity
- **result.title_ner[].text** (*string*) – text of the named entity
- **result.title_tokens[]** (*string*) – a token, possibly MWE
- **version** (*string*) – version of the service

Status Codes

- 200 OK – return the searchai doc with tokens
- 422 Unprocessable Entity – Something wrong on the request parameters
- 500 Internal Server Error – Internal Error, and exception occured

Response JSON Object

- **config** (*string*) – service configuration file
- **date** (*string*) – release date of the service
- **info** (*string*) – service info id

- **result.content** (*string*) – nested array of text representing the content of the document
- **result.content_morphosyntax[].lemma** (*string*) – lemma of token - the form
- **result.content_morphosyntax[].pos** (*string*) – Part Of Speech
- **result.content_ner[].end** (*integer*) – end offset of the token of the named entity
- **result.content_ner[].label** (*string*) – label of the named entity
- **result.content_ner[].start** (*integer*) – start offset of the token of the named entity
- **result.content_ner[].text** (*string*) – text of the named entity
- **result.content_tokens[]** (*string*) – a token, possibly MWE
- **result.data_source** (*string*) – source of the document (could be host file system, web, ...
- **result.embeddings[].content** (*string*) – text of the sentence
- **result.embeddings[].embeding[]** (*integer*) – embedding feature
- **result.embeddings[].field** (*string*) – Field from which the embedding is computed (can be empty)
- **result.embeddings[].position.length** (*integer*) – length of the sentence
- **result.embeddings[].position.start** (*integer*) – start offset of the sentence
- **result.kg[].property.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].property.content** (*string*) – Content of triple item
- **result.kg[].property.label** (*string*) – label of the content
- **result.kg[].property.lemma_content** (*string*) – Lemmatized content
- **result.kg[].property.pos[]** (*string*) – POS tag
- **result.kg[].property.positions[]** (*integer*) – Position in document
- **result.kg[].subject.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].subject.content** (*string*) – Content of triple item
- **result.kg[].subject.label** (*string*) – label of the content
- **result.kg[].subject.lemma_content** (*string*) – Lemmatized content
- **result.kg[].subject.pos[]** (*string*) – POS tag
- **result.kg[].subject.positions[]** (*integer*) – Position in document
- **result.kg[].value.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].value.content** (*string*) – Content of triple item
- **result.kg[].value.label** (*string*) – label of the content
- **result.kg[].value.lemma_content** (*string*) – Lemmatized content
- **result.kg[].value.pos[]** (*string*) – POS tag
- **result.kg[].value.positions[]** (*integer*) – Position in document

- **result.source_doc_id** (*string*) – uniq id of the document
- **result.title** (*string*) – title of document
- **result.title_morphosyntax[].lemma** (*string*) – lemma of token - the form
- **result.title_morphosyntax[].pos** (*string*) – Part Of Speech
- **result.title_ner[].end** (*integer*) – end offset of the token of the named entity
- **result.title_ner[].label** (*string*) – label of the named entity
- **result.title_ner[].start** (*integer*) – start offset of the token of the named entity
- **result.title_ner[].text** (*string*) – text of the named entity
- **result.title_tokens[]** (*string*) – a token, possibly MWE
- **version** (*string*) – version of the service

OPTIONS /api/nertagger/health

Ask the health of the service

This function allows to know if the service is up and healthy. It is use by client to see if they can use it.

Status Codes

- 200 OK – Service successfuly loaded
- 500 Internal Server Error – Service is not loaded

Response JSON Object

- **config** (*string*) – configuration file path
- **date** (*string*) – service development date
- **health** (*string*) – ok if service is healthy
- **info** (*string*) – service uniq id
- **version** (*string*) – version of the service

POST /api/nertagger/health

Ask the health of the service

This function allows to know if the service is up and healthy. It is use by client to see if they can use it.

Status Codes

- 200 OK – Service successfuly loaded
- 500 Internal Server Error – Service is not loaded

Response JSON Object

- **config** (*string*) – configuration file path
- **date** (*string*) – service development date
- **health** (*string*) – ok if service is healthy
- **info** (*string*) – service uniq id
- **version** (*string*) – version of the service

GET /api/nertagger/health

Ask the health of the service

This function allows to know if the service is up and healthy. It is use by client to see if they can use it.

Status Codes

- 200 OK – Service successfuly loaded
- 500 Internal Server Error – Service is not loaded

Response JSON Object

- **config** (*string*) – configuration file path
- **date** (*string*) – service development date
- **health** (*string*) – ok if service is healthy
- **info** (*string*) – service uniq id
- **version** (*string*) – version of the service

NER tagger configuration

Example of Configuration:

```
{
    "logger": {
        "logging-file": "ner.log",
        "logging-path": "/tmp",
        "logging-level": {"file": "info", "screen": "debug"}
    },
    "named-entities": {
        "label":[{
            "language":"en",
            "resources-base-path":"/home/searchai_svc/searchai/configs/default/
↪resources/modeling/tokenizer/en",
            "mwe": "searchai_mwe.pkl",
            "ner-rules": "ner-rules.json",
            "use-pre-label":true
        }],
        "network": {
            "host":"0.0.0.0",
            "port":10003,
            "associate-environment": {
                "host":"NERTAGGER_HOST",
                "port":"NERTAGGER_PORT"
            }
        },
        "runtime":{
            "request-max-size":100000000,
            "request-buffer-queue-size":100,
            "keep-alive":true,
            "keep-alive-timeout":500,
            "graceful-shutown-timeout":15.0,
            "request-timeout":600,
            "response-timeout":600,
            "workers":1
        },
        "serialize":{
            "input":{
                "path":"/tmp",
                "keep-service-info":true
            },
            "output":{
                "path":"/tmp",
                "keep-service-info":true
            }
        }
```

(continues on next page)

(continued from previous page)

```
    }
}
```

NER Tagger is an aggreation of network configuration, serialize configuration, runtime configuration (in field converter), logger (at top level). The label configuration allows to define validation rules file:

- **language** :the language of tokenizer
- **resources-base-path**: the path to the resources (containing file created by tools *createAnnotationResources.py*
- **ner-rules** : rules to filter labels

NER Rules allows to filter and validate rules according to their POS Tags.

Example of Configuration:

```
{
    "ner-pos-validation":[
        {"label":"person","possible-pos-in-syntagm":["PROPN","DET","ADJ","NOUN"],
↪"at-least":["PROPN"]},
        {"label":"organization","possible-pos-in-syntagm":["PROPN","DET","ADJ",
↪"NOUN"],"at-least":["PROPN"]},
        {"label":"location","possible-pos-in-syntagm":["PROPN","DET","ADJ","NOUN"],
↪"at-least":["PROPN"]},
        {"label":"location.city","possible-pos-in-syntagm":["PROPN","DET","ADJ",
↪"NOUN"],"at-least":["PROPN"]},
        {"label":"location.country","possible-pos-in-syntagm":["PROPN","DET","ADJ",
↪"NOUN"],"at-least":["PROPN"]},
        {"label":"product","possible-pos-in-syntagm":["PROPN","DET","ADJ","NOUN"],
↪"at-least":["PROPN"]},
        {"label":"facility","possible-pos-in-syntagm":["PROPN","DET","ADJ","NOUN"],
↪"at-least":["NOUN","PROPN"]},
        {"label":"event","possible-pos-in-syntagm":["PROPN","DET","ADJ","NOUN"],
↪"at-least":["NOUN","PROPN"]}]
}
```

The validation rule is a set of triple:

- **label**: label of named entity to validat
- **possible-pos-in-syntagm**: the list of the accepted POS tags in the syntagm associated to Named entity
- **at-least**: the minimal POS Tag

Configure NER tagger logger

Logger is configuration at top level of json in *logger* field.

Example of Configuration:

```
{
    "logger": {
        "logging-file": "test.log",
        "logging-path": "/tmp",
        "logging-level": {"file": "info", "screen": "debug"}
    }
}
```

The logger fields are:

- **logging-file** is the filename of the log file (notice that "-<logname>" will be added to this name=
- **logging-path** is the path to the logfile (if it does not exist it will be created)

- **logging-level** contains two fields:
  - **file** for the logging level of the file
  - **screen** for the logging level on screen output

  Both can be set to the following values:
  - **debug** for the debug level and developper information
  - **info** for the level of information
  - **warning** to display only warning and errors
  - **error** to display only error

Configure NER tagger Network

Example of Configuration:

```
{
    "network": {
        "host":"0.0.0.0",
        "port":8080,
        "associate-environment": {
            "host":"HOST_ENVNAME",
            "port":"PORT_ENVNAME"
        }
    }
}
```

The network fields:

- **host** : hostname
- **port** : port of the service
- **associated-environement** : is the "host" and "port" associated environment variables that allows to replace the default one. This field is not mandatory.
  - "host" : associated "host" environment variable
  - "port" : associated "port" environment variable

Configure NER tagger Serialize

Example of Configuration:

```
{
    "serialize":{
        "input":{
            "path":"/tmp",
            "keep-service-info":true
        },
        "output":{
            "path":"/tmp",
            "keep-service-info":true
        }
    }
}
```

The serialize fields:

- **input** : serialize input of service

  - **path** : path of serialized fields
  - **keep-service-info** : True if serialize info is kept
  - **associated-environement** : is the "path" and "keep-service-info" associated environment variables that allows to replace the default one. This field is not mandatory.
    * **path** : associated "path" environment variable
    * **keep-service-info** : associated "keep-service-info" environment variable
- output : serialize output of service
  - **path** : path of serialized fields
  - **keep-service-info** : True if serialize info is kept
  - **associated-environement** : if the "path" and "keep-service-info" associated environment variables that allows to replace the default one. This field is not mandatory.
    * **path** : associated "path" environment variable
    * **keep-service-info** : associated "keep-service-info" environment variable

Configure NER tagger runtime

Example of Configuration:

```
{
    "runtime":{
        "request-max-size":100000000,
        "request-buffer-queue-size":100,
        "keep-alive":true,
        "keep-alive-timeout":5,
        "graceful-shutown-timeout":15.0,
        "request-timeout":60,
        "response-timeout":60,
        "workers":1
    }
}
```

The Runtime fields:

- **request-max-size** : how big a request may be (bytes)
- **request-buffer-queue-size**: request streaming buffer queue size
- **request-timeout** : how long a request can take to arrive (sec)
- **response-timeout** : how long a response can take to process (sec)
- **keep-alive**: keep-alive
- **keep-alive-timeout**: how long to hold a TCP connection open (sec)
- **graceful-shutdown_timeout** : how long to wait to force close non-idle connection (sec)
- **workers** : number of workers for the service on a node
- **associated-environement** : if one of previous field is on the associated environment variables that allows to replace the default one. This field is not mandatory.
  - **request-max-size** : overwrite with environement variable
  - **request-buffer-queue-size**: overwrite with environement variable
  - **request-timeout** : overwrite with environement variable
  - **response-timeout** : overwrite with environement variable

- **keep-alive**: overwrite with environement variable
- **keep-alive-timeout**: overwrite with environement variable
- **graceful-shutdown_timeout** : overwrite with environement variable
- **workers** : overwrite with environement variable

NER tagger service

To run the command type simply from searchai directory:

```
python3 thot/nertagger_svc.py --config=<path to ner configuration file>
```

A light client can be run through the command

```
python3 thot/nertagger_client.py --config=<path to ner tagger configuration file> -
↪-input=<input directory> --output=<output directory>
```

NER Tagger Tests

The NER Tagger service come with unit and functional testing.

NERTagger Unit tests

Unittest allows to test Tokenizer classes only.

```
python3 -m unittest thot/tests/unittests/TestNERTaggerConfiguration.py
python3 -m unittest thot/tests/unittests/TestNERTagger.py
```

NER Tagger Functional tests

```
python3 -m unittest thot/tests/functional_tests/TestNERTaggerSvc.py
```

Syntactic tagger

The Named entity tagger is a tool allowing to extract Named Entities from “title_tokens” and “content_tokens” field of searchai document. This tools is a rest service where the API is described in **API section** and the configuration file is described in **Configuration section**.

Syntactic tagger API

POST /api/syntactictagger/run
run the syntactic tagger

Run the syntactic tagger according to input. It return searchai representation

Request JSON Object

- **config** (*string*) – service configuration file
- **date** (*string*) – release date of the service
- **info** (*string*) – service info id

- **result.content** (*string*) – nested array of text representing the content of the document
- **result.content_deps[].dep** (*string*) – Dependency name
- **result.content_deps[].head** (*integer*) – Head position
- **result.content_deps[].lefts[]** (*integer*) – position
- **result.content_deps[].lemma** (*string*) – lemmatized value
- **result.content_deps[].pos** (*string*) – part of speech
- **result.content_deps[].rights[]** (*integer*) – position
- **result.content_deps[].text** (*string*) – form of token
- **result.content_morphosyntax[].lemma** (*string*) – lemma of token - the form
- **result.content_morphosyntax[].pos** (*string*) – Part Of Speech
- **result.content_ner[].end** (*integer*) – end offset of the token of the named entity
- **result.content_ner[].label** (*string*) – label of the named entity
- **result.content_ner[].start** (*integer*) – start offset of the token of the named entity
- **result.content_ner[].text** (*string*) – text of the named entity
- **result.content_tokens[]** (*string*) – a token, possibly MWE
- **result.data_source** (*string*) – source of the document (could be host file system, web, ...
- **result.embeddings[].content** (*string*) – text of the sentence
- **result.embeddings[].embeding[]** (*integer*) – embedding feature
- **result.embeddings[].field** (*string*) – Field from which the embedding is computed (can be empty)
- **result.embeddings[].position.length** (*integer*) – length of the sentence
- **result.embeddings[].position.start** (*integer*) – start offset of the sentence
- **result.kg[].property.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].property.content** (*string*) – Content of triple item
- **result.kg[].property.label** (*string*) – label of the content
- **result.kg[].property.lemma_content** (*string*) – Lemmatized content
- **result.kg[].property.pos[]** (*string*) – POS tag
- **result.kg[].property.positions[]** (*integer*) – Position in document
- **result.kg[].subject.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].subject.content** (*string*) – Content of triple item
- **result.kg[].subject.label** (*string*) – label of the content
- **result.kg[].subject.lemma_content** (*string*) – Lemmatized content
- **result.kg[].subject.pos[]** (*string*) – POS tag

- **result.kg[].subject.positions[]** (*integer*) – Position in document
- **result.kg[].value.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].value.content** (*string*) – Content of triple item
- **result.kg[].value.label** (*string*) – label of the content
- **result.kg[].value.lemma_content** (*string*) – Lemmatized content
- **result.kg[].value.pos[]** (*string*) – POS tag
- **result.kg[].value.positions[]** (*integer*) – Position in document
- **result.source_doc_id** (*string*) – uniq id of the document
- **result.title** (*string*) – title of document
- **result.title_deps[].dep** (*string*) – Dependency name
- **result.title_deps[].head** (*integer*) – Head position
- **result.title_deps[].lefts[]** (*integer*) – position
- **result.title_deps[].lemma** (*string*) – lemmatized value
- **result.title_deps[].pos** (*string*) – part of speech
- **result.title_deps[].rights[]** (*integer*) – position
- **result.title_deps[].text** (*string*) – form of token
- **result.title_morphosyntax[].lemma** (*string*) – lemma of token - the form
- **result.title_morphosyntax[].pos** (*string*) – Part Of Speech
- **result.title_ner[].end** (*integer*) – end offset of the token of the named entity
- **result.title_ner[].label** (*string*) – label of the named entity
- **result.title_ner[].start** (*integer*) – start offset of the token of the named entity
- **result.title_ner[].text** (*string*) – text of the named entity
- **result.title_tokens[]** (*string*) – a token, possibly MWE
- **version** (*string*) – version of the service

Status Codes

- 200 OK – return the searchai doc with tokens
- 422 Unprocessable Entity – Something wrong on the request parameters
- 500 Internal Server Error – Internal Error, and exception occured

Response JSON Object

- **config** (*string*) – service configuration file
- **date** (*string*) – release date of the service
- **info** (*string*) – service info id
- **result.content** (*string*) – nested array of text representing the content of the document
- **result.content_deps[].dep** (*string*) – Dependency name
- **result.content_deps[].head** (*integer*) – Head position
- **result.content_deps[].lefts[]** (*integer*) – position

- **result.content_deps[].lemma** (*string*) – lemmatized value
- **result.content_deps[].pos** (*string*) – part of speech
- **result.content_deps[].rights[]** (*integer*) – position
- **result.content_deps[].text** (*string*) – form of token
- **result.content_morphosyntax[].lemma** (*string*) – lemma of token - the form
- **result.content_morphosyntax[].pos** (*string*) – Part Of Speech
- **result.content_ner[].end** (*integer*) – end offset of the token of the named entity
- **result.content_ner[].label** (*string*) – label of the named entity
- **result.content_ner[].start** (*integer*) – start offset of the token of the named entity
- **result.content_ner[].text** (*string*) – text of the named entity
- **result.content_tokens[]** (*string*) – a token, possibly MWE
- **result.data_source** (*string*) – source of the document (could be host file system, web, ...
- **result.embeddings[].content** (*string*) – text of the sentence
- **result.embeddings[].embeding[]** (*integer*) – embedding feature
- **result.embeddings[].field** (*string*) – Field from which the embedding is computed (can be empty)
- **result.embeddings[].position.length** (*integer*) – length of the sentence
- **result.embeddings[].position.start** (*integer*) – start offset of the sentence
- **result.kg[].property.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].property.content** (*string*) – Content of triple item
- **result.kg[].property.label** (*string*) – label of the content
- **result.kg[].property.lemma_content** (*string*) – Lemmatized content
- **result.kg[].property.pos[]** (*string*) – POS tag
- **result.kg[].property.positions[]** (*integer*) – Position in document
- **result.kg[].subject.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].subject.content** (*string*) – Content of triple item
- **result.kg[].subject.label** (*string*) – label of the content
- **result.kg[].subject.lemma_content** (*string*) – Lemmatized content
- **result.kg[].subject.pos[]** (*string*) – POS tag
- **result.kg[].subject.positions[]** (*integer*) – Position in document
- **result.kg[].value.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].value.content** (*string*) – Content of triple item
- **result.kg[].value.label** (*string*) – label of the content

- **result.kg[].value.lemma_content** (*string*) – Lemmatized content
- **result.kg[].value.pos[]** (*string*) – POS tag
- **result.kg[].value.positions[]** (*integer*) – Position in document
- **result.source_doc_id** (*string*) – uniq id of the document
- **result.title** (*string*) – title of document
- **result.title_deps[].dep** (*string*) – Dependency name
- **result.title_deps[].head** (*integer*) – Head position
- **result.title_deps[].lefts[]** (*integer*) – position
- **result.title_deps[].lemma** (*string*) – lemmatized value
- **result.title_deps[].pos** (*string*) – part of speech
- **result.title_deps[].rights[]** (*integer*) – position
- **result.title_deps[].text** (*string*) – form of token
- **result.title_morphosyntax[].lemma** (*string*) – lemma of token - the form
- **result.title_morphosyntax[].pos** (*string*) – Part Of Speech
- **result.title_ner[].end** (*integer*) – end offset of the token of the named entity
- **result.title_ner[].label** (*string*) – label of the named entity
- **result.title_ner[].start** (*integer*) – start offset of the token of the named entity
- **result.title_ner[].text** (*string*) – text of the named entity
- **result.title_tokens[]** (*string*) – a token, possibly MWE
- **version** (*string*) – version of the service

OPTIONS /api/syntactictagger/run

run the syntactic tagger

Run the syntactic tagger according to input. It return searchai representation

Request JSON Object

- **config** (*string*) – service configuration file
- **date** (*string*) – release date of the service
- **info** (*string*) – service info id
- **result.content** (*string*) – nested array of text representing the content of the document
- **result.content_deps[].dep** (*string*) – Dependency name
- **result.content_deps[].head** (*integer*) – Head position
- **result.content_deps[].lefts[]** (*integer*) – position
- **result.content_deps[].lemma** (*string*) – lemmatized value
- **result.content_deps[].pos** (*string*) – part of speech
- **result.content_deps[].rights[]** (*integer*) – position
- **result.content_deps[].text** (*string*) – form of token
- **result.content_morphosyntax[].lemma** (*string*) – lemma of token - the form

- **result.content_morphosyntax[].pos** (*string*) – Part Of Speech
- **result.content_ner[].end** (*integer*) – end offset of the token of the named entity
- **result.content_ner[].label** (*string*) – label of the named entity
- **result.content_ner[].start** (*integer*) – start offset of the token of the named entity
- **result.content_ner[].text** (*string*) – text of the named entity
- **result.content_tokens[]** (*string*) – a token, possibly MWE
- **result.data_source** (*string*) – source of the document (could be host file system, web, ...
- **result.embeddings[].content** (*string*) – text of the sentence
- **result.embeddings[].embeding[]** (*integer*) – embedding feature
- **result.embeddings[].field** (*string*) – Field from which the embedding is computed (can be empty)
- **result.embeddings[].position.length** (*integer*) – length of the sentence
- **result.embeddings[].position.start** (*integer*) – start offset of the sentence
- **result.kg[].property.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].property.content** (*string*) – Content of triple item
- **result.kg[].property.label** (*string*) – label of the content
- **result.kg[].property.lemma_content** (*string*) – Lemmatized content
- **result.kg[].property.pos[]** (*string*) – POS tag
- **result.kg[].property.positions[]** (*integer*) – Position in document
- **result.kg[].subject.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].subject.content** (*string*) – Content of triple item
- **result.kg[].subject.label** (*string*) – label of the content
- **result.kg[].subject.lemma_content** (*string*) – Lemmatized content
- **result.kg[].subject.pos[]** (*string*) – POS tag
- **result.kg[].subject.positions[]** (*integer*) – Position in document
- **result.kg[].value.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].value.content** (*string*) – Content of triple item
- **result.kg[].value.label** (*string*) – label of the content
- **result.kg[].value.lemma_content** (*string*) – Lemmatized content
- **result.kg[].value.pos[]** (*string*) – POS tag
- **result.kg[].value.positions[]** (*integer*) – Position in document
- **result.source_doc_id** (*string*) – uniq id of the document
- **result.title** (*string*) – title of document
- **result.title_deps[].dep** (*string*) – Dependency name

- **result.title_deps[].head** (*integer*) – Head position
- **result.title_deps[].lefts[]** (*integer*) – position
- **result.title_deps[].lemma** (*string*) – lemmatized value
- **result.title_deps[].pos** (*string*) – part of speech
- **result.title_deps[].rights[]** (*integer*) – position
- **result.title_deps[].text** (*string*) – form of token
- **result.title_morphosyntax[].lemma** (*string*) – lemma of token - the form
- **result.title_morphosyntax[].pos** (*string*) – Part Of Speech
- **result.title_ner[].end** (*integer*) – end offset of the token of the named entity
- **result.title_ner[].label** (*string*) – label of the named entity
- **result.title_ner[].start** (*integer*) – start offset of the token of the named entity
- **result.title_ner[].text** (*string*) – text of the named entity
- **result.title_tokens[]** (*string*) – a token, possibly MWE
- **version** (*string*) – version of the service

Status Codes

- 200 OK – return the searchai doc with tokens
- 422 Unprocessable Entity – Something wrong on the request parameters
- 500 Internal Server Error – Internal Error, and exception occured

Response JSON Object

- **config** (*string*) – service configuration file
- **date** (*string*) – release date of the service
- **info** (*string*) – service info id
- **result.content** (*string*) – nested array of text representing the content of the document
- **result.content_deps[].dep** (*string*) – Dependency name
- **result.content_deps[].head** (*integer*) – Head position
- **result.content_deps[].lefts[]** (*integer*) – position
- **result.content_deps[].lemma** (*string*) – lemmatized value
- **result.content_deps[].pos** (*string*) – part of speech
- **result.content_deps[].rights[]** (*integer*) – position
- **result.content_deps[].text** (*string*) – form of token
- **result.content_morphosyntax[].lemma** (*string*) – lemma of token - the form
- **result.content_morphosyntax[].pos** (*string*) – Part Of Speech
- **result.content_ner[].end** (*integer*) – end offset of the token of the named entity
- **result.content_ner[].label** (*string*) – label of the named entity

- **result.content_ner[].start** (*integer*) – start offset of the token of the named entity
- **result.content_ner[].text** (*string*) – text of the named entity
- **result.content_tokens[]** (*string*) – a token, possibly MWE
- **result.data_source** (*string*) – source of the document (could be host file system, web, ...
- **result.embeddings[].content** (*string*) – text of the sentence
- **result.embeddings[].embeding[]** (*integer*) – embedding feature
- **result.embeddings[].field** (*string*) – Field from which the embedding is computed (can be empty)
- **result.embeddings[].position.length** (*integer*) – length of the sentence
- **result.embeddings[].position.start** (*integer*) – start offset of the sentence
- **result.kg[].property.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].property.content** (*string*) – Content of triple item
- **result.kg[].property.label** (*string*) – label of the content
- **result.kg[].property.lemma_content** (*string*) – Lemmatized content
- **result.kg[].property.pos[]** (*string*) – POS tag
- **result.kg[].property.positions[]** (*integer*) – Position in document
- **result.kg[].subject.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].subject.content** (*string*) – Content of triple item
- **result.kg[].subject.label** (*string*) – label of the content
- **result.kg[].subject.lemma_content** (*string*) – Lemmatized content
- **result.kg[].subject.pos[]** (*string*) – POS tag
- **result.kg[].subject.positions[]** (*integer*) – Position in document
- **result.kg[].value.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].value.content** (*string*) – Content of triple item
- **result.kg[].value.label** (*string*) – label of the content
- **result.kg[].value.lemma_content** (*string*) – Lemmatized content
- **result.kg[].value.pos[]** (*string*) – POS tag
- **result.kg[].value.positions[]** (*integer*) – Position in document
- **result.source_doc_id** (*string*) – uniq id of the document
- **result.title** (*string*) – title of document
- **result.title_deps[].dep** (*string*) – Dependency name
- **result.title_deps[].head** (*integer*) – Head position
- **result.title_deps[].lefts[]** (*integer*) – position
- **result.title_deps[].lemma** (*string*) – lemmatized value
- **result.title_deps[].pos** (*string*) – part of speech

- **result.title_deps[].rights[]** (*integer*) – position
- **result.title_deps[].text** (*string*) – form of token
- **result.title_morphosyntax[].lemma** (*string*) – lemma of token - the form
- **result.title_morphosyntax[].pos** (*string*) – Part Of Speech
- **result.title_ner[].end** (*integer*) – end offset of the token of the named entity
- **result.title_ner[].label** (*string*) – label of the named entity
- **result.title_ner[].start** (*integer*) – start offset of the token of the named entity
- **result.title_ner[].text** (*string*) – text of the named entity
- **result.title_tokens[]** (*string*) – a token, possibly MWE
- **version** (*string*) – version of the service

GET /api/syntactictagger/health
Ask the health of the service

This function allows to know if the service is up and healthy. It is use by client to see if they can use it.

Status Codes

- 200 OK – Service successfuly loaded
- 500 Internal Server Error – Service is not loaded

Response JSON Object

- **config** (*string*) – configuration file path
- **date** (*string*) – service development date
- **health** (*string*) – ok if service is healthy
- **info** (*string*) – service uniq id
- **version** (*string*) – version of the service

POST /api/syntactictagger/health
Ask the health of the service

This function allows to know if the service is up and healthy. It is use by client to see if they can use it.

Status Codes

- 200 OK – Service successfuly loaded
- 500 Internal Server Error – Service is not loaded

Response JSON Object

- **config** (*string*) – configuration file path
- **date** (*string*) – service development date
- **health** (*string*) – ok if service is healthy
- **info** (*string*) – service uniq id
- **version** (*string*) – version of the service

OPTIONS /api/syntactictagger/health
Ask the health of the service

This function allows to know if the service is up and healthy. It is use by client to see if they can use it.

Status Codes

- 200 OK – Service successfuly loaded
- 500 Internal Server Error – Service is not loaded

Response JSON Object

- **config** (*string*) – configuration file path
- **date** (*string*) – service development date
- **health** (*string*) – ok if service is healthy
- **info** (*string*) – service uniq id
- **version** (*string*) – version of the service

Syntactic tagger configuration

Example of Configuration:

```
{
    "logger": {
        "logging-file": "syntax.log",
        "logging-path": "/tmp",
        "logging-level": {"file": "info", "screen": "debug"}
    },
    "syntax": {
        "taggers":[{
            "language":"en",
            "resources-base-path":"/home/searchai_svc/searchai/configs/default/
↪configs",
            "syntactic-rules": "syntactic-rules.json"
        }],
        "network": {
            "host":"0.0.0.0",
            "port":10004,
            "associate-environment": {
                "host":"SYNTAXTAGGER_HOST",
                "port":"SYNTAXTAGGER_PORT"
            }
        },
        "runtime":{
            "request-max-size":100000000,
            "request-buffer-queue-size":100,
            "keep-alive":true,
            "keep-alive-timeout":500,
            "graceful-shutown-timeout":15.0,
            "request-timeout":600,
            "response-timeout":600,
            "workers":1
        },
        "serialize":{
            "input":{
                "path":"/tmp",
                "keep-service-info":true
            },
            "output":{
                "path":"/tmp",
                "keep-service-info":true
            }
        }

    }
}
```

Syntactic is an aggreation of network configuration, serialize configuration, runtime configuration (in field converter), logger (at top level).

Syntactic Rules allows to define rule for triple Subject, Predicate, Object extraction

Example of Configuration:

```
{
    "pattern_syntagm_or_prep_group":{
        "rule":[[
            {"POS":{"IN":["PREP","DET","ADP","NOUN","PROPN","ADJ","PRON"]},"OP":"*
↪"},
            {"POS":{"IN":["SPACE"]},"OP":"*"},
            {"POS":{"IN":["NOUN","PROPN","ADJ"]},"OP":"+"}
        ],
        [
            {"POS":{"IN":["PREP","DET","ADP","NOUN","PROPN","ADJ","PRON"]},"OP":"*
↪"},
            {"POS":{"IN":["SPACE"]},"OP":"*"},
            {"POS":{"IN":["NOUN","PROPN","ADJ"]},"OP":"+"},
            {"POS":{"IN":["SPACE"]},"OP":"*"},
            {"POS":{"IN":["CONJ","CCONJ"]}},
            {"POS":{"IN":["SPACE"]},"OP":"*"},
            {"POS":{"IN":["NOUN","PROPN","ADJ","DET"]},"OP":"+"}
        ],
        [
            {"POS":{"IN":["ADP", "DET","NOUN","PROPN","ADJ"]},"OP":"+"},
            {"POS":{"IN":["SPACE"]},"OP":"*"},
            {"POS":{"IN":["DET","NOUN","PROPN","ADJ","PUNCT"]},"OP":"+"},
            {"POS":{"IN":["SPACE"]},"OP":"*"},
            {"POS":{"IN":["CONJ","CCONJ"]}},
            {"POS":{"IN":["SPACE"]},"OP":"*"},
            {"POS":{"IN":["DET","NOUN","PROPN","ADJ"]},"OP":"+"}
        ]],
        "type":["subject","object"]
    },
    "pattern_infinitive_verb":{
        "rule":[
            [
            {"LOWER":"to"},
            {"POS":{"IN":["SPACE"]},"OP":"*"},
            {"POS":{"IN":["VERB","AUX"]}}
            ]
        ],
        "type":["predicate"]
    },
    "pattern_pro":{
        "rule":[
            [
            {"POS":{"IN":["PRON"]}}
            ]
        ],
        "type":["subject"]
    },
    "pattern_verb_phrase":{
        "rule":[
                    [
                    {"POS":{"IN":["VERB","AUX"]},"OP":"+"},
                    {"POS":{"IN":["SPACE"]},"OP":"*"},
                    {"POS":{"IN":["ADV","ADP"]},"OP":"?"},
                    {"POS":{"IN":["SPACE"]},"OP":"*"},
                    {"POS":{"IN":["VERB","AUX"]},"OP":"*"}
                    ],
```

(continues on next page)

(continued from previous page)

```
                    [
                    {"POS":{"IN":["VERB","AUX"]},"OP":"+"},
                    {"POS":{"IN":["SPACE"]},"OP":"*"},
                    {"POS":{"IN":["ADV","ADP"]},"OP":"?"},
                    {"POS":{"IN":["SPACE"]},"OP":"*"},
                    {"POS":{"IN":["VERB","AUX"]},"OP":"*"},
                    {"POS":{"IN":["SPACE"]},"OP":"*"},
                    {"POS":{"IN":["CONJ","CCONJ"]}},
                    {"POS":{"IN":["SPACE"]},"OP":"*"},
                    {"POS":{"IN":["VERB","AUX"]},"OP":"+"},
                    {"POS":{"IN":["SPACE"]},"OP":"*"},
                    {"POS":{"IN":["ADV","ADP"]},"OP":"?"},
                    {"POS":{"IN":["SPACE"]},"OP":"*"},
                    {"POS":{"IN":["VERB","AUX"]},"OP":"*"}
                    ]
        ],
        "type":["predicate"]
    },
    "conj_rule":{
        "rule":
            [[{"POS":{"IN":["CONJ", "CCONJ"]},"OP":"+"}]],
        "type":["empty"]
    },
    "link_rule":{
        "type":["link"],
        "rule":[
            {"match-rule":"pattern_verb_phrase", "end-with":"ADP"},
            {"match-rule":"pattern_syntagm_or_prep_group", "start-with":"ADP"}
        ],
        "action": {
            "on":"span-right",
            "shift":"right"
        }
    },
    "available-name-entities": {
        "list": ["person","organization",
                 "location","location.city","location.country",
                 "product","facility","event",
                 "money","quantity","date","time","energyterm","financeterm",
                 "url","email","chemestry"],
        "type": ["named-entity-list"]
    },
    "triple_ner":{
        "type":["triple"],
        "rule":[
                    [{"subject":"pattern_syntagm_or_prep_group"}, {"predicate":
↪"pattern_verb_phrase"}, {"object":"pattern_syntagm_or_prep_group"}],
                    [{"subject":"available-name-entities"}, {"predicate":"pattern_
↪verb_phrase"}, {"object":"available-name-entities"}],
                    [{"subject":"pattern_pro"}, {"predicate":"pattern_verb_phrase"},
↪ {"object":"pattern_syntagm_or_prep_group"}],
                    [{"subject":"pattern_pro"}, {"predicate":"pattern_verb_phrase"},
↪ {"object":"available-name-entities"}],
                    [{"subject":"available-name-entities"}, {"predicate":"pattern_
↪verb_phrase"}, {"object":"pattern_syntagm_or_prep_group"}],
                    [{"subject":"pattern_syntagm_or_prep_group"}, {"predicate":
↪"pattern_verb_phrase"}, {"object":"available-name-entities"}]
        ]
    },
    "settings":{
            "suppress-bounds-sw": true,
```

(continues on next page)

(continued from previous page)

```
            "pos-to-suppress": ["ADP", "ADV", "AUX", "CONJ", "CCONJ", "DET", "INTJ
↪", "PART","SCONJ", "SYM", "SPACE", "X", "PRON", "PUNCT"]
    }
}
```

The rules allows to extract triple based on sequence matcher of spacy

The syntax of the field is:

- **<name of rule>** :
  - **rule** : matcher rule or triple rule
  - **type** : subject, object, predicate of triple

Configure Syntactic tagger logger

Logger is configuration at top level of json in *logger* field.

Example of Configuration:

```
{
    "logger": {
        "logging-file": "test.log",
        "logging-path": "/tmp",
        "logging-level": {"file": "info", "screen": "debug"}
    }
}
```

The logger fields are:

- **logging-file** is the filename of the log file (notice that "-<logname>" will be added to this name=
- **logging-path** is the path to the logfile (if it does not exist it will be created)
- **logging-level** contains two fields:
  - **file** for the logging level of the file
  - **screen** for the logging level on screen output

  Both can be set to the following values:
  - **debug** for the debug level and developper information
  - **info** for the level of information
  - **warning** to display only warning and errors
  - **error** to display only error

Configure Syntactic tagger Network

Example of Configuration:

```
{
    "network": {
        "host":"0.0.0.0",
        "port":8080,
        "associate-environment": {
            "host":"HOST_ENVNAME",
            "port":"PORT_ENVNAME"
        }
    }
```

(continues on next page)

(continued from previous page)

```
}
```

The network fields:

- **host** : hostname
- **port** : port of the service
- **associated-environement** : is the "host" and "port" associated environment variables that allows to replace the default one. This field is not mandatory.
  - "host" : associated "host" environment variable
  - "port" : associated "port" environment variable

Configure Syntactic tagger Serialize

Example of Configuration:

```
{
    "serialize":{
        "input":{
            "path":"/tmp",
            "keep-service-info":true
        },
        "output":{
            "path":"/tmp",
            "keep-service-info":true
        }
    }
}
```

The serialize fields:

- **input** : serialize input of service
  - **path** : path of serialized fields
  - **keep-service-info** : True if serialize info is kept
  - **associated-environement** : is the "path" and "keep-service-info" associated environment variables that allows to replace the default one. This field is not mandatory.
    - **path** : associated "path" environment variable
    - **keep-service-info** : associated "keep-service-info" environment variable
- output : serialize output of service
  - **path** : path of serialized fields
  - **keep-service-info** : True if serialize info is kept
  - **associated-environement** : if the "path" and "keep-service-info" associated environment variables that allows to replace the default one. This field is not mandatory.
    - **path** : associated "path" environment variable
    - **keep-service-info** : associated "keep-service-info" environment variable

Configure Syntactic tagger runtime

Example of Configuration:

```
{
    "runtime":{
        "request-max-size":100000000,
        "request-buffer-queue-size":100,
        "keep-alive":true,
        "keep-alive-timeout":5,
        "graceful-shutown-timeout":15.0,
        "request-timeout":60,
        "response-timeout":60,
        "workers":1
    }
}
```

The Runtime fields:

- **request-max-size** : how big a request may be (bytes)
- **request-buffer-queue-size**: request streaming buffer queue size
- **request-timeout** : how long a request can take to arrive (sec)
- **response-timeout** : how long a response can take to process (sec)
- **keep-alive**: keep-alive
- **keep-alive-timeout**: how long to hold a TCP connection open (sec)
- **graceful-shutdown_timeout** : how long to wait to force close non-idle connection (sec)
- **workers** : number of workers for the service on a node
- **associated-environement** : if one of previous field is on the associated environment variables that allows to replace the default one. This field is not mandatory.
  - **request-max-size** : overwrite with environement variable
  - **request-buffer-queue-size**: overwrite with environement variable
  - **request-timeout** : overwrite with environement variable
  - **response-timeout** : overwrite with environement variable
  - **keep-alive**: overwrite with environement variable
  - **keep-alive-timeout**: overwrite with environement variable
  - **graceful-shutdown_timeout** : overwrite with environement variable
  - **workers** : overwrite with environement variable

Syntactic tagger service

To run the command type simply from searchai directory:

```
python3 thot/syntactictagger_svc.py --config=<path to ner configuration file>
```

A light client can be run through the command

```
python3 thot/syntactictagger_client.py --config=<path to ner tagger configuration␣
↪file> --input=<input directory> --output=<output directory>
```

Syntactic tagger Tests

The Syntactic tagger service come with unit and functional testing.

Syntactic Tagger Unit tests

Unittest allows to test Tokenizer classes only.

```
python3 -m unittest thot/tests/unittests/TestSyntacticTaggerConfiguration.py
python3 -m unittest thot/tests/unittests/TestSyntacticTagger.py
```

Syntactic tagger Functional tests

```
python3 -m unittest thot/tests/functional_tests/TestSyntacticTaggerSvc.py
```

Keywords extractor

The Keywords extractor is a tool allowing to extract keywords from "title_morphosyntax" and "content_morphosyntax" field of searchai document. This tools is a rest service where the API is described in **API section** and the configuration file is described in **Configuration section**.

Keywords extractor API

POST /api/keywordsextractor/run
run the keyword extractor

Run the keyword extractor according to input. It return searchai representation

Request JSON Object

- **config** (*string*) – service configuration file
- **date** (*string*) – release date of the service
- **info** (*string*) – service info id
- **result.content** (*string*) – nested array of text representing the content of the document
- **result.content_deps[].dep** (*string*) – Dependency name
- **result.content_deps[].head** (*integer*) – Head position
- **result.content_deps[].lefts[]** (*integer*) – position
- **result.content_deps[].lemma** (*string*) – lemmatized value
- **result.content_deps[].pos** (*string*) – part of speech
- **result.content_deps[].rights[]** (*integer*) – position
- **result.content_deps[].text** (*string*) – form of token
- **result.content_morphosyntax[].lemma** (*string*) – lemma of token - the form
- **result.content_morphosyntax[].pos** (*string*) – Part Of Speech
- **result.content_ner[].end** (*integer*) – end offset of the token of the named entity
- **result.content_ner[].label** (*string*) – label of the named entity

- **result.content_ner[].start** (*integer*) – start offset of the token of the named entity
- **result.content_ner[].text** (*string*) – text of the named entity
- **result.content_tokens[]** (*string*) – a token, possibly MWE
- **result.data_source** (*string*) – source of the document (could be host file system, web, ...
- **result.embeddings[].content** (*string*) – text of the sentence
- **result.embeddings[].embeding[]** (*integer*) – embedding feature
- **result.embeddings[].field** (*string*) – Field from which the embedding is computed (can be empty)
- **result.embeddings[].position.length** (*integer*) – length of the sentence
- **result.embeddings[].position.start** (*integer*) – start offset of the sentence
- **result.keywords[].class_** (*string*) – Cluster class
- **result.keywords[].end** (*integer*) – end offset of the token of the keyword
- **result.keywords[].score** (*number*) – score of keyword
- **result.keywords[].start** (*integer*) – start offset of the token of the keyword
- **result.keywords[].text** (*string*) – text of the named entity
- **result.kg[].property.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].property.content** (*string*) – Content of triple item
- **result.kg[].property.label** (*string*) – label of the content
- **result.kg[].property.lemma_content** (*string*) – Lemmatized content
- **result.kg[].property.pos[]** (*string*) – POS tag
- **result.kg[].property.positions[]** (*integer*) – Position in document
- **result.kg[].subject.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].subject.content** (*string*) – Content of triple item
- **result.kg[].subject.label** (*string*) – label of the content
- **result.kg[].subject.lemma_content** (*string*) – Lemmatized content
- **result.kg[].subject.pos[]** (*string*) – POS tag
- **result.kg[].subject.positions[]** (*integer*) – Position in document
- **result.kg[].value.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].value.content** (*string*) – Content of triple item
- **result.kg[].value.label** (*string*) – label of the content
- **result.kg[].value.lemma_content** (*string*) – Lemmatized content
- **result.kg[].value.pos[]** (*string*) – POS tag
- **result.kg[].value.positions[]** (*integer*) – Position in document
- **result.source_doc_id** (*string*) – uniq id of the document

- **result.title** (*string*) – title of document
- **result.title_deps[].dep** (*string*) – Dependency name
- **result.title_deps[].head** (*integer*) – Head position
- **result.title_deps[].lefts[]** (*integer*) – position
- **result.title_deps[].lemma** (*string*) – lemmatized value
- **result.title_deps[].pos** (*string*) – part of speech
- **result.title_deps[].rights[]** (*integer*) – position
- **result.title_deps[].text** (*string*) – form of token
- **result.title_morphosyntax[].lemma** (*string*) – lemma of token - the form
- **result.title_morphosyntax[].pos** (*string*) – Part Of Speech
- **result.title_ner[].end** (*integer*) – end offset of the token of the named entity
- **result.title_ner[].label** (*string*) – label of the named entity
- **result.title_ner[].start** (*integer*) – start offset of the token of the named entity
- **result.title_ner[].text** (*string*) – text of the named entity
- **result.title_tokens[]** (*string*) – a token, possibly MWE
- **version** (*string*) – version of the service

Status Codes

- 200 OK – return the searchai doc with tokens
- 422 Unprocessable Entity – Something wrong on the request parameters
- 500 Internal Server Error – Internal Error, and exception occured

Response JSON Object

- **config** (*string*) – service configuration file
- **date** (*string*) – release date of the service
- **info** (*string*) – service info id
- **result.content** (*string*) – nested array of text representing the content of the document
- **result.content_deps[].dep** (*string*) – Dependency name
- **result.content_deps[].head** (*integer*) – Head position
- **result.content_deps[].lefts[]** (*integer*) – position
- **result.content_deps[].lemma** (*string*) – lemmatized value
- **result.content_deps[].pos** (*string*) – part of speech
- **result.content_deps[].rights[]** (*integer*) – position
- **result.content_deps[].text** (*string*) – form of token
- **result.content_morphosyntax[].lemma** (*string*) – lemma of token - the form
- **result.content_morphosyntax[].pos** (*string*) – Part Of Speech
- **result.content_ner[].end** (*integer*) – end offset of the token of the named entity

- **result.content_ner[].label** (*string*) – label of the named entity
- **result.content_ner[].start** (*integer*) – start offset of the token of the named entity
- **result.content_ner[].text** (*string*) – text of the named entity
- **result.content_tokens[]** (*string*) – a token, possibly MWE
- **result.data_source** (*string*) – source of the document (could be host file system, web, ...
- **result.embeddings[].content** (*string*) – text of the sentence
- **result.embeddings[].embeding[]** (*integer*) – embedding feature
- **result.embeddings[].field** (*string*) – Field from which the embedding is computed (can be empty)
- **result.embeddings[].position.length** (*integer*) – length of the sentence
- **result.embeddings[].position.start** (*integer*) – start offset of the sentence
- **result.keywords[].class_** (*string*) – Cluster class
- **result.keywords[].end** (*integer*) – end offset of the token of the keyword
- **result.keywords[].score** (*number*) – score of keyword
- **result.keywords[].start** (*integer*) – start offset of the token of the keyword
- **result.keywords[].text** (*string*) – text of the named entity
- **result.kg[].property.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].property.content** (*string*) – Content of triple item
- **result.kg[].property.label** (*string*) – label of the content
- **result.kg[].property.lemma_content** (*string*) – Lemmatized content
- **result.kg[].property.pos[]** (*string*) – POS tag
- **result.kg[].property.positions[]** (*integer*) – Position in document
- **result.kg[].subject.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].subject.content** (*string*) – Content of triple item
- **result.kg[].subject.label** (*string*) – label of the content
- **result.kg[].subject.lemma_content** (*string*) – Lemmatized content
- **result.kg[].subject.pos[]** (*string*) – POS tag
- **result.kg[].subject.positions[]** (*integer*) – Position in document
- **result.kg[].value.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].value.content** (*string*) – Content of triple item
- **result.kg[].value.label** (*string*) – label of the content
- **result.kg[].value.lemma_content** (*string*) – Lemmatized content
- **result.kg[].value.pos[]** (*string*) – POS tag
- **result.kg[].value.positions[]** (*integer*) – Position in document

- **result.source_doc_id** (*string*) – uniq id of the document
- **result.title** (*string*) – title of document
- **result.title_deps[].dep** (*string*) – Dependency name
- **result.title_deps[].head** (*integer*) – Head position
- **result.title_deps[].lefts[]** (*integer*) – position
- **result.title_deps[].lemma** (*string*) – lemmatized value
- **result.title_deps[].pos** (*string*) – part of speech
- **result.title_deps[].rights[]** (*integer*) – position
- **result.title_deps[].text** (*string*) – form of token
- **result.title_morphosyntax[].lemma** (*string*) – lemma of token - the form
- **result.title_morphosyntax[].pos** (*string*) – Part Of Speech
- **result.title_ner[].end** (*integer*) – end offset of the token of the named entity
- **result.title_ner[].label** (*string*) – label of the named entity
- **result.title_ner[].start** (*integer*) – start offset of the token of the named entity
- **result.title_ner[].text** (*string*) – text of the named entity
- **result.title_tokens[]** (*string*) – a token, possibly MWE
- **version** (*string*) – version of the service

OPTIONS /api/keywordsextractor/run
run the keyword extractor

Run the keyword extractor according to input. It return searchai representation

Request JSON Object

- **config** (*string*) – service configuration file
- **date** (*string*) – release date of the service
- **info** (*string*) – service info id
- **result.content** (*string*) – nested array of text representing the content of the document
- **result.content_deps[].dep** (*string*) – Dependency name
- **result.content_deps[].head** (*integer*) – Head position
- **result.content_deps[].lefts[]** (*integer*) – position
- **result.content_deps[].lemma** (*string*) – lemmatized value
- **result.content_deps[].pos** (*string*) – part of speech
- **result.content_deps[].rights[]** (*integer*) – position
- **result.content_deps[].text** (*string*) – form of token
- **result.content_morphosyntax[].lemma** (*string*) – lemma of token - the form
- **result.content_morphosyntax[].pos** (*string*) – Part Of Speech
- **result.content_ner[].end** (*integer*) – end offset of the token of the named entity

- **result.content_ner[].label** (*string*) – label of the named entity
- **result.content_ner[].start** (*integer*) – start offset of the token of the named entity
- **result.content_ner[].text** (*string*) – text of the named entity
- **result.content_tokens[]** (*string*) – a token, possibly MWE
- **result.data_source** (*string*) – source of the document (could be host file system, web, ...
- **result.embeddings[].content** (*string*) – text of the sentence
- **result.embeddings[].embeding[]** (*integer*) – embedding feature
- **result.embeddings[].field** (*string*) – Field from which the embedding is computed (can be empty)
- **result.embeddings[].position.length** (*integer*) – length of the sentence
- **result.embeddings[].position.start** (*integer*) – start offset of the sentence
- **result.keywords[].class_** (*string*) – Cluster class
- **result.keywords[].end** (*integer*) – end offset of the token of the keyword
- **result.keywords[].score** (*number*) – score of keyword
- **result.keywords[].start** (*integer*) – start offset of the token of the keyword
- **result.keywords[].text** (*string*) – text of the named entity
- **result.kg[].property.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].property.content** (*string*) – Content of triple item
- **result.kg[].property.label** (*string*) – label of the content
- **result.kg[].property.lemma_content** (*string*) – Lemmatized content
- **result.kg[].property.pos[]** (*string*) – POS tag
- **result.kg[].property.positions[]** (*integer*) – Position in document
- **result.kg[].subject.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].subject.content** (*string*) – Content of triple item
- **result.kg[].subject.label** (*string*) – label of the content
- **result.kg[].subject.lemma_content** (*string*) – Lemmatized content
- **result.kg[].subject.pos[]** (*string*) – POS tag
- **result.kg[].subject.positions[]** (*integer*) – Position in document
- **result.kg[].value.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].value.content** (*string*) – Content of triple item
- **result.kg[].value.label** (*string*) – label of the content
- **result.kg[].value.lemma_content** (*string*) – Lemmatized content
- **result.kg[].value.pos[]** (*string*) – POS tag
- **result.kg[].value.positions[]** (*integer*) – Position in document

- **result.source_doc_id** (*string*) – uniq id of the document
- **result.title** (*string*) – title of document
- **result.title_deps[].dep** (*string*) – Dependency name
- **result.title_deps[].head** (*integer*) – Head position
- **result.title_deps[].lefts[]** (*integer*) – position
- **result.title_deps[].lemma** (*string*) – lemmatized value
- **result.title_deps[].pos** (*string*) – part of speech
- **result.title_deps[].rights[]** (*integer*) – position
- **result.title_deps[].text** (*string*) – form of token
- **result.title_morphosyntax[].lemma** (*string*) – lemma of token - the form
- **result.title_morphosyntax[].pos** (*string*) – Part Of Speech
- **result.title_ner[].end** (*integer*) – end offset of the token of the named entity
- **result.title_ner[].label** (*string*) – label of the named entity
- **result.title_ner[].start** (*integer*) – start offset of the token of the named entity
- **result.title_ner[].text** (*string*) – text of the named entity
- **result.title_tokens[]** (*string*) – a token, possibly MWE
- **version** (*string*) – version of the service

Status Codes

- 200 OK – return the searchai doc with tokens
- 422 Unprocessable Entity – Something wrong on the request parameters
- 500 Internal Server Error – Internal Error, and exception occured

Response JSON Object

- **config** (*string*) – service configuration file
- **date** (*string*) – release date of the service
- **info** (*string*) – service info id
- **result.content** (*string*) – nested array of text representing the content of the document
- **result.content_deps[].dep** (*string*) – Dependency name
- **result.content_deps[].head** (*integer*) – Head position
- **result.content_deps[].lefts[]** (*integer*) – position
- **result.content_deps[].lemma** (*string*) – lemmatized value
- **result.content_deps[].pos** (*string*) – part of speech
- **result.content_deps[].rights[]** (*integer*) – position
- **result.content_deps[].text** (*string*) – form of token
- **result.content_morphosyntax[].lemma** (*string*) – lemma of token - the form
- **result.content_morphosyntax[].pos** (*string*) – Part Of Speech

- **result.content_ner[].end** (*integer*) – end offset of the token of the named entity
- **result.content_ner[].label** (*string*) – label of the named entity
- **result.content_ner[].start** (*integer*) – start offset of the token of the named entity
- **result.content_ner[].text** (*string*) – text of the named entity
- **result.content_tokens[]** (*string*) – a token, possibly MWE
- **result.data_source** (*string*) – source of the document (could be host file system, web, ...
- **result.embeddings[].content** (*string*) – text of the sentence
- **result.embeddings[].embeding[]** (*integer*) – embedding feature
- **result.embeddings[].field** (*string*) – Field from which the embedding is computed (can be empty)
- **result.embeddings[].position.length** (*integer*) – length of the sentence
- **result.embeddings[].position.start** (*integer*) – start offset of the sentence
- **result.keywords[].class_** (*string*) – Cluster class
- **result.keywords[].end** (*integer*) – end offset of the token of the keyword
- **result.keywords[].score** (*number*) – score of keyword
- **result.keywords[].start** (*integer*) – start offset of the token of the keyword
- **result.keywords[].text** (*string*) – text of the named entity
- **result.kg[].property.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].property.content** (*string*) – Content of triple item
- **result.kg[].property.label** (*string*) – label of the content
- **result.kg[].property.lemma_content** (*string*) – Lemmatized content
- **result.kg[].property.pos[]** (*string*) – POS tag
- **result.kg[].property.positions[]** (*integer*) – Position in document
- **result.kg[].subject.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].subject.content** (*string*) – Content of triple item
- **result.kg[].subject.label** (*string*) – label of the content
- **result.kg[].subject.lemma_content** (*string*) – Lemmatized content
- **result.kg[].subject.pos[]** (*string*) – POS tag
- **result.kg[].subject.positions[]** (*integer*) – Position in document
- **result.kg[].value.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].value.content** (*string*) – Content of triple item
- **result.kg[].value.label** (*string*) – label of the content
- **result.kg[].value.lemma_content** (*string*) – Lemmatized content

- **result.kg[].value.pos[]** (*string*) – POS tag
- **result.kg[].value.positions[]** (*integer*) – Position in document
- **result.source_doc_id** (*string*) – uniq id of the document
- **result.title** (*string*) – title of document
- **result.title_deps[].dep** (*string*) – Dependency name
- **result.title_deps[].head** (*integer*) – Head position
- **result.title_deps[].lefts[]** (*integer*) – position
- **result.title_deps[].lemma** (*string*) – lemmatized value
- **result.title_deps[].pos** (*string*) – part of speech
- **result.title_deps[].rights[]** (*integer*) – position
- **result.title_deps[].text** (*string*) – form of token
- **result.title_morphosyntax[].lemma** (*string*) – lemma of token - the form
- **result.title_morphosyntax[].pos** (*string*) – Part Of Speech
- **result.title_ner[].end** (*integer*) – end offset of the token of the named entity
- **result.title_ner[].label** (*string*) – label of the named entity
- **result.title_ner[].start** (*integer*) – start offset of the token of the named entity
- **result.title_ner[].text** (*string*) – text of the named entity
- **result.title_tokens[]** (*string*) – a token, possibly MWE
- **version** (*string*) – version of the service

GET /api/keywordsextractor/health
Ask the health of the service

This function allows to know if the service is up and healthy. It is use by client to see if they can use it.

Status Codes

- 200 OK – Service successfuly loaded
- 500 Internal Server Error – Service is not loaded

Response JSON Object

- **config** (*string*) – configuration file path
- **date** (*string*) – service development date
- **health** (*string*) – ok if service is healthy
- **info** (*string*) – service uniq id
- **version** (*string*) – version of the service

POST /api/keywordsextractor/health
Ask the health of the service

This function allows to know if the service is up and healthy. It is use by client to see if they can use it.

Status Codes

- 200 OK – Service successfuly loaded
- 500 Internal Server Error – Service is not loaded

Response JSON Object

- **config** (*string*) – configuration file path
- **date** (*string*) – service development date
- **health** (*string*) – ok if service is healthy
- **info** (*string*) – service uniq id
- **version** (*string*) – version of the service

OPTIONS /api/keywordsextractor/health

Ask the health of the service

This function allows to know if the service is up and healthy. It is use by client to see if they can use it.

Status Codes

- 200 OK – Service successfuly loaded
- 500 Internal Server Error – Service is not loaded

Response JSON Object

- **config** (*string*) – configuration file path
- **date** (*string*) – service development date
- **health** (*string*) – ok if service is healthy
- **info** (*string*) – service uniq id
- **version** (*string*) – version of the service

Keywords extractor configuration

Example of Configuration:

```
{
    "logger": {
        "logging-file": "test.log",
        "logging-path": "/tmp",
        "logging-level": {"file": "info", "screen": "debug"}
    },
    "keywords": {
        "extractors":[{
            "language":"en",
            "resources-base-path":"/home/searchai_svc/searchai/configs/default/
↪configs",
            "keywords-rules":"keywords-rules.json"
        }],
        "network": {
            "host":"0.0.0.0",
            "port":10005,
            "associate-environment": {
                "host":"HOST_ENVNAME",
                "port":"PORT_ENVNAME"
            }
        },
        "runtime":{
            "request-max-size":100000000,
            "request-buffer-queue-size":100,
            "keep-alive":true,
            "keep-alive-timeout":500,
            "graceful-shutown-timeout":15.0,
            "request-timeout":600,
            "response-timeout":600,
```

(continues on next page)

(continued from previous page)

```
            "workers":1
        },
        "serialize":{
            "input":{
                "path":"/tmp",
                "keep-service-info":true
            },
            "output":{
                "path":"/tmp",
                "keep-service-info":true
            }
        }

    }
}
```

Keywords extractor is an aggreation of network configuration, serialize configuration, runtime configuration (in field converter), logger (at top level). The extractor allows to define validation rules for keywords:

- **language** :the language of tokenizer
- **resources-base-path**: the path to the resources (containing file created by tools *createAnnotationResources.py*
- **keywords-rules** : validation rules

Keywords rules allows to filter and validate rules according to their POS Tags.

Example of Configuration:

```
{
    "keywords-pos-validation":{"possible-pos-in-syntagm":["VERB","PROPN","ADJ",
↪"NOUN","ADP", "ADV", "AUX", "CONJ", "CCONJ", "DET", "INTJ", "PART","SCONJ", "SYM
↪", "SPACE", "X"],"at-least":["PROPN","NOUN","ADJ"]},
    "settings":{
        "suppress-bounds-sw": true,
        "pos-to-suppress": ["ADP", "ADV", "AUX", "CONJ", "CCONJ", "DET", "INTJ",
↪"PART","SCONJ", "SYM", "SPACE", "X", "PRON", "PUNCT"]
    }
}
```

The validation rule:

- **possible-pos-in-syntagm**: the list of the accepted POS tags in the syntagm associated to Named entity
- **at-least**: the minimal POS Tag

Configure Keywords extractor logger

Logger is configuration at top level of json in *logger* field.

Example of Configuration:

```
{
    "logger": {
        "logging-file": "test.log",
        "logging-path": "/tmp",
        "logging-level": {"file": "info", "screen": "debug"}
    }
}
```

The logger fields are:

- **logging-file** is the filename of the log file (notice that "-<logname>" will be added to this name=
- **logging-path** is the path to the logfile (if it does not exist it will be created)
- **logging-level** contains two fields:
  - **file** for the logging level of the file
  - **screen** for the logging level on screen output

  Both can be set to the following values:
  - **debug** for the debug level and developper information
  - **info** for the level of information
  - **warning** to display only warning and errors
  - **error** to display only error

Configure Keywords extractor Network

Example of Configuration:

```
{
    "network": {
        "host":"0.0.0.0",
        "port":8080,
        "associate-environment": {
            "host":"HOST_ENVNAME",
            "port":"PORT_ENVNAME"
        }
    }
}
```

The network fields:

- **host** : hostname
- **port** : port of the service
- **associated-environement** : is the "host" and "port" associated environment variables that allows to replace the default one. This field is not mandatory.
  - "host" : associated "host" environment variable
  - "port" : associated "port" environment variable

Configure Keywords extractor Serialize

Example of Configuration:

```
{
    "serialize":{
        "input":{
            "path":"/tmp",
            "keep-service-info":true
        },
        "output":{
            "path":"/tmp",
            "keep-service-info":true
        }
    }
}
```

The serialize fields:

- **input** : serialize input of service
  - **path** : path of serialized fields
  - **keep-service-info** : True if serialize info is kept
  - **associated-environement** : is the "path" and "keep-service-info" associated environment variables that allows to replace the default one. This field is not mandatory.
    - **path** : associated "path" environment variable
    - **keep-service-info** : associated "keep-service-info" environment variable
- output : serialize output of service
  - **path** : path of serialized fields
  - **keep-service-info** : True if serialize info is kept
  - **associated-environement** : if the "path" and "keep-service-info" associated environment variables that allows to replace the default one. This field is not mandatory.
    - **path** : associated "path" environment variable
    - **keep-service-info** : associated "keep-service-info" environment variable

Configure Keywords extractor runtime

Example of Configuration:

```
{
    "runtime":{
        "request-max-size":100000000,
        "request-buffer-queue-size":100,
        "keep-alive":true,
        "keep-alive-timeout":5,
        "graceful-shutown-timeout":15.0,
        "request-timeout":60,
        "response-timeout":60,
        "workers":1
    }
}
```

The Runtime fields:

- **request-max-size** : how big a request may be (bytes)
- **request-buffer-queue-size**: request streaming buffer queue size
- **request-timeout** : how long a request can take to arrive (sec)
- **response-timeout** : how long a response can take to process (sec)
- **keep-alive**: keep-alive
- **keep-alive-timeout**: how long to hold a TCP connection open (sec)
- **graceful-shutdown_timeout** : how long to wait to force close non-idle connection (sec)
- **workers** : number of workers for the service on a node
- **associated-environement** : if one of previous field is on the associated environment variables that allows to replace the default one. This field is not mandatory.
  - **request-max-size** : overwrite with environement variable
  - **request-buffer-queue-size**: overwrite with environement variable

- **request-timeout** : overwrite with environement variable
- **response-timeout** : overwrite with environement variable
- **keep-alive**: overwrite with environement variable
- **keep-alive-timeout**: overwrite with environement variable
- **graceful-shutdown_timeout** : overwrite with environement variable
- **workers** : overwrite with environement variable

Keywords extractor service

To run the command type simply from searchai directory:

```
python3 thot/keywordextractor_svc.py --config=<path to keywords configuration file>
```

A light client can be run through the command

```
python3 thot/keywordextractor_client.py --config=<path to keywords configuration␣
↪file> --input=<input directory> --output=<output directory>
```

Keywords extractor Tests

The Keywords extractor service come with unit and functional testing.

Keywords Unit tests

Unittest allows to test Tokenizer classes only.

```
python3 -m unittest thot/tests/unittests/TestKeywordsConfiguration.py
python3 -m unittest thot/tests/unittests/TestKeywordsExtractor.py
```

Keywords extractor Functional tests

```
python3 -m unittest thot/tests/functional_tests/TestKeywordsExtractorSvc.py
```

Embbeding processing

The embeddings extraction is a tool allowing to extract embedding from "title_tokens" and "content_tokens", "ner", "svo" field of searchai document. This tools is a rest service where the API is described in **API section** and the configuration file is described in **Configuration section**.

Embeddings API

POST /api/embeddings/run
run the embedding extraction

Run the embedding extraction according to input. It return searchai representation

Request JSON Object

- **config** (*string*) – service configuration file
- **date** (*string*) – release date of the service

- **info** (*string*) – service info id
- **result.content** (*string*) – nested array of text representing the content of the document
- **result.content_deps[].dep** (*string*) – Dependency name
- **result.content_deps[].head** (*integer*) – Head position
- **result.content_deps[].lefts[]** (*integer*) – position
- **result.content_deps[].lemma** (*string*) – lemmatized value
- **result.content_deps[].pos** (*string*) – part of speech
- **result.content_deps[].rights[]** (*integer*) – position
- **result.content_deps[].text** (*string*) – form of token
- **result.content_morphosyntax[].lemma** (*string*) – lemma of token - the form
- **result.content_morphosyntax[].pos** (*string*) – Part Of Speech
- **result.content_ner[].end** (*integer*) – end offset of the token of the named entity
- **result.content_ner[].label** (*string*) – label of the named entity
- **result.content_ner[].start** (*integer*) – start offset of the token of the named entity
- **result.content_ner[].text** (*string*) – text of the named entity
- **result.content_tokens[]** (*string*) – a token, possibly MWE
- **result.data_source** (*string*) – source of the document (could be host file system, web, ...
- **result.embeddings[].content** (*string*) – text of the sentence
- **result.embeddings[].embeding[]** (*integer*) – embedding feature
- **result.embeddings[].field** (*string*) – Field from which the embedding is computed (can be empty)
- **result.embeddings[].position.length** (*integer*) – length of the sentence
- **result.embeddings[].position.start** (*integer*) – start offset of the sentence
- **result.keywords[].class_** (*string*) – Cluster class
- **result.keywords[].end** (*integer*) – end offset of the token of the keyword
- **result.keywords[].score** (*number*) – score of keyword
- **result.keywords[].start** (*integer*) – start offset of the token of the keyword
- **result.keywords[].text** (*string*) – text of the named entity
- **result.kg[].property.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].property.content** (*string*) – Content of triple item
- **result.kg[].property.label** (*string*) – label of the content
- **result.kg[].property.lemma_content** (*string*) – Lemmatized content
- **result.kg[].property.pos[]** (*string*) – POS tag

- **result.kg[].property.positions[]** (*integer*) – Position in document
- **result.kg[].subject.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].subject.content** (*string*) – Content of triple item
- **result.kg[].subject.label** (*string*) – label of the content
- **result.kg[].subject.lemma_content** (*string*) – Lemmatized content
- **result.kg[].subject.pos[]** (*string*) – POS tag
- **result.kg[].subject.positions[]** (*integer*) – Position in document
- **result.kg[].value.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].value.content** (*string*) – Content of triple item
- **result.kg[].value.label** (*string*) – label of the content
- **result.kg[].value.lemma_content** (*string*) – Lemmatized content
- **result.kg[].value.pos[]** (*string*) – POS tag
- **result.kg[].value.positions[]** (*integer*) – Position in document
- **result.source_doc_id** (*string*) – uniq id of the document
- **result.title** (*string*) – title of document
- **result.title_deps[].dep** (*string*) – Dependency name
- **result.title_deps[].head** (*integer*) – Head position
- **result.title_deps[].lefts[]** (*integer*) – position
- **result.title_deps[].lemma** (*string*) – lemmatized value
- **result.title_deps[].pos** (*string*) – part of speech
- **result.title_deps[].rights[]** (*integer*) – position
- **result.title_deps[].text** (*string*) – form of token
- **result.title_morphosyntax[].lemma** (*string*) – lemma of token - the form
- **result.title_morphosyntax[].pos** (*string*) – Part Of Speech
- **result.title_ner[].end** (*integer*) – end offset of the token of the named entity
- **result.title_ner[].label** (*string*) – label of the named entity
- **result.title_ner[].start** (*integer*) – start offset of the token of the named entity
- **result.title_ner[].text** (*string*) – text of the named entity
- **result.title_tokens[]** (*string*) – a token, possibly MWE
- **version** (*string*) – version of the service

Status Codes

- 200 OK – return the searchai doc with tokens
- 422 Unprocessable Entity – Something wrong on the request parameters
- 500 Internal Server Error – Internal Error, and exception occured

Response JSON Object

- **config** (*string*) – service configuration file

- **date** (*string*) – release date of the service
- **info** (*string*) – service info id
- **result.content** (*string*) – nested array of text representing the content of the document
- **result.content_deps[].dep** (*string*) – Dependency name
- **result.content_deps[].head** (*integer*) – Head position
- **result.content_deps[].lefts[]** (*integer*) – position
- **result.content_deps[].lemma** (*string*) – lemmatized value
- **result.content_deps[].pos** (*string*) – part of speech
- **result.content_deps[].rights[]** (*integer*) – position
- **result.content_deps[].text** (*string*) – form of token
- **result.content_morphosyntax[].lemma** (*string*) – lemma of token - the form
- **result.content_morphosyntax[].pos** (*string*) – Part Of Speech
- **result.content_ner[].end** (*integer*) – end offset of the token of the named entity
- **result.content_ner[].label** (*string*) – label of the named entity
- **result.content_ner[].start** (*integer*) – start offset of the token of the named entity
- **result.content_ner[].text** (*string*) – text of the named entity
- **result.content_tokens[]** (*string*) – a token, possibly MWE
- **result.data_source** (*string*) – source of the document (could be host file system, web, ...
- **result.embeddings[].content** (*string*) – text of the sentence
- **result.embeddings[].embeding[]** (*integer*) – embedding feature
- **result.embeddings[].field** (*string*) – Field from which the embedding is computed (can be empty)
- **result.embeddings[].position.length** (*integer*) – length of the sentence
- **result.embeddings[].position.start** (*integer*) – start offset of the sentence
- **result.keywords[].class_** (*string*) – Cluster class
- **result.keywords[].end** (*integer*) – end offset of the token of the keyword
- **result.keywords[].score** (*number*) – score of keyword
- **result.keywords[].start** (*integer*) – start offset of the token of the keyword
- **result.keywords[].text** (*string*) – text of the named entity
- **result.kg[].property.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].property.content** (*string*) – Content of triple item
- **result.kg[].property.label** (*string*) – label of the content
- **result.kg[].property.lemma_content** (*string*) – Lemmatized content

- **result.kg[].property.pos[]** (*string*) – POS tag
- **result.kg[].property.positions[]** (*integer*) – Position in document
- **result.kg[].subject.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].subject.content** (*string*) – Content of triple item
- **result.kg[].subject.label** (*string*) – label of the content
- **result.kg[].subject.lemma_content** (*string*) – Lemmatized content
- **result.kg[].subject.pos[]** (*string*) – POS tag
- **result.kg[].subject.positions[]** (*integer*) – Position in document
- **result.kg[].value.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].value.content** (*string*) – Content of triple item
- **result.kg[].value.label** (*string*) – label of the content
- **result.kg[].value.lemma_content** (*string*) – Lemmatized content
- **result.kg[].value.pos[]** (*string*) – POS tag
- **result.kg[].value.positions[]** (*integer*) – Position in document
- **result.source_doc_id** (*string*) – uniq id of the document
- **result.title** (*string*) – title of document
- **result.title_deps[].dep** (*string*) – Dependency name
- **result.title_deps[].head** (*integer*) – Head position
- **result.title_deps[].lefts[]** (*integer*) – position
- **result.title_deps[].lemma** (*string*) – lemmatized value
- **result.title_deps[].pos** (*string*) – part of speech
- **result.title_deps[].rights[]** (*integer*) – position
- **result.title_deps[].text** (*string*) – form of token
- **result.title_morphosyntax[].lemma** (*string*) – lemma of token - the form
- **result.title_morphosyntax[].pos** (*string*) – Part Of Speech
- **result.title_ner[].end** (*integer*) – end offset of the token of the named entity
- **result.title_ner[].label** (*string*) – label of the named entity
- **result.title_ner[].start** (*integer*) – start offset of the token of the named entity
- **result.title_ner[].text** (*string*) – text of the named entity
- **result.title_tokens[]** (*string*) – a token, possibly MWE
- **version** (*string*) – version of the service

OPTIONS /api/embeddings/run

run the embedding extraction

Run the embedding extraction according to input. It return searchai representation

Request JSON Object

- **config** (*string*) – service configuration file

- **date** (*string*) – release date of the service
- **info** (*string*) – service info id
- **result.content** (*string*) – nested array of text representing the content of the document
- **result.content_deps[].dep** (*string*) – Dependency name
- **result.content_deps[].head** (*integer*) – Head position
- **result.content_deps[].lefts[]** (*integer*) – position
- **result.content_deps[].lemma** (*string*) – lemmatized value
- **result.content_deps[].pos** (*string*) – part of speech
- **result.content_deps[].rights[]** (*integer*) – position
- **result.content_deps[].text** (*string*) – form of token
- **result.content_morphosyntax[].lemma** (*string*) – lemma of token - the form
- **result.content_morphosyntax[].pos** (*string*) – Part Of Speech
- **result.content_ner[].end** (*integer*) – end offset of the token of the named entity
- **result.content_ner[].label** (*string*) – label of the named entity
- **result.content_ner[].start** (*integer*) – start offset of the token of the named entity
- **result.content_ner[].text** (*string*) – text of the named entity
- **result.content_tokens[]** (*string*) – a token, possibly MWE
- **result.data_source** (*string*) – source of the document (could be host file system, web, ...
- **result.embeddings[].content** (*string*) – text of the sentence
- **result.embeddings[].embeding[]** (*integer*) – embedding feature
- **result.embeddings[].field** (*string*) – Field from which the embedding is computed (can be empty)
- **result.embeddings[].position.length** (*integer*) – length of the sentence
- **result.embeddings[].position.start** (*integer*) – start offset of the sentence
- **result.keywords[].class_** (*string*) – Cluster class
- **result.keywords[].end** (*integer*) – end offset of the token of the keyword
- **result.keywords[].score** (*number*) – score of keyword
- **result.keywords[].start** (*integer*) – start offset of the token of the keyword
- **result.keywords[].text** (*string*) – text of the named entity
- **result.kg[].property.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].property.content** (*string*) – Content of triple item
- **result.kg[].property.label** (*string*) – label of the content
- **result.kg[].property.lemma_content** (*string*) – Lemmatized content

- **result.kg[].property.pos[]** (*string*) – POS tag
- **result.kg[].property.positions[]** (*integer*) – Position in document
- **result.kg[].subject.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].subject.content** (*string*) – Content of triple item
- **result.kg[].subject.label** (*string*) – label of the content
- **result.kg[].subject.lemma_content** (*string*) – Lemmatized content
- **result.kg[].subject.pos[]** (*string*) – POS tag
- **result.kg[].subject.positions[]** (*integer*) – Position in document
- **result.kg[].value.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].value.content** (*string*) – Content of triple item
- **result.kg[].value.label** (*string*) – label of the content
- **result.kg[].value.lemma_content** (*string*) – Lemmatized content
- **result.kg[].value.pos[]** (*string*) – POS tag
- **result.kg[].value.positions[]** (*integer*) – Position in document
- **result.source_doc_id** (*string*) – uniq id of the document
- **result.title** (*string*) – title of document
- **result.title_deps[].dep** (*string*) – Dependency name
- **result.title_deps[].head** (*integer*) – Head position
- **result.title_deps[].lefts[]** (*integer*) – position
- **result.title_deps[].lemma** (*string*) – lemmatized value
- **result.title_deps[].pos** (*string*) – part of speech
- **result.title_deps[].rights[]** (*integer*) – position
- **result.title_deps[].text** (*string*) – form of token
- **result.title_morphosyntax[].lemma** (*string*) – lemma of token - the form
- **result.title_morphosyntax[].pos** (*string*) – Part Of Speech
- **result.title_ner[].end** (*integer*) – end offset of the token of the named entity
- **result.title_ner[].label** (*string*) – label of the named entity
- **result.title_ner[].start** (*integer*) – start offset of the token of the named entity
- **result.title_ner[].text** (*string*) – text of the named entity
- **result.title_tokens[]** (*string*) – a token, possibly MWE
- **version** (*string*) – version of the service

Status Codes

- 200 OK – return the searchai doc with tokens
- 422 Unprocessable Entity – Something wrong on the request parameters
- 500 Internal Server Error – Internal Error, and exception occured

Response JSON Object

- **config** (*string*) – service configuration file
- **date** (*string*) – release date of the service
- **info** (*string*) – service info id
- **result.content** (*string*) – nested array of text representing the content of the document
- **result.content_deps[].dep** (*string*) – Dependency name
- **result.content_deps[].head** (*integer*) – Head position
- **result.content_deps[].lefts[]** (*integer*) – position
- **result.content_deps[].lemma** (*string*) – lemmatized value
- **result.content_deps[].pos** (*string*) – part of speech
- **result.content_deps[].rights[]** (*integer*) – position
- **result.content_deps[].text** (*string*) – form of token
- **result.content_morphosyntax[].lemma** (*string*) – lemma of token - the form
- **result.content_morphosyntax[].pos** (*string*) – Part Of Speech
- **result.content_ner[].end** (*integer*) – end offset of the token of the named entity
- **result.content_ner[].label** (*string*) – label of the named entity
- **result.content_ner[].start** (*integer*) – start offset of the token of the named entity
- **result.content_ner[].text** (*string*) – text of the named entity
- **result.content_tokens[]** (*string*) – a token, possibly MWE
- **result.data_source** (*string*) – source of the document (could be host file system, web, ...
- **result.embeddings[].content** (*string*) – text of the sentence
- **result.embeddings[].embeding[]** (*integer*) – embedding feature
- **result.embeddings[].field** (*string*) – Field from which the embedding is computed (can be empty)
- **result.embeddings[].position.length** (*integer*) – length of the sentence
- **result.embeddings[].position.start** (*integer*) – start offset of the sentence
- **result.keywords[].class_** (*string*) – Cluster class
- **result.keywords[].end** (*integer*) – end offset of the token of the keyword
- **result.keywords[].score** (*number*) – score of keyword
- **result.keywords[].start** (*integer*) – start offset of the token of the keyword
- **result.keywords[].text** (*string*) – text of the named entity
- **result.kg[].property.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].property.content** (*string*) – Content of triple item
- **result.kg[].property.label** (*string*) – label of the content

- **result.kg[].property.lemma_content** (*string*) – Lemmatized content
- **result.kg[].property.pos[]** (*string*) – POS tag
- **result.kg[].property.positions[]** (*integer*) – Position in document
- **result.kg[].subject.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].subject.content** (*string*) – Content of triple item
- **result.kg[].subject.label** (*string*) – label of the content
- **result.kg[].subject.lemma_content** (*string*) – Lemmatized content
- **result.kg[].subject.pos[]** (*string*) – POS tag
- **result.kg[].subject.positions[]** (*integer*) – Position in document
- **result.kg[].value.class_** (*integer*) – Semantic class, comming from clustering prediction
- **result.kg[].value.content** (*string*) – Content of triple item
- **result.kg[].value.label** (*string*) – label of the content
- **result.kg[].value.lemma_content** (*string*) – Lemmatized content
- **result.kg[].value.pos[]** (*string*) – POS tag
- **result.kg[].value.positions[]** (*integer*) – Position in document
- **result.source_doc_id** (*string*) – uniq id of the document
- **result.title** (*string*) – title of document
- **result.title_deps[].dep** (*string*) – Dependency name
- **result.title_deps[].head** (*integer*) – Head position
- **result.title_deps[].lefts[]** (*integer*) – position
- **result.title_deps[].lemma** (*string*) – lemmatized value
- **result.title_deps[].pos** (*string*) – part of speech
- **result.title_deps[].rights[]** (*integer*) – position
- **result.title_deps[].text** (*string*) – form of token
- **result.title_morphosyntax[].lemma** (*string*) – lemma of token - the form
- **result.title_morphosyntax[].pos** (*string*) – Part Of Speech
- **result.title_ner[].end** (*integer*) – end offset of the token of the named entity
- **result.title_ner[].label** (*string*) – label of the named entity
- **result.title_ner[].start** (*integer*) – start offset of the token of the named entity
- **result.title_ner[].text** (*string*) – text of the named entity
- **result.title_tokens[]** (*string*) – a token, possibly MWE
- **version** (*string*) – version of the service

POST /api/embeddings/run_from_table
run the embedding extraction

Run the embedding extraction according to input. It return searchai representation

Request JSON Object

- **sentences[]** (*string*) – sentence to analyze

Status Codes

- 200 OK – return the embeddings
- 422 Unprocessable Entity – Something wrong on the request parameters
- 500 Internal Server Error – Internal Error, and exception occured

Response JSON Object

- **config** (*string*) – service configuration file
- **date** (*string*) – release date of the service
- **info** (*string*) – service info id
- **result[].content** (*string*) – text of the sentence
- **result[].embeding[]** (*integer*) – embedding feature
- **result[].field** (*string*) – Field from which the embedding is computed (can be empty)
- **result[].position.length** (*integer*) – length of the sentence
- **result[].position.start** (*integer*) – start offset of the sentence
- **version** (*string*) – version of the service

OPTIONS /api/embeddings/run_from_table

run the embedding extraction

Run the embedding extraction according to input. It return searchai representation

Request JSON Object

- **sentences[]** (*string*) – sentence to analyze

Status Codes

- 200 OK – return the embeddings
- 422 Unprocessable Entity – Something wrong on the request parameters
- 500 Internal Server Error – Internal Error, and exception occured

Response JSON Object

- **config** (*string*) – service configuration file
- **date** (*string*) – release date of the service
- **info** (*string*) – service info id
- **result[].content** (*string*) – text of the sentence
- **result[].embeding[]** (*integer*) – embedding feature
- **result[].field** (*string*) – Field from which the embedding is computed (can be empty)
- **result[].position.length** (*integer*) – length of the sentence
- **result[].position.start** (*integer*) – start offset of the sentence
- **version** (*string*) – version of the service

POST /api/embeddings/health

Ask the health of the service

This function allows to know if the service is up and healthy. It is use by client to see if they can use it.

Status Codes

- 200 OK – Service successfuly loaded
- 500 Internal Server Error – Service is not loaded

Response JSON Object

- **config** (*string*) – configuration file path
- **date** (*string*) – service development date
- **health** (*string*) – ok if service is healthy
- **info** (*string*) – service uniq id
- **version** (*string*) – version of the service

GET /api/embeddings/health
Ask the health of the service

This function allows to know if the service is up and healthy. It is use by client to see if they can use it.

Status Codes

- 200 OK – Service successfuly loaded
- 500 Internal Server Error – Service is not loaded

Response JSON Object

- **config** (*string*) – configuration file path
- **date** (*string*) – service development date
- **health** (*string*) – ok if service is healthy
- **info** (*string*) – service uniq id
- **version** (*string*) – version of the service

OPTIONS /api/embeddings/health
Ask the health of the service

This function allows to know if the service is up and healthy. It is use by client to see if they can use it.

Status Codes

- 200 OK – Service successfuly loaded
- 500 Internal Server Error – Service is not loaded

Response JSON Object

- **config** (*string*) – configuration file path
- **date** (*string*) – service development date
- **health** (*string*) – ok if service is healthy
- **info** (*string*) – service uniq id
- **version** (*string*) – version of the service

Embeddings configuration

Example of Configuration:

```
{
    "logger": {
        "logging-file": "embeddings.log",
        "logging-path": "/tmp",
        "logging-level": {"file": "info", "screen": "debug"}
    },
    "embeddings": {
        "models":[{
            "language":"multi",
            "use-cuda":true,
            "batch-size":256
        }],
        "network": {
            "host":"0.0.0.0",
            "port":10006,
            "associate-environment": {
                "host":"SENT_EMBEDDING_HOST",
                "port":"SENT_EMBEDDING_PORT"
            }
        },
        "runtime":{
            "request-max-size":100000000,
            "request-buffer-queue-size":100,
            "keep-alive":true,
            "keep-alive-timeout":500,
            "graceful-shutown-timeout":500.0,
            "request-timeout":600,
            "response-timeout":600,
            "workers":1
        },
        "serialize":{
            "input":{
                "path":"/tmp",
                "keep-service-info":true
            },
            "output":{
                "path":"/tmp",
                "keep-service-info":true
            }
        }

    }
}
```

Embedding configuration is an aggreation of network configuration, serialize configuration, runtime configuration (in field converter), logger (at top level). The models fiels allows to configure model access

“language”:”multi”,

“use-cuda”:true, “batch-size”:256

- **language** :the language of model (not yet implemented, use multi for language agnostic)
- **use-cuda**: use cuda to run the model
- **batch-size** : size of batch sent to the model

Configure Embeddings logger

Logger is configuration at top level of json in *logger* field.

Example of Configuration:

```
{
    "logger": {
        "logging-file": "test.log",
        "logging-path": "/tmp",
        "logging-level": {"file": "info", "screen": "debug"}
    }
}
```

The logger fields are:

- **logging-file** is the filename of the log file (notice that "-<logname>" will be added to this name=
- **logging-path** is the path to the logfile (if it does not exist it will be created)
- **logging-level** contains two fields:
  - **file** for the logging level of the file
  - **screen** for the logging level on screen output

  Both can be set to the following values:
  - **debug** for the debug level and developper information
  - **info** for the level of information
  - **warning** to display only warning and errors
  - **error** to display only error

Configure Embeddings Network

Example of Configuration:

```
{
    "network": {
        "host":"0.0.0.0",
        "port":8080,
        "associate-environment": {
            "host":"HOST_ENVNAME",
            "port":"PORT_ENVNAME"
        }
    }
}
```

The network fields:

- **host** : hostname
- **port** : port of the service
- **associated-environement** : is the "host" and "port" associated environment variables that allows to replace the default one. This field is not mandatory.
  - "host" : associated "host" environment variable
  - "port" : associated "port" environment variable

Configure Embeddings Serialize

Example of Configuration:

```
{
    "serialize":{
        "input":{
            "path":"/tmp",
            "keep-service-info":true
        },
        "output":{
            "path":"/tmp",
            "keep-service-info":true
        }
    }
}
```

The serialize fields:

- **input** : serialize input of service
  - **path** : path of serialized fields
  - **keep-service-info** : True if serialize info is kept
  - **associated-environement** : is the "path" and "keep-service-info" associated environment variables that allows to replace the default one. This field is not mandatory.
    - **path** : associated "path" environment variable
    - **keep-service-info** : associated "keep-service-info" environment variable
- output : serialize output of service
  - **path** : path of serialized fields
  - **keep-service-info** : True if serialize info is kept
  - **associated-environement** : if the "path" and "keep-service-info" associated environment variables that allows to replace the default one. This field is not mandatory.
    - **path** : associated "path" environment variable
    - **keep-service-info** : associated "keep-service-info" environment variable

Configure Embeddings runtime

Example of Configuration:

```
{
    "runtime":{
        "request-max-size":100000000,
        "request-buffer-queue-size":100,
        "keep-alive":true,
        "keep-alive-timeout":5,
        "graceful-shutown-timeout":15.0,
        "request-timeout":60,
        "response-timeout":60,
        "workers":1
    }
}
```

The Runtime fields:

- **request-max-size** : how big a request may be (bytes)

- **request-buffer-queue-size**: request streaming buffer queue size
- **request-timeout** : how long a request can take to arrive (sec)
- **response-timeout** : how long a response can take to process (sec)
- **keep-alive**: keep-alive
- **keep-alive-timeout**: how long to hold a TCP connection open (sec)
- **graceful-shutdown_timeout** : how long to wait to force close non-idle connection (sec)
- **workers** : number of workers for the service on a node
- **associated-environement** : if one of previous field is on the associated environment variables that allows to replace the default one. This field is not mandatory.
  - **request-max-size** : overwrite with environement variable
  - **request-buffer-queue-size**: overwrite with environement variable
  - **request-timeout** : overwrite with environement variable
  - **response-timeout** : overwrite with environement variable
  - **keep-alive**: overwrite with environement variable
  - **keep-alive-timeout**: overwrite with environement variable
  - **graceful-shutdown_timeout** : overwrite with environement variable
  - **workers** : overwrite with environement variable

Embeddings service

To run the command type simply from searchai directory:

```
python3 thot/embeddings_svc.py --config=<path to embeddings configuration␣
↪file>
```

A light client can be run through the command

```
python3 thot/embeddings_client.py --config=<path to embeddings configuration file>␣
↪--input=<input directory> --output=<output directory>
```

Embeddings Tests

The Embeddings service come with unit and functional testing.

Embeddings Unit tests

Unittest allows to test Tokenizer classes only.

```
python3 -m unittest thot/tests/unittests/TestEmbeddingConfiguration.py
python3 -m unittest thot/tests/unittests/TestEmbeddings.py
```

Embeddings Functional tests

```
python3 -m unittest thot/tests/functional_tests/TestEmbeddingsSvc.py
```

Relation clustering

Relation clustering allows to create class on SVO extracted during the Syntactic tagging phase.

Relations clustering configuration

Example of Configuration:

```
{
"logger": {
    "logging-file": "test.log",
    "logging-path": "/tmp",
    "logging-level": {"file": "info", "screen": "debug"}
},
"relations": {
    "cluster":{
        "algorithm":"kmeans",
        "number-of-classes":128,
        "number-of-iterations":16,
        "seed":123456,
        "batch-size":4096,
        "embeddings-server":{
            "host":"0.0.0.0",
            "port":10006
        }
    },
    "serialize":{
        "input":{
            "path":"/tmp",
            "keep-service-info":true
        },
        "output":{
            "path":"/tmp",
            "keep-service-info":true
        }
    }
}
}
```

Relation clustering configuration is an aggreation of serialize configuration, logger (at top level). The clustering configuration allows to define embedding server access and clustering algorithms settings:

- **algorithm**: ["kmeans","spericalkmeans" (Not yet available)],
- **number-of-classes**: number of cluster classes,
- **number-of-iterations**: number of kmeans iterations,
- **seed**:kmeans seed
- **batch-size**: we use mini batch kmeans, the batch size if the number of vectors send for partial fit,
- **embeddings-server** : embedding server network information (host and port)

Configure Relations clustering logger

Logger is configuration at top level of json in *logger* field.

Example of Configuration:

```
{
    "logger": {
        "logging-file": "test.log",
        "logging-path": "/tmp",
        "logging-level": {"file": "info", "screen": "debug"}
    }
}
```

The logger fields are:

- **logging-file** is the filename of the log file (notice that "-<logname>" will be added to this name=
- **logging-path** is the path to the logfile (if it does not exist it will be created)
- **logging-level** contains two fields:
  - **file** for the logging level of the file
  - **screen** for the logging level on screen output

  Both can be set to the following values:
  - **debug** for the debug level and developper information
  - **info** for the level of information
  - **warning** to display only warning and errors
  - **error** to display only error

Relation clustering tool

To run the command type simply from searchai directory:

```
python3 thot/relation_clustering.py --config=<path to relation␣
↪configuration file> -i <path to file with syntactic data extracted> -o
↪<path to output folder>
```

Search engine

Index

Document indexing is the step allowing to store in the elastic search the document analyzed during the steps of tokenization, tagging, keyword extraction ...

Indexing configuration

Example of Configuration:

```
{
    "logger": {
        "logging-file": "test.log",
        "logging-path": "/tmp",
        "logging-level": {"file": "info", "screen": "debug"}
    },
    "indexing": {
        "document":{
            "remove-knowledge-graph-duplicates":true
        },
        "elasticsearch":{
            "network": {
                "host": "opendistro",
                "port": 9200,
                "use_ssl": false,
                "verify_certs": false,
                "associate-environment": {
                    "host":"OPENDISTRO_HOST",
                    "port":"OPENDISTRO_PORT"
                }
            },
            "nms-index":{
               "name":"nms-index",
               "mapping-file":"/home/searchai_svc/searchai/resources/indices/
↪indices_mapping/nms_cache_index.json"
            },
            "text-index":{
                "name":"text-index",
                "mapping-file":"/home/searchai_svc/searchai/resources/indices/
↪indices_mapping/cache_index.json"
            },
            "relation-index":{
                "name":"relation-index",
                "mapping-file":"/home/searchai_svc/searchai/resources/indices/
↪indices_mapping/relation_index.json"
            }
        },
        "serialize":{
            "input":{
                "path":"/tmp",
                "keep-service-info":true
            },
            "output":{
                "path":"/tmp",
                "keep-service-info":true
            }
        }
     }
}
```

The indexing configuration is an aggreation of serialize configuration, logger (at top level). The the indexing configuration needs an ElasticSearch :

- **document/remove-knowledge-graph-duplicates**: in the analyzed document knowledge graph items are positions wise, to avoid duplication in the index you can suppress position an thus make items uniq
- **elasticsearch/network**: network configuration of E.S.
- **elasticsearch/nms-index**: index containing vectors (obsolete),

- **elasticsearch/text-index**: index containing analyzed textual document,
- **elasticsearch/relation-index**: index containing relations (obsolete),

Run Index process

To run the command type simply from searchai directory:

```
python3 thot/searchai2index --config=<path to indexing configuration file> -d
↪<directory to index> -t document
```

Search

Search component can be see as a proxy to ElasticSearch (E.S.). Nonetheless, it allows to create specifc E.S queries based on query analysis. It also allows to manipulate ranking scores.

Search Configuration

Example of Configuration:

```
{
    "logger": {
        "logging-file": "test.log",
        "logging-path": "/tmp",
        "logging-level": {"file": "info", "screen": "debug"}
    },
    "searching": {
        "document-index-name":"text-index",
        "disable-document-analysis":false,
        "search-policy":{
            "semantic-cluster":{
                "semantic-quantizer-model":"/home/searchai_svc/searchai/configs/
↪default/resources/modeling/relation_names.model.pkl"
            },
            "settings":{
                "basic-querying":{
                    "uniq-word-query":true,
                    "boosted-uniq-word-query":false,
                    "cut-query":4096
                },
                "advanced-querying":{
                    "use-lemma":false,
                    "use-keywords":false,
                    "use-knowledge-graph":false,
                    "use-semantic-keywords":false,
                    "use-semantic-knowledge-graph":true,
                    "use-concepts":true,
                    "use-sentences":false,
                    "querying":{
                        "match-phrase-slop":1,
                        "match-phrase-boosting":0.5,
                        "match-sentence":{
                            "number-and-symbol-filtering":true,
                            "max-number-of-words":30
                        },
                        "match-keyword":{
                            "number-and-symbol-filtering":true,
                            "semantic-skip-highest-ranked-classes":3,
```

(continues on next page)

(continued from previous page)

```
                        "semantic-max-boosting":5
                    },
                    "match-svo":{
                        "semantic-use-class-triple":false,
                        "semantic-use-lemma-property-object":false,
                        "semantic-use-subject-lemma-object":false,
                        "semantic-use-subject-property-lemma":false,
                        "semantic-use-lemma-lemma-object":true,
                        "semantic-use-lemma-property-lemma":true,
                        "semantic-use-subject-lemma-lemma":true,
                        "semanic-max-boosting":5
                    },
                    "match-concept":{
                        "concept-boosting":0.2,
                        "concept-pruning":10
                    }
                }
            },
            "query-expansion":{
                "term-pruning":128,
                "suppress-number":true,
                "keep-word-collection-thresold-under":0.4,
                "word-boost-thresold-above":0.25
            },
            "scoring":{
                "normalize-score":true,
                "document-query-intersection-penalty":"by-query-size",
                "run-clause-separately":true,
                "expand-results":50
            },
            "results":{
                "set-highlight":false,
                "excludes":[]
            }

        }
    },
    "elasticsearch":{
        "network": {
            "host": "http://searchai_opendistro:9200",
            "port": 0,
            "use_ssl": false,
            "verify_certs": false,
            "associate-environment": {
                "host":"HOST_ENVNAME",
                "port":"PORT_ENVNAME"
            }
        }
    },
    "network": {
        "host":"0.0.0.0",
        "port":8000,
        "associate-environment": {
            "host":"SEARCH_HOST",
            "port":"SEARCH_PORT"
        }
    },
    "runtime":{
        "request-max-size":100000000,
        "request-buffer-queue-size":100,
        "keep-alive":true,
```

(continues on next page)

(continued from previous page)

```
            "keep-alive-timeout":5,
            "graceful-shutown-timeout":15.0,
            "request-timeout":60,
            "response-timeout":60,
            "workers":1
        },
        "serialize":{
            "input":{
                "path":"/tmp",
                "keep-service-info":true
            },
            "output":{
                "path":"/tmp",
                "keep-service-info":true
            }
        }
    },
    "tokenizers": {
        "segmenters":[{
            "language":"en",
            "resources-base-path":"/home/searchai_svc/searchai/configs/default/
↪resources/modeling/tokenizer/en",
            "normalization-rules":"tokenizer-rules.json",
            "mwe": "searchai_mwe.pkl"
        }],
        "network": {
            "host":"0.0.0.0",
            "port":8080,
            "associate-environment": {
                "host":"HOST_ENVNAME",
                "port":"PORT_ENVNAME"
            }
        },
        "runtime":{
            "request-max-size":100000000,
            "request-buffer-queue-size":100,
            "keep-alive":true,
            "keep-alive-timeout":5,
            "graceful-shutown-timeout":15.0,
            "request-timeout":60,
            "response-timeout":60,
            "workers":1
        },
        "serialize":{
            "input":{
                "path":"/tmp",
                "keep-service-info":true
            },
            "output":{
                "path":"/tmp",
                "keep-service-info":true
            }
        }
    },
    "morphosyntax": {
        "taggers":[{
            "language":"en",
            "resources-base-path":"/home/searchai_svc/searchai/configs/default/
↪resources/modeling/tokenizer/en",
            "mwe": "searchai_mwe.pkl",
            "pre-sentencizer": true,
```

(continues on next page)

(continued from previous page)

```
            "pre-tagging-with-concept":true,
            "add-concept-in-knowledge-graph":true
        }],
        "network": {
            "host":"0.0.0.0",
            "port":8080,
            "associate-environment": {
                "host":"HOST_ENVNAME",
                "port":"PORT_ENVNAME"
            }
        },
        "runtime":{
            "request-max-size":100000000,
            "request-buffer-queue-size":100,
            "keep-alive":true,
            "keep-alive-timeout":5,
            "graceful-shutown-timeout":15.0,
            "request-timeout":60,
            "response-timeout":60,
            "workers":1
        },
        "serialize":{
            "input":{
                "path":"/tmp",
                "keep-service-info":true
            },
            "output":{
                "path":"/tmp",
                "keep-service-info":true
            }
        }
    },
    "named-entities": {
        "label":[{
            "language":"en",
            "resources-base-path":"/home/searchai_svc/searchai/configs/default/
↪resources/modeling/tokenizer/en",
            "mwe": "searchai_mwe.pkl",
            "use-pre-label":true
        }],
        "network": {
            "host":"0.0.0.0",
            "port":8080,
            "associate-environment": {
                "host":"HOST_ENVNAME",
                "port":"PORT_ENVNAME"
            }
        },
        "runtime":{
            "request-max-size":100000000,
            "request-buffer-queue-size":100,
            "keep-alive":true,
            "keep-alive-timeout":5,
            "graceful-shutown-timeout":15.0,
            "request-timeout":60,
            "response-timeout":60,
            "workers":1
        },
        "serialize":{
            "input":{
                "path":"/tmp",
```

(continues on next page)

(continued from previous page)

```
                "keep-service-info":true
            },
            "output":{
                "path":"/tmp",
                "keep-service-info":true
            }
        }
    },
    "embeddings": {
        "models":[
        { "language":"multi" }
        ],
        "network": {
            "host":"0.0.0.0",
            "port":8080,
            "associate-environment": {
                "host":"HOST_ENVNAME",
                "port":"PORT_ENVNAME"
            }
        },
        "runtime":{
            "request-max-size":100000000,
            "request-buffer-queue-size":100,
            "keep-alive":true,
            "keep-alive-timeout":5,
            "graceful-shutown-timeout":15.0,
            "request-timeout":60,
            "response-timeout":60,
            "workers":1
        },
        "serialize":{
            "input":{
                "path":"/tmp",
                "keep-service-info":true
            },
            "output":{
                "path":"/tmp",
                "keep-service-info":true
            }
        }
    },
    "syntax": {
        "taggers":[{
            "language":"en",
            "resources-base-path":"/home/searchai_svc/searchai/configs/default/
↪configs",
            "syntactic-rules": "syntactic-rules.json"
        }],
        "network": {
            "host":"0.0.0.0",
            "port":10004,
            "associate-environment": {
                "host":"HOST_ENVNAME",
                "port":"PORT_ENVNAME"
            }
        },
        "runtime":{
            "request-max-size":100000000,
            "request-buffer-queue-size":100,
            "keep-alive":true,
            "keep-alive-timeout":5,
```

(continues on next page)

(continued from previous page)

```
            "graceful-shutown-timeout":15.0,
            "request-timeout":60,
            "response-timeout":60,
            "workers":20
        },
        "serialize":{
            "input":{
                "path":"/tmp",
                "keep-service-info":true
            },
            "output":{
                "path":"/tmp",
                "keep-service-info":true
            }
        }

    },
    "keywords": {
        "extractors":[{
            "language":"en",
            "resources-base-path":"/home/searchai_svc/searchai/thot/tests/data",
            "stopwords":"en.stopwords.lst",
            "use-lemma":true,
            "use-pos":true,
            "use-form":false
        }],
        "network": {
            "host":"0.0.0.0",
            "port":8080,
            "associate-environment": {
                "host":"HOST_ENVNAME",
                "port":"PORT_ENVNAME"
            }
        },
        "runtime":{
            "request-max-size":100000000,
            "request-buffer-queue-size":100,
            "keep-alive":true,
            "keep-alive-timeout":5,
            "graceful-shutown-timeout":15.0,
            "request-timeout":60,
            "response-timeout":60,
            "workers":20
        },
        "serialize":{
            "input":{
                "path":"/tmp",
                "keep-service-info":true
            },
            "output":{
                "path":"/tmp",
                "keep-service-info":true
            }
        }

    }
}
```

The search configuration allows to set up the search behaviour according the query analysis. The other configuration are not specific to the service.

- searching/document-index-name: the name of the index where are stored the documents

- searching/disable-document-analysis : document analysis is not mandatory and can be disable
- searching/search-policy/semantic-cluster/semantic-quantizer-model : path to clustering model to use "statistical" semantic
- searching/search-policy/settings/basic-querying/uniq-word-query : transform query in bag of word
- searching/search-policy/settings/basic-querying/boosted-uniq-word-query : weigthening words according to their frequency in query
- searching/search-policy/settings/basic-querying/cut-query": maximum number of uniq word with the query
- searching/search-policy/settings/advanced-querying/use-lemma:use lemmatised field of index
- searching/search-policy/settings/advanced-querying/use-keywords: use keywords field of index
- searching/search-policy/settings/advanced-querying/use-knowledge-graph: use knowledge graph (the triple) of index
- searching/search-policy/settings/advanced-querying/use-concepts: use concepts of the index
- searching/search-policy/settings/advanced-querying/use-sentences: use sentence querying
- searching/search-policy/settings/advanced-querying/querying/match-phrase-slop: slop in match phrase clause
- searching/search-policy/settings/advanced-querying/querying/match-phrase-boosting: default boosting value for match phrase
- searching/search-policy/settings/advanced-querying/querying/match-sentence/number-and-symbol-filtering": filter symbol andnumber from sentences
- searching/search-policy/settings/advanced-querying/querying/match-sentence/max-number-of-words: set the maximum length (words) in the sentence
- searching/search-policy/settings/advanced-querying/querying/match-keywords/match-keyword/number-and-symbol-filtering": filter number and symbols
- searching/search-policy/settings/advanced-querying/querying/match-keywords/semantic-skip-highest-ranked-classes: when you use semantic class (comming from clustering) the most common classes are often irrelevant, you can skip this classes
- searching/search-policy/settings/advanced-querying/querying/match-keywords/semantic-max-boosting": query boosting in match-phrase clause
- searching/search-policy/settings/advanced-querying/querying/match-svo/semantic-use-class-triple : create query clause with all semantic classes
- searching/search-policy/settings/advanced-querying/querying/match-svo/semantic-use-lemma-property-object": use lemma on subject, class on property and object
- searching/search-policy/settings/advanced-querying/querying/match-svo/semantic-use-subject-lemma-object": use lemma on property, class on subject and object
- searching/search-policy/settings/advanced-querying/querying/match-svo/semantic-use-subject-property-lemma": use lemma on object, class on subject an property
- searching/search-policy/settings/advanced-querying/querying/match-svo/semantic-use-lemma-lemma-object": use lemma on subject and property, class on object
- searching/search-policy/settings/advanced-querying/querying/match-svo/semantic-use-lemma-property-lemma": use lemma on subject ad object, class on property
- searching/search-policy/settings/advanced-querying/querying/match-svo/semantic-use-subject-lemma-lemma": use lemma on property and object, class on subject
- searching/search-policy/settings/advanced-querying/querying/match-svo/semanic-max-boosting":no yet implement
- searching/search-policy/settings/advanced-querying/querying/match-concept/concept-boosting: boost concept clause

- searching/search-policy/settings/advanced-querying/querying/match-concept/concept-pruning: top N of concept used
- searching/search-policy/settings/query-expansion/term-pruning: max number of term used in expansion
- searching/search-policy/settings/query-expansion/suppress-number: filter number
- searching/search-policy/settings/query-expansion/suppress-numberkeep-word-collection-thresold-under : frequency of document where the word appear should lesser than this frequency
- searching/search-policy/settings/query-expansion/word-boost-thresold-above: frequency of word in the document should be greater than this number
- searching/search-policy/settings/scoring/normalize-score: normalize elastic search score max score max
- searching/search-policy/settings/scoring/document-query-intersection-penalty: document and query intersection normalized : no-normalization, by-query-size, by-union-size(jaccard)
- searching/search-policy/settings/scoring/run-clause-separately: clause can be run separately in this case the ranked lists are merged, or put in a uniq query
- searching/search-policy/settings/scoring/expand-results": when you run clause separately and merge result it is interesting to expand result list size to cover more ranked documents
- searching/search-policy/settings/results/set-highlight: highlight snippets
- searching/search-policy/settings/resultsexcludes : excluded some fields from returned list

Run Search engine service

To run the command type simply from searchai directory:

```
python3 thot/search_svc.py --config=<path to relation configuration file>
```

CHAPTER

TWO

ABOUT THIS DOCUMENTATION

This documentation has been generated by Sphinx 1.8 (sphinx documentation). We use the format 'ReSructured-Text' (sphinx ReStructruredText basic documentation).

CHAPTER

THREE

CHANGELOG

Date	Description	Authors
2021/03	First Version 1.0	Eric Blaudez
2021/09	Update Version 1.0	Eric Blaudez

HTTP ROUTING TABLE

/api